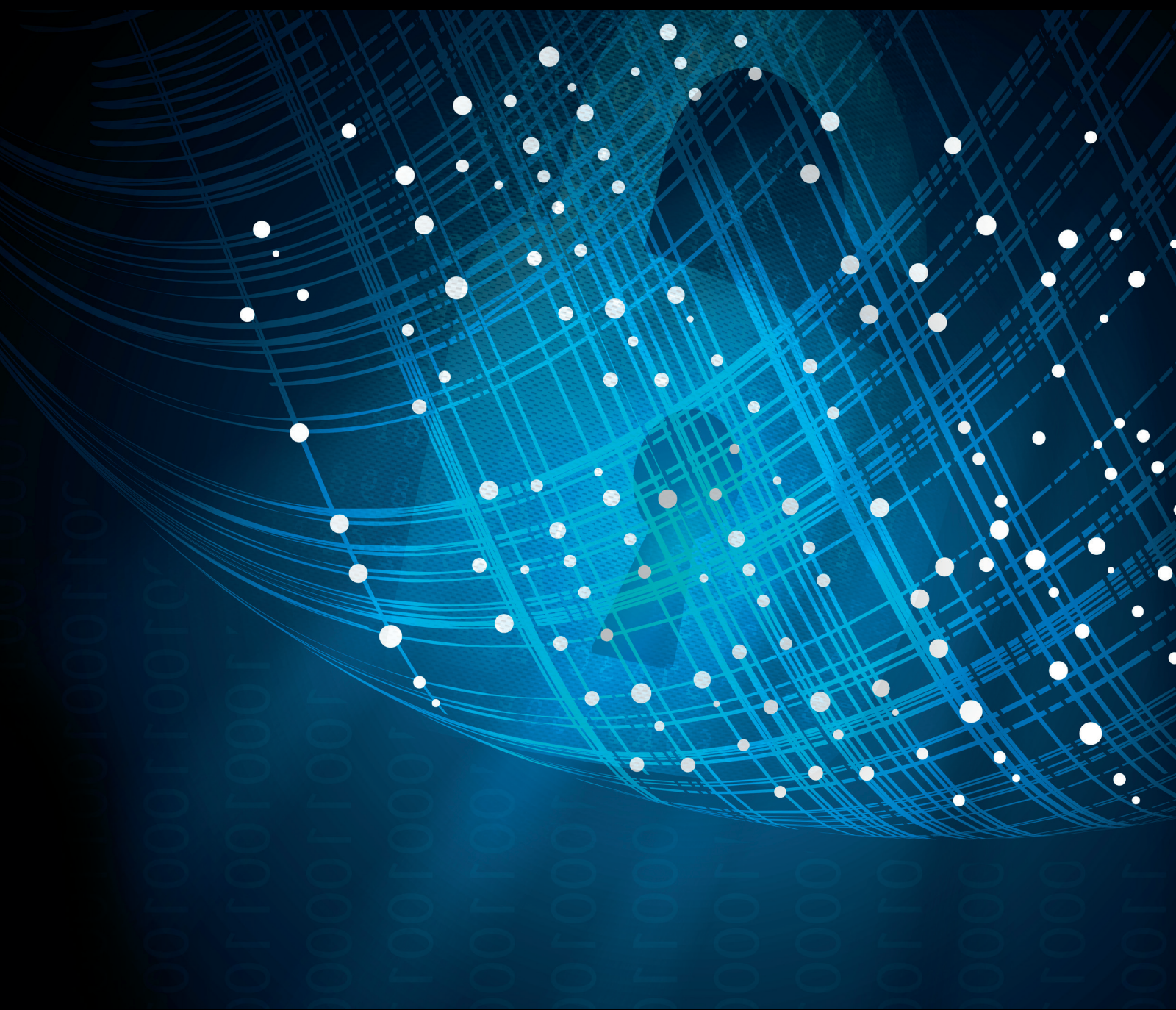
Security and Communication Networks

Applied Cryptography and Noise Resistant Data Security

Lead Guest Editor: Iqtadar Hussain
Guest Editors: Umar M. Khokhar, Amir Anees, and Fawad Ahmed



Applied Cryptography and Noise Resistant Data Security

Security and Communication Networks

Applied Cryptography and Noise Resistant Data Security

Lead Guest Editor: Iqtadar Hussain
Guest Editors: Umar M. Khokhar, Amir Anees, and Fawad Ahmed

Editorial Board

Contents

Hindawi
Security and Communication Networks
Volume 2018, Article ID 3962821, 2 pages
https://doi.org/10.1155/2018/3962821

Editorial

Applied Cryptography and Noise Resistant Data Security

Iqtadar Hussain,[1] **Fawad Ahmed**,[2] **Umar M. Khokhar**,[3] **and Amir Anees**[4]

[1]*Department of Mathematics, Statistics and Physics, College of Arts and Sciences, Qatar University, Doha, Qatar*
[2]*Department of Electrical Engineering, HITEC University, Taxila, Pakistan*
[3]*University System of Georgia, Georgia, USA*
[4]*Department of Computer Science, La Trobe University, Melbourne, Australia*

Correspondence should be addressed to Iqtadar Hussain; iqtadarqau@qu.edu.qa

Received 8 November 2018; Accepted 8 November 2018; Published 2 December 2018

There have been tremendous efforts exerted in field of secure communication by mathematicians and computer scientists. However, at the same time, there are security concerns that still must be properly addressed. Moreover, a critical requirement for a cipher is to be distortion tolerant. One of the most sophisticated ciphers, the Advanced Encrypted Standard (AES), is still unbreakable and widely used in all kinds of industrial applications. However, as the case with other modern and secured ciphers, the AES cannot tolerate any sort of channel noise. The solution for noise corruption is usage of error detection and correction modules, but it will significantly increase the overall computational complexity and may not be required in low profile applications. The other efficient solution can be the Noise Resistant Ciphers which should have the inherit property of tolerating any sort of noise. Besides for security protocols to be secure, robust, and noise resistant, power constraints should also be considered. In this regard, lightweight ciphers are of great importance to be synchronized with the modern hardware platforms. However, due importance to security attacks is also necessary in designing the lightweight ciphers. The goal followed in this special issue is to create a volume of recent works on advances in all aspects of cryptography, especially towards those ciphers which can tolerate channel noise. Besides data encryption, the focus is also on data hiding and copyright protection techniques as well. We have selected eleven research articles which deal with different aspects of cryptography:

In the paper entitled "High Embedding Capacity Data Hiding Algorithm for H.264/AVC Video Sequences without Intraframe Distortion Drift," D.-C. Nguyen et al. proposed a high-quality data hiding algorithm based on H.246/AVC without intraframe distortion drift in which quantized coefficients are clustered into two different groups.

In the paper entitled "A Robust Watermarking Scheme for Online Multimedia Copyright Protection Using New Chaotic Map," A. Anees et al. worked on copyright protection for contents of digital video. The proposed scheme can be applied on the other kinds of data as well, such as text, audio, and digital images.

In the paper entitled "A Vendor-Neutral Unified Core for Cryptographic Operations in GF(p) and GF(2^m) Based on Montgomery Arithmetic," M. Schramm et al. proposed a comprehensive adaptable hardware structure for efficient prime finite field and binary finite field arithmetic operations that expand the capabilities of single Montgomery Multiplier hardware designs.

In the paper entitled "A Novel Multiple-Bits Collision Attack Based on Double Detection with Error-Tolerant Mechanism," Y. Yuan et al. proposed a multiple-bits side-channel collision attack based on double distance voting detection and an improved version, involving the error-tolerant mechanism, which can find all 120 relations among 16 key bytes when applied to the AES.

In the paper entitled "An Efficient Certificateless Generalized Signcryption Scheme," B. Zhang et al. presented an

efficient certificateless generic signcryption scheme without utilizing bilinear pairing operations.

In the paper entitled "LWR-Based Fully Homomorphic Encryption, Revisited," F. Luo et al. presented the first workable LWR-based FHE scheme. Their FHE scheme erases the expensive Gaussian noise sampling and thus be an alternative to the LWE-based FHEs.

In the paper entitled "To Study the Effect of the Generating Polynomial on the Quality of Nonlinear Components in Block Ciphers," S. Mahmood et al. proposed a new design of S-box which has good properties such as nonlinearity, strict avalanche, bit independence, linear approximation probability, and differential approximation probability.

In the paper entitled "Side-Channel Attacks and Countermeasures for Identity-Based Cryptographic Algorithm SM9," Q. Zhang et al. discussed the implementation of SM9 algorithm and its Simple Power Attack and then presented the template attack and fault attack on SPA-resistant SM9.

In the paper entitled "Analysis of Software Implemented Low Entropy Masking Schemes," D. Li et al. analyzed the vulnerabilities on the mask sets of software Low Entropy Masking Schemes implementations.

In the paper entitled "Understanding Keystroke Dynamics for Smartphone Users Authentication and Keystroke Dynamics on Smartphones Built-In Motion Sensors," H. Lee et al. evaluated features with motion data and without motion data using various features including keystroke data such as time interval and motion data.

In the paper entitled "Under Quantum Computer Attack: Is Rainbow a Replacement of RSA and Elliptic Curves on Hardware?," H. Yi presented techniques to exploit Rainbow signature on hardware meeting the requirements of efficient high-performance applications.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

We would also like to thank all the reviewers who have participated in the review process of the articles submitted to this special issue and the special issue coordinators and acknowledge the technical support from the publishing team.

Iqtadar Hussain
Fawad Ahmed
Umar M. Khokhar
Amir Anees

Hindawi
Security and Communication Networks
Volume 2018, Article ID 2029869, 11 pages
https://doi.org/10.1155/2018/2029869

Research Article

High Embedding Capacity Data Hiding Algorithm for H.264/AVC Video Sequences without Intraframe Distortion Drift

Dinh-Chien Nguyen,[1,2] Thai-Son Nguyen,[3] Chin-Chen Chang,[1] Huan-Sheng Hsueh,[4] and Fang-Rong Hsu[1]

[1]*Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan*
[2]*Faculty of Information Technology, Hung Yen University of Technology and Education, Dan Tien, Khoai Chau, Hung Yen, Vietnam*
[3]*School of Engineering and Technology, Tra Vinh University, Tra Vinh Province, Vietnam*
[4]*Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan*

Correspondence should be addressed to Thai-Son Nguyen; thaison@tvu.edu.vn

Received 6 December 2017; Revised 5 June 2018; Accepted 18 July 2018; Published 1 August 2018

Academic Editor: Fawad Ahmed

Data hiding is a technique that allows secret data to be delivered securely by embedding the data into cover digital media. In this paper, we propose a new data hiding algorithm for H.264/advanced video coding (AVC) of video sequences with high embedding capacity. In the proposed scheme, to embed secret data into the quantized discrete cosine transform (QDCT) coefficients of *I* frames without any intraframe distortion drift, some embeddable coefficient pairs are selected in each block, and they are divided into two different groups, i.e., the embedding group and the averting group. The embedding group is used to carry the secret data, and the averting group is used to prevent distortion drift in the adjacent blocks. The experimental results show that the proposed scheme can avoid intraframe distortion drift and guarantee low distortion of video sequences. In addition, the proposed scheme provides enhanced embedding capacity compared to previous schemes. Moreover, the embedded secret data can be extracted completely without the requirement of the original secret data.

1. Introduction

With the rapid development of multimedia and network technology, a huge amount of digital media, i.e., images, video, audio, and texts, is transmitted each second in the public network, such as the Internet. Such transmitted media are easily modified or illegally copied by malicious attackers. As a result, the security of private information has become a very important issue. Therefore, many solutions have been proposed in the literature, and they can be divided into two different categories, i.e., encryption and data hiding techniques. Since the encryption technique converts the media into a meaningless form that will emphasize the importance of the media's content, this technique can actually result in attracting the attention of attackers. Data hiding is one promising technique to protect the security and privacy of the digital media because it embeds the secret data into the cover media. The embedded media that contain the secret data have a meaningful form, which helps avoid attracting the attention of attackers and can guarantee the security and privacy of the secret data.

Many data hiding schemes [1–7] have been proposed for different digital data in the last decade. After Richardson introduced the H.264/AVC video compression standard [8] in 2003, this standard has been used extensively for hiding secret data [9–19]. In 2005, Noorkami et al. [9] proposed a low complexity watermarking algorithm by using the relative change of the DC coefficients of the 4×4 block. In their scheme, a public key is used for determining the embedded data, while the copyright owner possesses a secret key. In 2006, Nguyen et al. [10] proposed a fast watermarking system based on H.264/AVC motion vectors. However, their scheme

offered low embedding capacity; i.e., the average embedding capacity was only about 2,000 bits. Then, to achieve robustness, Zhang et al. [11] proposed a new data hiding scheme with a 2D, 8-bit watermark in the compressed domain. Noorkami and Mersereau [12] introduced a framework for robust watermarking of H.264 encoded video by using the quantized AC coefficient to obtain optimal detection of video watermarking. By using a texture masking-based perceptual model, Gong et al. [13] proposed a fast and robust watermarking scheme for H.264 video. In this scheme, the quantized DC coefficients were used to conceal the watermark. However, in these two schemes [12, 13], the original watermark is required for detection. In 2007, Kapotas et al. [15] proposed blind data hiding in an H.264 stream by using the difference of block sizes during the interframe prediction stage to increase the embedding capacity. However, Kapotas et al.'s scheme had a high bit-rate increment. To solve this issue, Kim et al. [16] embedded the watermark bit into the sign bit of the trailing ones in Context Adaptive Variable Length Coding (CAVLC). However, embedding the watermark in the discrete cosine transform (DCT) coefficients of the I-frames in these previous schemes result in stego videos that have low visual quality, which is caused by the distortion drift in the I-frame prediction. To overcome this shortcoming, in 2010, Ma et al. [17] proposed a new algorithm for data hiding in H.264/AVC based on DCT coefficients. Their scheme used several paired-coefficients of a 4 × 4 macroblock to embed the secret data to avoid distortion drift. However, this scheme obtained limited visual quality. To improve the visual quality of stego videos, Huo et al. [14] proposed a new data hiding scheme by using the controllable error drift-elimination technique. However, unsatisfactory embedding capacity was obtained by their scheme. To increase the embedding capacity of Ma et al.'s scheme, Lin et al. [18] fully used the remaining luminance blocks to hide the secret data. Their experimental results indicated that their embedding capacity was improved further when the increase of embedding capacity obtained by Lin et al.'s scheme is 0.15 bits per pixel (bpp) more than that of Ma et al.'s scheme. However, the average embedding capacity of these two schemes [17, 18] is still low, i.e., always smaller than 0.68 bpp, because they use a pair of DCT coefficients to hide the secret bit separately. By doing so, as the amount of embedded secret data increases, the the distortion of the video becomes greater. To overcome these shortcomings, we propose in this paper a new data hiding scheme for video sequences without intraframe distortion drift. Instead of using the coefficient pair separately for embedding data, all embeddable coefficient pairs in each luminance block are determined and classified into two different clusters, i.e., the embedding group and the averting group. The secret data are hidden in the embedding group by minimum modification, while the averting group is used to avoid distortion drift. The experimental results showed that the proposed algorithm further improved the embedding capacity while maintaining good visual quality and no distortion drift.

The remainder of this paper is organized as follows. Section 2 provides information about intraframe prediction and introduces the previous data hiding schemes. The proposed scheme is explained in Section 3. In Section 4, experimental results are presented that illustrate the performance of the proposed scheme in comparison with some previous schemes. Our conclusion is presented in Section 5.

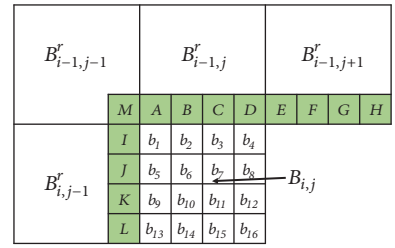


Figure 1: Predicted pixels in block, $B_{i,j}$, and the reference pixels in the adjacent region.

2. Related Work

2.1. Intraframe Prediction. Intraframe prediction is a technique in H.264/AVC coding [7] that is used to reduce the spatial redundancies of H.264/AVC intraframes. In H.264/AVC coding, for each block, some previously encoded adjacent blocks are used to predict the pixels of the current block. Figure 1 shows the current 4 × 4 block, $B_{i,j}$, with its pixels labeled from b_1 to b_{16}. These pixels are predicted based on the reference pixels (labeled from A to M) of four adjacent blocks. These four blocks were encoded previously by using a prediction formula corresponding to the selected optimal prediction mode from nine prediction modes of each 4 × 4 block in Figure 2.

In H.264/AVC encoding, the current block, $B_{\mathrm{i,j}}$, is subtracted from its prediction block, $P_{\mathrm{i,j}}$, to obtain the residual block, $R_{i,j} = B_{i,j} - P_{i,j}$. Then, the residual block, $R_{\mathrm{i,j}}$, is encoded further by the following processes of H.264/AVC coding, i.e., transformation, quantization, and entropy coding. For simplicity, 4 × 4 integer DCT transformation and quantization processes are implemented on $R_{i,j}$ to generate the corresponding quantized DCT coefficient matrix $R^q_{i,j}$ as follows:

$$R^q_{i,j} = \left(C_f R_{i,j} C_f^T\right) . \times \frac{PF}{Q} = \begin{bmatrix} Y_{00} & Y_{01} & Y_{02} & Y_{03} \\ Y_{10} & Y_{11} & Y_{12} & Y_{13} \\ Y_{20} & Y_{21} & Y_{22} & Y_{23} \\ Y_{30} & Y_{31} & Y_{32} & Y_{33} \end{bmatrix}, \quad (1)$$

where $C_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$, $PF = \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix}$, $a = 1/2$, $b = \sqrt{2/5}$, and Q is the size of the quantization step which is defined by quantization parameter (QP).

After H.264/AVC coding, the block $B_{\mathrm{i,j}}$ is represented by the prediction block $P_{\mathrm{i,j}}$ and the quantized DCT coefficients $R^q_{i,j}$. To encode the next block, $B_{\mathrm{i,j+1}}$, the encoded block, $B_{\mathrm{i,j}}$, should be decompressed from $P_{\mathrm{i,j}}$ and $R^q_{i,j}$. The dequantization and inverse DCT operations are implemented on $R^q_{i,j}$ to reconstruct the values of $R_{\mathrm{i,j}}$, denoted as $R^r_{i,j}$, and the

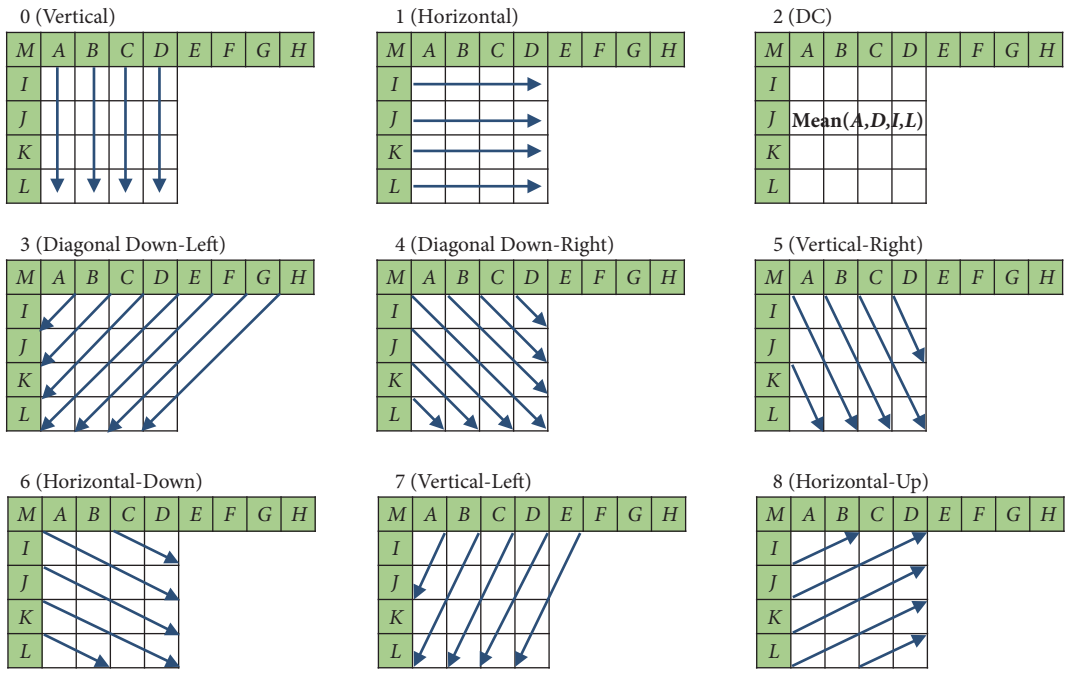


Figure 2: Nine prediction modes.

reconstructed block, denoted as $B^r_{i,j}$, is calculated as $B^r_{i,j} = P_{i,j} + R^r_{i,j}$.

In the decoding phase, the residual block, $R^r_{i,j}$, is reconstructed using the dequantization process and the 4×4 integer inverse DCT transformation on $R^q_{i,j}$ by (2), and the reconstructed block is obtained as $B^r_{i,j} = P_{i,j} + R^r_{i,j}$.

$$R^r_{i,j} = C^T_r \left(R^q_{i,j}. \times Q. \times PF\right) C_r, \tag{2}$$

where $C_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix}$.

2.2. Data Hiding Schemes Based on Intraframe Prediction. In 2010, Ma et al. [17] used three pairs of quantized DCT coefficients in $R^q_{i,j}$ for embedding data into H.264/AVC video sequences. To prevent intraframe distortion drift during the embedding process, they analyzed the use of seven pixels, i.e., $b^r_4, b^r_8, b^r_{12}, b^r_{13}, b^r_{14}, b^r_{15}, b^r_{16}$, in the reconstructed block, $B^r_{i,j}$, for the intraprediction process. Figure 3 shows the four adjacent blocks, i.e., $B_{i,j+1}$, $B_{i+1,j-1}$, $B_{i+1,j}$, and $B_{i+1,j+1}$, that are affected directly by the above process. For example, if the selected prediction mode of $B_{i+1,j}$ is 0, as shown in Figure 2, then $b^r_{13}, b^r_{14}, b^r_{15}, b^r_{16}$ are modified by embedding data into some coefficients of $R^q_{i,j}$.

During the embedding process, the seven pixels, b^r_4, b^r_8, $b^r_{12}, b^r_{13}, b^r_{14}, b^r_{15}, b^r_{16}$, and the selected prediction modes of the four adjacent blocks are classified into three different conditions, i.e., Con_1, Con_2, and Con_3, which are defined as follows. Con_1 consists of the prediction modes in $\{1, 2, 4, 5, 6, 8\}$. This means the pixels located at the position of b^r_4, b^r_8, b^r_{12}, and, b^r_{16} are referenced for predicting $B_{i,j+1}$. Con_2 consists of the prediction modes in $\{0, 2, 3, 4, 5, 6, 7\}$. This indicates that

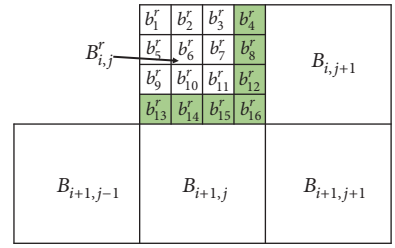


Figure 3: The current block and the four blocks directly affected by the encoding process.

the pixels located at the positions of $b^r_{13}, b^r_{14}, b^r_{15}$, and b^r_{16} are referenced for predicting $B_{i+1,j}$ or $B_{i+1,j-1}$. Con_3 consists of the prediction modes in $\{4, 5, 6\}$. This means that the pixel located at the position of b^r_{16} is referenced for predicting $B_{i+1,j+1}$.

To take advantage of the relationship of the reference pixels and the selected prediction modes for embedding data, these three conditions also can be classified into the five categories presented in Table 1.

To solve the drift distortion problem during embedding the secret data, Ma et al. classified the current block into three different conditions, i.e., Con_1, Con_2, and Con_3. Then three specific pairs of quantized DCT coefficients are selected for embedding three secret data bits. Take the category Con_2 as an example, three coefficient pairs, i.e., $\{(Y_{01}, Y_{21}), (Y_{02}, Y_{22}), (Y_{03}, Y_{23})$, are used for embedding. The main reason is that these three pairs have the same property; i.e., when the values of quantized DCT coefficients are modified in a pair, the modification will be concentrated only on the two middle columns or the two middle rows of the block.

Table 1: Five general categories of reference pixels and selected prediction modes.

	Con_1	Con_2	Con_3	Reference pixels
Cat_1	T	F	X	b_4^r, b_8^r, b_{12}^r, b_{16}^r
Cat_2	F	T	X	$b_{13}^r, b_{14}^r, b_{15}^r, b_{16}^r$
Cat_3	F	F	F	Nrp
Cat_4	F	F	T	b_{16}^r
Cat_5	T	T	X	All

T: true, F: false, X: do not care, Nrp: not reference pixel, and all: all seven reference pixels.

For example, consider the quantized DCT coefficient pair (Y_{03}, Y_{23}) of $R_{i,j}^q$. Assume that Y_{03} is added by the value of v to embed the secret data, i.e., $\Delta = \begin{bmatrix} 0 & 0 & 0 & v \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$. Then, the modification will be spread to all pixels of the 4×4 block as $D = \begin{bmatrix} abtQ/4 & -abtQ/2 & abtQ/2 & -abtQ/4 \\ abtQ/4 & -abtQ/2 & abtQ/2 & -abtQ/4 \\ abtQ/4 & -abtQ/2 & abtQ/2 & -abtQ/4 \\ abtQ/4 & -abtQ/2 & abtQ/2 & -abtQ/4 \end{bmatrix}$, which will propagate to other adjacent blocks. However, in the Ma et al.'s scheme, to embed a hidden bit v in (Y_{03},Y_{23}), the quantized DCT coefficient pair (Y_{03},Y_{23}) is perturbed to $(Y_{03}+v, Y_{23} - v)$, i.e., $\Delta = \begin{bmatrix} 0 & 0 & 0 & v \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -v \\ 0 & 0 & 0 & 0 \end{bmatrix}$. Then, the modification will be $D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ abtQ/2 & -abtQ & abtQ & -abtQ/2 \\ abtQ/4 & -abtQ & abtQ & -abtQ/2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.

In this scenario, it is clear that the modification is concentrated only on the two middle rows, while the modifications in the first and the last rows are zeros. Therefore, the distortion drift is avoided. However, they did not fully explore all of the cases for embedding data, so their average embedding rate was less than 0.45 bpp.

To further improve the embedding capacity of Ma et al.'s scheme while avoiding the distortion drift in the H.264/AVC video sequences, Lin et al. [18] have divided the relationship of the reference pixels and the selected prediction modes into five categories presented in Table 1. Then, for the block belongs to three categories, i.e., Cat_1, Cat_2, and Cat_4, Lin et al. extracted one more coefficient pair for embedding one more secret bit by the same way as was done by Ma et al.'s scheme. In addition, each block belonging to Cat_5 is also used for embedding one more secret bit to increase embedding capacity. As a result, the embedding capacity obtained by Lin et al.'s scheme is 0.15 bits per pixel (bpp) higher than that of Ma et al.'s scheme. However, in these two schemes [17, 18], each pair of quantized DCT coefficients is subsequently perturbed to embed only one secret bit. Thus, to embed n secret bits, n selected pairs of quantized DCT coefficients are modified. This means that the more secret bits are embedded, the more distortion will cause in the video frames, leading to low visual quality of the video frames. In the schemes [17, 18], to guarantee the higher visual quality, if (Y_{mn}, Y_{pq}) is a zero coefficient pair, it is not used to embed any secret data bits. Therefore, their embedding capacity is still low when the average embedding capacity is smaller than 0.68 bpp when QP = 28 is used.

It is obvious that, to maintain the high visual quality of video sequence, the previous schemes [17, 18] have selected quantized DCT coefficient pairs (excluding the zero coefficient pairs) of three categories, Cat_1, Cat_2, and Cat_4 for embedding data. However, their schemes still obtained low embedding capacity, while the visual quality of video sequences is not guaranteed. Therefore, in this paper, to overcome their shortcoming, instead of modifying each coefficient pair for embedding data, the group of coefficients are selected and altered at the same time. In particular, all suitable pairs of quantized DCT coefficients are extracted and classified into two different groups, one group is used for embedding data and the other one is used to prevent distortion drift of video sequences. This means that, in the proposed scheme, the group of n coefficient pairs is modified to embed n secret bits. By modifying by the group, at most $n/2$ coefficient pairs are modified which guarantees the better visual quality of the embedding video sequences. In addition, to increase embedding capacity, in the proposed scheme, zero coefficient pairs are still used for embedding data. The details of the proposed scheme are described in the next section.

3. The Proposed Scheme

Figure 4 shows all of the main processes of the proposed embedding phase and extracting phase. In the embedding phase, the original H.264/AVC video sequence is first decoded by entropy coding. Then, the 4×4 quantized DCT blocks meet three cases, i.e., Cat1, Cat2, and Cat4; four secret data bits are embedded by based on group modulation. And if the blocks belong to the category Cat3, each coefficient is used to contain one secret bit. Then, all the quantized DCT coefficients are entropy encoded to get the embedded H.264/AVC video sequences. In this phase, to prevent the distortion drift in the proposed scheme, quantized DCT coefficients of each block are partitioned into two groups, i.e., the embedding group and the averting group. Then, to achieve high embedding capacity and to ensure good image quality, the embedding group is used for embedding data, while the averting group is used for preventing the proliferation of errors. Figure 4(b) shows the detail of the extracting phase. The embedded H.264/AVC video sequence is entropy decoded. Then, the category of the 4×4 quantized DCT blocks is determined. After that according to the determined category, the corresponding secret bits are extracted.

3.1. Category Selection and Coefficient Grouping. To prevent intraframe distortion drift in the proposed scheme, all blocks that belong to the first four categories are selected for embedding data by suitable ways. Therefore, the pixels that are used for intraframe prediction are not used during the embedding process so that the embedding distortion would not affect the other adjacent blocks. Figure 5 shows the percentage of the blocks which meet the conditions of the first four categories of the 14 H.264/AVC test video sequences. It is obvious that most of the blocks in each video sequence belong to Cat_1 and Cat_2. Therefore, the proposed scheme is designed to embed more secret bits into these two categories with small distortion and without distortion drift.

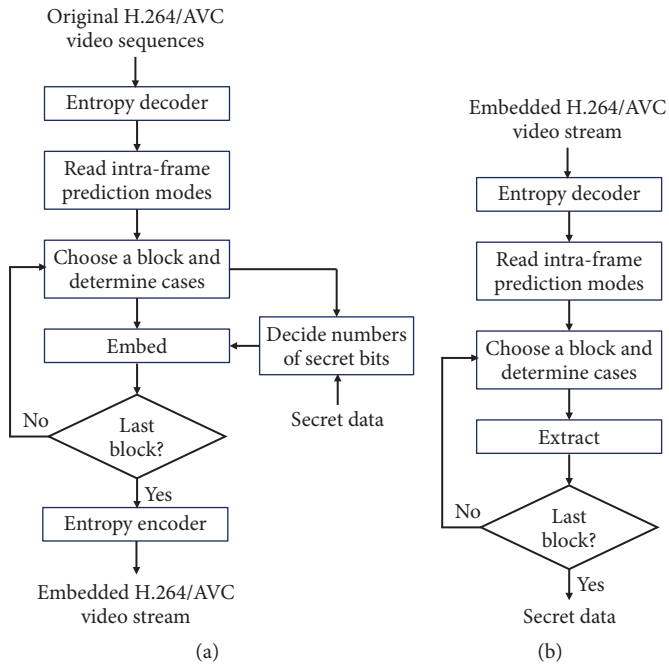


Figure 4: Main processes of the proposed scheme: (a) embedding phase and (b) extracting phase.

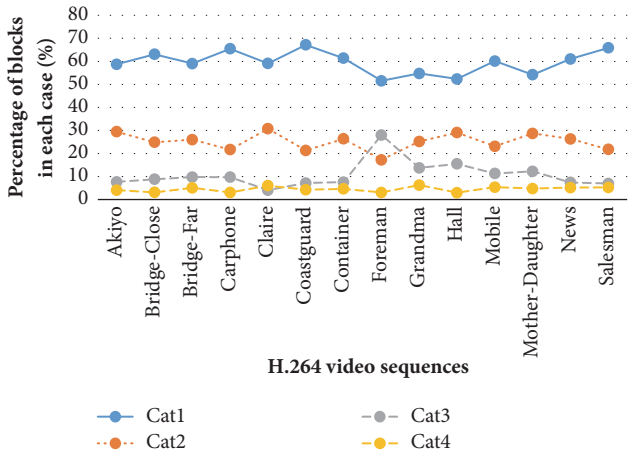


Figure 5: Percentage of blocks that meet the first four categories.

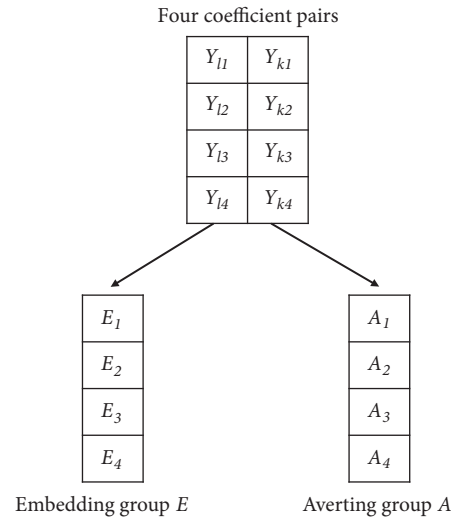


Figure 6: Example of coefficient grouping.

In the proposed scheme, four quantized DCT coefficient pairs of these three categories are also selected and divided into two groups, i.e., an embedding group E and an averting group A, to conceal the secret data. Take the block B which belongs to category Cat_1 as an example, four coefficient pairs, $(Y_{li}, Y_{ki}) \in \{(Y_{00}, Y_{02}), (Y_{10}, Y_{12}), (Y_{20}, Y_{22}), (Y_{30}, Y_{32})\}$, are selected for embedding data. Then, all of the first coefficients Y_{li} of each pair are grouped to construct the embedding group, E, which is used to carry the secret bits by modifying, at most, two coefficients, whereas the remaining coefficients Y_{ki} of four pairs are clustered into the averting group A, which is modified to prevent intraframe distortion drift during embedding process. Figure 6 shows an example of the grouping process of Cat_1.

3.2. Embedding Phase. In this subsection, the embedding algorithm is described in detail. First, for security reasons, the secret data are encrypted in advance to $S = s_1, s_2, \ldots, s_{|S|}$, and $s_r \in \{0, 1\}$. Original video sequences are decoded by entropy decoder to get the intraframe prediction modes and quantized DCT coefficients. Then, if the 4×4 quantized DCT blocks belong to Cat_1, Cat_2, Cat_4, the secret data are embedded by based on group modulation. Otherwise, if the blocks belong to the category Cat_3, each coefficient is used to contain one secret bit. Then, all the quantized DCT coefficients are entropy encoded to get the target embedded

video sequences. To make the embedding algorithm clearer, two cases are used for embedding data as an example:

Case 1. If the blocks belong to Cat_1, Cat_2, and Cat_4, the four steps are implemented for embedding n secret bits as follows.

Step 1. Read four appropriate pair-coefficients and classify them into two different groups, i.e., E and A, as was done in Section 3.1.

Step 2. Read n secret bits, s_1, s_2,..., s_n, from the encrypted message and embed them into group E. For embedding, the weight value W of group E is calculated by (3). In this paper, four coefficient pairs are used for embedding; therefore, the value of n could be set to at most 4.

$$W = \left(\sum_{i=1}^{n} E_i \times i\right) \bmod (2^n) \tag{3}$$

And, the difference value, d, is calculated by

$$d = dec(s_1 s_2 \dots s_n) - W, \tag{4}$$

where $dec(s_1 s_2 \dots s_n)$ is the decimal value of n secret bits, $s_1 s_2 \dots s_n$.

Step 3. If the value of d is 0, all elements of group E are kept unchanged. Otherwise, we can arbitrarily increase or decrease the elements of group E by 1. If E_i increases by 1, the values of W will be increased by 1 to 4. Whereas, if E_i decreases by 1, the values of W will be decreased by $2^n - 4$ to $2^n - 1$. It can be observed that, in the proposed scheme, at most, two elements in group E are modified by the value 1. Therefore, we can alter E to E' by changing; at most, two elements in E to satisfy $W' \equiv dec(s_1 s_2 \dots s_n) \bmod (2^n)$.

Step 4. Preventing the drift of intraframe distortion, the inverse operations of modifying the group E are performed on the corresponding elements of the group A. For example, if the element E_i increases by 1, the element A_i will decrease by 1 and vice versa.

Similarly, when the block $B_{i,j}$ satisfies one of two categories, i.e., Cat_2 and Cat_4, four coefficient pairs, $(Y_{li}, Y_{ki}) \in \{(Y_{00}, Y_{20}), (Y_{01}, Y_{21}), (Y_{02}, Y_{22}),$ and $(Y_{03}, Y_{23})\}$, are selected to generate the embedding and averting groups, E and A; then n secret bits are embedded into the group by the same manner as was done for Cat_1.

Case 2 (the block belongs to category Cat_3). In such blocks, each coefficient is used to embed one secret bit because this category does not have any effect on other adjacent blocks during the encoding process. Then, the stego coefficient, Y'_{ij}, is calculated by

$$Y'_{ij} = \begin{cases} Y_{ij} + 1 & if\ \left(Y_{ij}\ is\ even,\ s_r = 1,\ and\ Y_{ij} \geq 0\right) \\ & or\ \left(Y_{ij}\ is\ odd,\ s_r = 0,\ and\ Y_{ij} \geq 0\right) \\ Y_{ij} - 1 & if\ \left(Y_{ij}\ is\ even\ s_r = 1,\ and\ Y_{ij} < 0\right) \\ & or\ \left(Y_{ij}\ is\ odd,\ s_r = 0,\ and\ Y_{ij} < 0\right) \\ Y_{ij} & otherwise \end{cases} \tag{5}$$

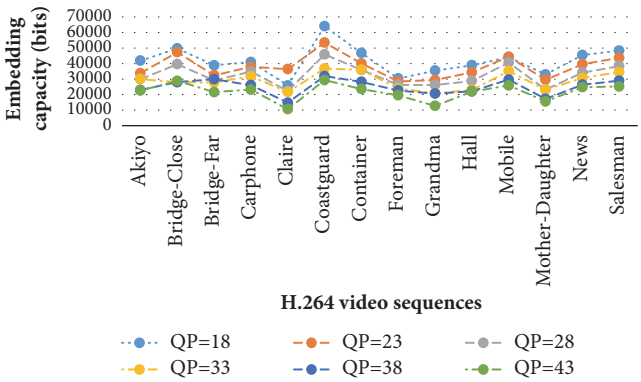


Figure 7: Embedding capacity of the proposed scheme for different QPs.

where Y_{ij} is one of 16 quantized DCT coefficients in $R^q_{i,j}$. Here, if the block $B_{i,j}$ belongs to Cat_5, we leave it without embedding any secret bits.

3.3. Extracting Phase. In this subsection, the extracting algorithm is used to extract the secret data from the embedded H.264/AVC video sequences. If the current block belongs to Cat_1, Cat_2, or Cat_4, the embedding group E will be reconstructed as was done in the embedding phase. Then, n embedded bits are extracted by

$$(s_1 s_2 \dots s_n) = bin\left(\left(\sum_{i=1}^{n} E_i \times i\right) \bmod (2^n)\right). \tag{6}$$

For the block that belongs to Cat_3, each coefficient is checked to determine the embedded bit, s_r, by using

$$s_r = \begin{cases} 1 & if\ Y_{ij}\ is\ odd \\ 0 & otherwise. \end{cases} \tag{7}$$

4. Experimental Results

In this section, we describe the experimental evaluation of the performance of the proposed scheme. Fourteen video sequences, i.e., Akiyo, Bridge-Close, Bridge-Far, Carphone, Claire, Coastguard, Container, Foreman, Grandma, Hall, Mobile, Mother-Daughter, News, and Salesman, were used as test samples. The size 30 of the group of pictures (GOP) and the structure of "IBPBP" were used in the experiment. Six different quantization steps (QP), i.e., 18, 23, 28, 33, 38, and 43, were checked for the 14 video sequences mentioned above. In principle, in H.264/AVC coding, if a small value of QP is used, the better visual quality of video sequences is obtained and the more encoded bits are required.

4.1. Embedding Capacity Evaluation. Figure 7 shows the performance of the proposed scheme, in terms of embedding capacity, using six QPs. It is apparent that using a smaller value of QP resulted in a higher embedding capacity.

Table 2 compares the embedding capacity of the proposed scheme and two state-of-the-art schemes [17, 18]. As shown

Table 2: Comparison of embedding capacity of the proposed scheme and two previous schemes [17, 18] in terms of embedded bits per 4 × 4 block.

QP	Ma et al.	Lin et al.	Proposed	Improvement rate	
				Ma et al.	Lin et al.
18	0.97	1.32	1.46	51%	11%
23	0.69	0.94	1.33	93%	41%
28	0.49	0.67	1.14	133%	70%
33	0.24	0.31	1.00	317%	223%
38	0.10	0.13	0.87	770%	569%
43	0.03	0.04	0.76	2433%	1800%
Average	***0.42***	***0.57***	***1.09***	***160%***	***91%***

Table 3: Embedding capacity comparison of the proposed scheme and two previous schemes [17, 18] for QP = 28.

Sequences	Ma et al.	Lin et al.	Proposed	Improvement rate	
				Ma et al.	Lin et al.
Akiyo	0.36	0.47	1.04	189%	121%
Bridge-Close	0.58	0.76	1.39	140%	83%
Bridge-Far	0.14	0.20	1.02	629%	410%
Carphone	0.44	0.59	1.22	177%	107%
Claire	0.19	0.26	0.78	311%	200%
Coastguard	1.00	1.27	1.61	61%	27%
Container	0.49	0.65	1.26	157%	94%
Foreman	0.38	0.56	0.92	142%	64%
Grandma	0.30	0.43	0.92	207%	114%
Hall	0.55	0.74	1.05	91%	42%
Mobile	1.16	1.27	1.42	22%	12%
Mother-Daughter	0.27	0.36	0.81	200%	125%
News	0.56	0.74	1.19	113%	61%
Salesman	0.55	0.76	1.34	144%	76%
Average	***0.49***	***0.67***	***1.14***	***133%***	***70%***

in the table, the average embedding capacity of the proposed scheme was considerably higher than those of the two state-of-the-art schemes. The average improvement in embedding capacity of our proposed scheme over the schemes of Ma et al. [17] and Lin et al. [18] were 160 and 91%, respectively. The main reason for this improvement over Ma et al.'s scheme was that they only selected three coefficient pairs for carrying secret bits, which resulted in low embedding capacity. Lin et al.'s scheme also had a lower embedding capacity than the proposed scheme because they did not use coefficient pairs with value of zero for embedding data to avoid significant distortion in the video intraframes. In our proposed scheme, all of the blocks that belonged to the first four categories were used to embed secret data with a small modification. As a result, the proposed scheme achieved higher embedding capacity. Specifically, Table 3 shows the embedding capacity of the three schemes for 14 test video sequences when the value of QP was 28. As Table 3 shows, the gain in embedding capacity of the proposed scheme ranged from 22 to 629% better than that of Ma et al.'s scheme and from 12 to 410% better than that of Lin et al.'s scheme. The value of the improvement rate was different for the 14 video sequences because the selected prediction modes of blocks were based on the content of each video sequence.

Table 4: Comparison of bit rate increment ratio of the proposed scheme and two previous schemes [17, 18].

QP	Ma et al.	Lin et al.	Proposed
18	0.24%	0.65%	0.64%
23	0.82%	1.22%	1.12%
28	1.71%	2.06%	1.78%
33	2.29%	2.53%	2.71%
38	2.19%	2.28%	2.96%
43	1.39%	1.33%	2.87%
Average	***1.44%***	***1.68%***	***2.01%***

4.2. Bit-Rate Increment Ratio. Table 4 shows the ratio of the bit-rate increment of different QPs. The average ratios of the bit-rate increment of Ma et al.'s scheme, Lin et al.'s scheme, and the proposed scheme were 1.44%, 1.68%, and 2.01%, respectively. These results indicated that the degradation was quite small in all three schemes.

Table 5: Comparison of visual quality (PSNR: dB) of the proposed scheme and two previous schemes [17, 18].

QP	Ma et al.	Lin et al.	Proposed
18	42.66	42.21	42.32
23	39.22	38.74	38.21
28	35.71	35.21	35.26
33	32.94	32.65	31.57
38	30.07	29.91	29.27
43	27.53	27.49	25.68
Average	***34.69***	***34.37***	***33.71***

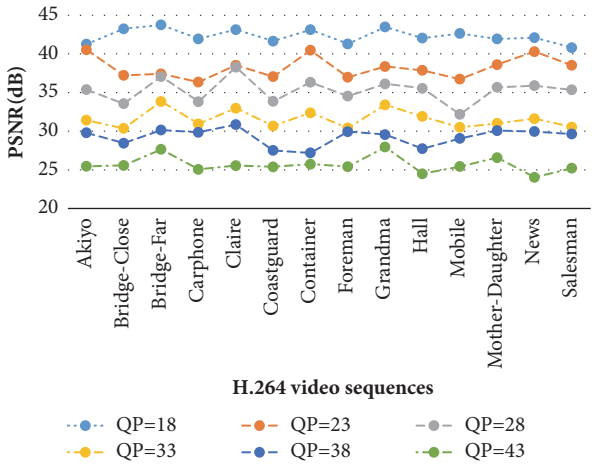


Figure 8: Visual quality of the proposed scheme for different QPs.

4.3. Visual Quality Evaluation. To evaluate the visual quality of video frames in the three schemes, the peak signal-to-noise ratio (PSNR) [20] is calculated by comparing the original frame to the embedded frame. Figure 8 shows the performance of the proposed scheme in the visual quality with different value of QP. It clear to see that the higher PSNR is obtained when the smaller QP is used.

Table 5 shows comparison results of the proposed scheme, Ma et al.'s scheme, and Lin et al.'s scheme in terms of visual quality of the video frames. It can be observed that the PSNR of the proposed scheme was slightly smaller than that of other two schemes [17, 18]. However, the proposed scheme can improve embedding capacity significantly; i.e., the average improvement rates were 160 and 91% over Ma et al.'s scheme and Lin et al.'s scheme, respectively. Table 6 compares the visual quality of the three schemes for QP = 28, corresponding to the embedding capacity in Table 3. Compared with Ma et al.'s scheme, the average degradation of the proposed scheme was larger than 0.45 dB. However, the proposed scheme provided better visual quality of video frames than Lin et al.'s scheme.

For a fair comparison, both values of PSNR and the structural similarly (SSIM) [21] index are used to evaluate the visual quality of the proposed scheme and two other schemes [17, 18]. Here, the max embedding capacity obtained by Lin et al.'s scheme [17] is embedded into the proposed scheme.

Table 6: Comparison of visual quality (PSNR: dB) of the proposed scheme and two previous schemes [17, 18] for QP = 28.

Sequences	Ma et al.	Lin et al.	Proposed
Akiyo	37.32	36.77	35.38
Bridge-Close	34.01	33.55	33.57
Bridge-Far	37.58	37.20	37.08
Carphone	36.05	35.57	33.82
Claire	39.48	39.02	38.30
Coastguard	33.45	32.96	33.86
Container	35.25	34.72	36.33
Foreman	35.73	35.19	34.54
Grandma	36.17	35.67	36.13
Hall	35.70	35.04	35.56
Mobile	32.35	31.67	32.18
Mother-Daughter	36.47	36.41	35.68
News	35.74	35.12	35.89
Salesman	34.67	34.08	35.35
Average	***35.71***	***35.21***	***35.26***

Table 7, Figures 9 and 10 show that the proposed scheme successfully preserved the high visual quality of the video sequences. The average PSNR of the proposed scheme was better than those of Ma et al.'s and Lin et al.'s schemes, at 3.39 and 3.89 dB, respectively. As can be seen in Figure 9, the PSNR obtained by the proposed scheme is always better than that of two previous schemes [17, 18]. In terms of SSIM, in Figure 10, the proposed schemes showed better performance in some video sequences, i.e., Bridge-Close, Bridge-Far, and Coastguard, while two previous schemes [17, 18] obtained higher values of SSIM in some other video sequences, i.e., Carphone and Claire. However, the SSIM obtained by the proposed scheme also was higher than those of the two previous schemes [17, 18]. It can be concluded that, based on coefficient grouping for embedding data, the proposed scheme prevented intraframe distortion drift and improved embedding capacity further while maintaining good visual quality of the embedded video sequences. In addition, Figure 11 shows the visual effect in video sequences by three schemes. It is obvious that three schemes reveal the same visual quality observation.

5. Conclusion

In this paper, a high-quality data hiding algorithm based on H.246/AVC is proposed without intraframe distortion drift. In the proposed scheme, the quantized coefficients are clustered into two different groups, i.e., the embedding group and the averting group. Then, the embedding group is used to carry secret bits to preserve high visual quality and further improve embedding capacity. In addition, the averting group is used to avoid the intraframe distortion drift. The experimental results demonstrated that the embedding capacity increased significantly in the proposed scheme while guaranteeing the good visual quality of embedded video sequences.

Table 7: Comparison of visual quality (PSNR and SSIM) of the proposed scheme and two previous schemes [17, 18] for QP = 28.

Sequences	Capacity	Ma et al.		Capacity	Lin et al.		Proposed	
		PSNR	SSIM		PSNR	SSIM	PSNR	SSIM
Akiyo	0.36	37.32	0.990	0.47	36.77	0.989	41.41	0.990
Bridge-Close	0.58	34.01	0.965	0.76	33.55	0.963	37.18	0.982
Bridge-Far	0.14	37.58	0.965	0.20	37.20	0.964	47.89	0.977
Carphone	0.44	36.05	0.992	0.59	35.57	0.991	36.44	0.983
Claire	0.19	39.48	0.994	0.26	39.02	0.994	44.21	0.985
Coastguard	1.00	33.45	0.935	1.27	32.96	0.931	35.40	0.975
Container	0.49	35.25	0.958	0.65	34.72	0.957	40.28	0.982
Foreman	0.38	35.73	0.989	0.56	35.19	0.987	35.37	0.980
Grandma	0.30	36.17	0.982	0.43	35.67	0.980	37.74	0.974
Hall	0.55	35.70	0.990	0.74	35.04	0.988	37.19	0.975
Mobile	1.16	32.35	0.981	1.27	31.67	0.979	33.91	0.991
Mother-Daughter	0.27	36.47	0.985	0.36	36.41	0.985	43.07	0.982
News	0.56	35.74	0.988	0.74	35.12	0.987	39.37	0.976
Salesman	0.55	34.67	0.976	0.76	34.08	0.974	37.96	0.986
Average	***0.49***	***35.71***	***0.978***	***0.65***	***35.21***	***0.976***	***39.10***	***0.981***

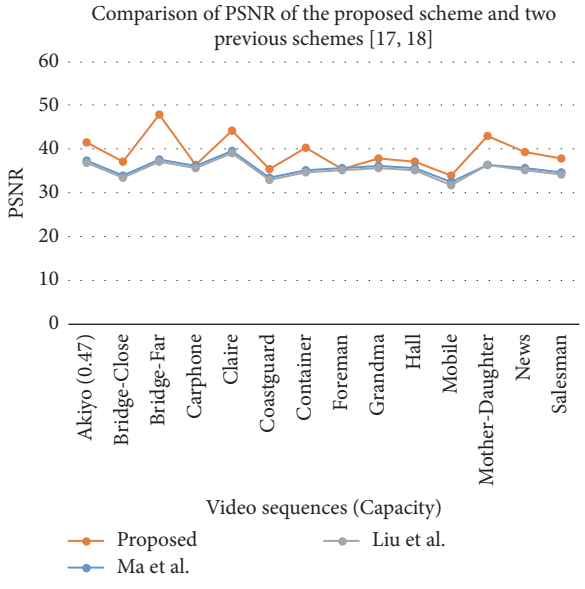


Figure 9: Visual quality (PSNR) performances of the proposed scheme and two previous schemes [17, 18] with different video sequences.

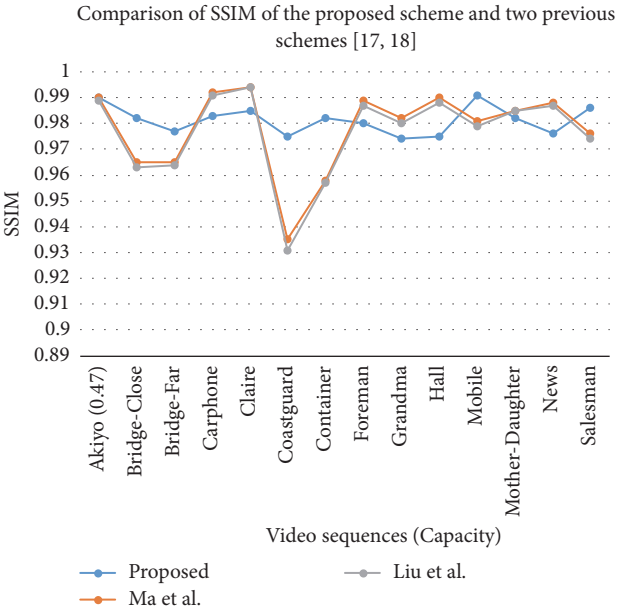


Figure 10: Visual quality (SSIM) performances of the proposed scheme and two previous schemes [17, 18] with different video sequences.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant no. 102.01-2016.06. The authors also would like to thank the Ministry of Science and Technology of the Republic of China for financially supporting this research under Contract nos. MOST 106-2218-E-035-011 and MOST 106-2627-M-035-007.

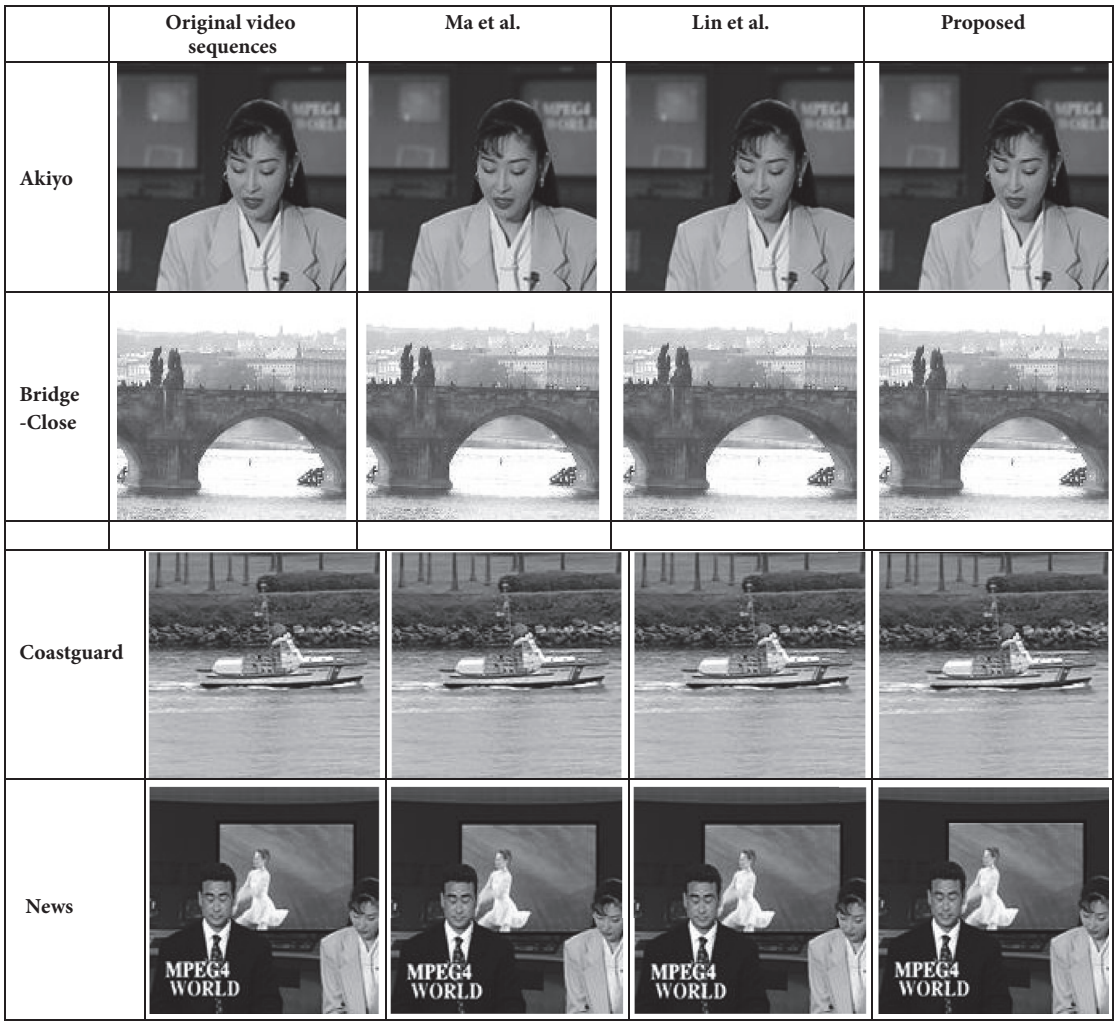


Figure 11: Visual effect of the resultant intraframes by the proposed scheme and two previous schemes [17, 18].

References

[1] Y. Liu, X. Qu, and G. Xin, "A ROI-based reversible data hiding scheme in encrypted medical images," *Journal of Visual Communication and Image Representation*, vol. 39, pp. 51–57, 2016.

[2] W. Zhang, H. Wang, D. Hou, and N. Yu, "Reversible data hiding in encrypted images by reversible image transformation," *IEEE Transactions on Multimedia*, vol. 18, no. 8, pp. 1469–1479, 2016.

[3] J. Bai, C.-C. Chang, T.-S. Nguyen, C. Zhu, and Y. Liu, "A high payload steganographic algorithm based on edge detection," *Displays*, vol. 46, pp. 42–51, 2017.

[4] Z. Pan and L. Wang, "Novel reversible data hiding scheme for Two-stage VQ compressed images based on search-order coding," *Journal of Visual Communication and Image Representation*, vol. 50, pp. 186–198, 2018.

[5] K. Chen and D. Xu, "An Efficient Reversible Data Hiding Scheme for Encrypted Images," *International Journal of Digital Crime and Forensics*, vol. 10, no. 2, pp. 1–22, 2018.

[6] C.-C. Chang and T.-S. Nguyen, "A reversible data hiding scheme for SMVQ indices," *Informatica (Netherlands)*, vol. 25, no. 4, pp. 523–540, 2014.

[7] X. P. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 826–832, 2012.

[8] I. E. G. Richardson, *H.264 and MPEG-4 video compression: video coding for next-generation multimedia*, Wiley, Chichester, U.K, 2003.

[9] M. Noorkami and R. Mersereau, "Compressed-domain video watermarking for H.264," in *Proceedings of the rnational Conference on Image Processing*, pp. II–890, Genova, Italy, September 2005.

[10] C. Nguyen, D. B. Tay, and G. Deng, "A Fast Watermarking System for H.264/AVC Video," in *Proceedings of the APCCAS 2006 - 2006 IEEE Asia Pacific Conference on Circuits and Systems*, pp. 81–84, Singapore, December 2006.

[11] J. Zhang, A. T. S. Ho, G. Qiu, and P. Marziliano, "Robust video watermarking of H.264/AVC," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, no. 2, pp. 205–209, 2007.

[12] M. Noorkami and R. M. Mersereau, "A framework for robust watermarking of H.264-encoded video with controllable detection performance," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 1, pp. 14–23, 2007.

[13] X. Gong and H.-M. Lu, "Towards fast and robust watermarking scheme for H.264 video," in *Proceedings of the 10th IEEE International Symposium on Multimedia (ISM '08)*, pp. 649–653, Berkeley, Calif, USA, December 2008.

[14] W. Huo, Y. Zhu, and H. Chen, "A controllable error-drift elimination scheme for watermarking algorithm in H.264/AVC stream," *IEEE Signal Processing Letters*, vol. 18, no. 9, pp. 535–538, 2011.

[15] S. K. Kapotas, E. E. Varsaki, and A. N. Skodras, "Data hiding in H. 264 encoded video sequences," in *Proceedings of the 9th IEEE Workshop on Multimedia Signal Processing (MMSP '07)*, pp. 373–376, October 2007.

[16] S. M. Kim, S. B. Kim, Y. Hong, and C. S. Won, "Data hiding on H," in *Proceedings of the 4th International Conference on Image Analysis and Recognition*, pp. 698–707, 2007.

[17] X. Ma, Z. Li, H. Tu, and B. Zhang, "A data hiding algorithm for h.264/AVC video streams without intra-frame distortion drift," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 10, pp. 1320–1330, 2010.

[18] T.-J. Lin, K.-L. Chung, P.-C. Chang, Y.-H. Huang, H.-Y. M. Liao, and C.-Y. Fang, "An improved DCT-based perturbation scheme for high capacity data hiding in H.264/AVC intra frames," *The Journal of Systems and Software*, vol. 86, no. 3, pp. 604–614, 2013.

[19] Y. Liu, Z. Li, X. Ma, and J. Liu, "A robust without intra-frame distortion drift data hiding algorithm based on H.264/AVC," *Multimedia Tools and Applications*, vol. 72, no. 1, pp. 613–636, 2014.

[20] VQEG, "Final Report From the Video Quality Experts Group on the Validation of Objective Models of Video Quality Assessment," http://www.vqeg.org/.

[21] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

Hindawi
Security and Communication Networks
Volume 2018, Article ID 1840207, 20 pages
https://doi.org/10.1155/2018/1840207

Research Article

A Robust Watermarking Scheme for Online Multimedia Copyright Protection Using New Chaotic Map

Amir Anees,[1] **Iqtadar Hussain**,[2] **Abdulmohsen Algarni**,[3] **and Muhammad Aslam**[4]

[1]*Department of Electrical Engineering, HITEC University, Pakistan*
[2]*Department of Mathematics, College of Science, King Khalid University, Abha, Saudi Arabia*
[3]*College of Computer Science, King Khalid University, Abha, Saudi Arabia*
[4]*Department of Mathematics, Quaid-i-Azam University, Islamabad, Pakistan*

Correspondence should be addressed to Amir Anees; a.anees@latrobe.edu.au

Received 22 November 2017; Accepted 14 May 2018; Published 28 June 2018

Academic Editor: Salvatore D'Antonio

The protection of copyrights of digital media uploaded to the Internet is a growing problem. In this paper, first, we present a unified framework for embedding and detecting watermark in digital data. Second, a new robust watermarking scheme is proposed considering this concern. The proposed work incorporates three chaotic maps which specify the location for embedding the watermark. Third, a new chaotic map, the Extended Logistic map, is proposed in this work. The proposed map has a bigger range than logistic and cubic maps. It has shown good results in a bifurcation, sensitivity to initial conditions, and randomness tests. Furthermore, with the detailed analysis of initial parameters, it is justified that Extended Logistic map can be used in secure communication, particularly watermarking. Fourth, to check the robustness of proposed watermarking scheme, we have done a series of analyses and standard attacks. The results confirm that the proposed watermarking scheme is robust against visual and statistical analysis and can resist the standard attacks.

1. Introduction

With the exponential growth of multimedia and their applications, most of the digital data is exchanged thorough the insecure channel [1]. It is important to protect the digital data, in particular secret data. To ensure the security of digital data, much effort has been put in the area of secure communication [2–4]. At the top algorithmic level, secure communication can be divided into three categories: cryptography [5–7], steganography, and watermarking [8–10]. The purpose of cryptography and stenography is to conceal the secret data; this is why these fields also lie in the information hiding subcategory [11]. Although the goal is the same, the methods are differently employed in cryptography and steganography. In cryptography, the data is transformed into an unreadable format so that, after transformation (encryption), the attacker can not deduce any information from it [12]. In steganography, the data is inserted or embedded into a carrier which can be a text, audio, image, or video. It is expected that the texture of carrier before and after the insertion of data will remain the same so that the attacker can not deduce any information regarding the data from the carrier [13, 14]. The methodology of steganography and watermarking is same, but the purposes are different. Steganography deals with the data hiding and watermarking concerns with the copyright protection of data [15, 16]. Instead of secret data, a watermark is inserted into the carrier to claim the copyrights when needed. Let us suppose that C denotes the carrier and w represents the watermark to be inserted and let Δ be the watermarking function, then the watermarked W data is given as

$$W = \Delta(C, w). \tag{1}$$

At the time of claiming the copyrights, inverse watermarking function Δ^{-1} is applied on W to extract the watermark as follows:

$$w = \Delta^{-1}(W). \tag{2}$$

This kind of watermarking scheme is known as private watermarking in which only W is required to extract the

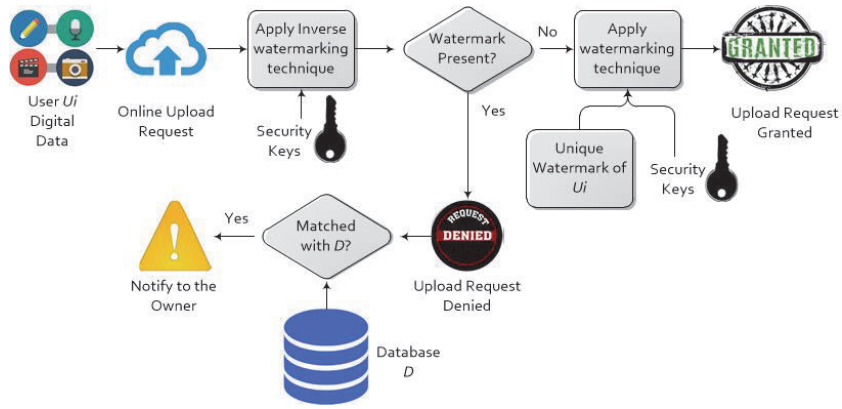


Figure 1: A top level block description of the proposed unified framework for protecting the rightful ownership of digital data. If an unauthenticated user tries to download and upload a data of user U_i, not only will the permission be denied but the domain will also notify U_i to legally make a case against that unauthenticated user.

watermark. However, some watermarking schemes do need the C as well with the W as defined as follows:

$$w = \Delta^{-1}(W, C). \tag{3}$$

This sort of watermarking scheme is known as public watermarking. In this manuscript, we have employed the private watermarking scheme using three chaotic maps which serve the functionality of Δ.

1.1. Motivations and Proposed Framework. Although there have been tremendous works on watermarking for the copyrights protection still the practical and real-life applications do need much attention, specifically in the area of online privacy of digital data. For example, a public domain of video uploader, youtube.com, does not have the sophisticated framework for the protection of rightful ownership. Let say, for instance, Bob has uploaded his video and after some time (days), Alice has downloaded that video and uploaded it again by her name. At the time of uploading, Alice has been granted the permission of uploading that video which should not be the case. The management of that public domain eventually remove that video after some days, and if the name of the video uploaded by Alice is significantly different from the one given by Bob, then it can take even more time to look up for that video and eventually in removing the video. Similarly, other than videos, piracy of digital images face the same problem.

In this work, we have proposed a unified framework for protecting the rightful ownership of digital data. The proposed framework is shown in Figure 1. It works as follows: let say a user U_i wants to upload his data on a public domain. First, he requests that domain upload that data. The public domain upon receiving the request applies the proposed inverse watermarking function Δ^{-1} to see if there is any watermark present in it. If there is any watermark present in that data, the domain will deny the upload request. Furthermore, the public domain will match the extracted watermark with the database. If the extracted watermark is matched with someone else watermark, the domain will notify the rightful owner of that data so that he or she can move with the legal case. In the case in which there is no watermark present in it, the domain will grant the permission of uploading that data and also will apply the proposed watermarking function Δ with the unique watermark w_i associated with the user U_i. In that way, let say, in future, if Alice tries to download and upload that data, not only will the permission be denied, but the domain will also notify U_i to make a case against her legally.

1.2. Contributions of This Work. In this paper, we develop a framework for watermarking using three different chaotic maps. The proposed work is effective and efficient as evident from the different analysis and attacks examined. The contributions of this paper are summarized as follows:

(1) We have proposed a unified framework for the protection of rightful ownership. In this framework, the wrong owner is not only denied but also reported to the right owner for any legal case.

(2) We have proposed a new chaotic map, Extended Logistic map in this work. The proposed map has a bigger range than logistic and cubic maps. It has shown good results in a bifurcation, sensitivity to initial conditions, and randomness tests. Furthermore, with the detailed analysis of initial parameters, it is justified that Extended Logistic map can be used in secure communication, particularly watermarking.

(3) We have proposed a watermarking scheme for any digital data specifically image data. The proposed watermarking is primarily based on the embedding of a watermark in lease significant bits of the carrier using three different chaotic maps. The initial conditions of these three chaotic maps are considered as secret keys [19–22].

(4) We have done noise resistant analysis in which we have shown that if the watermarked image is corrupted by the channel noise or by an unauthorized user, the inverse watermarking algorithm can

correctly extract the watermark from the corrupted watermarked image with some minor changes.

(5) To evaluate the strength of proposed watermarking algorithm, we have done a series of analyses along with standard attacks. The results of these analyses showed the superior performance and robustness of our proposed watermarking algorithm.

The remaining part of the manuscript is planned as follows: Section 2 presents the new chaotic map with its significance, Section 3 is devoted to the proposed watermarking scheme in detail. Section 4 is dedicated to simulation results and statistical analysis; Section 5 presents the performance analysis that will evaluate the proposed scheme with standard analysis and attacks and Section 6 concludes the manuscript.

2. A New Chaotic Map and Its Significance in Watermarking

This section will briefly introduce a new chaotic map which is a combination of logistic and cubic chaotic maps. The importance of usage of the proposed chaotic map is illustrated through the bifurcation diagrams, randomness, and sensitivity tests. The basics of other chaotic maps employed in the proposed watermarking algorithm are also given. These maps are piecewise linear chaotic map and Tangent Delay Ellipse Reflecting Cavity Map System (TD-ERCS).

The logistic chaotic map is given as [23]

$$x_n = r x_{n-1} \left(1 - x_{n-1}\right). \tag{4}$$

where $x_0 \in (0, 1)$ and $r \in (0, 4)$ are the initial seed parameters.

The cubic chaotic map is given as [24]

$$x_n = r x_{n-1} \left(1 - x_{n-1}^2\right). \tag{5}$$

where $x_0 \in (0, 1)$ and $r \in (0, 2.6)$ are the initial seed parameters.

Combining the above two maps, the new proposed Extended Logistic (EL) chaotic map is given as

$$x_n = r x_{n-1} \left(1 - x_{n-1}\right)\left(c + x_{n-1}\right). \tag{6}$$

where $x_0 \in (0, 1)$, $c \in (0, \infty)$, and r which is c dependable are the initial seed parameters.

We have analyzed the bifurcation diagrams of the EL map considering different values of parameter c plotted in Figure 2. These figures are the combination of all the x-vectors plotted against r. The r values are plotted on x-axis with spacing of 0.01 and x values are plotted on y-axis for each and every r value. Furthermore, the x values are plotted after 500 iterations to see the long-term effect of EL map. It can be observed that, by increasing the parameter c, the chaotic region of EL map is shrinking. Nonetheless, with parameter c in addition to x and r, the key space is huge enough to resist brute force attack and thus is sufficient, relevant, and appropriate to be used in secure communication.

In addition, we have examined a bifurcation diagram shown in Figure 2(a) to observe the overall behavior of EL map. Based on Figure 2(a), the interval considering r can be divided into following segments:

(i) When $r \in (0, 1.2)$ for $c = 2$, the iteration sequence of EL map shows a stable behavior, that is, after few iterations, the sequence comes to a constant point. The stable behavior can be shown in Figure 3. Figure 3(a) shows the bifurcation diagram of EL map for $r = 0$ to $r = 1.2$ when $c = 2$; Figures 3(b)–3(f) demonstrate the iteration sequence of this chaotic map for initial conditions of $r = 0.2$ to $r = 1.0$, $x_0 = 0.5$, and $c = 2$ with a spacing of $r = 0.2$. It can be observed that the iteration sequences come to a constant point after some iterations and thus cannot be used in secure communication.

(ii) When $r \in (1.2, 1.41)$ for $c = 2$, the iteration sequence of EL map shows a periodic behavior; that is, after few iterations, the sequence iterates between two or four points. The periodic behavior can be shown in Figure 4. Figure 4(a) shows the bifurcation diagram of EL map for $r = 1.2$ to $r = 1.41$ when $c = 2$; Figures 4(b)–4(f) demonstrate the iteration sequence of this chaotic map for initial conditions of $r = 1.22$ to $r = 1.40$, $x_0 = 0.5$, and $c = 2$ with an approximate spacing of $r = 0.04$. It can be observed that the iteration sequences iterate between two or four points after some iterations and thus cannot be used in secure communication.

(iii) When $r \in (1.41, 1.60)$, the iteration sequence for EL map shows random points exhibiting a chaotic behavior. Figure 5(a) shows the bifurcation diagram of EL map for $r = 1.41$ to $r = 1.60$; Figures 5(b)–5(f) demonstrate the iteration sequence of this chaotic map for initial conditions of $r = 1.41$ to $r = 1.60$, $x_0 = 0.5$, and $c = 2$ with an approximate spacing of $r = 0.03$. As can be seen from the Figure 5(a), the whole interval does not produces chaotic behavior as further elaborated in Figure 5(d). Figure 5(d) demonstrates the iteration sequence of this chaotic map for initial conditions of $r = 1.51$, $x_0 = 0.5$, and $c = 2$. It can be observed that the iteration sequences iterate between two points after some iterations and thus cannot be used in secure communication. Nonetheless, majority of this interval does exhibit as chaotic behavior, the iteration sequences are completely random and thus can be used in secure communication.

The chaotic behavior is further tested using the standard NIST-800-22 statistical tests [25]. This test suite includes 15 statistical tests which were originally designed for a larger stream of binary bits (usually > 10000); however these tests can also be applied on shorter streams as well as on integer values. The results of these tests on EL map for initial conditions of $r = 1.48$, $x_0 = 0.5$, and $c = 2$ are listed in Table 1 showing that the generated chaotic sequences passes the randomness tests.

For the appropriate usage in secure communication, randomness alone is not enough for a chaotic map. One of the essential requirements for any random generator in security is to be sensitive to the initial conditions. We have plotted chaotic sequence for EL map for initial conditions of $r = 1.56$, $x_0 = 0.500000000$, and $c = 2$, shown in Figure 6(a); in the

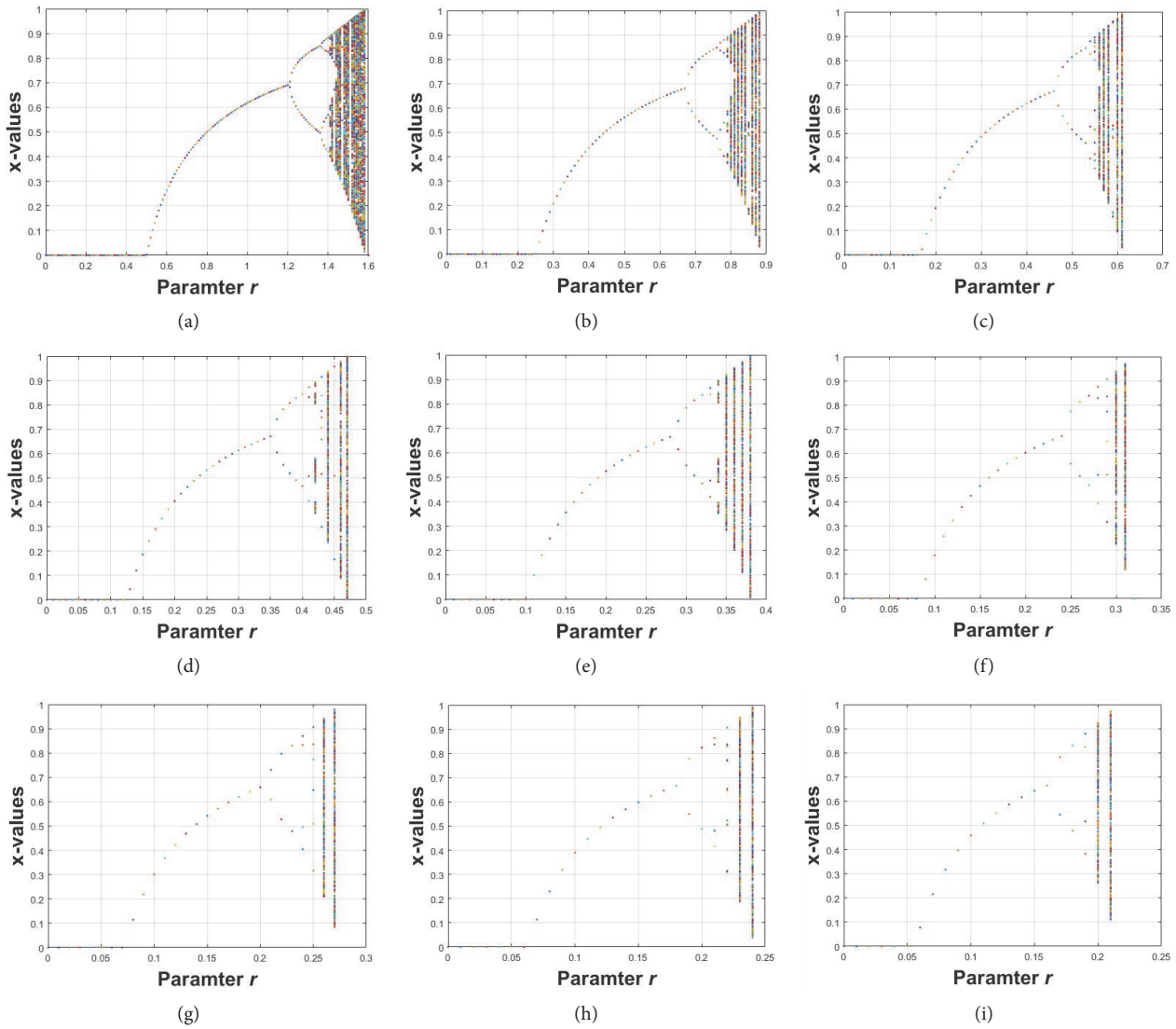


Figure 2: Bifurcation diagrams of the EL map considering different values of parameter c. (a) $c = 2$, (b) $c = 4$, (c) $c = 6$, (d) $c = 8$, (e) $c = 10$, (f) $c = 12$, (g) $c = 14$, (h) $c = 16$, and (i) $c = 18$. These figures are the combination of all the x-vectors plotted against r. The r values are plotted on x-axis with spacing of 0.01 and x values are plotted on y-axis for each and every r values. Furthermore, the x values are plotted after 500 iterations to see the long-term effect of EL map.

same figure, we have also plotted the chaotic sequences for EL map for slightly change initial conditions of $r = 1.56$, $x_0 = 0.500000001$, and $c = 2$ and $r = 1.56$, $x_0 = 0.500000002$, and $c = 2$. For the first 35 iterations, all the three sequences are same; however after that point, all sequences are completely out of phase to each other. Also, the EL map is sensitive to the initial conditions for r as well. Figure 6(b) shows the chaotic sequence for EL map for initial conditions of $r = 1.5600000000$, $x_0 = 0.5$, and $c = 2$ and in the same figure, two graphs are plotted for EL map for initial conditions of $r = 1.5600000001$, $x_0 = 0.5$, and $c = 2$ and $r = 1.5600000002$, $x_0 = 0.5$, and $c = 2$. It can be observed that at the beginning all the three sequences are almost same but with the increase in iterations; all the sequences can be recognized individually.

It can be shown that the other two chaotic maps, TD-ERCS and piecewise linear map, also have the same properties as EL map. The TD-ERCS map gives two chaotic sequences, x and k; mathematically, it is given as [26]

$$x_n = -\frac{2k_{n-1}y_{n-1} + x_{n-1}\left(\mu^2 - k_{n-1}^2\right)}{\mu^2 + k_{n-1}^2}.$$
$$k_n = -\frac{2k'_{n-m} - k_{n-1} + k_{n-1}k'^2_{n-m}}{1 + 2k_{n-1}k'_{n-m} - k'^2_{n-m}} \tag{7}$$

where

$$k'_n = -\frac{x_n}{x_n}\mu^2.$$

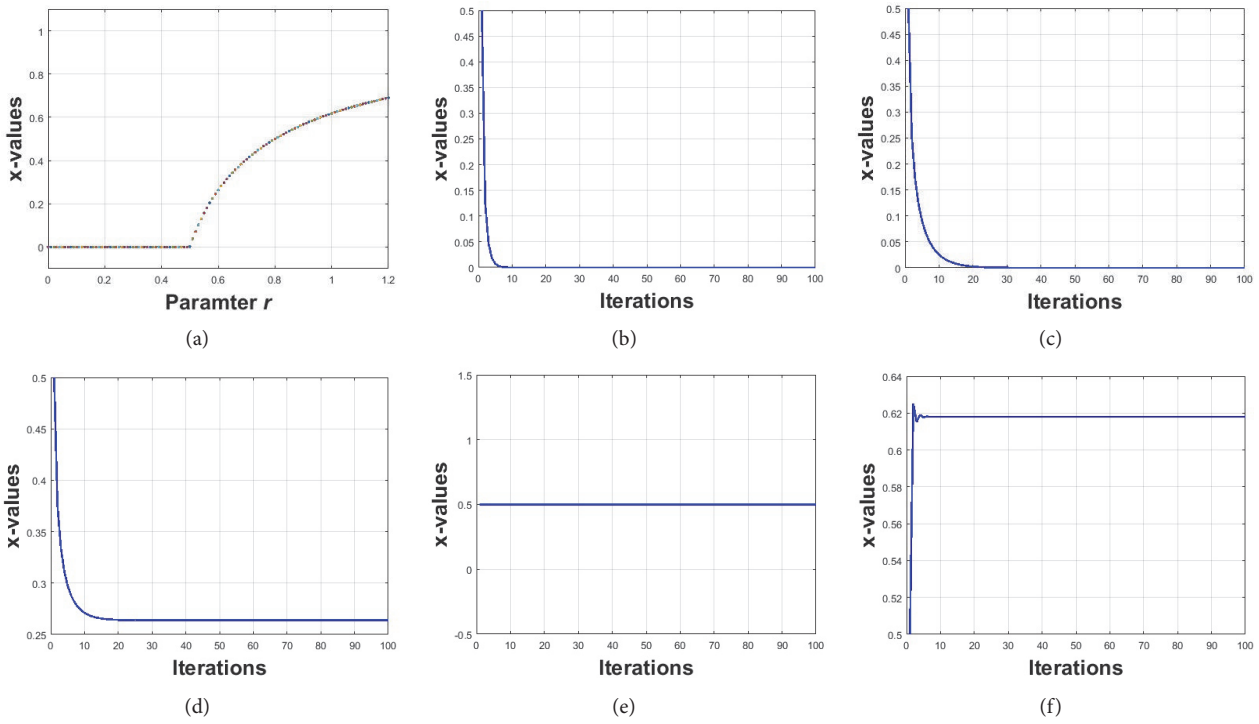


FIGURE 3: (a) Bifurcation diagram of EL map for $r = 0.2$ to $r = 1.0$, with initial conditions of $x_0 = 0.5$ and $c = 2$; (b)-(f) the iteration sequence diagrams of EL map for initial conditions of $r = 0.2$ to $r = 1.0$, $x_0 = 0.5$, and $c = 2$ with a spacing of $r = 0.2$. It can be observed that the iteration sequences come to a constant point after some iterations and thus cannot be used in secure communication.

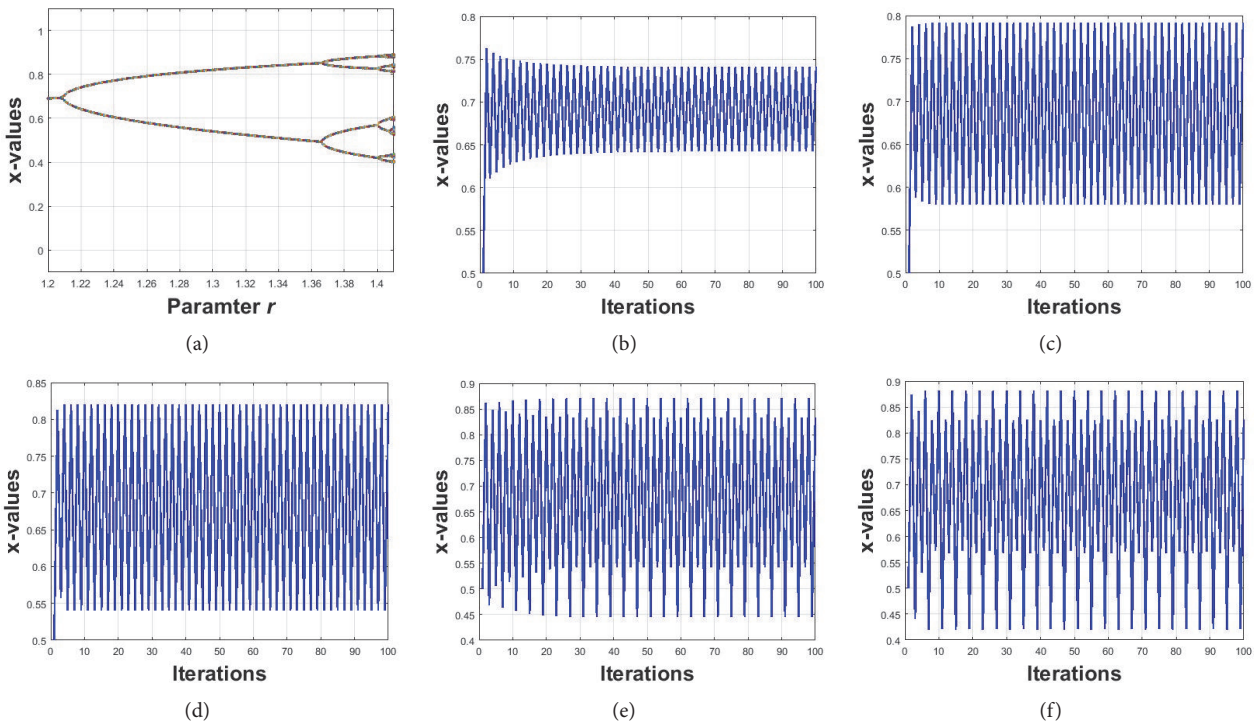


FIGURE 4: (a) Bifurcation diagram of EL map for $r = 1.2$ to $r = 1.41$, with initial conditions of $x_0 = 0.5$ and $c = 2$; (b)-(f) the iteration sequence diagrams of EL map for initial conditions of $r = 1.22$ to $r = 1.40$, $x_0 = 0.5$, and $c = 2$ with an approximate spacing of $r = 0.04$. It can be observed that the iteration sequences iterate between two or four points after some iterations and thus cannot be used in secure communication.

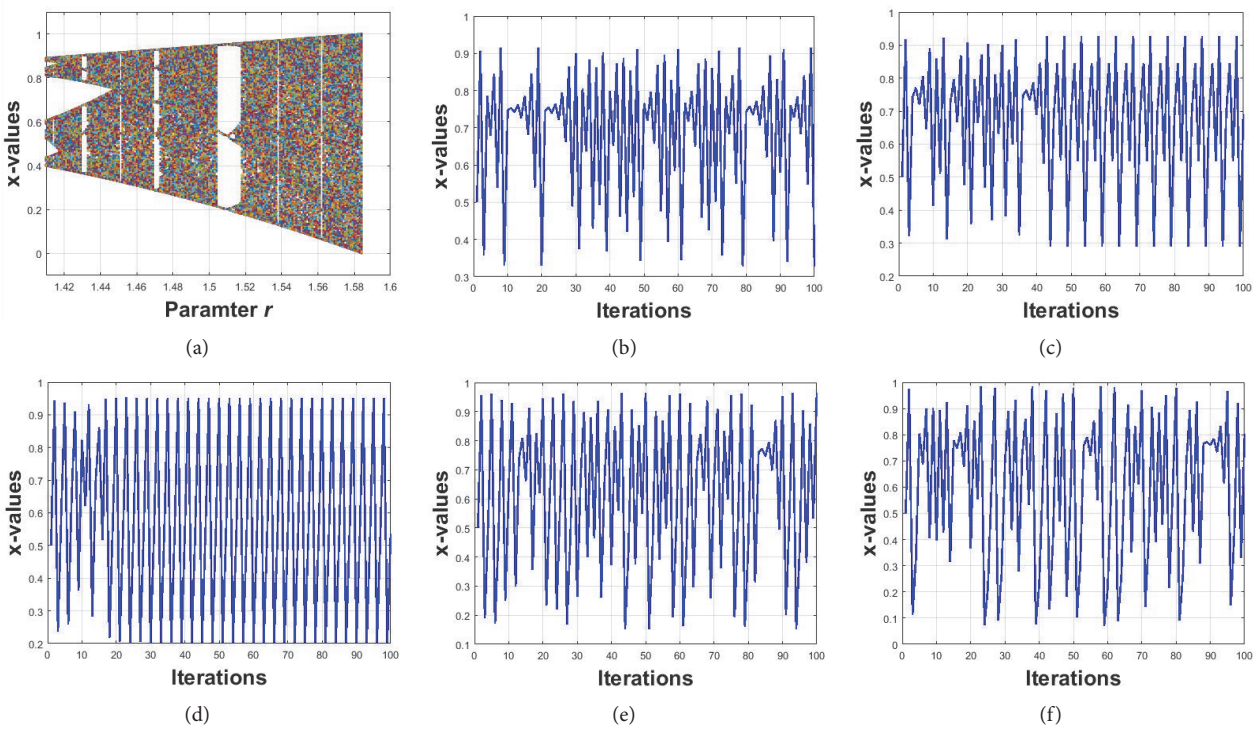
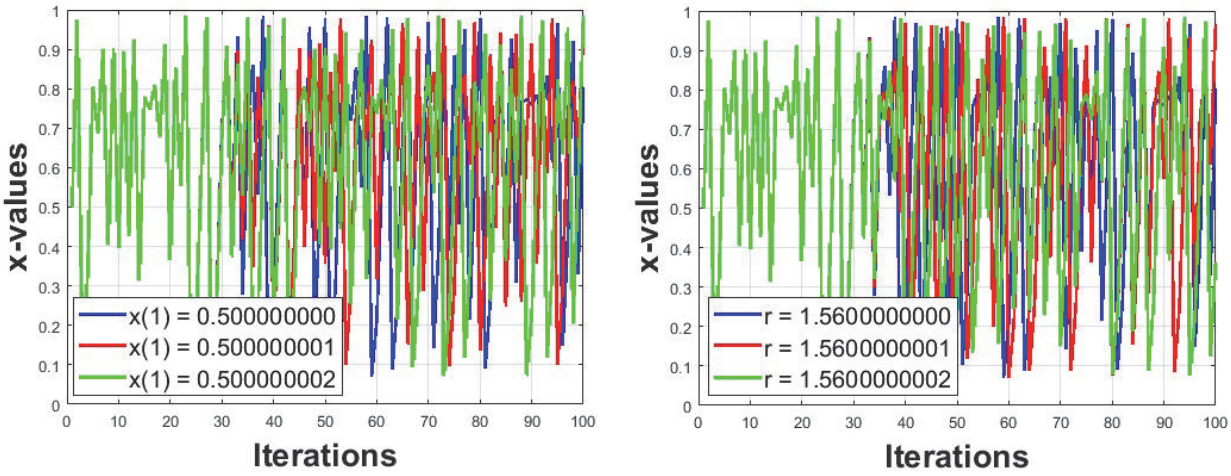


FIGURE 5: (a) Bifurcation diagram of EL map for $r = 1.41$ to $r = 1.60$, with initial conditions of $x_0 = 0.5$ and $c = 2$; (b)-(f) the iteration sequence diagrams of EL map for initial conditions of $r = 1.41$ to $r = 1.60$, $x_0 = 0.5$, and $c = 2$ with an approximate spacing of $r = 0.03$. The whole interval does not produce chaotic behavior as elaborated in (d). Nonetheless, majority of this interval does exhibit as chaotic behavior; the iteration sequences are completely random and thus can be used in secure communication.

TABLE 1: NIST statistical tests to check the randomness of chaotic sequences generated by EL map for initial conditions of $r = 1.48$, $x_0 = 0.5$, and $c = 2$. The results show that the chaotic sequences passes the randomness statistical tests.

Statistical Test	*P* Value	Decision
Frequency (Mono Bit) Test	0.9984	Passed
Frequency Test within a Block		
(Blocks sizes: 3, 4, 5, 6, 7, 8)	0.6845	Passed
The Runs Test	0.0409	Passed
Tests for the Longest-Run-of-Ones in a Block		
(Blocks size: 8)	0.3089	Passed
The Binary Matrix Rank Test		
(4 Matrices, Rows: 8, Columns: 8)	0.0886	Passed
(16 Matrices, Rows: 4, Columns: 4)	0.1547	Passed
The Discrete Fourier Transform (Spectral) Test	0.1007	Passed
The Non-overlapping Template Matching Test		
(Template length = 4, Blocks = 2, 4, 8)	0.0314	Passed
The Overlapping Template Matching Test		
(Template length = 4, Blocks = 4, 8)	0.5947	Passed
Maurer's Universal Statistical Test	0.2514	Passed
The Approximate Entropy Test	0.3108	Passed
The Cumulative Sums (CUSUM) Test	0.4512	Passed



(a) (b)

FIGURE 6: (a) Chaotic sequence for EL map for initial conditions of $r = 1.56$, $x_0 = 0.500000000$, and $c = 2$ and in the same figure, two graphs are plotted for EL map for initial conditions of $r = 1.56$, $x_0 = 0.500000001$, and $c = 2$ and $r = 1.56$, $x_0 = 0.500000002$, and $c = 2$. (b) Chaotic sequence for EL map for initial conditions of $r = 1.5600000000$, $x_0 = 0.5$, and $c = 2$ and in the same figure, two graphs are plotted for EL map for initial conditions of $r = 1.5600000001$, $x_0 = 0.5$, and $c = 2$ and $r = 1.5600000002$, $x_0 = 0.5$, and $c = 2$. It can be seen that initially up to 35 iterations, all the sequences of both graphs are same; however after that point, the sequences are completely out of phase to each other.

$$y_n = k_{n-1}\left(x_n - x_{n-1}\right) + y_{n-1}.$$

$$k_{n-m} = \begin{cases} \dfrac{x_{n-1}}{y_{n-1}}\mu^2 & if\ n < m \\ \dfrac{x_{n-m}}{y_{n-m}}\mu^2 & if\ n \geq m. \end{cases} \tag{8}$$

And the initial seed parameters are

$$\begin{aligned} x_0 &\in [-1, 1], \\ \tan\alpha &\in (-\infty, \infty), \\ \mu &\in (0.05, 1), \\ m &= 2, 3, \ldots, n. \end{aligned} \tag{9}$$

Given these initial parameters, we have

$$\begin{aligned} y_0 &= \mu\sqrt{1 - x_0^2}. \\ k_0' &= -\frac{x_0}{y_0}\mu^2. \\ k_0 &= \frac{\tan\alpha + k_0'}{1 - k_0'\tan\alpha}. \end{aligned} \tag{10}$$

The third chaotic map that will be employed is piecewise linear chaotic map, given as [27]

$$x_{n+1} = \begin{cases} \dfrac{x_n}{p} & if\ x_n \in [0, p], \\ \dfrac{1 - x_n}{1 - p} & if\ x_n \in (p, 1], \end{cases} \tag{11}$$

where $x_0 \in (0, 1)$ and $p \in (0, 1)$ are the initial seed parameters.

3. Proposed Watermarking Algorithm

The proposed work is intended for the online multimedia copyright protection. The watermark is inserted into a carrier based on the decisions made by three chaotic maps. The watermark which we want to insert is taken as a byte of length with eight binary bits, even if it is a digital signal, sound data, image data, video data, or simply text. Beside the watermark insertion, a substitution operation is also performed on watermark to enhance the security of the overall system. The watermarking and inverse watermarking processes are explained as follows.

3.1. Watermarking. The watermarking algorithm in the form of a flowchart is shown in Figure 8. We consider the carrier as a digital image denoted as C with size ro, co, fo, where ro is the number of rows, co is the number of columns, and fo is the number of frames in C; that is, a color image has three frames whereas a gray image has one frame. An image pixel at a specific location of i^{th} row, j^{th} column, and k^{th} frame of that image is denoted as $C(i, j, k)$. The image pixel $C(i, j, k)$ has the range of [0 255] and can be expressed as a binary of 8 bits. We consider the watermark as a digital image too denoted as W with size rw, cw, fw, where rw is the number of rows, cw is the number of columns, and fw is the number of frames in W. Here we consider a two-dimensional watermark image so that we will denote the image pixel of the watermark as $W(i, j)$.

Before the watermark insertion, the watermark is divided into eight binary bits in which 4 Least Significant Bits (LSBs) of each pixel are discarded, keeping the 4 Most Significant Bits (MSBs). The reason for reducing the size of watermark image is to decrease the computational complexity while keeping the texture of watermark image as same as possible; this is

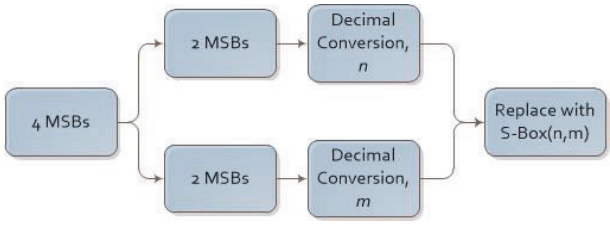


Figure 7: Graphical illustration of 4 × 4 S-Box substitution. The 4 MSBs are divided into two parts, each having two bits. The decimal value of first two bits represents the row number of 4 × 4 S-Box and the decimal value of last two bits represents the column number of 4 × 4 S-Box. The value at that position of S-Box will be replaced with those 4 MSBs.

Table 2: Presentation of S-Box in 4 × 4 matrix.

R/C	0	1	2	3
0	11	1	7	13
1	5	15	4	2
2	6	12	9	14
3	10	0	3	8

shown in the simulated results carried later. The 4 MSBs are substituted with the 4 × 4 S-Box as defined below:

$$W(i,j) = substitution(W(i,j), S-Box) \quad \forall i \in rw,\ \forall j \in cw. \tag{12}$$

For simplicity and convenience, we write substituted watermark image as W as well. The 4 × 4 S-Box is shown in Table 2 and S-Box substitution operation is shown in Figure 7. The 4 MSBs are divided into two parts, each having two bits. The decimal value of first two bits represents the row number of 4 × 4 S-Box and the decimal value of last two bits represents the column number of 4 × 4 S-Box. The value at that position of S-Box will be replaced with those 4 MSBs. After the substitution, the image pixels of W are inserted into C such that the first 2 MSBs are inserted in the left part of C, and the next 2 MSBs are inserted in the right part of C. These MSBs inserted in the positions of left or right parts of C are defined by three chaotic maps; the first chaotic map will define the row number, the second chaotic map will define the column number, and the third chaotic map will define the frame number.

Let x be the chaotic sequence resulted from piece wise chaotic map with initial values of x_0 and p. These initial values are considered as first two secret keys of the proposed algorithm, that is, $k_1 = x_0$ and $k_2 = p$. The length of x is $rw * cw * 100$ and values having under modulo ro. As the chaotic sequence initially has the range of $[0, 1]$, we have amplified that range by multiplying that sequence with 1000 and then limiting the sequence under modulo ro. The x values defined the row positions of C while inserting W in C.

Let x and k be the two chaotic sequences resulting from TD-ERCS chaotic map with initial values of x_0, $\tan\alpha$, μ, and m. For simplicity and convenience, we denote x as y. These four initial values are considered as next four secret keys of the proposed algorithm, that is, $k_3 = x_0$, $k_4 = \tan\alpha$, $k_5 = \mu$, and $k_6 = a$. Moreover, we only need one chaotic sequence from this map; therefore we will only use y sequence. The length of y is $rw * cw * 100$ and values having under modulo $co/2$. As the chaotic sequence initially has the range of [-1, 1], we have amplified that range by first shifting the range from [-1, 1] to [0, 2] and then multiplying that shifted sequence with 1000 and then limiting the sequence under modulo $co/2$. The y values defined the column positions of C while inserting W in C.

Let x be the chaotic sequence resulted from EL chaotic map with initial values of x_0, r, and c. For simplicity and convenience, we denote x as z. These three initial values are considered as next three secret keys of the proposed algorithm, that is, $k_7 = x_0$, $k_8 = r$, and $k_9 = c$. The length of z is $rw * cw * 100$ with under modulo $fo/2$. As the chaotic sequence initially has the range of $[0, 1]$, we have amplified that range by multiplying that sequence with 1000 and then limiting the sequence under modulo $fo/2$. The z values defined the frame positions of C while inserting W in C.

In parallel, the C is divided into C_1 and C_2, where C_1 denotes the left part of C and C_2 denotes the right part of C such that $C_1 = C(:, 1 : co/2, :)$ and $C_2 = C(:, co/2 + 1 : co, :)$. Based on the x, y and z sequences, the image pixels of C_1 and C_2 at $x(i), y(i), z(i)$, i.e., $C_1(x(i), y(i), z(i))$ and $C_2(x(i), y(i), z(i))$, are selected. These image pixels are converted into binary having 8 bits each. The 2 LSBs of these two image pixels are replaced with the 4 MSBs of W. The 2 MSBs of W will be replaced with the 2 LSBs of C_1 at $C_1(x(i), y(i), z(i))$ and the next 2 MSBs of W will be replaced with the 2 LSBs of C_2 at $C_2(x(i), y(i), z(i))$. After the replacement, the binary bits are joined and converted into decimal. This process will continue for all the image pixels present in W. After the replacement of all the image pixels, the two parts are joined together to form a watermarked image, denoted as Wd as explained in Algorithm 1.

3.2. Inverse Watermarking. The inverse watermarking algorithm which is exactly the inverse of the watermarking algorithm in the form of a flowchart is shown in Figure 9. The purpose of inverse watermarking is to extract the watermark to claim the copyright of that digital carrier. The input to the inverse watermarking algorithm is the watermarked image, Wd with size $ro * co * fo$. To have the successful extraction of W, the same set of secret keys will be used.

As a first step of the inverse watermarking algorithm, the chaotic sequences are generated from three chaotic maps using the same set of secret keys. In parallel, for the extraction of watermark, the Wd is first divided into C_1 and C_2, where C_1 denotes the left part of C and C_2 denotes the right part of C such that $C_1 = C(:, 1 : co/2, :)$ and $C_2 = C(:, co/2 + 1 : co, :)$. Based on the x, y and z sequences, the image pixels of C_1 and C_2 at $x(i), y(i), z(i)$, i.e., $C_1(x(i), y(i), z(i))$ and $C_2(x(i), y(i), z(i))$, are selected. These image pixels are converted into binary having 8 bits each. The 2 LSBs of these two image pixels are selected, combined into 4 MSBs, and converted into decimal. This process will continue until all the image pixels present in the W are extracted as shown in

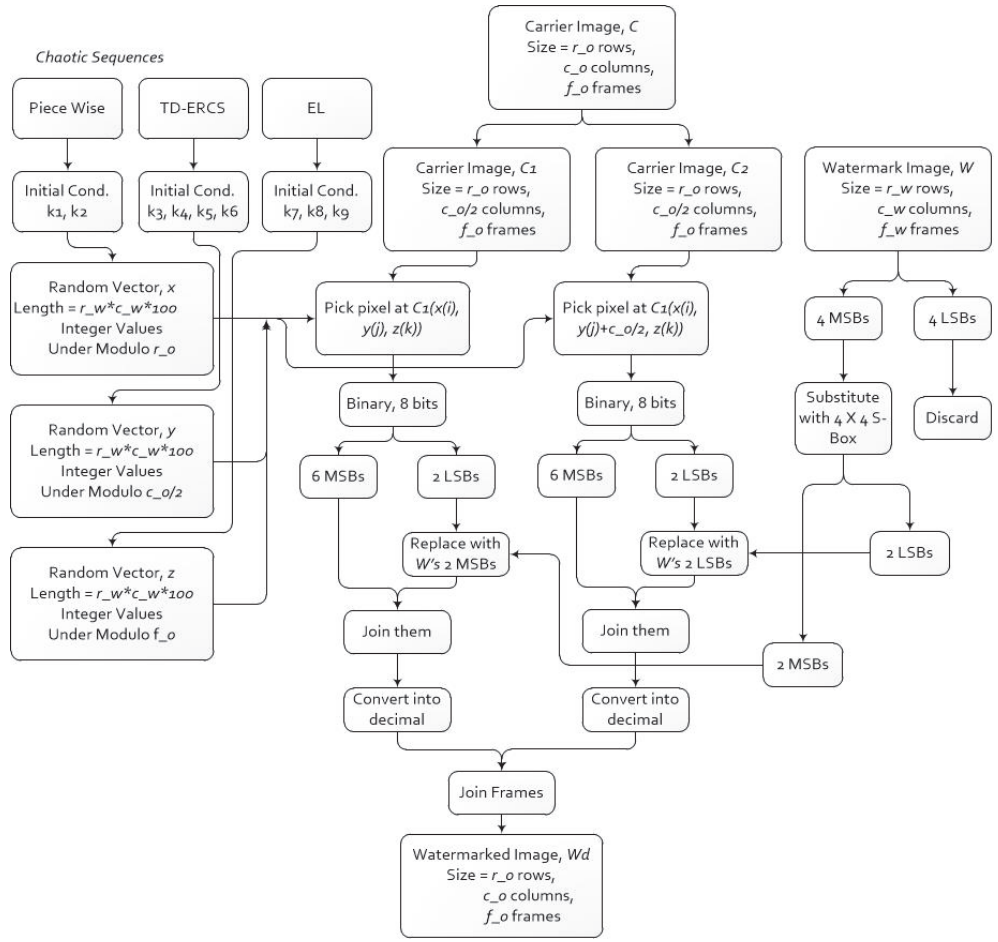


Figure 8: Flowchart of the watermarking algorithm is comprised of three chaotic maps with the help of a 4 × 4 S-Box. For each operation performed with the assistance of a chaotic map, a different set of secret keys is used.

Inputs: Carrier image $C_{(ro\times co\times fo)}$, Watermark image $W_{(rw\times cw)}$, 4 × 4 S-Box, piece wise map, td-ercs map, EL map, and secret keys: $k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$.
Output: Watermarked image $Wd_{(ro\times co\times fo)}$.

(1) $x = piece\_map(k_1, k_2)$
(2) $y = tdercs\_map(k_3, k_4, k_5, k_6)$
(3) $z = EL\_map(k_7, k_8, k_9)$
(4) $W = substitution(W, S - box)$
(5) $ii = 1, jj = 1, kk = 1.$
(6) $Wd = C$
(7) **for** $i \longleftarrow 1 : rw$ **do**
(8) **for** $j \longleftarrow 1 : cw$ **do**
(9) $W\_a = dec2bin(W(i, j), 8)$
(10) $Wd\_a = dec2bin(C(x(ii), y(jj), z(kk), 8)$
(11) $Wd\_b = dec2bin(C(x(ii), y(jj) + co/2, z(kk), 8)$
(12) $Wd(x(ii), y(jj), z(kk) = bin2dec(Wd\_a(1 : 6), W\_a(1 : 2))$
(13) $Wd(x(ii), y(jj) + co/2, z(kk) = bin2dec(Wd\_b(1 : 6), W\_a(3 : 4))$
(14) $ii = ii + 1, jj = jj + 1, kk = kk + 1.$
(15) **Endfor**
(16) **Endfor**

Algorithm 1: Watermarking procedure is comprised of three chaotic maps with the help of a 4 × 4 S-Box.

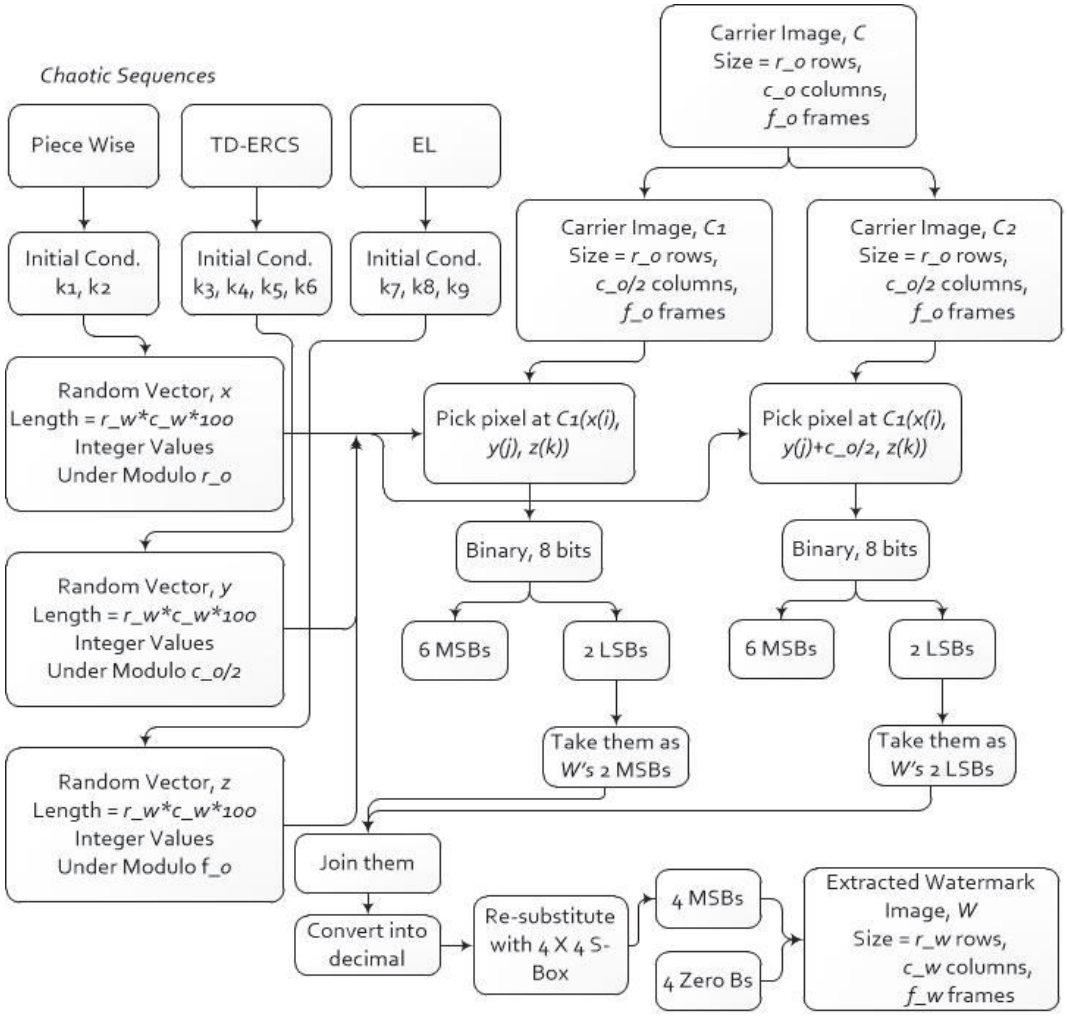


Figure 9: Flowchart of the inverse watermarking algorithm is comprised of three chaotic maps with the help of a 4 × 4 S-Box. For each operation performed with the assistance of a chaotic map, a different set of secret keys is used. However, the secret keys are the same as those used in the watermarking algorithm.

Algorithm 2. As the last step, the image pixels of W are then resubstituted with the same S-Box shown in Table 2 as defined below:

$$W(i,j) = inv_{s}ubstitution\left(W(i,j), S-Box\right) \quad \forall i \in rw,\ \forall j \in cw. \tag{13}$$

4. Simulated Results and Statistical Analysis

The simulations are conducted taking the first baboon as carrier image with size 512 × 512 × 3; i.e., there are 512 rows, 512 columns, and three frames in the baboon color image. We take cameraman image as the watermark with size 225 × 225 × 1; i.e., there are 225 rows, 225 columns, and one frame in the cameraman gray scale image. The size of the watermark image is almost the half of the carrier which is very large; the reason for taking a large watermark is to check the robustness of proposed work. The secret keys which are the initial values of the three chaotic maps employed are listed in Table 3. Before inserting the watermark into the carrier, the watermark is divided into eight binary bits in which 4 Least Significant Bits (LSBs) of each pixel are discarded. The watermark cameraman image is shown in Figure 10(a) and after discarding 4 Least Significant Bits (LSBs) of each pixel, the watermark cameraman image is shown in Figure 10(b). It can be seen that there is not much difference in texture between these two images. Furthermore, we have substituted this after-discarded version of the watermark with the S-Box shown in Table 2. The substituted version of watermark is shown in Figure 10(c). Although the information is visually available in the substituted image, it still provides better security. It is worth mentioning here that we can increase the security by employing multiple S-Boxes instead of a single S-Box but at the cost of extra computational complexity. Before the insertion, the baboon image which is considered as a carrier is shown in Figure 11(a). The watermark which is to

Inputs: Watermarked image $Wd_{(r\_o \times c\_o \times f\_o)}$, 4 × 4 S-Box, piece wise map, td-ercs map, EL map, and secret keys: $k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$.
Output: Inverse Watermark image $InvW_{(r\_w \times c\_w)}$.
(1) $x = piece\_map(k_1, k_2)$
(2) $y = tdercs\_map(k_3, k_4, k_5, k_6)$
(3) $z = EL\_map(k_7, k_8, k_9)$
(4) $ii = 1, jj = 1, kk = 1.$
(5) **for** $i \longleftarrow 1 : r\_w$ **do**
(6) **for** $j \longleftarrow 1 : c\_w$ **do**
(7) $InvW\_a = dec2bin(Wdx((ii), y(jj), z(kk)), 8)$
(8) $InvW\_b = dec2bin(Wdx((ii), y(jj) + c\_o/2, z(kk)), 8)$
(9) $InvW = bin2dec(InvW\_a(7:8), InvW\_b(7:8), 0000)$
(10) **Endfor**
(11) **Endfor**
(12) $InvW = inv\_substitution(InvW)$

Algorithm 2: Inverse watermarking procedure is comprised of three chaotic maps with the help of a 4 × 4 S-Box.

Table 3: Values of secret keys which are the initial conditions of the three chaotic maps employed in the proposed watermarking algorithm.

Maps	Parameters						
	x_0	$\tan\alpha$	μ	m	r	p	c
EL Map	0.5	-	-	-	1.46	-	2
TD-ERCS Map	0.5	1	0.4	50	-	-	-
Piece Wise Map	0.4	-	-	-	3.7	0.9	-

be inserted is shown in Figure 11(b) and after insertion, the watermarked baboon image is shown in Figure 11(c). The histogram of the carrier image is shown in Figure 11(d) and the histogram of watermarked image is shown in Figure 11(e). We can see from the images and histograms of the carrier and watermarked that they are visually similar to each other. After the extraction of the watermark from the watermarked image, we expect to get the cameraman image shown in Figure 11(b). The visual strength of proposed watermarking is further elaborated through the statistical analysis done in the next section.

4.1. Statistical Analysis. To examine the visual strength of the proposed watermarking algorithm, different statistical analyses are performed on carrier and watermark to compare the visual appearance of these two images. The statistical analysis considered in this work is as follows.

4.1.1. Correlation. The correlation of an image is given as [28]

$$Corr. = \sum_{i,j} \frac{(i - \mu i)(j - \mu j)\rho(i,j)}{\varphi_i \varphi_j}, \tag{14}$$

where i, j corresponds to image pixels positions, $\rho(i, j)$ is pixel value at i^{th} row and j^{th} column of image, μ is the variance, and φ is the standard deviation. The correlation analysis determines the similarity between two neighbor image pixels over the whole image having range between [−1 1] with 1 showing the perfect correlation.

4.1.2. Entropy. The entropy of an image is given as [28]

$$Entropy = -\sum_{i,j} pr(\rho(i,j)) \log_2 pr(\rho(i,j)), \tag{15}$$

where i, j corresponds to image pixels positions, $\rho(i, j)$ is pixel value at i^{th} row and j^{th} column of image, and $pr(\rho(i, j))$ is the probability of image pixel. Entropy shows the randomness of image having range between [0 8] for an image having 256 gray scales. A greater value of entropy shows the greater amount of randomness.

4.1.3. Contrast. The contrast of an image is given as [28]

$$Contrast = \sum_{i,j} |i - j|^2 \rho(i,j), \tag{16}$$

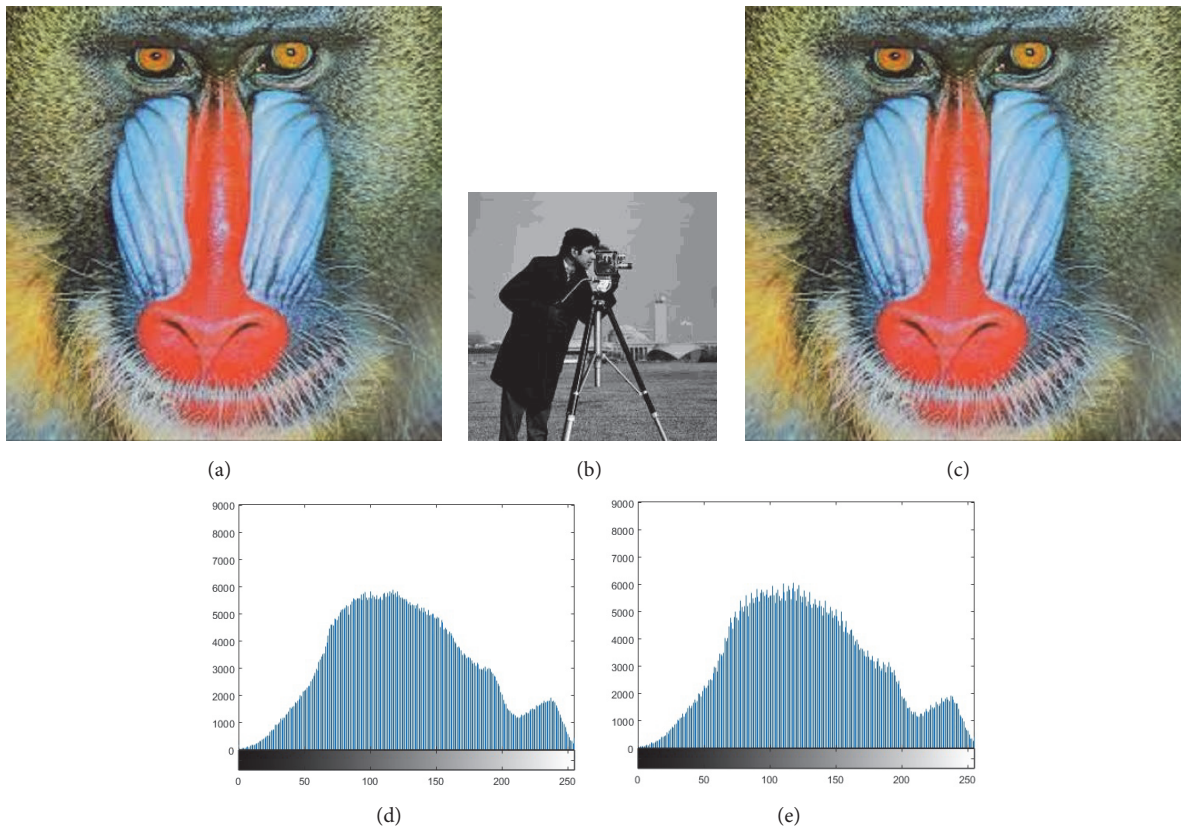
where i, j corresponds to image pixels positions and $\rho(i, j)$ is pixel value at i^{th} row and j^{th} column of image. The contrast analysis of the image enables the viewer to vividly identify the objects in texture of an image. The contrast values ranges from $[0\ (size(Image) - 1)^2]$. The contrast value of a constant image is 0. The greater value of the contrast shows greater variation in image pixels.

4.1.4. Homogeneity. The homogeneity of an image is given as [28]

$$Homo. = \sum_{i,j} \frac{\rho(i,j)}{1 + |i - j|}, \tag{17}$$

(a) (b) (c)

Figure 10: (a) Watermark cameraman image; (b) watermark cameraman image after discarding 4 Least Significant Bits (LSBs) of each pixel. It can be seen that there is not much difference of texture between these two images (a and b). (c) Substituted version of watermark by substituting the after-discarded version of watermark with the S-Box shown in Table 2. Although the information is visually available in substituted image, it still provides better security as compared to (b).



(a) (b) (c)

(d) (e)

Figure 11: (a) The baboon image which is considered as a carrier. (b) The watermark which is to be inserted. (c) After insertion, the watermarked baboon image. We can see that these two images, carrier and watermarked, are visually similar to each other. After the extraction of watermark from the watermarked image, we expect to get the cameraman image shown in (b). (d) Histogram of the carrier image and (e) histogram of the watermarked image.

where i, j corresponds to image pixels positions. The homogeneity analysis processes the closeness of the distribution in the gray level cooccurrence matrix (GLCM) to GLCM diagonal. The range of homogeneity is [0 1].

4.1.5. Energy. The energy of an image is given as [28]

$$Energy = \sum_{i,j} \rho\left(i, j\right)^2, \tag{18}$$

Table 4: Comparative statistical analysis on the carrier and watermarked images. The analyses are done on the individual three frames of these two images. It can be seen that, except the entropy analysis, the values of all the other analysis are same for all three frames of these two images showing very good performance.

Frame No.	Images	Analysis				
		Corr.	Entropy	Homo.	Contrast	Energy
1	Carrier	0.9570	7.6605	0.8872	0.2369	0.1062
	Watermark	0.9570	7.6602	0.8872	0.2369	0.1062
2	Carrier	0.9330	7.3575	0.8844	0.2434	0.1244
	Watermark	0.9330	7.3569	0.8844	0.2434	0.1244
3	Carrier	0.9610	7.6779	0.8806	0.2518	0.1040
	Watermark	0.9610	7.6772	0.8806	0.2518	0.1040

Table 5: A comparison of statistical analysis of other watermarking techniques [16–18] with the proposed work applied on different images. It can be seen that the proposed work has superior performance over the other works.

Images	Other Works	Analysis				
		Homo.	Contrast	Energy.	Entropy	Corr.
Pepper	Original	0.8902	0.3311	0.1330	7.5612	0.9207
	Watermarked [16]	0.8917	0.3181	0.1233	7.6003	0.9295
	Watermarked [17]	0.8902	0.3311	0.1330	7.5613	0.9207
	Watermarked [18]	0.8512	0.3241	0.1520	7.4521	0.6241
	Watermarked, Proposed	0.8902	0.3311	0.1330	7.5641	0.9207
Lena	Original	0.8651	0.4141	0.0942	7.7021	0.9444
	Watermarked [16]	0.8687	0.3857	0.1288	7.2512	0.8933
	Watermarked [17]	0.8811	0.3371	0.1130	7.7023	0.9443
	Watermarked [18]	0.9277	0.2688	0.3208	7.6745	0.9688
	Watermarked, Proposed	0.8651	0.4141	0.0942	7.7045	0.9444
Baboon	Original	0.7294	1.0004	0.0817	7.3903	0.6607
	Watermarked [16]	0.8427	0.3531	0.1387	7.1872	0.8933
	Watermarked [17]	0.7294	1.0004	0.0817	7.3903	0.6607
	Watermarked [18]	0.7669	0.7179	0.1028	7.4521	0.6788
	Watermarked, Proposed	0.7294	1.0004	0.0817	7.3945	0.6607

where i, j corresponds to image pixels positions. The energy analysis returns the sum of squared elements in the GLCM. The range of energy is [0 1]. The energy of a constant image is 1.

The results of these analyses considering our proposed algorithm for the carrier and watermarked images are shown in Table 4. The analyses are done on the individual three frames of these two images. It can be seen that except the entropy analysis, the values of all the other analyses are same for all three frames of these two images showing very good performance.

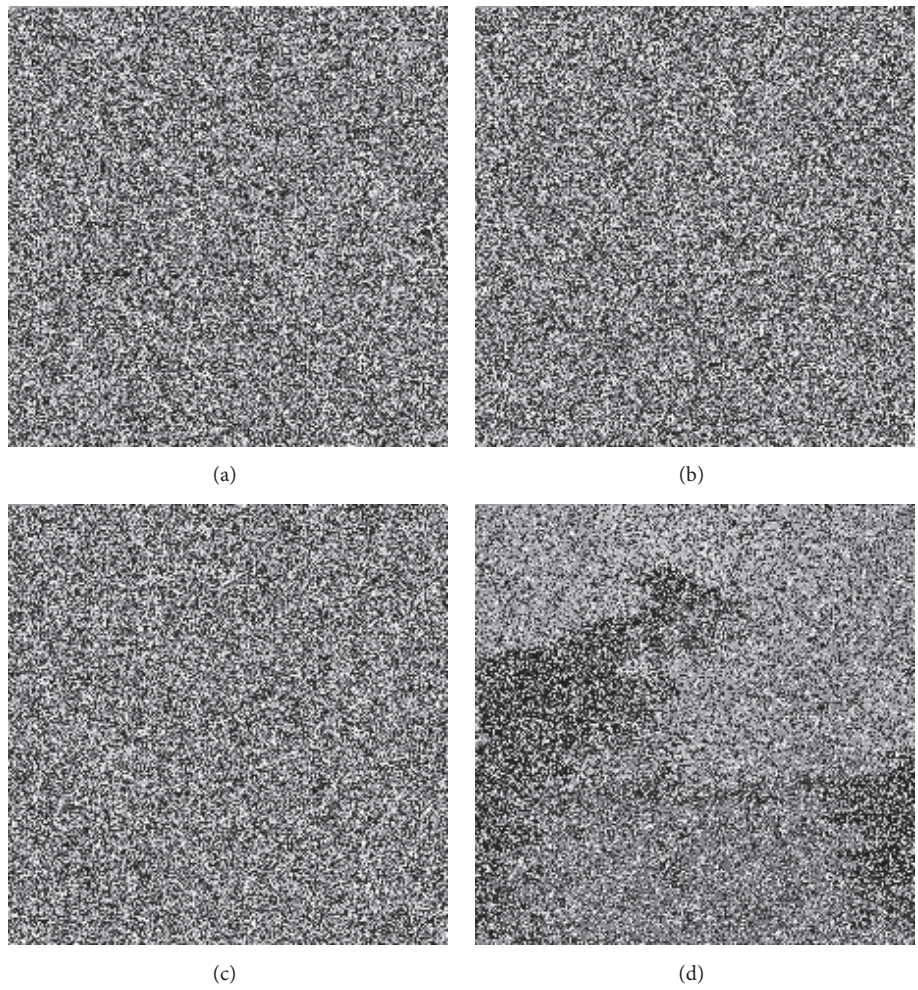
Table 5 presents a comparison of statistical analysis of other watermarking techniques [16–18] with the proposed work applied on different images. It can be seen that the proposed work has superior performance over the other works.

5. Security Analysis

The security analysis assists in determining the strength of any security algorithm. In this section, we have done detailed security analysis which includes key security, noise resistant analysis, and different attacks. These security analyses are described as follows.

5.1. Key Space and Key Sensitivity. Key space refers to the total number of keys that can be used in the watermarking algorithm. We have used initial conditions of three chaotic maps as the secret keys. There are nine secret keys used; the values of these secret keys with their ranges are mentioned earlier. If the average range of a secret key is 10^8, then the total number of different keys that can be used is $10^{8\times9} = 10^{72}$. This is equivalent or more than the 256 binary bits. With this key space, a modern computer will take more than 10^{20} years to check all the combinations.

The key space will only be effective if every key is effective regarding successful extraction of the watermark. For example, key space will only be effective if we tried to extract the watermark from the watermarked image with a slightly change (even change of a single bit) secret key(s) that was used in the embedding of a watermark in carrier image, then the extraction should not be successful. This is known as key sensitivity. In this work, we have embedded the cameraman image with the secret keys mentioned in Table 3

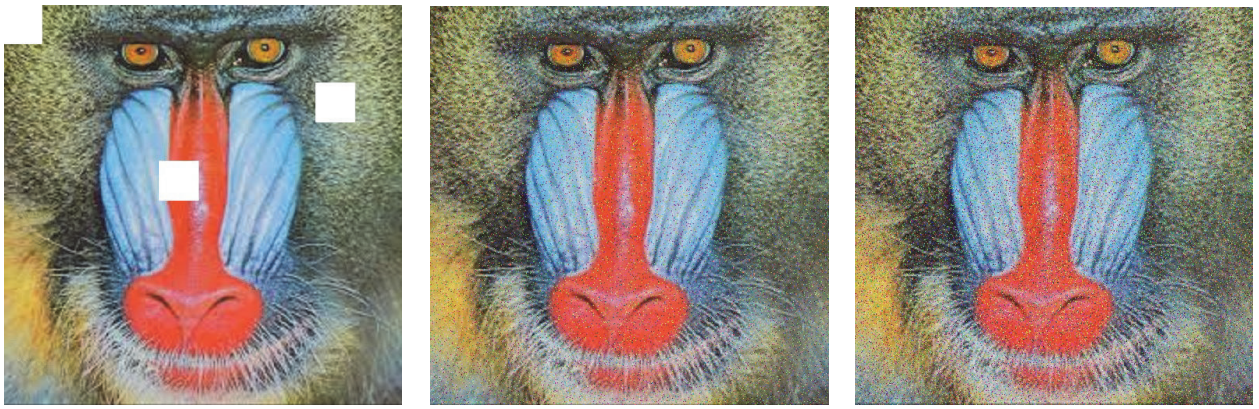


(a) (b) (c) (d)

Figure 12: Four different cases of key sensitivity. (a) Key used in extraction of watermark is changed from $k_1 = x_0 = 0.4$ to $k_1' = x_0 = 0.40000000001$, (b) key used in extraction of watermark is changed from $k_3 = x_0 = 0.5$ to $k_3' = x_0 = 0.50000000001$, (c) key used in extraction of watermark is changed from $k_5 = \mu = 0.4$ to $k_5' = \mu = 0.40000000001$, and (d) key used in extraction of watermark is changed from $k_8 = r = 1.46$ to $k_8' = r = 1.4600001$. In all these images, successful extraction is not done despite a minor change in the secret keys showing the strong results of key sensitivity of our proposed watermarking algorithm.

and watermarked image is shown in Figure 11(c). Then we have tried to extract the watermark image with a slightly change different keys. We have considered four cases. In the first case, we have changed $k_1 = x_0 = 0.4$ to $k_1' = x_0 = 0.40000000001$ while keeping the remaining eight keys as they are. The extracted image with this changed key, k_1', is shown in Figure 12(a); we can see that the extraction is not successful despite a slight change in one of the secret keys. In the second case, we have changed $k_3 = x_0 = 0.5$ to $k_3' = x_0 = 0.50000000001$ and extracted image is shown in Figure 12(b). In the third case, we have changed $k_5 = \mu = 0.4$ to $k_5' = \mu = 0.40000000001$ and extracted image is shown in Figure 12(c). In the fourth case, we have changed $k_8 = r = 1.46$ to $k_8' = r = 1.4600001$ and extracted image is shown in Figure 12(d). In all these images, successful extraction is not done despite a minor change in the secret keys showing the strong results of the key sensitivity of our proposed watermarking algorithm.

5.2. Robustness: Noise Resistant Analysis. One of the necessary features of a modern security system is to be noise resistant [29, 30]. The noise can be added intentionally by an unauthenticated user or can be caused due to the channel noise; channel can be wired or wireless as well. It is a well-known fact that the data that is to be transmitted through any channel is effected by the channel noise. If the watermarked data (image) is corrupted even a little bit, the renown security systems of watermarking do not tend to successfully extract the watermark from the corrupted watermarked image. To handle this situation, error detection and correction are used in parallel to watermarking at transmitter before sending or uploading that watermarked image. When extracting, error correction takes place before the extraction to successfully extract the watermark from the corrupted watermarked image. However, this adds the computational complexity of the whole system. The systems at the transmitter as well as at receiver need more



(a) (b) (c)

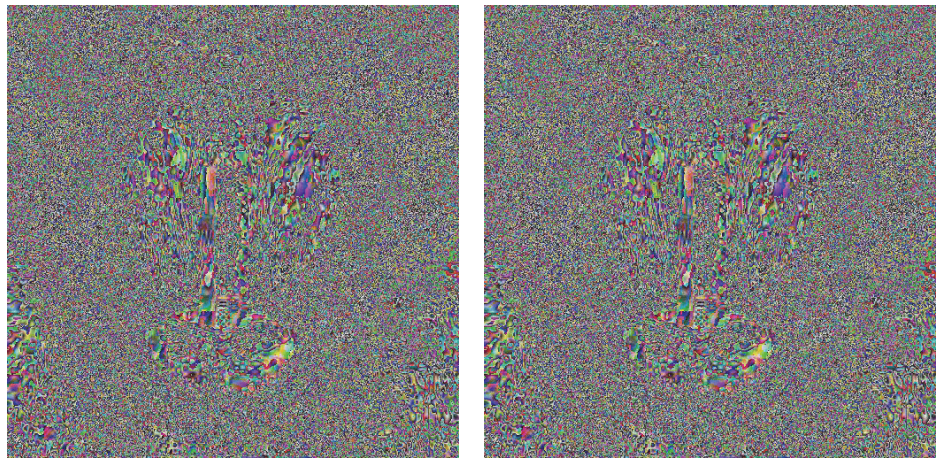
Figure 13: Noise addition results on watermarked images. (a) The watermarked image in which pixels from different locations are corrupted or cropped with the white pixels. (b) The watermarked image in which salt and pepper noise with density equal to 0.05 is added. (c) The watermarked image in which salt and pepper noise with density equal to 0.1 is added to further test the robustness.

(a) (b) (c)

Figure 14: The watermark images after extracting the watermark form the noisy watermarked images. (a) The watermark extracted from Figure 13(a), (b) the watermark extracted from Figure 13(b), and (c) the watermark extracted from Figure 13(c). It can be observed that the extraction is successful with minor changes. The results confirm the robustness of our proposed watermarking algorithm.

time to process the information and therefore need more computational resources. This may not be required in some low profile applications where speed is more required as compared to security. Our proposed watermarking algorithm can extract the watermark from the corrupted watermarked image correctly with some minor changes despite the fact that the watermarked image is undergone through noise addition. We have conducted experiments considering noise addition in watermarked images and then tried to extract the watermarks from those noisy watermarked images. We have embedded the cameraman image in the carrier image of baboon shown in Figure 11(a) with the secret keys mentioned in Table 3 and watermarked image is shown in Figure 11(c). Then we have added different types of noises in it. Figure 13(a) shows the watermarked image in which pixels from different locations are corrupted or cropped with the white pixels. Similarly, we have added salt and pepper noise with a density equal to 0.05. The noisy watermarked image is shown in Figure 13(b). Furthermore, to further test the robustness, we have added salt and pepper noise with a density equal to 0.1. The noisy watermarked image with that density is shown in Figure 13(c). After extracting the watermark form these noisy watermarked images, the watermark images are shown in Figure 14. The watermark extracted from Figure 13(a) is shown in Figure 14(a), the watermark extracted from Figure 13(b) is shown in Figure 14(b) and the watermark extracted from Figure 13(c) is shown in Figure 14(c). It can be observed that the extraction is successful with minor changes. The results confirm the robustness of our proposed watermarking algorithm.

The robustness to noise attacks can be numerically measured as well through the confidence measure proposed



(a) (b)

Figure 15: (a) LSBs of carrier image before the embedding of watermark and (b) LSBs of watermarked image after the embedding of watermark. For the plot, the 4 LSBs of carrier and watermarked images are picked and consider them as 4 MSBs of these two images and for 4 LSBs, 4 zeros are simply considered. It can be seen that there is no visual difference between these two images and thus our proposed algorithm is robust against LSB attack.

Table 6: A comparison results of similarity of different watermarking techniques resulting in applying various noise attacks. Again, it can be seen that the proposed work has superior performance.

Attacks	Other Works	Images/Similarity Index (%)		
		Baboon	Lena	Pepper
Noise	Ref. [17]	72	74	73
	Ref. [16]	72	74	73
	Proposed	88	89	88
Compression	Ref. [17]	67	69	69
	Ref. [16]	67	69	70
	Proposed	85	81	84
Cropping	Ref. [17]	40	42	39
	Ref. [16]	40	42	39
	Proposed	68	69	67

by [31] which returns a numeric value of similarity. The confidence measure is given as [31]

$$Sim = \frac{\sum t_i \cdot s_i}{\sqrt{\sum t_i^2 \cdot \sum s_i^2}} \tag{19}$$

We have performed this confidence measure on our proposed technique as comparison to other works as well. However, for comparison, we have taken the value of similarity as a percentage instead of exact value. Table 6 lists the results of similarity of different watermarking techniques resulting in applying various noise attacks. Again, it can be seen that the proposed work has superior performance.

5.3. LSBs Attack. In this attack, the attacker attempts to find the visual difference between the LSBs of carrier image and watermarked image. As the watermark is embedded into the LSBs of carrier image, this attack has the significance importance. It is assumed that the attacker has access to the watermarked image and carrier image as well (although this assumption leads to the compromise of the carrier and correspondingly the copyrights of this carrier image). The attacker plots the LSBs of the carrier and watermarked images and then tries to find the differences and subsequently tries to extract the watermark. It is required in a good watermarking algorithm that no visual difference should be visible. For the plot, we have picked the 4 LSBs of the carrier and watermarked images and consider them as 4 MSBs of these two images, and for 4 LSBs, we simply consider four zeros. Figure 15(a) shows the LSBs of carrier image before the embedding of watermark and Figure 15(b) shows the LSBs of the watermarked image after the embedding of the watermark. It can be seen that there is no visual difference between these two images and thus our proposed algorithm is robust against LSB attack.

5.4. Adjacent Pixel Difference Analysis. One of the other methods to analyze the LSBs in a watermarked image is to

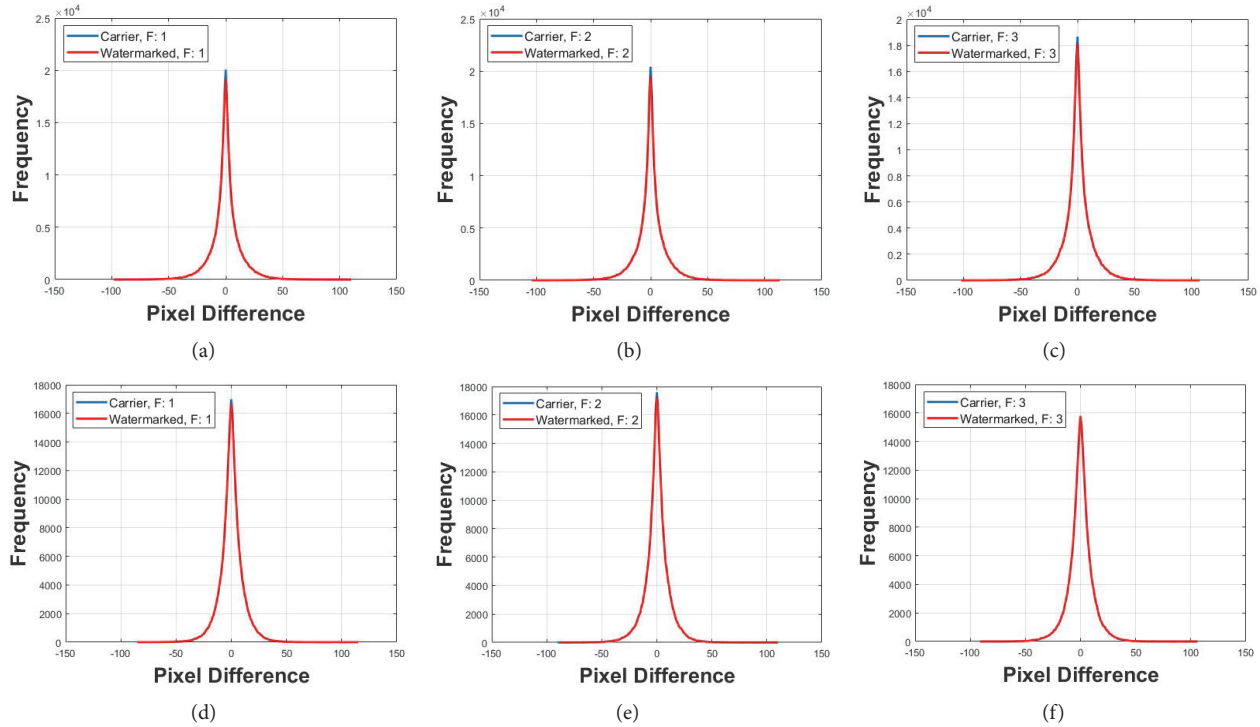
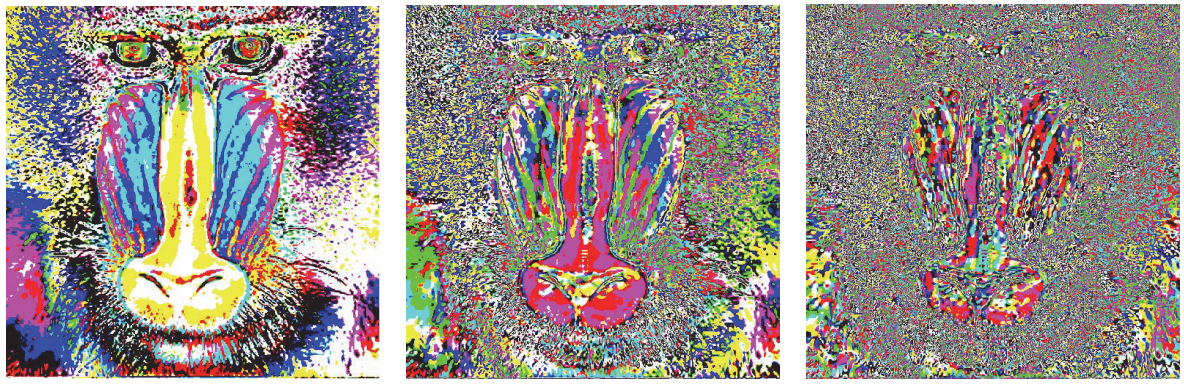


FIGURE 16: (a)-(c) The difference of adjacent pixels for frames 1-3 of carrier and watermarked images when the difference is considered row-wise. In row-wise, the difference of those two image pixels is considered whose positions are in same row but separated by a single column. Similarly, (d)-(f) show the difference of adjacent pixels for frames 1-3 of carrier and watermarked images when the difference is considered column-wise. In column-wise, the difference of those two image pixels is considered whose positions are in same column but separated by a single row. In all these images, the differences of adjacent pixels of carrier and watermarked images are almost the same showing the robustness of proposed watermarking algorithm.

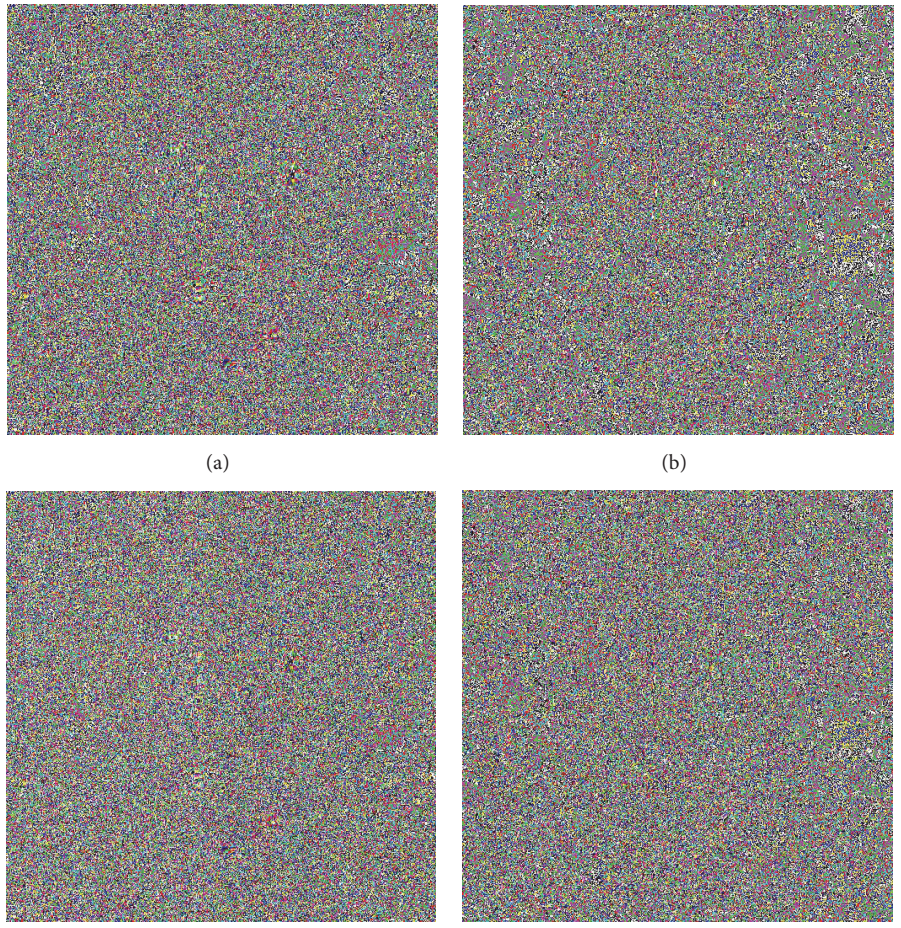
see the difference between adjacent pixels. In an image, the correlation between two adjacent pixels is very high, and thus the difference between these two image pixels is usually close to zero. However, when the watermark is embedded into the LSBs of carrier image to generate a watermarked image, the correlation between adjacent pixels decreases where the watermark is embedded, and the difference of these two image pixels moves away from zero. It is required that the difference between adjacent pixels of both the carrier and watermarked images should be same to each other or near to each other. We have plotted the difference between adjacent pixels of the carrier and watermarked images to see the similarity between these two images. Particularly, we have plotted the difference of adjacent pixels of individual frames of all three frames of the carrier and watermarked images. Figures 16(a)–16(c) show the difference of adjacent pixels for frames 1-3 of the carrier and watermarked images when the difference is considered row-wise. In row-wise, the difference between those two image pixels is considered whose positions are in the same row but separated by a single column. Similarly, Figures 16(d)–16(f) show the difference of adjacent pixels for frames 1-3 of the carrier and watermarked images when the difference is considered column-wise. In column-wise, the difference between those two image pixels is considered whose positions are in the same column but separated by a single row. In all these images, the differences of adjacent pixels of the carrier and watermarked images are almost same showing the robustness of the proposed watermarking algorithm.

5.5. Visual Attack Examining Bit by Bit. It is assumed for a long time since the sophistication of watermarking techniques that the LSBs of a digital image do not contain the important information regarding the image. However, this is not the case for all the images. There are works in the literature on watermarking attacks that suggest examining the LSBs of the carrier and watermarked images to possibly extract the watermark from the watermarked image. To further elaborate this point, we have plotted the single bits of carrier image of a baboon to examine the important information. Figure 17(a) shows the plot of 7th LSB ('xBxxxxxx,' B is the 7th bit) of carrier image; the information is visible due to the high percentage of information (50%) present in it. However, as we move towards the next LSBs, the information tends to lose as can be seen in Figures 17(b) and 17(c) which shows the plot of sixth and 5th LSB, respectively. To show the robustness of proposed watermarking algorithm, we have plotted the single bits of those LSBs of the carrier and watermarked images in which the watermark is embedded, that is, first and second bits. Figure 18(a) shows the plot of second LSB and Figure 18(b) shows the plot of first LSB of carrier image of baboon. As the watermark is embedded in these 2 LSBs



(a) (b) (c)

Figure 17: (a) The plot of 7th LSB ('xBxxxxxx', B is the 7th bit) of carrier image; the information is clearly visible due to high percentage of information (50%) present in it. However as we move towards the next LSBs, the information tends to lose as can be seen in (b) and (c) which show the plot of 6th and 5th LSB, respectively.



(a) (b)

(c) (d)

Figure 18: (a) The plot of second LSB and (b) the plot of first LSB of carrier image of baboon. As the watermark is embedded in these 2 LSBs of carrier image to get the watermarked image therefore it is necessary that the plot of individual bits of first and second LSBs of watermarked image should be similar to the plots of bits of carrier image. (c) The plot of second LSB and (d) the plot of first LSB of watermarked image. It can be seen that these two plots are very similar to the plots of carrier image and they do not give any information about the watermark, thus showing the robustness of proposed watermarking algorithm.

of carrier image to get the watermarked image, therefore, it is necessary that the plot of individual bits of first and second LSBs of the watermarked image should be similar to the plots of bits of carrier image. Figure 18(c) shows the plot of second LSB and Figure 18(d) shows the plot of first LSB of watermarked image. It can be seen that these two plots are very similar to the plots of carrier image and they do not give any information about the watermark thus showing the robustness of the proposed watermarking algorithm.

6. Conclusion

We have developed a unified watermarking algorithm using three different and distinct chaotic maps in which one map is proposed in this work. The embedding of the watermark is operated by the individual chaotic sequence generated by a different chaotic map. The simulation results and security analysis confirmed that the proposed algorithm is secure against well-known attacks. Like all new proposals, we strongly encourage the analysis of our framework before its immediate deployment. The proposed algorithm is a generalized watermarking model that can incorporate changes as required. For instance, the number of substitution boxes can be increased for better security but at the expense of more computational complexity. Furthermore, the work can be extended for the application of steganography as well in which instead of the watermark, the secret message can be inserted for information hiding.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through research groups program under Grant no. R.G.P-1/5/38.

References

[1] Y.-M. Chu, N.-F. Huang, and S.-H. Lin, "Quality of service provision in cloud-based storage system for multimedia delivery," *IEEE Systems Journal*, vol. 8, no. 1, pp. 292–303, 2014.

[2] I. Hussain, T. Shah, M. A. Gondal, and H. Mahmood, "An efficient approach for the construction of LFT S-boxes using chaotic logistic map," *Nonlinear Dynamics*, vol. 71, no. 1-2, pp. 133–140, 2013.

[3] L. M. Pecora and T. L. Carroll, "Synchronization in chaotic systems," *Physical Review Letters*, vol. 64, no. 8, pp. 821–824, 1990.

[4] L. Kocarev, "Chaos-based cryptography: a brief overview," *IEEE Circuits and Systems Magazine*, vol. 1, no. 3, pp. 6–21, 2001.

[5] I. Hussain, A. Anees, M. Aslam, R. Ahmed, and N. Siddiqui, "A noise resistant symmetric key cryptosystem based on S8 S-boxes and chaotic maps," *The European Physical Journal Plus*, vol. 133, no. 4, 2018.

[6] I. Hussain, A. Anees, A. H. AlKhaldi, A. Algarni, and M. Aslam, "Construction of chaotic quantum magnets and matrix Lorenz systems S-boxes and their applications," *Chinese Journal of Physics*, 2018.

[7] I. Hussain, A. Anees, and A. Algarni, "A novel algorithm for thermal image encryption," *Journal of integrative neuroscience*, pp. 1–15, 2018.

[8] A. Anees, A. M. Siddiqui, and F. Ahmed, "Chaotic substitution for highly autocorrelated data in encryption algorithm," *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, no. 9, pp. 3106–3118, 2014.

[9] I. A. Al-Kadi, "Origins of cryptology: the Arab contributions," *Cryptologia*, vol. 16, no. 2, pp. 97–126, 1992.

[10] T. T. Mapoka, S. J. Shepherd, and R. A. Abd-Alhameed, "A new multiple service key management scheme for secure wireless mobile multicast," *IEEE Transactions on Mobile Computing*, vol. 14, no. 8, pp. 1545–1559, 2015.

[11] A. Anees, W. A. Khan, M. A. Gondal, and I. Hussain, "Application of mean of absolute deviation method for the selection of best nonlinear component based on video encryption," *Zeitschrift fur Naturforschung - Section A Journal of Physical Sciences*, vol. 68, no. 6-7, pp. 479–482, 2013.

[12] H. Liu and X. Wang, "Color image encryption based on one-time keys and robust chaotic maps," *Computers & Mathematics with Applications. An International Journal*, vol. 59, no. 10, pp. 3320–3327, 2010.

[13] A. Anees and Z. Ahmed, "A Technique for Designing Substitution Box Based on Van der Pol Oscillator," *Wireless Personal Communications*, vol. 82, no. 3, pp. 1497–1503, 2015.

[14] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-The Advanced Encryption Standard*, Springer, Berlin, Germany, 2002.

[15] A. Anees and M. A. Gondal, "Construction of Nonlinear Component for Block Cipher Based on One-Dimensional Chaotic Map," *3D Research*, vol. 6, no. 2, 2015.

[16] A. Anees and A. M. Siddiqui, "A technique for digital watermarking in combined spatial and transform domains using chaotic maps," in *Proceedings of the 2013 2nd National Conference on Information Assurance, NCIA 2013*, pp. 119–124, pak, December 2013.

[17] S. S. Jamal, M. U. Khan, and T. Shah, "A Watermarking Technique with Chaotic Fractional S-Box Transformation," *Wireless Personal Communications*, vol. 90, no. 4, pp. 2033–2049, 2016.

[18] S. S. Jamal, T. Shah, and I. Hussain, "An efficient scheme for digital watermarking using chaotic map," *Nonlinear Dynamics*, vol. 73, no. 3, pp. 1469–1474, 2013.

[19] W. Sheng, S. Chen, G. Xiao, J. Mao, and Y. Zheng, "A Biometric Key Generation Method Based on Semisupervised Data Clustering," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 9, pp. 1205–1217, 2015.

[20] M.-H. Lim, A. B. J. Teoh, and K.-A. Toh, "An efficient dynamic reliability-dependent bit allocation for biometric discretization," *Pattern Recognition*, vol. 45, no. 5, pp. 1960–1971, 2012.

[21] W. Sheng, G. Howells, M. Fairhurst, and F. Deravi, "Template-free biometric-key generation by means of fuzzy genetic clustering," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 2, pp. 183–191, 2008.

[22] E. J. Kelkboom, G. G. Molina, J. Breebaart, R. N. Veldhuis, T. A. Kevenaar, and W. Jonker, "Binary Biometrics: An Analytic Framework to Estimate the Performance Curves Under Gaussian Assumption," *IEEE Transactions on Systems, Man, and*

Cybernetics - Part A: Systems and Humans, vol. 40, no. 3, pp. 555–571, 2010.

[23] P. F. Verhulst, "Recherches mathématiques sur la loi daccroissement de la population," *Nouveaux mémoires de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles*, vol. 18, pp. 14–54, 1845.

[24] N. K. Pareek, V. Patidar, and K. K. Sud, "Cryptography using multiple one-dimensional chaotic maps," *Communications in Nonlinear Science and Numerical Simulation*, vol. 10, no. 7, pp. 715–723, 2005.

[25] A. Rukhin, J. Sota, J. Nechvatal et al., "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Special Publication NIST 800-22, National Institute of Standards and Technology, 2010.

[26] S. L-Yuan, S. K-Hui, and L. C-Bing, "Study of a discrete chaotic system based on tangent-delay for elliptic reflecting cavity and its properties," *Acta Physica Sinica*, vol. 53, no. 9, pp. 2871–2876, 2004.

[27] X. Wang and D. Chen, "A parallel encryption algorithm based on piecewise linear chaotic map," *Mathematical Problems in Engineering*, vol. 2013, 2013.

[28] A. Anees, A. M. Siddiqui, J. Ahmed, and I. Hussain, "A technique for digital steganography using chaotic maps," *Nonlinear Dynamics*, vol. 75, no. 4, pp. 807–816, 2014.

[29] F. Ahmed, A. Anees, V. U. Abbas, and M. Y. Siyal, "A noisy channel tolerant image encryption scheme," *Wireless Personal Communications*, vol. 77, no. 4, pp. 2771–2791, 2014.

[30] F. Ahmed and A. Anees, "Hash-Based Authentication of Digital Images in Noisy Channels," *Robust Image Authentication in the Presence of Noise*, pp. 1–42, 2015.

[31] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1673–1687, 1997.

Hindawi
Security and Communication Networks
Volume 2018, Article ID 4983404, 18 pages
https://doi.org/10.1155/2018/4983404

Research Article

A Vendor-Neutral Unified Core for Cryptographic Operations in GF(p) and GF(2^m) Based on Montgomery Arithmetic

Martin Schramm,[1,2] **Reiner Dojen**,[1] **and Michael Heigl**[2]

[1]*Department of Electronic and Computer Engineering, University of Limerick, Limerick, Ireland*
[2]*Institute ProtectIT, Deggendorf Institute of Technology, 94469 Deggendorf, Germany*

Correspondence should be addressed to Martin Schramm; martin.schramm@th-deg.de

Received 6 October 2017; Revised 14 March 2018; Accepted 17 May 2018; Published 21 June 2018

Academic Editor: Fawad Ahmed

In the emerging IoT ecosystem in which the internetworking will reach a totally new dimension the crucial role of efficient security solutions for embedded devices will be without controversy. Typically IoT-enabled devices are equipped with integrated circuits, such as ASICs or FPGAs to achieve highly specific tasks. Such devices must have cryptographic layers implemented and must be able to access cryptographic functions for encrypting/decrypting and signing/verifying data using various algorithms and generate true random numbers, random primes, and cryptographic keys. In the context of a limited amount of resources that typical IoT devices will exhibit, due to energy efficiency requirements, efficient hardware structures in terms of time, area, and power consumption must be deployed. In this paper, we describe a scalable word-based multivendor-capable cryptographic core, being able to perform arithmetic operations in prime and binary extension finite fields based on Montgomery Arithmetic. The functional range comprises the calculation of modular additions and subtractions, the determination of the Montgomery Parameters, and the execution of Montgomery Multiplications and Montgomery Exponentiations. A prototype implementation of the adaptable arithmetic core is detailed. Furthermore, the decomposition of cryptographic algorithms to be used together with the proposed core is stated and a performance analysis is given.

1. Introduction

The next generation of embedded systems and IoT devices will exhibit a much higher degree of internetworking which gives rise to security considerations [1]. As a logical consequence, such devices must become cryptographic nodes, besides others, being capable of encrypting/decrypting and signing/verifying data as well as establishing spontaneous secured communications by exchanging common secrets used for secret key calculation. While many embedded chips already have support for hardware-accelerated symmetric algorithms (mainly AES) [2] and hash functions, due to various reasons, such as complexity, space, and costs, they lack in hardware support especially for supporting a wide range of public-key and key exchange algorithms with different precision widths. Besides, many modern cryptographic primitives necessitate the capability for producing true random numbers and random prime numbers. Typical IoT devices furthermore very often only exhibit a limited amount of resources which requires efficient cryptographic hardware structures in terms of area, power consumption, and calculation performance [3]. In general enterprises developing IoT products basically have three options to include application functionalities in high integrated devices, using Application Specific Standard Products (ASSP), Application Specific Integrated Circuits (ASIC), or Field Programmable Gate Arrays (FPGA). Today FPGAs have become promising components for IoT applications [4], compared to ASSP solutions which often cannot provide the required functionality and can provide a better Total Cost of Ownership (TCO) compared to ASIC solutions. Thus for devices which are equipped with a FPGA device, it is valuable to examine how efficient hardware structures for performing cryptographic operations can be included.

In matters of algorithm agility an arithmetic engine with minimal hardware footprint, which can handle the arithmetic

operations of a great variety of cryptographic algorithms, is of great importance for IoT based devices. Especially the calculability of the individual operations leading to lower and upper calculation time bounds is quite important.

This paper proposes a tiny-held vendor-neutral cryptographic arithmetic core exemplarily implemented in FPGA-logic. For efficiency, time-intensive modular operations, such as multiplication and exponentiation operations, Montgomery Arithmetic is used. Without the need of any expensive software precalculations the core is able to perform a high number of cryptographic algorithms and handle various key sizes by simply processing operation lists. Furthermore the core architecture is unified and can perform calculations in both prime finite fields ($GF(p)$) and binary extension fields ($GF(2^m)$). To illustrate the versatility of the developed core, well-established cryptographic algorithms have been rewritten and fragmented into operation lists to be processed by the arithmetic engine.

The paper is organized as follows. Section 2 states the related work of this research. In Section 3 the design of the proposed Enhanced Montgomery Multiplication Core is stated; the specified functional range of the core is given in Section 4. In Section 5 some exemplary application descriptions for the core are mentioned and in Section 6 the results of the performance analysis are stated. Finally, Section 7 concludes the paper.

2. Related Works

The efficiency of cryptographic algorithms when implemented on reconfigurable hardware is mainly determined by the fact of how the underlying finite field arithmetic operations are realized [5]. Several applications in cryptography such as ciphering and deciphering of asymmetric algorithms, the creation and verification of digital signatures, and secure key exchange mechanisms require excessive use of the basic finite field modular arithmetic operations addition, multiplication, and the calculation of the multiplicative inverse. Especially the field multiplication operation is crucial to the efficiency of a design, since it is the core operation of many cryptographic algorithms [6].

In [7] P. L. Montgomery introduced a representation of residue classes in order to speed up modular multiplications without affecting modular additions and subtractions. Over the years numerous designs have been proposed implementing modular multiplications based on Montgomery's multiplication algorithm [8]. The foundation for these architectures was presented by A. Tenca and Ç. Koç in [9]. The architecture is based on a word-based Montgomery Multiplication algorithm for prime finite fields in which multiplications are performed in a bit-serial fashion. E. Savaş et al. in [10] have proposed an extension which, in addition to the standard integer modulo arithmetic, also allows polynomial computations over binary finite fields. An overview about algorithms and hardware architectures for Montgomery Multiplication can be found in [11]. Optimizations of the original design have been proposed concerning the hardware implementation of the Montgomery Multiplication algorithm [12] as well as by utilizing special arithmetic hardcore extensions of FPGAs to accelerate digital signal processing applications [13]. Some designs only focus on utilizing the Montgomery Multiplication method to accelerate modular exponentiation operations as required by the RSA algorithm [14, 15].

However, no publication focuses on how the Montgomery Multiplication architecture can be embedded into a comprehensive solution. In this paper we propose an enhanced version of a bit-serial word-based unified Montgomery Multiplication core based on logic elements only which is controlled by a state machine and offers the functional range to be able to perform complete cryptographic algorithms without additional complex processing required in software.

3. Enhanced Montgomery Multiplication Core

3.1. Requirements. Today a high number of different public-key algorithms are in use. To ensure compatibility, cryptographic applications must support a large portion of those algorithms. While typical software implementations often can easily be upgraded in order to adapt new algorithms and larger key sizes, the same is not necessarily true for hardware implementations. Therefore following requirements have been identified for the Enhanced Montgomery Multiplication Core:

(i) *Use of Montgomery Arithmetic*. The design must be able to perform modulo operations in a time-efficient manner by using Montgomery Arithmetic. At least the core must support Montgomery Multiplications and Montgomery Exponentiations. Furthermore the core must support standard modulo additions and modulo subtractions.

(ii) *Works on Both Finite Fields $GF(p)$ and $GF(2^m)$*. The architecture must exhibit an unified structure supporting both standard integer modulo operations of prime finite fields as well as polynomial calculations of binary finite fields.

(iii) *Montgomery Parameter Calculation*. In general the Montgomery Parameters (r and r^2) can be precomputed for previously known moduli. However, as a requirement the core must be able to handle arbitrary moduli. Therefore it must be capable of calculating the Montgomery Parameters $r \bmod n$, $r^2 \bmod n$ and $r^{-1} \bmod n$ without the need of precalculations done in software.

(iv) *Scalable Design*. The architecture must be scalable in terms of timing, area, and power consumption. This includes the parametrisation of the word width, the internal storage size, and the amount of processing units within the pipeline.

(v) *Multialgorithm Support*. The core must be based on a building-block design. The functional range provided by the arithmetic unit should empower algorithm agility, by fragmenting cryptographic algorithms into a list of core operations. At least the core must be capable of performing RSA [16] operations, (safe) prime number generation and primality testing (MR)

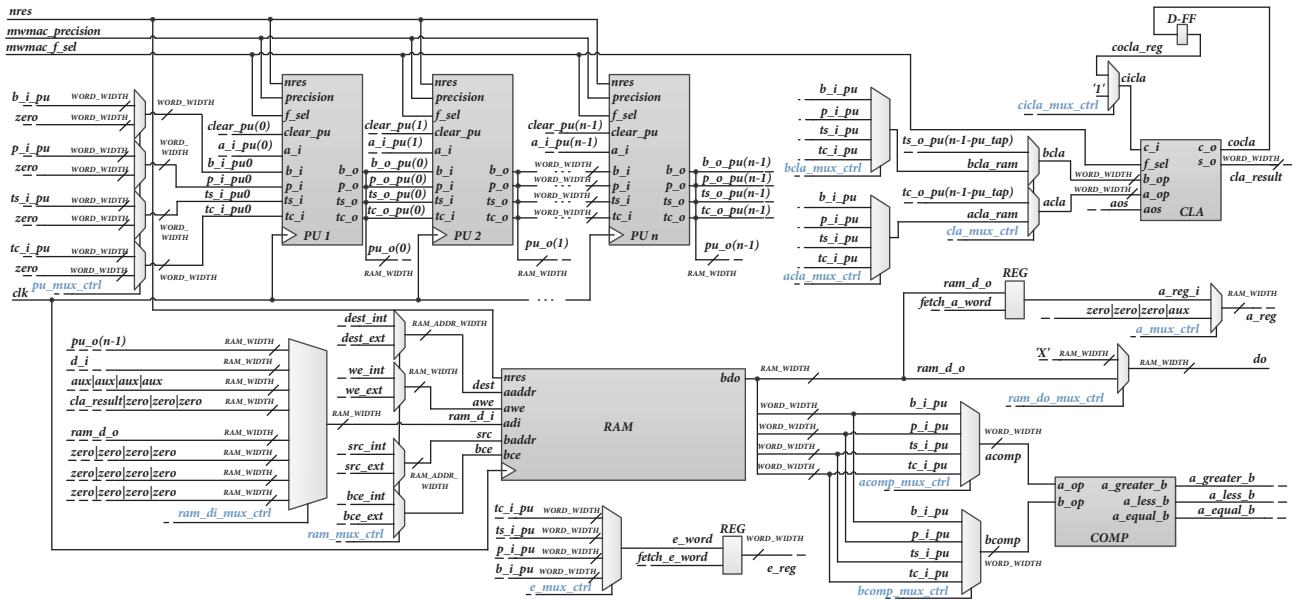


Figure 1: Overall architecture of the Enhanced Montgomery Multiplication Core.

[17, 18], key exchange operations (DH) [19], and elliptic curve calculations (EC) [20] over both prime and binary finite fields.

(vi) *Supporting as Many Precision Widths as Possible*. The design must support a wide range of different precision widths determining the security level of the cryptographic algorithm. If a certain security level, due to increased attacking computing power, becomes inadequate, the precision width can be adjusted accordingly which makes the hardware less prone to become obsolete due to higher security demands. The core must support the current recommendations for minimum key sizes [21] and should also support larger key sizes. For RSA algorithm and Diffie-Hellman key exchange support the architecture should be able to handle precisions up to 4096 bit moduli, for elliptic curve cryptography support precisions up to 512 bits for prime finite fields and precisions up to 571 bits for binary finite fields should be possible.

(vii) *Time-Invariant Operations*. The architecture must be capable of performing its operations in a time-invariant manner. If security sensitive information, such as private keys, will be processed, it must be ensured that all operations exhibit the same execution time to prevent side-channel attacks based on timing analysis.

3.2. Overall Core Architecture. Figure 1 illustrates the overall architecture of the proposed Enhanced Montgomery Multiplication Core which is capable of meeting all requirements as specified above.

Besides the pipeline of processing units handling the main part of the word-based Montgomery Multiplication algorithm, the core features an enhanced word-based Carry Look-Ahead adder being responsible for the calculation of the final result after the pipeline has processed all bits of an operand as well as for performing single modular addition and subtraction operations. The register files of the original design have been replaced with an internal dual-ported RAM which holds the operands as well as intermediate results of the core operations. Furthermore a word-based comparator component has been described which is queried during operations to decide if a modular addition or subtraction step must be performed. Two additional r-bit words for the A operand and the exponent E have been introduced with $|r|$ being the RAM width ($|r| = 4 \cdot |w|$) which will be fetched from RAM in case of Montgomery Multiplication and Montgomery Exponentiation operations. An auxiliary w-bit word *aux* is used for RAM reorganisation operations as well as for the calculation of the Montgomery Parameters r and r^2.

The intelligence of the core is the controlling state machine which utilizes the defined components to perform standard modular addition and subtraction operations, Montgomery Multiplications, Montgomery Exponentiations, Montgomery Parameter calculation, and RAM reorganisation operations. Therefore it is responsible for controlling the RAM write and read access, the source and destination address signals of RAM, as well as the values passed through to the first processing unit, to the CLA adder, and to the comparator component. Furthermore it controls the assignments of A operand, E exponent, and *aux* words.

The described core can be parametrised in three ways. The parameter named *MAX_PRECISION_WIDTH* specifies the highest supported precision width $|p|$, whereas the parameter *WORD_WIDTH* is used to specify the word width $|w|$ of the operands involved in the calculations. These two parameters determine the size and the address space of the internal core RAM. The third parameter *MAX_NUM_PUS* specifies

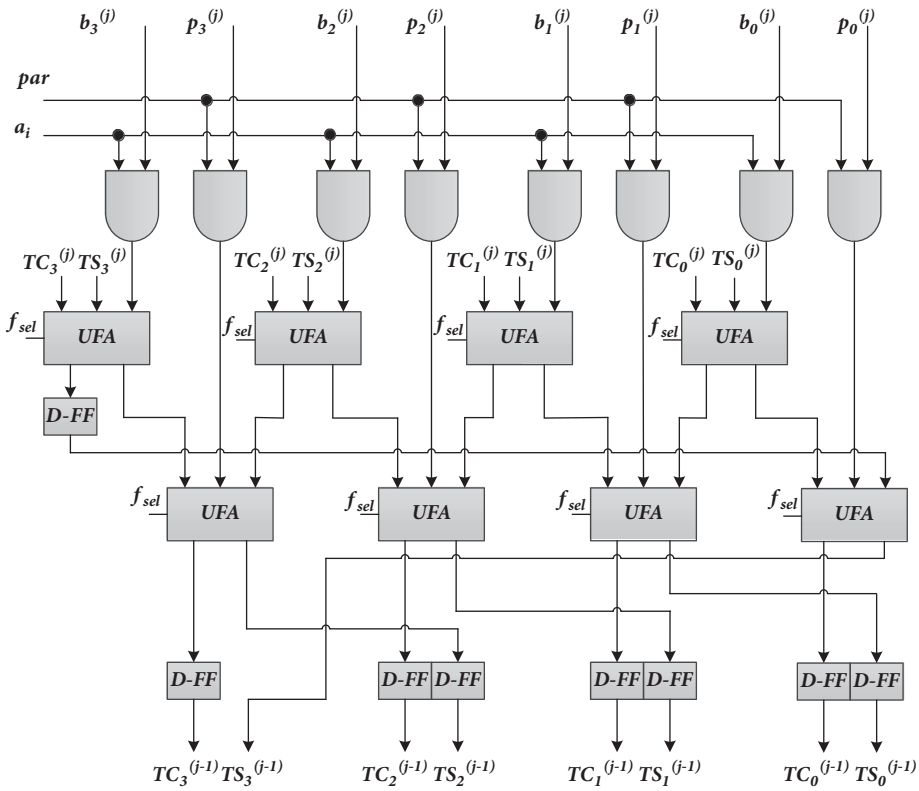


Figure 2: Processing unit with word size $w = |4|$.

the maximum number of processing units of the pipeline implemented for a specific core variation mainly affecting the performance and the size in terms of area consumption.

3.2.1. Processing Units. The heart of the core is the pipeline of processing units implementing the multiple word version of the Montgomery Multiplication algorithm. Therefore the processing unit structure has been described from scratch. The processing unit can be held in reset and keeps track of the cycle number according to the number of words to be processed depending on the supplied parameters. This control logic is needed to determine whether the supplied modulus has to be added to the processed words in this cycle or not, depending on the value of the signal *par* denoting an odd intermediate result. Note that buffering the output of a processing unit between two processing units is not required in this design. Compared to the original design presented in [10] for a given precision width $|p|$ and a word size $|w|$, $e = \lceil |p|/|w| \rceil + 1$ number of words are required for a unified solution and the pipeline must consist of a power of two (2^x) number of processing units with a maximum number of $2^x < (e-1)$ in order to avoid pipeline stalls. Figure 2 illustrates the internal architecture of an exemplary processing unit with word size $|w| = 4$.

Each processing unit consists of a cascade of two layers of so-called Unified Full Adder (UFA) cells. The Unified Full Adder cells basically consist of simple full adder cells which have been enhanced by an additional finite field selection input f_{sel}. This allows for the creation of a unified multiplier architecture which can not only be used in prime fields $GF(p)$ ($f_{sel} = 1$) but also in binary fields $GF(2^m)$ ($f_{sel} = 0$) in which additions will be simple bitwise XOR calculations without any carry output.

3.2.2. Carry Look-Ahead Adder. Since the pipeline generates the result in carry save form, an additional step is necessary at the end of each calculation to obtain a nonredundant version of the result. For the sake of uniformity a circuit is required that can operate in both finite fields $GF(p)$ and $GF(2^m)$. Furthermore, since the calculation in $GF(p)$ could require one further subtraction step, the Carry Look-Ahead adder in the design has been formulated to be able to perform word-based modular additions and subtractions. Figure 3 illustrates the logic of the proposed enhanced n–bit wide CLA adder of the core.

The internal signal b' of the second operand will be calculated as $b_i' = b_i \oplus (aos \cdot f_{sel})$ in which *aos* denotes an add-or-subtract signal ($aos = 0$ means addition, $aos = 1$ represents subtraction by performing an addition in two's complement representation). The modified CLA adder involves the same common Carry Look-Ahead adder logic for the calculation of the generate ($g_i = a_i \cdot b_i'$) and propagate ($p_i = a_i + b_i'$) functions. The output values c_i of the CLA adder logic will be calculated as $c_0 = c_{in}$ for the least-significant bit and $c_i = g_{i-1} + (p_{i-1} \cdot c_{i-1})$ for all further bits. The final sum output bits s_i will be calculated as $s_i = (c_i \cdot f_{sel}) \oplus a_i \oplus b_i'$ the carry output bit will be determined as $c_{out} = c_n \cdot f_{sel}$. If the selected finite field is $GF(2^m)$ ($f_{sel} = 0$), then the add-or-subtract input will

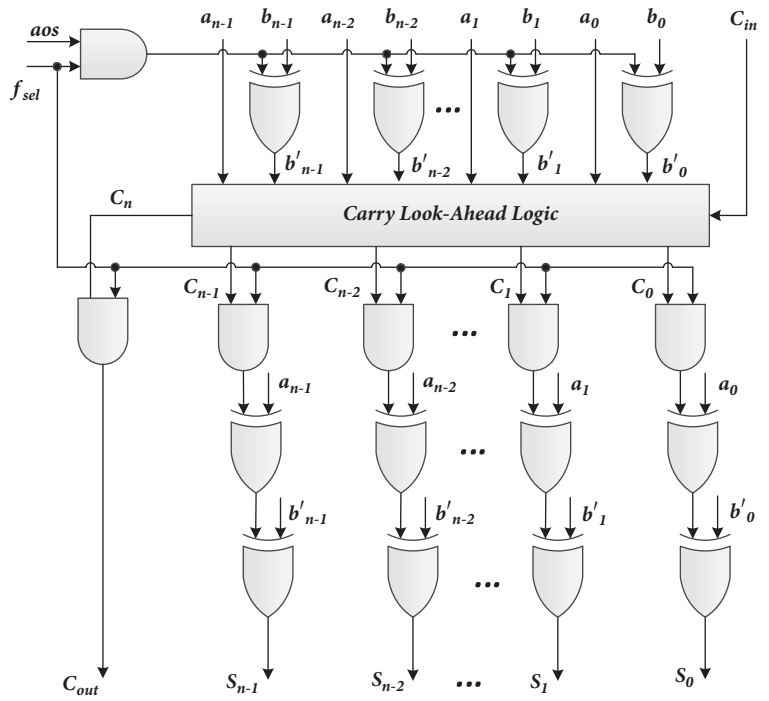


Figure 3: Enhanced n-bit wide CLA adder.

X_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1	E_8	E_7	E_6	E_5	E_4	E_3	E_2	E_1	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	$B_{8\ 7}$	$B_{7\ 6}$	$B_{6\ 5}$	$B_{5\ 4}$	$B_{4\ 3}$	$B_{3\ 2}$	$B_{2\ 1}$	$B_{1\ 0}$
																								$P_{8\ 15}$	$P_{7\ 14}$	$P_{6\ 13}$	$P_{5\ 12}$	$P_{4\ 11}$	$P_{3\ 10}$	$P_{2\ 9}$	$P_{1\ 8}$
																								$TS_{8\ 23}$	$TS_{7\ 22}$	$TS_{6\ 21}$	$TS_{5\ 20}$	$TS_{4\ 19}$	$TS_{3\ 18}$	$TS_{2\ 17}$	$TS_{1\ 16}$
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	$TC_{8\ 31}$	$TC_{7\ 30}$	$TC_{6\ 29}$	$TC_{5\ 28}$	$TC_{4\ 27}$	$TC_{3\ 26}$	$TC_{2\ 25}$	$TC_{1\ 24}$

Figure 4: Enhanced Montgomery Multiplication Core RAM organization.

be ignored, the final sum will simply be the bitwise modulo-2 addition of the two input values *TC* and *TS* and the carry output bit will be forced to zero.

3.2.3. Core RAM Structure. The RAM of the core must be capable of holding all the necessary operands and intermediate values required during the execution of cryptographic algorithms. The basic structure of the described RAM is pictured in Figure 4.

It features four symbolic horizontal RAM operand locations with *MAX_PRECISION_WIDTH* bit each which are organized as eight pieces of *MAX_PRECISION_WIDTH*/8 bit each. The location named *B* is intended to hold operand *B* in Montgomery Multiplication and Montgomery Exponentiation operations; the location named *P* is intended to hold the modulus. The location *TS* usually holds the temporary sum value during Montgomery Multiplications and Montgomery Exponentiation or the first operand in modular addition or subtraction operations. The location *TC* usually holds the temporary carry stream during Montgomery Multiplications and Montgomery Exponentiation or the second operand in modular addition or subtraction operations.

Besides the horizontal RAM operand locations three symbolic vertical RAM operand locations with *MAX_PRECISION_WIDTH* bit each have been defined which are organized as eight pieces of *MAX_PRECISION_WIDTH*/8 bit each. The locations named *A*, *E*, and *X* for convenience usually are used to hold operand *A* in Montgomery Multiplication and Montgomery Exponentiation operations as well as the exponent operand *E* and the auxiliary operand *X* in Montgomery Exponentiation operations. In addition all RAM slots are intended to hold intermediate values during the execution of cryptographic algorithms.

4. Functional Range of the Core

This section provides a description of the functional range of the proposed core. The following precisions (denoted in bit-length) are supported:

(i) *EC over GF(p), RSA, MR, DH*: 192, 224, 256, 320, 384, 448, 512, 768, 1024, 1536, 2048, 3072, 4096

(ii) *EC over GF*(2^m): 131, 163, 176, 191, 193, 208, 233, 239, 272, 283, 304, 359, 368, 409, 431, 571

If further or other precision widths should be supported, the described core can easily be adjusted in an appropriate manner. For the parametrisation and the execution/abortion of an operation a 32-bit wide command input word has been defined. Besides the start, abort, and finite field selection signals also the encoded precision width, operation code as well as RAM offsets for the specified operation can be supplied. The following operations have been specified.

4.1. MontMult Operation. The *MontMult* operation code instructs the core to perform a single Montgomery Multiplication with the supplied elements in the given finite field. A Montgomery Multiplication will start by reading the first r-bit word of operand A from RAM. Afterwards the pipeline will be started and the appropriate bits of A operand will be fed to the individual processing unit. If all bits of the A operand word have been fed to the processing units, a new word will be read from RAM. Once the last bit of A operand has been processed, the temporary sum and temporary carry words will be fed into the CLA adder in order to reunite the two streams. After the last words of temporary sum and temporary carry have been brought together, the carry output bit of the CLA adder will be evaluated. If a carry bit is set the modulus will be subtracted once; otherwise the result will be compared to the given modulus. If the result is equal or greater than the modulus the given modulus will be subtracted once.

4.2. MontR Operation. The *MontR* operation code instructs the core to calculate the Montgomery Parameter $r = 2^k$ regarding a supplied modulus in the given finite field, with k being the bit-length of the given precision.

In the case of prime field arithmetic the Montgomery Parameter r will be $r \equiv 2^k \bmod p$, so r can be calculated as two's complement of p as bitwise inverse of the given modulus plus 1. Therefore the individual words of the modulus will be XOR-ed with a constant word consisting of all-ones. In addition the least-significant bit of the first word will be set to one.

In the case of binary field arithmetic the Montgomery Parameter r will be $r \equiv 2^k \bmod n(x)$, so r is equal to binary expression of the irreducible polynomial $n(x)$ with the most significant bit set to zero. Therefore the individual words of the modulus will be scanned and the appropriate most significant bit will be set to zero, depending on the given precision.

4.3. MontR2 Operation. The *MontR2* operation code instructs the core to calculate the Montgomery Parameter r^2 with $r^2 = 2^{2 \cdot k}$ for a supplied modulus in the given finite field with k being the bit-length of the given precision.

In the case of prime field arithmetic the Montgomery Parameter r^2 will be given by $r^2 \equiv r \cdot r \equiv 2^k \cdot 2^k \equiv 2^{2 \cdot k} \bmod p$. Therefore in a first step the Montgomery Parameter $r \equiv 2^k \bmod p$ will be calculated for prime fields as described above. In order to calculate r^2 one possible way is to calculate $2^l \cdot 2^k \bmod p$ with l being a small divider of k. In the given implementation $l = 1$. Therefore the bits of r will be shifted to the left by one bit. If the result is equal or greater than the modulus, p will be subtracted once. By using a square-and-multiply-like algorithm, multiple Montgomery Multiplications will be performed in order to calculate $r^2 \equiv 2^k \cdot 2^k \bmod p$.

In the case of binary field arithmetic the Montgomery Parameter r^2 will be given by $r^2 \equiv r \cdot r \equiv 2^k \cdot 2^k \equiv 2^{2 \cdot k} \bmod n(x)$. Therefore in a first step the Montgomery Parameter $r \equiv 2^k \bmod n(x)$ will be calculated for binary fields as described above. In order to calculate r^2 the resulting parameter r will be shifted k-times bitwise to the left. After each shift, the most significant bit as given by the precision parameter will be evaluated. If the bit is one, the irreducible polynomial will be added to the intermediate result which represents a modulo reduction with $n(x)$. Once the shift has been performed k-times the result will be $r^2 \equiv 2^k \cdot 2^k \bmod n(x)$

4.4. MontExp Operation. The *MontExp* operation code instructs the core to perform a Montgomery Exponentiation consisting of multiple Montgomery Multiplication steps in the given finite field. A Montgomery Exponentiation will start by reading the first r-bit word of exponent E from RAM. Afterwards the first appearing one of the exponent word will be searched starting from the most significant bit. If the first word consists of all-zeros then the next word of exponent E will be read and evaluated. Once the highest bit of exponent E has been found, multiple Montgomery Multiplications will be performed until all bits of the exponent have been processed following a square-and-multiply algorithm.

4.5. ModAdd Operation. The *ModAdd* operation code instructs the core to perform a modular addition of the supplied elements in the given finite field. After preparing the core for the addition operation, the CLA adder will add the given operands using the appropriate arithmetic given by the finite field selection input. Once the last words of the given operands have been added the carry output bit of the CLA adder will be evaluated. If a carry bit is set, the modulus will be subtracted once; otherwise the result will be compared to the given modulus. If the result is equal to or greater than the modulus, it will also be subtracted once.

4.6. ModSub Operation. The *ModSub* operation code instructs the core to perform a modular subtraction of the supplied elements in prime fields. After preparing the core for the subtraction operation the CLA adder will be used to perform a word-based subtraction by performing an addition in two's complement representation with prime field arithmetic. After the last words of the given operands have been processed, the carry output bit of the CLA adder will be evaluated. If the carry bit signals a negative result, the modulus will be added once; otherwise the result will be compared to the given modulus. If the result is equal to or greater than the modulus, it will be subtracted once.

4.7. RAM Copy Operations. In order to support cryptographic algorithms which have been disassembled into a list

of instructions, RAM copy operations are needed. According to the proposed RAM layout stated above four individual copy operations have been defined.

The *CopyH2V* operation code instructs the core to copy a number of words, according to the supplied precision parameter, from the horizontal RAM layout starting from the given source address to the vertical RAM layout starting from the given destination address.

The *CopyV2V* operation code instructs the core to copy a number of words, according to the supplied precision parameter, from the vertical RAM layout starting from the given source address to the vertical RAM layout starting from the given destination address.

The *CopyH2H* operation code instructs the core to copy a number of words, according to the supplied precision parameter, from the horizontal RAM layout starting from the given source address to the horizontal RAM layout starting from the given destination address.

The *CopyV2H* operation code instructs the core to copy a number of words, according to the supplied precision parameter, from the vertical RAM layout starting from the given source address to the horizontal RAM layout starting from the given destination address.

4.8. MontMult1 Operation. The *MontMult1* operation code instructs the core to perform a single Montgomery Multiplication of the supplied element with the constant 1 in the given finite field. This type of operation is needed when a montgomerized value should be transformed back from the Montgomery Domain and has been implemented as an independent operation since an operand $A = 1$ will unnecessarily occupy a vertical RAM slot. A Montgomery Multiplication with the constant 1 will be executed in an analogous manner as the *MontMult* operation with the only exception that, instead of the RAM words, constant words will be used for the A operand.

5. Exemplary Core Application Descriptions

This section gives exemplary descriptions of how the specified functional range of the proposed building-block Enhanced Montgomery Multiplication Core design can be utilized to support a wide range of cryptographic algorithms demanding the least possible memory capacity yet at the same time supporting as much precision widths as possible. Information is given of how to perform Chinese Remainder Theorem [22] (CRT) accelerated RSA private key operations and how to use the core in order to test/generate prime numbers.

For the support of elliptic curve cryptography over prime and binary finite fields modular functions are given for preparing and conducting point operations for arbitrary elliptic curves for the supported precision widths. For all these algorithms a list of operations and the quantity of different operations is given allowing to perform cryptographic algorithms by simply processing these operation lists.

5.1. CRT-Accelerated RSA Operation. In order to speed up RSA private key operations the CRT-accelerated version is also supported by the core. Therefore some operations have to be performed with full precision whereas most of the operations have to be performed with half precision. Algorithm 1 lists the necessary steps to utilize the core for CRT-accelerated RSA private key operations.

Requires: $(c, d, p, q, exp1, exp2, coeff, n)$
Calculates: (m)

(h) $rq^2 = MontR2(q)$;
(f) $cq = MontMult(rq^2, c, q)$;
(h) $cq_{MD} = MontMult(rq^2, cq, q)$;
(h) $mq_{MD} = MontExp(cq_{MD}, cq_{MD}, exp2, cq_{MD}, q)$;
(h) $mq = MontMult1(mq_{MD}, q)$;
(h) $rp^2 = MontR2(p)$;
(f) $cp = MontMult(rp^2, c, p)$;
(h) $cp_{MD} = MontMult(rp^2, cp, p)$;
(h) $mp_{MD} = MontExp(cp_{MD}, cp_{MD}, exp1, cp_{MD}, p)$;
(h) $mp = MontMult1(mp_{MD}, p)$;
(h) $coeff_{MD} = MontMult(rp^2, cp, p)$;
(h) $t1 = ModSub(mp, mq, p)$;
(h) $t2 = MontMult(coeff_{MD}, t1, p)$;
(f) $r^2 = MontR2(n)$;
(f) $q_{MD} = MontMult(q, r^2, n)$;
(f) $t3 = MontMult(t2, q_{MD}, n)$;
(f) $m = ModAdd(t3, mq, n)$;
Provides: (m)

Algorithm 1: CRT-RSA private key operation.

Table 1 illustrates the abstract operations lists of the core for CRT-accelerated RSA application using the private key portion for all supported precision widths (512, 768, 1024, 1536, 2048, 3072, 4096). The number given in the index of the RAM locations denotes the offset given by the corresponding src_addr, dest_addr, src_addr_e, src_addr_x input signals. The width of the processed values depends on the supplied mwmac_precision input signal which depends on the operation. In the table operations requiring full precision (the precision of the RSA modulus) are marked by (f), operations requiring half precision are marked by (h). The mwmac_f_sel signal must be set to $GF(p)$ arithmetic.

CRT-accelerated RSA private key operations require $2 \times MontR2$, $4 \times MontMult$, $2 \times MontMult1$, $2 \times MontExp$, $1 \times ModSub$, $9 \times CopyH2V$, $4 \times CopyV2H$, $2 \times CopyV2V$ and $1 \times CopyH2H$ performed on half precision and $1 \times MontR2$, $4 \times MontMult$, $1 \times ModAdd$, $1 \times CopyH2V$ and $1 \times CopyH2H$ performed on full precision.

5.2. Prime Generation/Testing Operation. Algorithm 2 lists the necessary steps to utilize the core, in conjunction with a TRNG generator as Miller-Rabin Primality Tester. In the algorithm n denotes the random integer to be tested for primality and k denotes the confidence parameter determining the accuracy of the test, i.e., the amount of Miller-Rabin loops. In a precomputation step the parameters s and d with $2^s \cdot d = (n - 1)$ must be calculated which can be done by simple shift operations and counter increments in software.

Table 1: Core operations list CRT-RSA.

Step Nr.	Precision	Operation
1	-	Clear RAM
2	-	Write $q \mapsto P_1$, $exp1 \mapsto E_1$, $exp2 \mapsto E_5$, $coeff \mapsto X_1$, $p \mapsto X_5$
3	(h)	$MontR2(P_1, A_1)$
4	(h)	$CopyH2V(B_1, A_1)$
5	-	Write $c \mapsto B_1$
6	(f)	$MontMult(A_1, B_1, P_1)$
7	(h)	$MontMult(A_1, B_1, P_1)$
8-9	(h)	$CopyH2V(B_1, A_1)$, $CopyV2V(A_1, A_5)$
10	(h)	$MontExp(A_5, B_1, E_5, A_1, P_1)$
11	(h)	$MontMult1(B_1, P_1)$
12 - 14	(h)	$CopyH2V(P_1, E_5)$, $CopyV2H(X_5, P_1)$, $CopyH2V(B_1, X_5)$
15	(h)	$MontR2(P_1, A_1)$
16	(f)	$CopyH2V(B_1, A_1)$
17	(h)	$CopyV2H(X_1, B_1)$
18	(h)	$MontMult(A_1, B_1, P_1)$
19	(h)	$CopyH2V(B_1, X_1)$
20	-	Write $c \mapsto B_1$
21	(f)	$MontMult(A_1, B_1, P_1)$
22	(h)	$MontMult(A_1, B_1, P_1)$
23 - 24	(h)	$CopyH2V(B_1, A_1)$, $CopyV2V(A_1, A_5)$
25	(h)	$MontExp(A_5, B_1, E_1, A_1, P_1)$
26	(h)	$MontMult1(B_1, P_1)$
27 - 28	(h)	$CopyH2H(B_1, TS_1)$, $CopyV2H(X_5, TC_1)$
29	(h)	$ModSub(TS_1, TC_1, P_1)$
30	(h)	$MontMult(X_1, B_1, P_1)$
31 - 32	(h)	$CopyH2V(B_1, A_1)$, $CopyH2V(TS_1, X_1)$
33	-	Write $n \mapsto P_1$
34	(f)	$MontR2(P_1, A_5)$
35	(h)	$CopyV2V(X_1, A_5)$
36	(f)	$MontMult(E_5, B_1, P_1)$
37	(f)	$MontMult(A_1, B_1, P_1)$
38	(f)	$CopyH2H(B_1, TS_1)$
39	(h)	$CopyV2H(X_5, TC_1)$
40	(f)	$ModAdd(TS_1, TC_1, P_1)$

The test furthermore requires an amount of random integers $\{a_1, \ldots, a_k\}$ serving as random bases.

Table 2 illustrates the operations list of utilizing the core for Miller-Rabin Primality Test steps for all supported precision widths (192, 224, 256, 320, 384, 512, 768, 1024, 1536, 2048, 3072, 4096). The number given in the index of the

Precomputation: $(s, d \text{ with } 2^s \cdot d = (n-1))$
Input: $(n, s, d, k\ \{a_1, \ldots, a_k\})$
Output: (*eval*, composite or probably prime)

$r = MontR(n)$;
$(n - r) = ModSub(n, r, n)$;
for i ***from*** 1 ***to*** k **do**
  $t1_{MD} = MontExp(a_i, a_i, d, a_i, n)$;
  $t2_{MD} = ModSub(t1_{MD}, r, n)$;
  $t3_{MD} = ModSub(t1_{MD}, (n - r), n)$;
  if $t2_{MD} = 0$ ***or*** $t3_{MD} = 0$ **then**
    continue;
  for j ***from*** 0 ***to*** $(s - 1)$ **do**
    $t1_{MD} = MontMult(t1_{MD}, t1_{MD}, n)$;
    $t2_{MD} = ModSub(t1_{MD}, r, n)$;
    if $t2_{MD} = 0$ **then**
      return (*eval* = composite);
    $t3_{MD} = ModSub(t1_{MD}, (n - r), n)$;
    if $t3_{MD} = 0$ **then**
      continue;
  return (*eval* = composite);
return (*eval* = probably prime)

Algorithm 2: Modified Miller-Rabin Primality Test.

RAM locations denotes the offset given by the corresponding src_addr, dest_addr, src_addr_e, src_addr_x input signals. The width of the processed values depends on the supplied mwmac_precision input signal. The mwmac_f_sel signal must be set to $GF(p)$ arithmetic. Note that since the results of the performed operations will be in the Montgomery Domain, they will be checked against the Montgomery Parameter r and $(n - r)$ instead of 1 and $(n - 1)$. Also note that the random bases a_i that will be checked must not necessarily be transformed into the Montgomery Domain first, they simply will be interpreted as random montgomerized values.

The total number of needed core operations depends on the security parameter k and the value s resulting from the factorization of $(n-1)$. Within the outer for loop from writing a new a_i to the RAM until the evaluation of $t2_{MD}$ $1 \times MontR$, $1 \times MontExp$, $1 \times ModSub$, $1 \times CopyH2V$, $2 \times CopyV2H$, $1 \times CopyV2V$ and $1 \times CopyH2H$ and until evaluation of $t3_{MD}$ $2 \times ModSub$, $2 \times CopyV2H$ and $2 \times CopyH2H$ operations are required. Within the inner for loop until evaluation of updated $t2_{MD}$ $1 \times MontMult$, $1 \times ModSub$, $1 \times CopyH2V$, $2 \times CopyV2H$ and $1 \times CopyH2H$ and until evaluation of updated $t3_{MD}$ $2 \times ModSub$, $2 \times CopyV2H$ and $2 \times CopyH2H$ operations is required.

5.3. Elliptic Curve Operations. Unlike modular exponentiation which only is based on modular multiplications, elliptic curve Point Addition and Point Doubling operations also in the Jacobian projective coordinate representation [23] involve modular additions, subtractions, and multiplications. The algorithms for prime field elliptic curve Point Addition and Point Doubling using Jacobian coordinates furthermore involve multiplications by some constants. Since the described core performs multiplication operations by

Table 2: Core operations list for Miller-Rabin Primality Test.

Step Nr.	Operation
1	Clear RAM
2	Write $n \mapsto P_1$, $d \mapsto E_1$
3	Write $a \mapsto X_1$
4-5	$CopyV2V(X_1, A_1)$, $CopyV2H(X_1, B_1)$
6	$MontExp(X_1, B_1, E_1, A_1, P_1)$
7	$CopyH2V(B_1, X_1)$
8	$MontR(P_1, B_1)$
9-11	$CopyH2V(B_1, A_1)$, $CopyV2H(X_1, TS_1)$, $CopyH2H(B_1, TC_1)$
12	$ModSub(TS_1, TC_1, P_1)$
13	Read B_1, if $t2_{MD} = 0$ continue at Step Nr. 3 else continue at Step Nr. 14
14-15	$CopyH2H(P_1, TS_1)$, $CopyV2H(A_1, TC_1)$
16	$ModSub(TS_1, TC_1, P_1)$
17-18	$CopyV2H(X_1, TS_1)$, $CopyH2H(B_1, TC_1)$
19	$ModSub(TS_1, TC_1, P_1)$
20	Read B_1, if $t3_{MD} = 0$ continue at Step Nr. 3 else if $(s-1) = 0$ stop test with *eval* = composite else continue at Step Nr. 21
21	$CopyV2H(X_1, B_1)$
22	$MontMult(X_1, B_1, P_1)$
23-25	$CopyH2V(B_1, X_1)$, $CopyH2H(B_1, TS_1)$, $CopyV2H(A_1, TC_1)$
26	$ModSub(TS_1, TC_1, P_1)$
27	Read B_1, if $t2_{MD} = 0$ stop test with *eval* = composite else continue at Step Nr. 28
28-29	$CopyH2H(P_1, TS_1)$, $CopyV2H(A_1, TC_1)$
30	$ModSub(TS_1, TC_1, P_1)$
31-32	$CopyV2H(X_1, TS_1)$, $CopyH2H(B_1, TC_1)$
33	$ModSub(TS_1, TC_1, P_1)$
34	Read B_1, if $t3_{MD} = 0$ and $i = k$ stop test with *eval* = probably prime else if $t3_{MD} = 0$ and $i \neq k$ continue at Step Nr. 3 else if $j = (s-1)$ stop test with *eval* = composite else continue at Step Nr. 21

using Montgomery Arithmetic, these constants must be transformed into the Montgomery Domain first for the intermediate values to remain montgomerized.

In order to utilize the core for elliptic curve operations the following modular functions have been specified for both $GF(p)$ and $GF(2^m)$ support:

(i) *EC Preparation*.

(ii) *EC Montgomery Transformation*.

(iii) *EC Affine-to-Jacobi Transformation*.

Requires: $(2, 3, 4, 8, a, b, p)$
Calculates: $(r^2, 2_{MD}, 3_{MD}, 4_{MD}, 8_{MD}, a_{MD}, b_{MD}, exp)$

$r^2 = MontR2(p)$;
$2_{MD} = MontMult(r^2, 2, p)$;
$3_{MD} = MontMult(r^2, 3, p)$;
$4_{MD} = MontMult(r^2, 4, p)$;
$8_{MD} = MontMult(r^2, 8, p)$;
$a_{MD} = MontMult(r^2, a, p)$;
$b_{MD} = MontMult(r^2, b, p)$;
$exp = ModSub(p, 2, p)$;
Provides: $(r^2, 2_{MD}, 3_{MD}, 4_{MD}, 8_{MD}, a_{MD}, b_{MD}, exp)$

Algorithm 3: Core $GF(p)$ EC Preparation.

Requires: (x_P, y_P, r^2)
Calculates: (x_{PMD}, y_{PMD})

$x_{PMD} = MontMult(x_P, r^2, p)$;
$y_{PMD} = MontMult(y_P, r^2, p)$;
Provides: (x_{PMD}, y_{PMD})

Algorithm 4: Core $GF(p)$ EC Montgomery Transformation.

(iv) *EC Point Validation*.

(v) *EC Point Doubling*.

(vi) *EC Point Addition*.

(vii) *EC Jacobi-to-Affine Transformation*.

(viii) *EC Montgomery Backtransformation*.

In the following, algorithms for utilizing the core to perform EC operations in $GF(p)$ are stated. For $GF(2^m)$ EC support, similar algorithms have been derived.

5.3.1. GF(p) EC Preparation. The prime field EC Preparation steps include the calculation of the Montgomery Parameter $r^2 \bmod p$, the exponent $\exp = p - 2$ as well as the montgomerized versions of the constants $2, 3, 4, 8$ and the EC Domain Parameters a and b for a given elliptic curve $E : y^2 \equiv x^3 + a \cdot x + b \bmod p$ over $GF(p)$. Algorithm 3 lists the necessary steps to utilize the core for EC prime field preparation.

A core prime field EC preparation operation requires $1 \times MontR2$, $6 \times MontMult$, $1 \times ModSub$, $8 \times CopyH2V$, and $8 \times CopyH2H$.

5.3.2. GF(p) EC Montgomery Transformation. The prime field EC Montgomery Transformation steps are responsible for the transformation of the supplied affine point coordinates x_P and y_P of a Point P in the case of a Point Doubling or Point Multiplication operation, x_P, y_P, x_Q and y_Q of the curve Points P and Q in the case of a Point Addition operation into the Montgomery Domain. Algorithm 4 lists

Requires: (x_{PMD}, y_{PMD})
Calculates: $(x_{PMDj}, y_{PMDj}, z_{PMDj})$

$x_{PMDj} \coloneqq x_{PMD}$;
$y_{PMDj} \coloneqq y_{PMD}$;
$z_{PMDj} = MontR(p)$;
Provides: $(x_{PMDj}, y_{PMDj}, z_{PMDj})$

Algorithm 5: Core $GF(p)$ EC Affine-to-Jacobi Transformation.

Requires: $(x_{PMD}, y_{PMD}, p, a_{MD}, b_{MD})$
Output: (*eval*, point on curve or point not on curve)

$y^2_{PMD} = MontMult(y_{PMD}, y_{PMD}, p)$;
$x^2_{PMD} = MontMult(x_{PMD}, x_{PMD}, p)$;
$ax_{PMD} = MontMult(a_{MD}, x_{PMD}, p)$;
$x^3_{PMD} = MontMult(x^2_{PMD}, x_{PMD}, p)$;
$t1_{MD} = ModAdd(x^3_{PMD}, ax_{PMD}, p)$;
$t2_{MD} = ModAdd(t1_{MD}, b_{MD}, p)$;
$t3_{MD} = ModSub(y^2_{PMD}, t2_{MD}, p)$;
if $t3_{MD} = 0$ **then**
    return: (*eval* = point on curve);
else
    return: (*eval* = point not on curve);

Algorithm 6: Core $GF(p)$ EC Point Validation.

the steps to utilize the core for prime field EC Montgomery Transformation for an arbitrary curve Point P.

A core prime field EC Montgomery Transformation operation requires $2 \times MontMult$, $2 \times CopyH2V$, and $2 \times CopyV2V$ in the case of an intended Point Doubling or Point Multiplication operation and $4 \times MontMult$ and $4 \times CopyH2V$ in the case of an intended Point Addition operation.

5.3.3. GF(p) EC Affine-to-Jacobi Transformation. The prime field EC Affine-to-Jacobi Transformation steps are responsible for transforming the supplied montgomerized affine point coordinates x_{PMD} and y_{PMD} of a curve Point P into Jacobian coordinates. Algorithm 5 lists the necessary steps to utilize the core for prime field EC Affine-to-Jacobi Transformation for an arbitrary montgomerized curve Point P.

A core prime field EC Affine-to-Jacobi Transformation operation requires $1 \times MontR$, $1 \times CopyH2V$, and $1 \times CopyV2V$ in the case of an intended Point Addition, Point Doubling, or Point Multiplication operation.

5.3.4. GF(p) EC Point Validation. The prime field EC Point Validation performs a check, if a supplied (or calculated) point indeed is a valid point of the elliptic curve given by the equation $y^2 \equiv x^3 + a \cdot x + b \bmod p$. As a requirement the Point Validation must be conducted on montgomerized points in affine coordinate representation. Algorithm 6 lists the necessary steps to utilize the core for prime field EC Point Validation.

A core prime field EC Point Validation operation requires $4 \times MontMult$, $2 \times ModAdd$, $1 \times ModSub$, $4 \times CopyV2H$, and $5 \times CopyH2H$.

5.3.5. GF(p) EC Point Doubling. The prime field EC Point Doubling steps perform a single Point Doubling operation of a Point P with montgomerized Jacobi coordinates, resulting in $2 \cdot P = R$ also represented in montgomerized Jacobi coordinates. The original algorithm for Point Doubling with Jacobi coordinate representation has been modified to be suitable for the proposed core and is given in Algorithm 7.

A core prime field EC Point Doubling operation requires $15 \times MontMult$, $1 \times ModAdd$, $3 \times ModSub$, $6 \times CopyH2V$, $6 \times CopyV2H$, and $7 \times CopyH2H$.

5.3.6. GF(p) EC Point Addition. The prime field EC Point Addition steps perform a single Point Addition operation of two Points P and Q with montgomerized Jacobi coordinates, resulting in $P + Q = R$ also represented in montgomerized Jacobi coordinates. The original algorithm for Point Addition with Jacobi coordinate representation has been modified to be suitable for the proposed core and is given in Algorithm 8.

A core prime field EC Point Addition operation requires $17 \times MontMult$, $6 \times ModSub$, $9 \times CopyH2V$, $7 \times CopyV2H$, and $12 \times CopyH2H$.

5.3.7. GF(p) EC Jacobi-to-Affine Transformation. The prime field EC Jacobi-to-Affine Transformation steps are responsible for the transformation of the supplied montgomerized Jacobi coordinates x_{RMDj}, y_{RMDj}, and z_{RMDj} of the curve Point R back into affine coordinate representation. This transformation step requires the calculation of a modular multiplicative inverse element which will be performed by a Montgomery modular exponentiation according to Euler's theorem since the modulus is a prime number. Algorithm 9 lists the necessary steps to utilize the core for prime field EC Jacobi-to-Affine Transformation.

A core prime field EC Jacobi-to-Affine Transformation operation requires $4 \times MontMult$, $1 \times MontExp$, $3 \times CopyH2V$, $1 \times CopyV2H$, $1 \times CopyV2V$ and $1 \times CopyH2H$.

5.3.8. GF(p) EC Montgomery Backtransformation. The prime field EC Montgomery Backtransformation steps are responsible for the transformation of the supplied montgomerized point coordinates x_{RMD} and y_{RMD} of a Point R out of the Montgomery Domain. Algorithm 10 lists the necessary steps to utilize the core for prime field EC Montgomery Backtransformation for an arbitrary curve Point R.

A core prime field EC Montgomery Backtransformation operation requires $2 \times MontMult1$ and $2 \times CopyV2H$.

6. Performance Analysis

In this section parameter-dependent formulas for the calculation of the computation times in clock cycles of the described basic core operations are given which allows specifying upper and lower calculation boundaries. Furthermore for the supported precision widths in both finite fields the number of words to be processed and the possible numbers

Requires: $(x_{PMDj}, y_{PMDj}, z_{PMDj}, p, 2_{MD}, 3_{MD}, 4_{MD}, 8_{MD}, a_{MD})$
Calculates: $(x_{RMDj}, y_{RMDj}, z_{RMDj})$

$t1_{MD} = MontMult(4_{MD}, x_{PMDj}, p);$
$y^2_{PMDj} = MontMult(y_{PMDj}, y_{PMDj}, p);$
$S_{MD} = MontMult(t1_{MD}, y^2_{PMDj}, p);$
$x^2_{PMDj} = MontMult(x_{PMDj}, x_{PMDj}, p);$
$t2_{MD} = MontMult(3_{MD}, x^2_{PMDj}, p);$
$z^2_{PMDj} = MontMult(z_{PMDj}, z_{PMDj}, p);$
$z^4_{PMDj} = MontMult(z^2_{PMDj}, z^2_{PMDj}, p);$
$t3_{MD} = MontMult(a_{MD}, z^4_{PMDj}, p);$
$M_{MD} = ModAdd(t2_{MD}, t3_{MD}, p);$
$M^2_{MD} = MontMult(M_{MD}, M_{MD}, p);$
$t4_{MD} = MontMult(2_{MD}, S_{MD}, p);$
$x_{RMDj} = ModSub(M^2_{MD}, t4_{MD}, p);$
$t5_{MD} = ModSub(S_{MD}, x_{RMDj}, p);$
$t6_{MD} = MontMult(M_{MD}, t5_{MD}, p);$
$y^4_{PMDj} = MontMult(y^2_{PMDj}, y^2_{PMDj}, p);$
$t7_{MD} = MontMult(8_{MD}, y^4_{PMDj}, p);$
$y_{RMDj} = ModSub(t6_{MD}, t7_{MD}, p);$
$t8_{MD} = MontMult(y_{PMDj}, z_{PMDj}, p);$
$z_{RMDj} = MontMult(2_{MD}, t8_{MD}, p);$
Provides: $(x_{RMDj}, y_{RMDj}, z_{RMDj})$

Algorithm 7: Core $GF(p)$ EC Point Doubling.

Table 3: Core RAM copy operations computation time in clock cycles (CC).

	GF(p)	**GF(2^m)**
$CC_{CopyH2V}$	$\lceil(\lvert p\rvert/\lvert w\rvert)\rceil + 3$	$\lceil(m/\lvert w\rvert)\rceil + 3$
$CC_{CopyV2V}$	$\lceil(\lvert p\rvert/\lvert r\rvert)\rceil + 2$	$\lceil(m/\lvert r\rvert)\rceil + 2$
$CC_{CopyH2H}$	$\lceil(\lvert p\rvert/\lvert w\rvert)\rceil + 3$	$\lceil(m/\lvert w\rvert)\rceil + 3$
$CC_{CopyV2H}$	$\lceil(\lvert p\rvert/\lvert w\rvert)\rceil + 3$	$\lceil(m/\lvert w\rvert)\rceil + 3$

of processing units is given. In order to estimate the size ratio of different core variations the number of logic elements and dedicated logic registers for exemplary Altera and Xilinx FPGAs is stated. Furthermore results of power estimation are given. Depending on the resulting clock cycle times of core variations a reference implementation exhibiting a balance of performance and area consumption has been defined. For this reference implementation the computation times in clock cycles for the described exemplary cryptographic algorithms are given.

6.1. Core Computation Time Formulas. Table 3 lists the RAM copy operations computation time formulas in clock cycles of the proposed core. Note that the resulting calculation times of RAM reorganisation operations are only dependent on the specified precision ($|p|$ for $GF(p)$ and m for $GF(2^m)$), the word width $|w|$ parameter for which the core variation has been generated and the resulting RAM width parameter $|r|$ with $|r| = 4 \cdot |w|$. The operations $CopyH2V$, $CopyH2H$, and $CopyV2H$ exhibit the same computation time, whereas the operation $CopyV2V$ will be performed in less clock cycles.

Table 4: Core $GF(p)$ operations computation time in clock cycles (CC).

	GF(p)
$CC_{MontMult\_b_{GF(p)}}$	$\frac{\lvert p\rvert - k}{k} \cdot e + k + e + 4$
$CC_{MontMult\_w_{GF(p)}}$	$\frac{\lvert p\rvert - k}{k} \cdot e + k + 3 \cdot e + 4$
$CC_{MontR_{GF(p)}}$	$\lceil(\lvert p\rvert/\lvert w\rvert)\rceil + 3$
$CC_{MontR2\_b_{GF(p)}}$	$2 \cdot (\lceil(\lvert p\rvert/\lvert w\rvert)\rceil + 2) + 1 + x \cdot (CC_{CopyH2V} - 1) + y \cdot (CC_{MontMult\_b_{GF(p)}} - 1) + 1$
$CC_{MontR2\_w_{GF(p)}}$	$2 \cdot (\lceil(\lvert p\rvert/\lvert w\rvert)\rceil + 2) + 2 \cdot \lceil(\lvert p\rvert/\lvert w\rvert)\rceil + 3 + x \cdot (CC_{CopyH2V} - 1) + y \cdot (CC_{MontMult\_w_{GF(p)}} - 1) + 1$
$CC_{MontExp\_b_{GF(p)}}$	$[\lceil(\lvert p\rvert/\lvert w\rvert)\rceil \cdot (\lvert w\rvert + 2)] + (CC_{CopyV2V} - 1) + [2 \cdot (CC_{MontMult\_b_{GF(p)}} - 1)]$
$CC_{MontExp\_w_{GF(p)}}$	$[\lceil(\lvert p\rvert/\lvert w\rvert)\rceil] + [\lceil(\lvert p\rvert/\lvert w\rvert)\rceil \cdot \lvert w\rvert - \lvert p\rvert + 3] + [2 \cdot (\lvert p\rvert - 2) \cdot (CC_{MontMult\_w_{GF(p)}} - 1)] + [(\lvert p\rvert - 2) \cdot (CC_{CopyV2V} - 1)] + [(\lvert p\rvert - 3) \cdot (CC_{CopyH2V} - 1)] + 1$
$CC_{ModAdd\_b_{GF(p)}}$	$\lceil(\lvert p\rvert/\lvert w\rvert)\rceil + 4$
$CC_{ModAdd\_w_{GF(p)}}$	$3 \cdot \lceil(\lvert p\rvert/\lvert w\rvert)\rceil + 6$
$CC_{ModSub\_b_{GF(p)}}$	$\lceil(\lvert p\rvert/\lvert w\rvert)\rceil + 4$
$CC_{ModSub\_w_{GF(p)}}$	$2 \cdot \lceil(\lvert p\rvert/\lvert w\rvert)\rceil + 4$
$CC_{ModSub\_aw_{GF(p)}}$	$3 \cdot \lceil(\lvert p\rvert/\lvert w\rvert)\rceil + 6$

The computing time formulas of prime field core operations given in clock cycles are listed in Table 4.

Requires: $(x_{PMDj}, y_{PMDj}, z_{PMDj}, x_{QMDj}, y_{QMDj}, z_{QMDj}, p, 2_{MD})$
Calculates: $(x_{RMDj}, y_{RMDj}, z_{RMDj})$

$z^2_{QMDj} = MontMult(z_{QMDj}, z_{QMDj}, p);$
$U1_{MD} = MontMult(x_{PMDj}, z^2_{QMDj}, p);$
$z^2_{PMDj} = MontMult(z_{PMDj}, z_{PMDj}, p);$
$U2_{MD} = MontMult(x_{QMDj}, z^2_{PMDj}, p);$
$z^3_{QMDj} = MontMult(z^2_{QMDj}, z_{QMDj}, p);$
$S1_{MD} = MontMult(y_{PMDj}, z^3_{QMDj}, p);$
$z^3_{PMDj} = MontMult(z^2_{PMDj}, z_{PMDj}, p);$
$S2_{MD} = MontMult(y_{QMDj}, z^3_{PMDj}, p);$
$H_{MD} = ModSub(U2_{MD}, U1_{MD}, p);$
$R_{MD} = ModSub(S2_{MD}, S1_{MD}, p);$
if $H_{MD} = 0$ **then**
  if $R_{MD} \neq 0$ **then**
    Notify: $(x_{RMDj}, y_{RMDj}, z_{RMDj}) = (0, 1, 0)$
      Point_at_Infinity ($\mathcal{O}$);
  else
    Perform: PointDoubling Operation;
else
  $R^2_{MD} = MontMult(R_{MD}, R_{MD}, p);$
  $H^2_{MD} = MontMult(H_{MD}, H_{MD}, p);$
  $H^3_{MD} = MontMult(H^2_{MD}, H_{MD}, p);$
  $t1_{MD} = MontMult(U1_{MD}, H^2_{MD}, p);$
  $t2_{MD} = MontMult(2_{MD}, t1_{MD}, p);$
  $t3_{MD} = ModSub(R^2_{MD}, H^3_{MD}, p);$
  $x_{RMDj} = ModSub(t3_{MD}, t2_{MD}, p);$
  $t4_{MD} = MontMult(S1_{MD}, H^3_{MD}, p);$
  $t5_{MD} = ModSub(t1_{MD}, x_{RMDj}, p);$
  $t6_{MD} = MontMult(R_{MD}, t5_{MD}, p);$
  $y_{RMDj} = ModSub(t6_{MD}, t4_{MD}, p);$
  $t7_{MD} = MontMult(z_{PMDj}, H_{MD}, p);$
  $z_{RMDj} = MontMult(z_{QMDj}, t7_{MD}, p);$
  Provides: $(x_{RMDj}, y_{RMDj}, z_{RMDj})$

Algorithm 8: Core $GF(p)$ EC Point Addition.

Requires: $(x_{RMDj}, y_{RMDj}, z_{RMDj}, p, exp = p - 2)$
Calculates: (x_{RMD}, y_{RMD})

$t1_{MD} = MontExp(z_{RMDj}, z_{RMDj}, exp, z_{RMDj}, p);$
$t2_{MD} = MontMult(t1_{MD}, t1_{MD}, p);$
$t3_{MD} = MontMult(t2_{MD}, t1_{MD}, p);$
$x_{RMD} = MontMult(x_{RMDj}, t2_{MD}, p);$
$y_{RMD} = MontMult(y_{RMDj}, t3_{MD}, p);$
Provides: (x_{RMD}, y_{RMD})

Algorithm 9: Core $GF(p)$ EC Jacobi-to-Affine Transformation.

The computation time of the *MontMult* operation in $GF(p)$ depends on the specified precision $|p|$, the number of active processing units k as well as the number of words $e = \lceil (|p|/|w|) \rceil + 1$ running through the pipeline. In order to specify lower and upper computation times a best case and worst case formula is given. In the best case the carry-out bit of the CLA adder after reuniting TS and TC words is not set, the comparator only has to evaluate the most significant word, and a modular subtraction is not necessary. In the worst case the carry-out bit of the CLA adder is also not set but the comparator has to evaluate all words and a reduction of the resulting value is necessary.

The *MontR* operation in $GF(p)$ only depends on the chosen precision $|p|$ and specified word width $|w|$ parameters.

Requires: (x_{RMD}, y_{RMD})
Calculates: (x_R, y_R)

$x_R = MontMult1(x_{RMD}, p)$;
$y_R = MontMult1(y_{RMD}, p)$;
Provides: (x_R, y_R)

Algorithm 10: Core $GF(p)$ EC Montgomery Backtransformation.

Table 5: Precision-dependent values of x and y for $GF(p)$ $MontR2$ operation.

	192	224	256	320	384	448	512
x	7	6	8	7	8	7	9
y	8	11	8	10	9	12	9
	768	**1024**	**1536**	**2048**	**3072**	**4096**	
x	9	10	10	11	11	12	
y	10	10	11	11	12	12	

For the $MontR2$ operation computation time a best case and worst case formula is given. In the best case, after the shift operation, the comparator will only evaluate one word and an initial modular subtraction operation is not necessary. For the involved Montgomery Multiplication operations the best case formula is used. In the worst case, after the shift operation the comparator has to evaluate all words and decide that an initial modular subtraction operation is needed. For the involved Montgomery Multiplication operations the worst case formula is used. The amount x of $CopyH2V$ and y of $MontMult$ operations depends on the chosen precision. Table 5 lists the values for all supported $GF(p)$ precisions.

For the $MontExp$ operation, computation time in a best case and worst case formula is given. In the best case the exponent operand is 3; therefore only two Montgomery Multiplications and one $CopyV2V$ operation is necessary. For the involved Montgomery Multiplication operations the best case formula is used. In the worst case the exponent is assumed to be $2^{(|p|-1)}$; therefore $2\cdot(|p|-2)\times MontMult$, $(|p|-2)\times CopyV2V$ and $(|p|-3)\times CopyH2V$ operations have to be performed. For the involved Montgomery Multiplication operations the worst case formula is used.

For the $ModAdd$ operation computation time a best case and worst case formula is given. In the best case, after the modular addition the CLA adder carry-out bit will not be set, the comparator will only have to evaluate one word and an additional modular subtraction is not needed. In the worst case the CLA adder carry-out bit will also not be set, but the comparator will have to evaluate all words to decide that an additional modular subtraction is necessary.

For the $ModSub$ operation computation time a best case, worst case, and absolute worst case formula is given. In the best case, after the modular subtraction the CLA adder carry-out bit will not be set, the comparator will only have to evaluate one word and an additional modular subtraction is not needed. In the worst case, after the modular subtraction the CLA adder carry-out bit will be set and a modular addition must be performed. In the absolute worst case after the modular subtraction the CLA adder carry-out bit will not be set, the comparator will evaluate all words, and an additional modular subtraction step is necessary. Note that this will only occur if the resulting value after the first subtraction operation will be identical to the modulus, which under normal operation conditions will not be the case.

Table 6: Core $GF(2^m)$ operations computation time in clock cycles (CC).

Operation	$GF(2^m)$
$CC_{MontMult_{GF(2^m)}}$	$\frac{m-(m \bmod k)}{k}\cdot e + (m \bmod k) + e + 4$
$CC_{MontR_{GF(2^m)}}$	$\lceil (m/\lvert w\rvert)\rceil + 3$
$CC_{MontR2\_b_{GF(2^m)}}$	$(m+1)\cdot(\lceil (m/\lvert w\rvert)\rceil+2)+m+1$
$CC_{MontR2\_w_{GF(2^m)}}$	$(m+1)\cdot(\lceil (m/\lvert w\rvert)\rceil+2)+m\cdot\lceil (m/\lvert w\rvert)\rceil+m+1$
$CC_{MontExp\_b_{GF(2^m)}}$	$[\lceil (m/\lvert w\rvert)\rceil\cdot(\lvert w\rvert+2)]+(CC_{CopyV2V}-1)+[2\cdot(CC_{MontMult_{GF(2^m)}})]$
$CC_{MontExp\_w_{GF(2^m)}}$	$[\lceil (m/\lvert w\rvert)\rceil]+[\lceil (m/\lvert w\rvert)\rceil\cdot\lvert w\rvert-m+3]+[2\cdot(m-2)\cdot(CC_{MontMult_{GF(2^m)}}-1)]+[(m-2)\cdot(CC_{CopyV2V}-1)]+[(m-3)\cdot(CC_{CopyH2V}-1)]+1$
$CC_{ModAdd\_b_{GF(2^m)}}$	$\lceil (m/\lvert w\rvert)\rceil + 4$
$CC_{ModAdd\_w_{GF(2^m)}}$	$3\cdot\lceil (m/\lvert w\rvert)\rceil + 6$

The prime field $MontMult1$ operation is identical to the $GF(p)$ $MontMult$ operation; therefore the same best and worst case formulas apply.

The computing time formulas of binary field core operations given in clock cycles are listed in Table 6.

The computation time of the $MontMult$ operation in $GF(2^m)$ depends on the specified precision parameter m, the number of active processing units k, and the number of words $e = \lceil (m/|w|)\rceil + 1$ running through the pipeline. Since the additions in $GF(2^m)$ are simple XOR-operations and the most significant bit of the resulting value will never be set after calculation only one formula is given.

While the determination of the Montgomery Parameter r in $GF(2^m)$ differs from the calculation rule for $GF(p)$ it also only depends on the chosen precision m and specified word width $|w|$ parameters.

The $MontR2$ operation in $GF(2^m)$ is based on shifts and possible modular additions whenever the most significant bit of the intermediate value will be set after a shift. The amount of modular additions depends on the Montgomery Parameter r which itself depends on the irreducible polynomial. In order to specify lower and upper computation times a best case and worst case formula is given. The best case assumes that no modular addition operation is required at all, whereas the worst case assumes that a modular addition operation is required after each shift operation.

For the $MontExp$ operation computation time a best case and worst case formula is given. In the best case the exponent

operand is 3 therefore only two Montgomery Multiplications and one $CopyV2V$ operation is necessary. In the worst case the exponent is assumed to be $2^{(m-1)}$ therefore $2 \cdot (m-2) \times MontMult$, $(m-2) \times CopyV2V$ and $(m-3) \times CopyH2V$ operations are required.

For the *ModAdd* operation computation time a best case and absolute worst case formula is given. In the best case the comparator will only have to evaluate one word. In the absolute worst case the comparator will have to evaluate all words to decide that an additional modular addition is necessary. Note that this will only occur if the resulting value after the first addition operation will be identical to the modulus polynomial, which under normal operation conditions will not be the case.

The binary field $MontMult1$ operation is identical to the $GF(2^m)$ $MontMult$ operation; therefore the same formula applies.

6.2. Core Variations. Depending on the needs, in terms of performance, area consumption, supported precisions, and the interfacing structure, different variations of the core can be generated by defining the parameters *MAX_PRECISION_WIDTH*, *WORD_WIDTH* and *MAX_NUM_PUS*. Table 7 lists the resulting number of words e and the possible number of processing units k for the supported prime field precisions $|p|$ and typical word widths $|w|$ of 16, 32 and 64 bit.

In contrast, Table 8 lists the resulting number of words e and the number of possible processing units k for the supported binary field precisions m and typical word widths $|w|$ of 16, 32, and 64 bits. Note that the number of possible processing units for binary fields within the defined core is subjected to a further constraint. Once all bits of A operand have been processed the remaining processing units in the pipeline must be bypassed and the TS and TC words must be directly fed into the CLA adder. Since the result of the CLA adder will be written back to RAM but remaining words must still be read from RAM and fed into the first processing unit, the RAM source and destination signals must never address the same memory location at one time. Therefore the equation $(k - (m \bmod k)) \bmod k$ must hold true to $(k - (m \bmod k)) \equiv 0 \bmod k$, meaning that no processing unit will be bypassed, or $(k - (m \bmod k)) \equiv 1 \bmod k$, meaning that the very last processing unit will be bypassed at the last cycle of A operand bits.

6.3. Core Hardware Footprint. Since all components of the design consist of simple logic elements, the proposed arithmetic core is vendor-neutral. In order to estimate the hardware footprint of different core implementations the design variations have been compiled on Altera and Xilinx FPGAs. Table 9 lists the amount of total logic elements and comprised logic registers for varied values of *WORD_WIDTH* ($|w|$) and *MAX_NUM_PUS* generated for an Altera Cyclone IV (EP4CE115F29C9L) device featuring 114,480 logic elements and 3,981,312 memory bits.

Table 10 lists the amount of total logic elements and comprised logic registers for varied values of *WORD_WIDTH* ($|w|$) and *MAX_NUM_PUS* generated for an Xilinx XC7Z020 (xc7z020clg484-1) device featuring 53,200 logic elements and 106,400 registers. The resulting values demonstrate that the design can compete with other proposed designs, for instance, the one compared in [14, 15]. Furthermore instead of being restricted to only one cryptographic application, the core can handle various algorithms. According to the needs, in terms of area, a suitable solution for a specific implementation can be chosen. The choice will have an impact on power consumption and computing time.

Table 7: Number of words e and amount of possible processing units k depending on precision $|p|$ and word width $|w|$ for $GF(p)$.

GF(p)	**$\|w\|$ = 16**	**$\|w\|$ = 32**	**$\|w\|$ = 64**
$\|p\|$ = 192	$e = 13$ $k = 2, 4, 8$	$e = 7$ $k = 2, 4$	$e = 4$ $k = 2$
$\|p\|$ = 224	$e = 15$ $k = 2, 4, 8$	$e = 8$ $k = 2, 4$	$e = 5$ $k = 2$
$\|p\|$ = 256	$e = 17$ $k = 2, 4, 8$	$e = 9$ $k = 2, 4$	$e = 5$ $k = 2$
$\|p\|$ = 320	$e = 21$ $k = 2, 4,$ $8, 16$	$e = 11$ $k = 2, 4, 8$	$e = 6$ $k = 2, 4$
$\|p\|$ = 384	$e = 25$ $k = 2, 4,$ $8, 16$	$e = 13$ $k = 2, 4, 8$	$e = 7$ $k = 2, 4$
$\|p\|$ = 448	$e = 29$ $k = 2, 4,$ $8, 16$	$e = 15$ $k = 2, 4, 8$	$e = 8$ $k = 2, 4$
$\|p\|$ = 512	$e = 33$ $k = 2, 4,$ $8, 16$	$e = 17$ $k = 2, 4, 8$	$e = 9$ $k = 2, 4$
$\|p\|$ = 768	$e = 49$ $k = 2, 4, 8,$ $16, 32$	$e = 25$ $k = 2, 4,$ $8, 16$	$e = 13$ $k = 2, 4, 8$
$\|p\|$ = 1024	$e = 65$ $k = 2, 4, 8,$ $16, 32$	$e = 33$ $k = 2, 4, 8, 16$	$e = 17$ $k = 2, 4, 8$
$\|p\|$ = 1536	$e = 97$ $k = 2, 4, 8,$ $16, 32, 64$	$e = 49$ $k = 2, 4, 8,$ $16, 32$	$e = 25$ $k = 2, 4,$ $8, 16$
$\|p\|$ = 2048	$e = 129$ $k = 2, 4, 8,$ $16, 32, 64$	$e = 65$ $k = 2, 4, 8,$ $16, 32$	$e = 33$ $k = 2, 4,$ $8, 16$
$\|p\|$ = 3072	$e = 193$ $k = 2, 4, 8,$ $16, 32, 64$	$e = 97$ $k = 2, 4, 8,$ $16, 32, 64$	$e = 49$ $k = 2, 4, 8,$ $16, 32$
$\|p\|$ = 4096	$e = 257$ $k = 2, 4, 8,$ $16, 32, 64$	$e = 129$ $k = 2, 4, 8,$ $16, 32, 64$	$e = 65$ $k = 2, 4, 8,$ $16, 32$

6.4. Core Power Estimation. In order to evaluate the suitability of the proposed core for the application in the IoT area, a power estimation has been conducted using two common frequencies of 100 MHz and 200 MHz for various core variations. Timing analysis yields that the design can reliably be operated with these frequencies. The power consumption characteristics have been derived by applying the PowerPlay Power Analyzer Tool of the Quartus Prime IDE to the

Table 8: Number of words e and amount of possible processing units k depending on precision m and word width $|w|$ for $GF(2^m)$.

$GF(2^m)$	$\|w\| = 16$	$\|w\| = 32$	$\|w\| = 64$
m = 131	$e = 10$ $k = 2, 4$	$e = 6$ $k = 2, 4$	$e = 4$ $k = 2$
m = 163	$e = 12$ $k = 2, 4$	$e = 7$ $k = 2, 4$	$e = 4$ $k = 2$
m = 176	$e = 12$ $k = 2, 4, 8$	$e = 7$ $k = 2, 4$	$e = 4$ $k = 2$
m = 191	$e = 13$ $k = 2, 4, 8$	$e = 7$ $k = 2, 4$	$e = 4$ $k = 2$
m = 193	$e = 14$ $k = 2$	$e = 8$ $k = 2$	$e = 5$ $k = 2$
m = 208	$e = 14$ $k = 2, 4, 8$	$e = 8$ $k = 2, 4$	$e = 5$ $k = 2$
m = 233	$e = 16$ $k = 2$	$e = 9$ $k = 2$	$e = 5$ $k = 2$
m = 239	$e = 16$ $k = 2, 4, 8$	$e = 9$ $k = 2, 4$	$e = 5$ $k = 2$
m = 272	$e = 18$ $k = 2, 4, 8, 16$	$e = 10$ $k = 2, 4, 8$	$e = 6$ $k = 2, 4$
m = 283	$e = 19$ $k = 2, 4$	$e = 10$ $k = 2, 4$	$e = 6$ $k = 2, 4$
m = 304	$e = 20$ $k = 2, 4, 8, 16$	$e = 11$ $k = 2, 4, 8$	$e = 6$ $k = 2, 4$
m = 359	$e = 24$ $k = 2, 4, 8$	$e = 13$ $k = 2, 4, 8$	$e = 7$ $k = 2, 4$
m = 368	$e = 24$ $k = 2, 4, 8, 16$	$e = 13$ $k = 2, 4, 8$	$e = 7$ $k = 2, 4$
m = 409	$e = 27$ $k = 2$	$e = 14$ $k = 2$	$e = 8$ $k = 2$
m = 431	$e = 28$ $k = 2, 4, 8, 16$	$e = 15$ $k = 2, 4, 8$	$e = 8$ $k = 2, 4$
m = 571	$e = 37$ $k = 2, 4$	$e = 19$ $k = 2, 4$	$e = 10$ $k = 2, 4$

final design using default settings of a power toggle rate as well as a power input I/O toggle rate of 12.5%, using a vectorless estimation and a board temperature of 25°C. Table 11 lists the Total Thermal Power Dissipation values for varied *WORD_WIDTH* ($|w|$) and *MAX_NUM_PUS* parameters generated for the Altera Cyclone IV (EP4CE115F29C9L) device. The values are comparable to the ones given in [24] for RSA calculation.

Furthermore it has to be mentioned that the optimization mode in the compiler settings was set to balanced and no specific compiler optimizations regarding power have been turned on. The results show that the core is quite suitable for applications which have special constraints regarding power consumption. According to such needs as well as the desired clock frequency a suitable variation can be implemented. The choice will have an impact on computing time and hardware footprint.

Table 9: Amount of logic elements and logic registers for different core variations (Altera Cyclone IV).

$\|w\|$	MAX_NUM_PUS	Logic Elements	Registers
16	2	3,128	706
16	4	3,523	904
16	8	4,344	1,300
16	16	5,960	2,092
16	32	9,198	3,676
16	64	15,568	6,844
32	2	4,086	988
32	4	4,721	1,314
32	8	5,935	1,966
32	16	8,473	3,270
32	32	13,498	5,878
32	64	23,484	11,094
64	2	6,114	1,557
64	4	7,151	2,139
64	8	9,346	3,303
64	16	13,624	5,631
64	32	22,113	10,287

Table 10: Amount of logic elements and logic registers for different core variations (Xilinx XC7Z020).

$\|w\|$	MAX_NUM_PUS	Logic Elements	Registers
16	2	2,587	755
16	4	2,874	956
16	8	3,467	1,352
16	16	4,623	2,146
16	32	7,208	3,724
16	64	11,934	6,895
32	2	3,367	1,114
32	4	4,014	1,429
32	8	4,694	2,082
32	16	6,645	3,386
32	32	10,177	5,999
32	64	17,770	11,222
64	2	5,098	1,833
64	4	5,869	2,421
64	8	7,585	3,591
64	16	10,654	5,928
64	32	16,871	10,624

6.5. Core Reference Implementation. For the reference implementation a word width of *WORD_WIDTH* = 32 bit was chosen and the maximum number of processing units of the pipeline was set to *MAX_NUM_PUS* = 32. The maximum supported precision width parameter *MAX_PRECISION_WIDTH* was set to 4096 leading to a RAM consisting of 28,672 bits. Table 12 lists the computation time in clock cycles of the reference implementation for RSA application. For RSA public-key operations best case and worst case computation times are given under the assumption that the public exponent is $e = 0x10001$. Therefore during the

Table 11: Total Thermal Power Dissipation (TTPD) values for different core variations (Altera Cyclone IV).

$\|w\|$	MAX_NUM_PUS	TTPD 100 MHz	TTPD 200 MHz
16	2	237.36mW	266.38mW
16	4	239.72mW	297.73mW
16	8	245.97mW	304.40mW
16	16	261.07mW	358.22mW
16	32	289.61mW	389.32mW
16	64	358.76mW	549.60mW
32	2	287.18mW	362.94mW
32	4	294.21mW	375.02mW
32	8	311.36mW	401.65mW
32	16	343.68mW	454.94mW
32	32	398.65mW	568.00mW
32	64	490.90mW	760.71mW

Table 12: Core reference implementation RSA computation times.

	$\|p\|$ = 2048	$\|p\|$ = 3072	$\|p\|$ = 4096
CC_{RSA_pub}	134,128	302,309	529,783
CC_{RSA_priv}	17,925,156	59,094,721	138,504,443
$CC_{CRT-RSA}$	9,194,091	15,667,698	36,131,248

Table 13: Core reference implementation Miller-Rabin computation times.

	$\|p\|$ = 512	$\|p\|$ = 768	$\|p\|$ = 1024
CC_{MR_o1}	1,167,529	1,969,583	4,534,869
CC_{MR_o2}	148	212	276
CC_{MR_i1}	1,246	1,430	2,406
CC_{MR_i2}	148	212	276
	$\|p\|$ = 1536	$\|p\|$ = 2048	$\|p\|$ = 3072
CC_{MR_o1}	7,720,993	17,868,205	58,960,069
CC_{MR_o2}	404	532	788
CC_{MR_i1}	2,790	4,726	10,134
CC_{MR_i2}	404	532	788
	$\|p\|$ = 4096		
CC_{MR_o1}	138,267,613		
CC_{MR_o2}	1,044		
CC_{MR_i1}	17,590		
CC_{MR_i2}	1,044		

MontExp operation a total of $17 \times MontMult$, $15 \times CopyH2V$ and $1 \times CopyV2V$ operations will be performed. Since the private exponent is different for varied RSA keys only worst case computation times for the supported precision widths are given. The worst case RSA private key and CRT-accelerated private key computation times assume the worst case clock cycle times of the underlying operations given in previous section.

Table 13 lists the worst computation times in clock cycles of the reference implementation for Miller-Rabin prime testing application for one iteration. Note that the most time consuming operation is part one of the outer loop of Algorithm 2 which will always be performed for each iteration. Depending on the evaluation of the result it might be necessary to execute part two of the outer loop. Furthermore depending on the structure of the prime in question it might be necessary to execute part one and two of the inner loop multiple times.

Table 14: Core reference implementation prime field EC computation times.

	$\|p\|$ = 192	$\|p\|$ = 224	$\|p\|$ = 256
CC_{EC_prep}	5,252	8,281	8,736
$CC_{EC\_mont\_doub}$	742	972	1,234
$CC_{EC\_mont\_add}$	1,468	1,928	2,452
CC_{EC_a2j}	22	24	26
$CC_{EC\_point\_val}$	1,577	2,050	2,587
$CC_{EC\_point\_doub}$	5,613	7,351	9,329
$CC_{EC\_point\_add}$	6,434	8,412	10,662
$CC_{EC\_point\_mult}$	2,288,930	3,499,386	5,077,714
CC_{EC_j2a}	139,233	213,732	311,079
CC_{EC_demont}	734	964	1,226
	$\|p\|$ = 320	$\|p\|$ = 384	$\|p\|$ = 512
CC_{EC_prep}	7,938	10,357	17,575
$CC_{EC\_mont\_doub}$	984	1,364	2,318
$CC_{EC\_mont\_add}$	1,948	2,708	4,612
CC_{EC_a2j}	31	35	44
$CC_{EC\_point\_val}$	2,109	2,895	4,851
$CC_{EC\_point\_doub}$	7,465	10,341	17,533
$CC_{EC\_point\_add}$	8,566	11,842	20,026
$CC_{EC\_point\_mult}$	5,097,858	8,473,906	19,155,090
CC_{EC_j2a}	307,884	514,610	1,172,029
CC_{EC_demont}	974	1,354	2,306

Table 14 lists the computation time in clock cycles of the reference implementation for prime field EC operations for all supported precision widths. The Affine-to-Jacobi Transformation step requires a precision dependent number of clock cycles. For the remaining steps worst case clock cycle times are given. For the Point Multiplication operation an absolute worst case computation time is stated in which a theoretical scalar is hypothesized to be $2^{|p|-1}$, therefore a maximum of $(|p| - 1)$ Point Doubling and $(|p| - 1)$ Point Addition operations would be necessary assuming a simple double and add algorithm.

7. Conclusion and Future Work

A comprehensive adaptable hardware structure for efficient prime finite field and binary finite field arithmetic operations that expand the capabilities of single Montgomery Multiplier hardware designs has been proposed which allows carrying out cryptographic calculations for a large range of different algorithms all based on the same arithmetic unit operations with arbitrary parameters. The approach taken by the proposed core is to combine standard modulo addition / subtraction support with the capability of performing Montgomery Multiplications, full Montgomery Exponentiations,

and the calculation of Montgomery Parameters r and r^2 for arbitrary moduli, bringing together all required arithmetic operations for carrying out a wide range of cryptographic algorithms used today. Through the breakdown of these algorithms individual operation lists have been derived for the arithmetic unit rendering extra precomputations in software unnecessary.

The given values of possible hardware footprint and power consumption for specific core variations allow choosing the proper configuration for a specific implementation. The reference implementation showed that with an internal RAM of merely 3.5 kB the core is capable of performing complete prime field and binary field EC operations for various precision widths of standardised curves. Furthermore the same core configuration is capable of performing (CRT-accelerated) RSA operations for typical precision widths required today, (safe) prime testing/generation, and Diffie-Hellman key exchange operations up to 4096 bit precision widths. The design should further be optimized in terms of power consumption.

However the type of implementation of some core operations, such as the Montgomery Multiplication and especially the Montgomery Exponentiation operation, necessitates additional security considerations, since the calculation times depend on the structure of the processed operands. This makes the design prone to side-channel attacks if security sensitive information, such as private keys, will be processed. But not all operations are critical and must be secured, such as the calculation of the Montgomery Parameters. Therefore during the writing of this article the core will be enhanced to provide a secure calculation bit within the command input word, which, if set, instructs the core to perform the specified arithmetic operation in a time-invariant fashion. In addition, special care has to be taken when defining core operation lists, for instance, for performing elliptic curve Point Multiplication operations. Descriptions performing in a fixed amount of time, e.g., the Montgomery ladder [25], mitigating the risk of timing, and power analysis attacks must be chosen.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] D. Minoli, K. Sohraby, and J. Kouns, "IoT security (IoTSec) considerations, requirements," in *Proceedings of 14th IEEE Annual Consumer Communications Networking Conference (CCNC'17)*, pp. 1006-1007, 2017.

[2] G.-L. Guo, Q. Qian, and R. Zhang, "Different implementations of AES cryptographic algorithm," in *Proceedings of the 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, pp. 1848–1853, 2015.

[3] M. Bafandehkar, S. M. Yasin, R. Mahmod, and Z. M. Hanapi, "Comparison of ECC and RSA algorithm in resource constrained devices," in *Proceedings of the 2013 3rd International Conference on IT Convergence and Security (ICITCS'13)*, pp. 1–3, 2013.

[4] A. Rupani and G. Sujediya, "A Review of FPGA implementation of Internet of Things," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 4, no. 9, 2016.

[5] B. Halak, S. S. Waizi, and A. Islam, "A survey of hardware implementations of elliptic curve cryptographic systems," Cryptology ePrint Archive 2016/712, 2016.

[6] N. Nedjah and L. de Macedo Mourelle, "A review of modular multiplication methods and respective hardware implementations," *Informatica*, vol. 30, no. 1, pp. 111–129, 2006.

[7] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519–521, 1985.

[8] H. Kaur and C. Madhu, "Montgomery multiplication methods - A review," *Journal of Application or Innovation in Engineeting & Management, IJAIEM*, vol. 2, no. 2, pp. 229–235, 2013.

[9] A. F. Tenca and Ç. K. Koç, "A Scalable Architecture for Montgomery Nultiplication," in *Cryptographic Hardware and Embedded Systems*, vol. 1717 of *Lecture Notes in Computer Science*, pp. 94–108, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[10] E. Savas, A. F. Tenca, and Ç. K. Koç, "A Scalable and Unified Multiplier Architecture for Finite Fields GF(p) and GF(2m)," in *Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000)*, pp. 277–292.

[11] M. Morales-Sandoval and A. D. Perez, "Novel algorithms and hardware architectures for Montgomery Multiplication over GF(p)," Cryptology ePrint Archive 2015/696, 2015.

[12] R. Cramer, *Public Key Cryptography – PKC 2008*, vol. 4939, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[13] A. Mrabet, "A Systolic Hardware Architectures of Montgomery Modular Multiplication for Public Key Cryptosystems, Cryptology ePrint Archive," *Report 2016/487*, 2016.

[14] Z. Liu et al., "A tiny RSA coprocessor based on optimized systolic Montgomery architecture," in *Proceedings of the International Conference on Security and Cryptography (SECRYPT'11)*, 2011.

[15] G. D. Sutter, J.-P. Deschamps, and J. L. Imana, "Modular multiplication and exponentiation architectures for fast RSA cryptosystem based on digit serial computation," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 7, pp. 3101–3109, 2011.

[16] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[17] M. O. Rabin, "Probabilistic algorithm for testing primality," *Journal of Number Theory*, vol. 12, no. 1, pp. 128–138, 1980.

[18] J. von zur Gathen and I. E. Shparlinski, "Generating safe primes," *Journal of Mathematical Cryptology*, vol. 7, no. 4, pp. 333–365, 2013.

[19] W. Diffie, W. Diffie, and M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[20] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[21] BSI - Technical Guidline, "Cryptographic Mechanisms: Recommendations and Key Lengths," BSI TR-02102-1, 2017.

[22] D. Wulansari, M. A. Muslim, and E. Sugiharti, "Implementation of RSA algorithm with chinese remainder theorem for modulus n 1024 bit and 4096 bit," *International Journal of Computer Science and Security*, vol. 10, no. 5, pp. 186–194, 2016.

[23] V. S. Miller, "Use of elliptic curves in cryptography," in *Proceedings of the Conference on the Theory and Application of Cryptographic Techniques, CRYPTO 1985*, pp. 417–426, 1985.

[24] B. Zhou, M. Egele, and A. Joshi, "High-performance low-energy implementation of cryptographic algorithms on a programmable SoC for IoT devices," in *Proceedings of the 2017 IEEE High-Performance Extreme Computing Conference (HPEC)*, 2017.

[25] M. Joye and S. Yen, "The Montgomery Powering Ladder," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES'02)*, pp. 291–302, 2002.

Hindawi
Security and Communication Networks
Volume 2018, Article ID 2483619, 13 pages
https://doi.org/10.1155/2018/2483619

Research Article

A Novel Multiple-Bits Collision Attack Based on Double Detection with Error-Tolerant Mechanism

Ye Yuan[1,2], **Liji Wu**[1,2], **Yijun Yang**[1,2] **and Xiangmin Zhang**[1,2]

[1] *Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 10084, China*
[2] *Institute of Microelectronics, Tsinghua University, Beijing 10084, China*

Correspondence should be addressed to Liji Wu; lijiwu@tsinghua.edu.cn

Received 3 November 2017; Revised 6 April 2018; Accepted 3 May 2018; Published 5 June 2018

Academic Editor: Umar M. Khokhar

Side-channel collision attacks are more powerful than traditional side-channel attack without knowing the leakage model or establishing the model. Most attack strategies proposed previously need quantities of power traces with high computational complexity and are sensitive to mistakes, which restricts the attack efficiency seriously. In this paper, we propose a multiple-bits side-channel collision attack based on double distance voting detection (DDVD) and also an improved version, involving the error-tolerant mechanism, which can find all 120 relations among 16 key bytes when applied to AES (Advanced Encryption Standard) algorithm. In addition, we compare our collision detection method called DDVD with the Euclidean distance and the correlation-enhanced collision method under different intensity of noise, which indicates that our detection technique performs better in the circumstances of noise. Furthermore, 4-bit model of our collision detection method is proven to be optimal in theory and in practice. Meanwhile the corresponding practical attack experiments are also performed on a hardware implementation of AES-128 on FPGA board successfully. Results show that our strategy needs less computation time but more traces than LDPC method and the online time for our strategy is about 90% less than CECA and 96% less than BCA with 90% success rate.

1. Introduction

Although modern cryptographic algorithms have been proven to be safe mathematically, this does not mean that the physical implementation is safe enough, where attacker can obtain some physical information from side channel. Side-channel attack (SCA) was proposed almost 20 years ago, which was first put forward in 1996 by Kocher [1] and became a powerful cryptanalysis technique. Power consumption analyses are widely used in SCA, which utilizes the relation between power consumption or electromagnetic signal of the executing device and processed data in order to recover the key value. Since Differential Power Analysis (DPA) was proposed in 1997 [2], whose distinguisher is the difference of the mean traces, various distinguishers have been designed and improved to enhance attack ability and efficiency, for example, Pearson correlation coefficient as a distinguisher for Correlation Power Analysis (CPA)[3], mutual information for Mutual Information Analysis (MIA)[4], and maximum likelihood for Template Attack [5, 6] (TA) and Template Based DPA [7]. However, the necessity of estimating and establishing the leakage model has been a serious restriction for SCA, which collision attack can ignore. Collision attack was first proposed to analyze Hash algorithm [8] and has become a branch of mathematical cryptanalysis, but it only reveals relation between input and output without exploiting internal information as SCA.

As a combination of SCA and collision attack, side-channel collision attack can exploit the information of internal leakage without a large number of power traces as well as the knowledge of the leakage model. Side-channel collision attack showed strong ability of attack, when first presented [9] against Data Encryption Standard (DES) by Schramm et al., which was applied to AES [10] successfully later. Then all kinds of improved versions [11–17] of side-channel collision attack sprang up, and most of these methods show high sensitivity to errors, where the recovered key is totally wrong even when error occurs only in 1 bit under the high noise level circumstance, leading to a low efficiency. Bogdanov presented some voting detection methods that seemed to be more practical [14], but they need too many traces in a profiling phase and encrypting the same plaintexts repeatedly

for decreasing the influence of noise may not be realistic. In 2010, Moradi proposed a correlation-enhanced method [15] that improves the probability of collision, but it may need lots of average power traces to process an attack and is sensitive to errors. In 2011, Bogdanov proposed an attack strategy [17] that uses the results of DPA to test chain separately. This method can improve the success probability in a sense that it cannot check the mistakes in collision detection which highly impact the attack results. Then Gérard et al. combined Low Density Parity Check (LDPC) decoding with correlation-enhanced and Euclidean Distance detection method in 2012 [16], which can be a globally efficient attack strategy in noisy settings. Two side-channel collision attack procedures based on bitwise collision detection were proposed, respectively, by Ren et al [18] in 2015 and by Wang et al [19] in 2017, which may have a poor performance on the detection success rate with high level noise. However, efficiency of collision detection and lack of error-tolerant and check mechanism are two main issues of existing side-channel collision attack.

Our Contribution. In this paper, we propose a novel multiple-bits collision attack framework. In particular, double distance voting detection (DDVD) and the error-tolerant and check mechanism are presented to ensure the high accuracy. In addition, we compare our collision detection method called DDVD with the Euclidean Distance and the correlation-enhanced collision methods under different intensity of noise, which indicates that our detection technique has a better performance in the circumstances of noise. Furthermore, 4-bit collision attack is proven to be optimal in theory and experiments. Practical attack experiments are performed successfully on a hardware implementation of AES in FPGA board.

The remainder of this paper is organized as follows. In Section 2, for a better understanding, we introduce some notations of our method as well as the basic linear collision attacks and then review the binary and ternary voting detection methods, correlation-enhanced collision attack, and LDPC decoding method in collision attack. In Section 3, a novel framework of multiple-bits collision attack is presented and we take the 4-bit model as an example to explain the attack procedure. In Section 4, we propose an improved version with an error-tolerant and check mechanism. In Section 5, we compare our collision detection method with other widely used detection techniques under different intensity of noise and analyze our model, and the experiments as well as the comparisons are also shown. Finally, we give the conclusion in Section 6.

2. Preliminaries

In order to understand the strategy easily, AES is chosen as the target block cipher to perform the attack method. As for the hardware implementation of this paper, it operates each of 16 S-boxes, which are used for the SubBytes operation, sequentially one by one. The following proposed statements and techniques can be successfully utilized in other cryptographic symmetric algorithms.

2.1. Notations. For a better description of the proposed method, we define some notations as follows. First we use letters k and p for 16-byte plaintext and first round subkey, with subscripts indicating a particular byte:

$$P \triangleq \{p_1, p_2, p_3, \cdots, p_{16}\},$$
$$K \triangleq \{k_1, k_2, k_3, \cdots, k_{16}\}. \tag{1}$$

Then we use the superscripts letters m and l for the 4 most significant bits and 4 least significant bits separately, meaning that $p_i^m \triangleq p_i[7{:}\,4]$, $k_i^m \triangleq k_i[7{:}\,4]$, $p_i^l \triangleq p_i[3{:}\,0]$, $k_i^l \triangleq k_i[3{:}\,0]$. Next, the attacker is able to choose the value of plaintext with key value all the same. The superscripts m_f and l_g state that the 4 most significant bits and 4 least significant bits are equal to values f and g in decimal format:

$$p_i^{m\_f} \triangleq \{p_i^m = f\},$$
$$p_i^{l\_g} \triangleq \{p_i^l = g\}. \tag{2}$$

Each trace acquired corresponding to first-round encryption contains 16 subtraces due to 16 sequential S-boxes, with subscripts indicating a particular S-box and each subtrace contains a number p of points, which are denoted by the subscripts:

$$T \triangleq \{T_1, T_2, T_3, \cdots, T_{16}\},$$
$$T_a \triangleq \{t_{a,1}, t_{a,2}, t_{a,3}, \cdots t_{a,p}\}. \tag{3}$$

Furthermore, we use T^{m_f} (T^{l_g}) to denote the power trace corresponding to the plaintext, where the value of 4 most (least) significant bits of all 16 bytes is $f(g)$ in decimal format; namely, $\{p_i^m = f\}_{i=1}^{16}$ ($\{p_i^l = g\}_{i=1}^{16}$).

However, if the superscript is a certain digit, it shows that the plaintext is this value or power trace is corresponding to the plaintext with this value. For example, p_1^{128} means that the first byte of plaintext equals 128 in decimal format and T_1^{128} is denoted as the power trace of the first S-box operation with the corresponding plaintext byte being 128. Meanwhile, we use $T(n)$ and $P(n)$ for the *nth* acquisition of power traces and plaintexts, respectively.

2.2. Linear Collision Attack. The internal collision was first presented for attacking DES [9]. It is based on the fact that if a collision on a key-dependent function can be detected, the attacker can acquire some relations between the different inputs.

Linear collision is based on the internal collision. When it is applied to AES, if a collision between two S-boxes operations of the first round is detected (e.g., the collision between the *ith* and the *jth S-boxes* in Figure 1), it is obvious that (4) is tenable:

$$S\_box(p_i \oplus k_i) = S\_box(p_j \oplus k_j) \tag{4}$$

Then one can obtain a linear equation about the relation between plaintexts and first round subkey:

$$p_i \oplus p_j = k_i \oplus k_j = \Delta k_{i,j} \tag{5}$$

Online Stage:
(1) $\{P^n \mid n = 0, 1, 2, 3, \cdots, N\} \leftarrow$ **Plaintexts**
(2) $\{T^n \mid n = 0, 1, 2, 3, \cdots, N\} \leftarrow$ **AcquireTrace**$(\{P^n\}_{n=0}^{N})$
Offline Stage:
(3) $\{T_1(n) \mid n = 0, 1, 2, 3, \cdots, N\} \leftarrow$ **CutTrace**$(\{T^n\}_{n=0}^{N})$
(4) $\{T_2(n) \mid n = 0, 1, 2, 3, \cdots, N\} \leftarrow$ **CutTrace**$(\{T^n\}_{n=0}^{N})$
(5) $\{\overline{T_1^b} \mid b = 0, 1, 2, 3, \cdots, 255\} \leftarrow$ **AverageTraces**$(\{T_1(n)\}_{n=0}^{N})$
(6) $\{\overline{T_2^b} \mid b = 0, 1, 2, 3, \cdots, 255\} \leftarrow$ **AverageTraces**$(\{T_2(n)\}_{n=0}^{N})$
(7) **for** $\Delta \in \{0, 1, 2, 3, \cdots, 255\}$
(8) $\quad \rho(\Delta) \leftarrow$ **Correlation**$(\{\overline{T_1^b}\}_{b=0}^{255}, \{\overline{T_2^{b\oplus\Delta}}\}_{b=0}^{255})$
(9) **end for**
(10) **return** $\arg\max_{\Delta}\rho(\Delta)$

Algorithm 1: Correlation enhanced detection of S-box 1 and S-box 2.

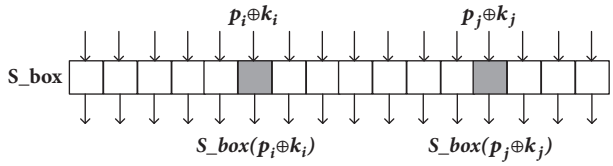


Figure 1: Collisions between two S_boxes.

If the attacker can find all possible relations among 16 key bytes by detecting the collision of *S-boxes*, then he will obtain an equation set about the key bytes of the first round containing 15 linear equations:

$$\begin{aligned} k_1 \oplus k_2 &= \Delta k_{1,2} \\ k_2 \oplus k_3 &= \Delta k_{2,3} \\ &\vdots \\ k_{15} \oplus k_{16} &= \Delta k_{15,16} \end{aligned} \tag{6}$$

Note that all the equations in the set are relevant, and there is only one free variable. Thus, this equation set only has 2^8 possible solutions, which means that we just need to enumerate all 256 possible candidates of 1 key byte to recover the whole key value.

However, under the noisy experiment setting, the collision detection method may detect wrong collisions, which lead to some incorrect equations in (6). For all equations in (6) are relevant, even if only one bit error occurs in any equation, the equation system will have no solution. Thus, in this paper, we propose a detection method called DDVD which is to ensure a high detection success rate.

2.3. Voting Detection Methods. In [14], Bogdanov proposed the voting detection containing binary voting test and ternary voting test. Both binary and ternary voting tests are based on Euclidean Distance. For binary voting test, if an attacker can acquire a lot of power traces of the same plaintexts, he will calculate the Euclidean Distance between two trace pairs of different plaintexts. Then the attacker should calculate the total number of the trace pairs whose distance is less than a predetermined threshold. When the total number is more than the predetermined voting value, it can be confirmed that one collision is detected. However, the basic strategy of ternary voting test is the same as binary voting test, but instead of calculating the Euclidean Distance directly, this method requires calculating the distance between each of the obtained power traces with certain plaintexts and the reference power traces that are obtained during a profiling phase preparing a set of reference traces without knowing related encrypting values.

2.4. Correlation-Enhanced Collision Attack. Correlation-enhanced collision attack was one of the last major advanced detection techniques proposed by Moradi et al in 2010 [15]. This method compares the correlation coefficient between two sets of power traces corresponding to two different S-boxes rather than detecting the collision between two single power traces.

As can be seen in Algorithm 1, we take the detection between S-box 1 and S-box 2 as an example. In the online stage, an attacker should obtain N power traces corresponding to N plaintexts. When in offline stage, the attacker cuts the power trace into 16 sections based on the operation of 16 S-boxes and takes the section for S-box 1 and S-box 2. Then $T_1(n)$ is divided into 256 groups according to the plaintext byte value and the attacker can get the averaged power traces of each group $(\{\overline{T_1^b}\}_{b=0}^{255})$, which is the same for S-box 2. Next, for each value of $\Delta \in GF(2^8)$, the attacker rearranges $\{\overline{T_2}\}$ based on the value of $b \oplus \Delta$ and calculates correlation coefficients $\rho(\Delta)$ between $\{\overline{T_1^b}\}_{b=0}^{255}$ and $\{\overline{T_2^{b\oplus\Delta}}\}_{b=0}^{255}$. If $\Delta = k_1 \oplus k_2$, the correlation $\rho(\Delta)$ shall reach a maximum value; otherwise, it should have a pretty low value.

2.5. LDPC Decoding Problem in Collision Attack. In [16], Gérard et al. proposed a unified and optimized collision attack method. The proposed method rewrote the linear collision attack as a LDPC decoding problem, according to the linear relationship:

$$\Delta k_{i_1,i_2} \oplus \Delta k_{i_2,i_3} = \Delta k_{i_1,i_3} \quad \left(\forall 1 \le i_1 < i_2 < i_3 \le 16\right) \tag{7}$$

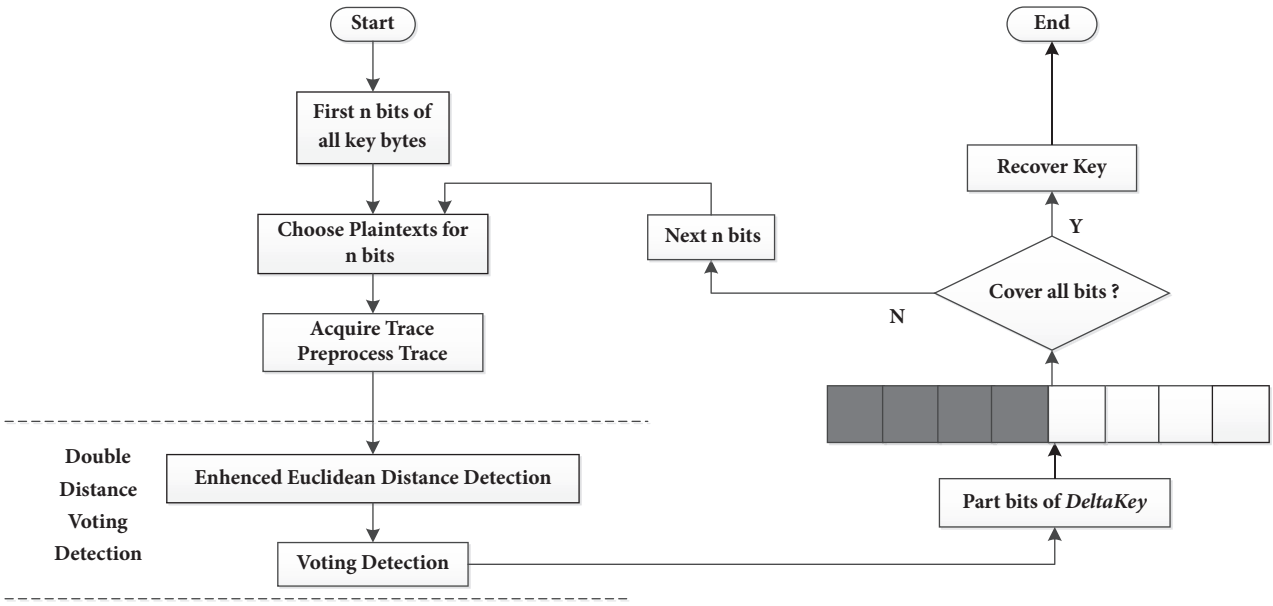


Figure 2: Framework of multiple-bits (n bits) collision attack.

(1) $\{P(n) \mid n = 1, 2, 3, \cdots, 16N\} \leftarrow$ **ChoosePlaintexts**()
(2) $\{T^{m\_j}(n) \mid n = 1, 2, 3, \cdots, N\}_{j=0}^{15} \leftarrow$ **AcquireTrace**$(\{\mathbf{P(n)}\}_{\mathbf{n=1}}^{\mathbf{16N}})$
(3) $\{\bar{t}_i^{m\_j} \mid 1 \le i \le 16\}_{j=0}^{15} \leftarrow$ **PreProTrace**$(\{\mathbf{T^{m\_j}(n)} \mid \mathbf{0 \le n \le N}\}_{\mathbf{j=0}}^{\mathbf{15}})$
(4) **for** each $(\{(i_1, i_2) \mid 1 \le i_1 < i_2 \le 16\})$
(5) $\quad \Delta k_{i_1,i_2} \leftarrow$ ***DDVD***$(\{\bar{t}_{i_1}^{m\_j}\}_{j=0}^{15}, \{\bar{t}_{i_2}^{m\_j}\}_{j=0}^{15})$
(6) **end for**
(7) **Recoverkey**$(\Delta_{i_1,i_2}(1 \le i_1 < i_2 \le 16))$

Algorithm 2: Multiple-bits side-channel collision attack.

The vector $\Delta K = (\Delta k_{1,2}, \Delta k_{2,3}, \cdots, \Delta k_{15,16})$ can be seen as an LDPC codeword whose dimension is 15 and length is 120. Finding the right key value is just to decode the LDPC code. Furthermore, in order to make the attack method have a better performance in the noisy settings, actual posteriori probability value of each code was used for LDPC decoding, which is called soft decision decoding. Unlike soft decision decoding, hard decision decoding uses bit value for decoding. Compared to soft decision decoding, the effect of hard decision decoding is worse in a noise setting but the computation complexity is lower.

In this paper, for the error-tolerant and check mechanism, we choose top three possible values for each $\Delta k_{i_1,i_3}$ as candidates and find the likeliest value based on (7). It may be seen as a kind of hard decision decoding procedure. However, due to the DDVD, the detection success rate may remain in a high level in some noisy settings.

3. A Novel Framework of Multiple-Bits Collision Attack

In this section, a framework of multiple-bits side-channel collision attack is presented. As can be seen in Figure 2, plaintexts need to be chosen based on multiple-bits (n-bits) model and then we prepare the power trace. Double distance voting detection is the important part of the framework ensuring the high success probability along with high efficiency. However, the principal part of the framework is based on a circulation. Each iteration stands for an attack, where we obtain only n-bits of a byte relation between all key bytes. After several iterations, the whole byte value of Δ between all key bytes can be acquired, which will be utilized to recover the key value.

Due to the fact that the 4-bit collision attack leads to the highest efficiency, which will be proven and verified in Section 5, we take the 4-bit model as an example to explain our attack method. According to our attack framework, for 4-bit model, 2 iterations are enough to recover the key value, whereof one is for the four most significant bits and the other is for the four least significant bits. In the rest of this paper, we only describe the strategy for the four most significant bits of one byte. The remaining four least significant bits can be found using the same technique.

3.1. The Idea in a Nutshell. For a better understanding, we describe the main flow of our attack strategy in Algorithm 2. As our description is based on 4-bit model, all the following

Input: the total number of plaintexts 16N
Output: 16N plaintexts: $\{\mathbf{P(n)}\}$ $(\mathbf{1 \le n \le 16N})$
(1) **for** $j = 0$: 15
(2) **for** $h = 1$: N
(3) $P(16j + h) = \{p_i \mid p_i^m = j, p_i^l = random(16)\}_{i=1}^{16}$
(random(n) is to generate an integer ranging from 0 to n-1)
(4) **end for**
(5) **end for**
(6) **return** $\{P(n)\}$ $(1 \le n \le 16N)$

Algorithm 3: *ChoosePlaintexts*.

Input: 16 sets of power traces:$\{\mathbf{T^{m\_j}(n) \mid n = 1, 2, 3, \cdots, N}\}_{\mathbf{j=0}}^{\mathbf{15}}$
Output: 16 averaged traces:$\{\overline{\mathbf{T}}^{\mathbf{m\_j}}\}_{\mathbf{j=0}}^{\mathbf{15}} = \{\bar{\mathbf{t}}_{\mathbf{i}}^{\mathbf{m\_j}} \mid \mathbf{i = 1, 2, 3, \cdots, 16}\}_{\mathbf{j=0}}^{\mathbf{15}}$
(1) **for** $j = 0$: 15
(2) $\overline{T}^{m\_j} = (1/N) \sum_{n=1}^{N} T^{m\_j}(n)$
(3) Cut $\overline{T}^{m\_j}$ into 16 sub-traces: $\overline{T}^{m\_j} = \{\bar{t}_i^{m\_j} \mid i = 1, 2, 3, \cdots, 16\}$
(4) **end for**
(5) **return** $\{\overline{T}^{m\_j}\}_{j=0}^{15} = \{\bar{t}_i^{m\_j} \mid i = 1, 2, 3, \cdots, 16\}_{j=0}^{15}$

Algorithm 4: *PreprocessTraces*.

statements can be applied to our multiple-bits models. For example, some parameters of 4-bit model are 15 or 16, which can be interpreted as $2^4 - 1$ or 2^4, and thus for other n-bits model, the parameters should be $2^n - 1$ or 2^n.

Like most of other attack strategies, our method also first gets some traces and proceeds with some preprocessing, seen from Steps1, 2, and 3. Double distance voting detection is the core of our method combining enhanced Euclidean Distance detection and voting detection, which ensures our success rate. Finally, based on the main idea in Section 2.2, when we find all possible relations among 16 key bytes, the brute force way is able to find the right key value quickly, for we only need to enumerate 256 key values. Some details will be presented in the following sections.

3.2. Choose Plaintexts. According to our attack strategy, we assume that the attacker is able to choose the plaintext. The 4-bit side-channel collision attack model aims to detect the collision of 4 bits between 2 different S-boxes, and in this situation other 4 bits are the noise for the detection. It is important for improving the efficiency of our method to determine how to choose the value of $\{p_i^m\}_{i=1}^{16}$. Algorithm 3 presents the strategy of plaintexts choice, which can generate $16N$ plaintexts. For $\{p_i^m\}_{i=1}^{16}$ equal to each of the values belonging to $GF(2^4)$, N plaintexts can be obtained with the other 4 bits ($\{p_i^l\}_{i=1}^{16}$) being random. Due to the fact that the value belonging to $GF(2^4)$ ranges from $(0000)_2$ to $(1111)_2$, $16N$ plaintexts should be obtained.

3.3. Acquire Traces and Preprocess Traces. We can obtain $16N$ power traces of the first round operation corresponding to $16N$ plaintexts. The obtained power traces can be divided into 16 sets according to the values of $\{p_i^m\}_{i=1}^{16}$. For the values of $\{p_i^m\}_{i=1}^{16}$ are the same all the time ranging from 0 to 15 referred to Section 3.2 and each value corresponds to N random value for $\{p_i^l\}_{i=1}^{16}$, the number of the sets is 16 and each set contains N power traces.

This can be easily expanded to the attack for the four least significant bits with $\{p_i^l\}_{i=1}^{16}$ ranging from 0 to 15 and $\{p_i^m\}_{i=1}^{16}$ being random.

As for preprocessing the power traces, the detailed procedures are stated in Algorithm 4. For each of the trace sets, we can average all N power traces in this set to a single averaged trace. Each of the averaged power traces is composed of 16 subtraces corresponding to 16 sequential S-boxes operations and can be cut into 16 subtraces. Thus, we can obtain 16 averaged power traces containing 16 subtraces.

3.4. Double Distance Voting Detection. Double distance voting detection is the core of our attack technique ensuring the high success rate and stability. As is seen in Algorithm 5, DDVD is composed of enhanced Euclidean Distance detection and voting detection. For a better understanding of our DDVD technique, a diagrammatic sketch is shown in Figure 3. Taking S-boxes i_1 and i_2 as an example, there shall be 16 subtraces $\{\bar{\mathbf{t}}_{\mathbf{i_1}}^{\mathbf{m\_j_1}}\}_{\mathbf{j_1=0}}^{\mathbf{15}}$ and $\{\bar{\mathbf{t}}_{\mathbf{i_2}}^{\mathbf{m\_j_2}}\}_{\mathbf{j_2=0}}^{\mathbf{15}}$ for S-boxes i_1 and i_2, respectively, after the former operation. Each single trace $\bar{\mathbf{t}}_{\mathbf{i_1}}^{\mathbf{m\_j_1}}$ of $\{\bar{\mathbf{t}}_{\mathbf{i_1}}^{\mathbf{m\_j_1}}\}_{\mathbf{j_1=0}}^{\mathbf{15}}$ should operate the enhanced Euclidean Distance detection with the trace set $\{\bar{\mathbf{t}}_{\mathbf{i_1}}^{\mathbf{m\_j_1}}\}_{\mathbf{j_1=0}}^{\mathbf{15}}$ with 16 traces, which is seen as a decision making unit.

For example, we compute the Euclidean distance between $\bar{\mathbf{t}}_{\mathbf{i_1}}^{\mathbf{m\_6}}$ and each trace of $\{\bar{\mathbf{t}}_{\mathbf{i_2}}^{\mathbf{m\_j_2}}\}_{\mathbf{j_2=0}}^{\mathbf{15}}$, and if the minimum distance is between $\bar{\mathbf{t}}_{\mathbf{i_1}}^{\mathbf{m\_6}}$ and $\bar{\mathbf{t}}_{\mathbf{i_2}}^{\mathbf{m\_j_2}}$, this decision making unit generates one of the possible values for $\Delta k_{i_1,i_2}^m$, namely, $\Delta_6 = (0110)_2 \oplus$

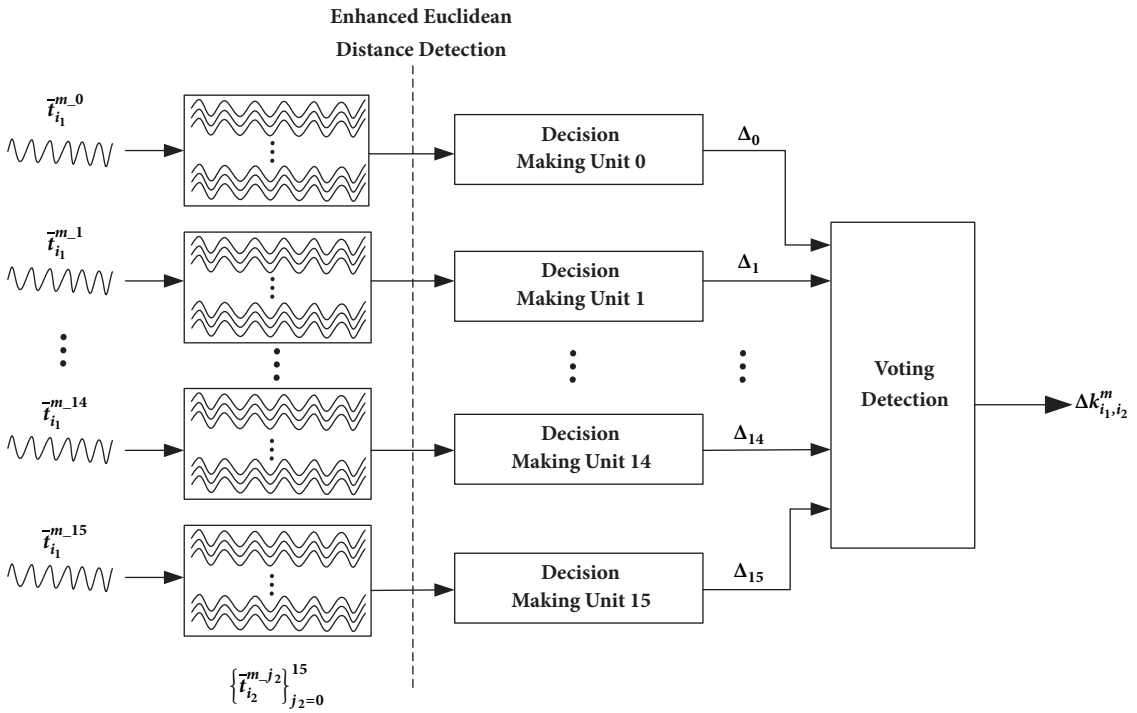


Figure 3: Flow of double distance voting detection.

Input: 2 sets of sub-traces: $\{\bar{\mathbf{t}}_{\mathbf{i}_1}^{\mathbf{mj_1}}\}_{\mathbf{j_1}=0}^{\mathbf{15}}, \{\bar{\mathbf{t}}_{\mathbf{i}_2}^{\mathbf{mj_2}}\}_{\mathbf{j_2}=0}^{\mathbf{15}}$
Output: the 4 most significant bits of $\Delta\mathbf{k}_{\mathbf{i_1},\mathbf{i_2}}$: $\Delta\mathbf{k}_{\mathbf{i_1},\mathbf{i_2}}^{\mathbf{m}}$
Enhanced Euclidean Distance Detection:
(1) **for** $(0 \le j_1 \le 15)$
(2) **for** $(0 \le j_2 \le 15)$
(3) $Distance(j_1 \oplus j_2) = \sum_{l=1}^{L} (\bar{t}_{i_1,l}^{mj_1} - \bar{t}_{i_2,l}^{mj_2})^2$
(4) **end for**
(5) $\Delta_{j_1} = \arg\min_{j_1 \oplus j_2} Distance(j_1 \oplus j_2)$
(6) **end for**
Voting Detection:
(7) $num_n = 0 (0 \le n \le 15)$
(8) **for** $(0 \le j \le 15)$
(9) **for** $(0 \le n \le 15)$
(10) **if** $(\Delta_j = n)$
(11) $num_n = num_n + 1$
(12) **else**
(13) $num_n = num_n$
(14) **end if**
(15) **end for**
(16) **end for**
(17) $\Delta k_{i_1,i_2}^m = \arg\max_n num_n$
(18) **return** $\Delta k_{i_1,i_2}^m$

Algorithm 5: Double distance voting detection.

j_2. This must be done for all 16 single traces of $\{\bar{\mathbf{t}}_{\mathbf{i}_1}^{\mathbf{m\_j_1}}\}_{\mathbf{j_1}=0}^{\mathbf{15}}$; therefore there shall be 16 decision making units generating 16 possible values $\{\Delta_{j_1}\}_{j_1=0}^{15}$ for the candidates of $\Delta k_{i_1,i_2}^m$. During voting detection stage, the value that occurs the maximum times among $\{\Delta_{j_1}\}_{j_1=0}^{15}$ will be voted as the final value of $\Delta k_{i_1,i_2}^m$.

4. Improved Framework

In this section, we propose an improved framework of multiple-bits side-channel collision attack, where we modify our double distance voting detection and insert the error-tolerant and check mechanism. As is shown in Figure 4, in this new framework, the modified double distance detection works with the error-tolerant and check mechanism, which leads to a remarkable promotion in the success rate as well as the attack efficiency.

We still take the 4-bit model as an example to describe the improved attack framework of our method. The procedure is shown in Algorithm 6. Like Section 3, we only care about the four most significant bits, with the four least significant bits being almost the same. Algorithms of *ChoosePlaintexts*, *AcquireTrace*, and *PreprocessTrace* are all the same. In the rest of this section, we only explain the modified double distance voting detection and the fresh error-tolerant and check mechanism.

4.1. Modified Double Distance Voting Detection. The modified detection method is shown in Algorithm 7. Just like the original one, the input of the DDVD is still 2 sets of subtraces corresponding to 2 different S-boxes, but the output changes from a single value to a 1×3 matrix including 3 candidate values of $\Delta k_{i_1,i_2}^m$. Euclidean Distance between each subtrace of a certain S-box and a set of subtraces of another S-box also should be calculated first. Then, instead of choosing n with the maximum number as the result, we prefer three values whose number is in the top three $\Delta k_{i_1,i_2}^m$ where i_1 and i_2 range from 1 to 15.

(1)$\{P(n)|n = 1, 2, 3, \cdots, 16N \leftarrow$ **ChoosePlaintexts**$()\}$
(2)$\{T^{mj}(n) \mid n = 0, 1, 2, 3, \cdots, N\}_{j=0}^{15} \leftarrow$ **AcquireTrace**$(\{P(n)\}_{n=1}^{16N})$
(3)$\{\bar{t}_i^{mj} \mid 1 \le i \le 16\}_{j=0}^{15} \leftarrow$ **PreProcessTrace**$(\{T^{mj}(n) \mid 0 \le n \le N\}_{j=0}^{15})$
(4) **for** each $(\{(i_1, i_2) \mid 1 \le i_1 < i_2 \le 16\})$
(5) $\Delta k_{i_1,i_2}^{m}[1: 3] \leftarrow$ **DDVD**$(\{\bar{t}_{i_1}^{mj}\}_{j=0}^{15}, \{\bar{t}_{i_2}^{mj}\}_{j=0}^{15})$
(6) **end for**
(7)$\{\Delta k^{m}[1: 15], pass\} \leftarrow$ **Error_tolerant**$(\Delta k_{i_1,i_2}^{m}[1: 3] \mid 1 \le i_1 < i_2 \le 16\})$
(8) **if** $(pass=1)$
(9) **Recoverkey** $(\Delta k^{m}[1: 15])$
(10) **else**
(11) **Back to (1)**
(12) **end if**

Algorithm 6: Improved framework.

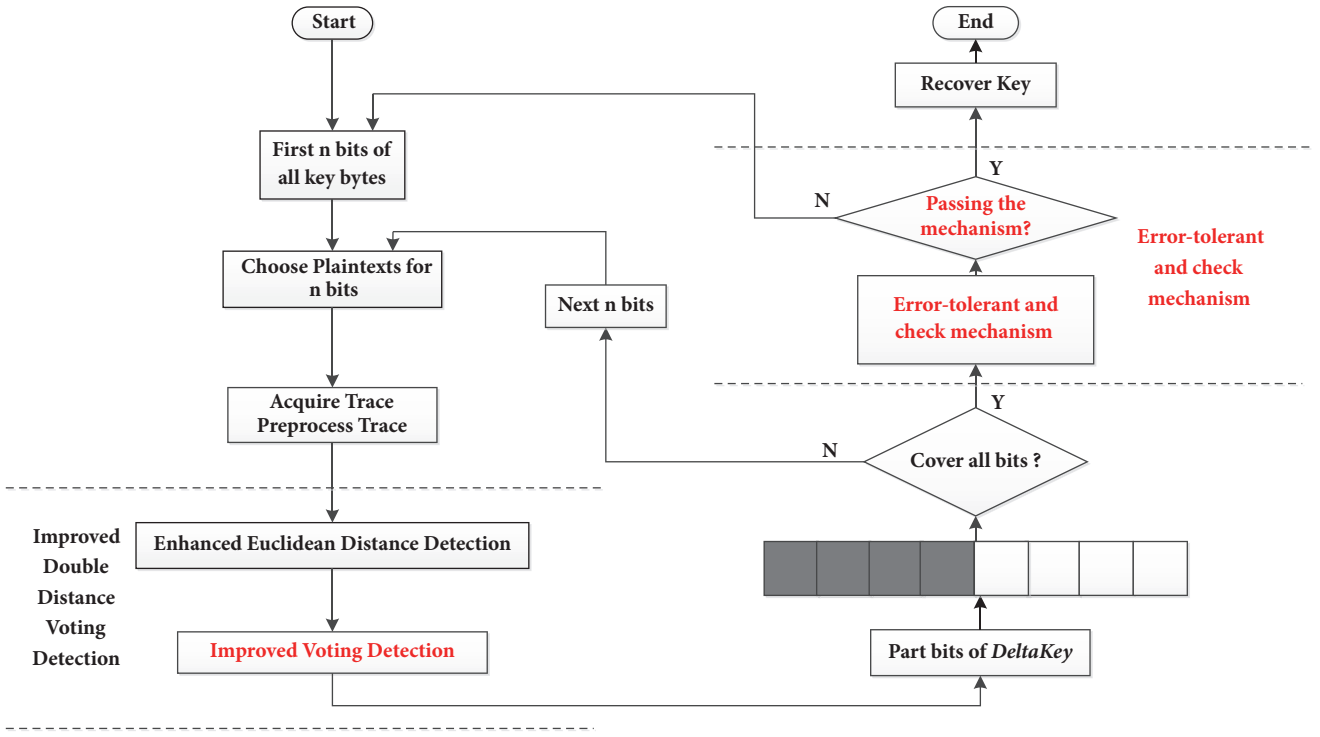


Figure 4: Improved framework with error-tolerant mechanism.

4.2. Error-Tolerant and Check Mechanism. Error-tolerant and check mechanism is presented in Algorithm 8. Three candidate values ensure the error-tolerant mechanism. The main thought of the error detection and tolerance is based on (6), which provides a way to find errors occurring in collision detections.

In order to recover the key value correctly, 15 delta values $(\Delta k_{1,2}, \Delta k_{2,3}, \ldots, \Delta k_{15,16})$ should be right. Thus, for each candidate of $\Delta k_{n,n+1}$, any exiting relations in (7) should be checked. If there exists a candidate of $\Delta k_{n,n+1}$ that can pass the check, it will be the final result of $\Delta k_{n,n+1}$; otherwise this attack is considered to have failed and should start from the beginning again.

5. Model Analysis and Experiments Results

5.1. Comparison of Detection Success Rate under Noise. Collision detection technique has played an important role in side-channel collision attack. In this section, we compare double distance voting detection (DDVD) proposed in this paper with other two widely used detection techniques, which are correlation-enhanced detection[15] and traditional Euclidean Distance detection with dimension reduction[17], respectively.

The detailed procedure of correlation-enhanced detection is already proposed in Section 2.4. In [17], Bogdanov presented a collision attack method based on Euclidean Distance combining DPA. However, according to the method, if

Input : 2 sets of sub-traces:$\{\bar{\mathbf{t}}_{\mathbf{i_1}}^{\mathbf{m,j_1}}\}_{\mathbf{j_1=0}}^{\mathbf{15}}, \{\bar{\mathbf{t}}_{\mathbf{i_2}}^{\mathbf{m,j_2}}\}_{\mathbf{j_2=0}}^{\mathbf{15}}$
Output: the 4 most significant bits of $\Delta k_{i_1,i_2}^{m}$: $\mathbf{\Delta k_{i_1,i_2}^{m}}$**(1 × 3 matrix)**
Enhanced Euclidean Distance Detection:
(1) **for** $(0 \le j_1 \le 15)$
(2) **for** $(0 \le j_2 \le 15)$
(3) $\text{Distance}(j_1 \oplus j_2) = \sum_{l=1}^{L}(\bar{t}_{i_1,l}^{m,j_1} - \bar{t}_{i_2,l}^{m,j_2})^2$
(4) **end for**
(5) $\Delta_{j_1} = \arg\min_{j_1 \oplus j_2} \text{Distance}(j_1 \oplus j_2)$
(6) **end for**
Improved Voting Detection:
(7) $\text{num}_n = 0 (0 \le n \le 15)$
(8) **for** $(0 \le j \le 15)$
(9) **for** $(0 \le n \le 15)$
(10) **if** $(\Delta_j = n)$
(11) $num_n = num_n + 1$
(12) **else**
(13) $num_n = num_n$
(14) **end if**
(15) **end for**
(16) **end for**
(17) $\Delta k_{i_1,i_2}^{m}[1] = \arg\max_n num_n$
(18) $\Delta k_{i_1,i_2}^{m}[2] = \arg\max_n num_n \ (n \ne \Delta k_{i_1,i_2}^{m}[1])$
(19) $\Delta k_{i_1,i_2}^{m}[3] = \arg\max_n num_n \ (n \ne \Delta k_{i_1,i_2}^{m}[1], \Delta k_{i_1,i_2}^{m}[2])$
(20) **return** $\Delta k_{i_1,i_2}^{m}$

Algorithm 7: Double distance voting detection (modified).

Input : 3 candidates for each $\Delta k_{i_1,i_2}^{m}$: $\{\mathbf{\Delta k_{i_1,i_2}^{m}[1:3]} \mid \mathbf{1 \le i_1 < i_2 \le 16}\}$
Output: $\mathbf{\Delta k^{m}[1:15]}$, **pass**
(1) **for** $(1 \le i_1 \le 14)$
(2) $i_2 = i_1 + 1$
(3) **for** $(1 \le t \le 3)$
(4) **if** (existing $x \in \Delta k_{i_2,h}^{m}[1:3]$ *and* $y \in \Delta k_{i_1,h}^{m}[1:3] (i_2 < h \le 16)$
satisfying $\mathbf{\Delta k_{i_1,i_2}^{m}[t] = x \oplus y}$)
(5) $\Delta k^{m}[i_1] = \Delta k_{i_1,i_2}^{m}[t]$ $pass = 1$
(6) **else**
(7) $pass = 0$ exiting all loops
(8) **end if**
(9) **end for**
(10)**end for**

Algorithm 8: Error-tolerant.

collision detection generates incorrect results, DPA makes no sense for recovering the right key value. Therefore, the success rate of Euclidean Distance detection for that method is the key part.

The power traces are obtained from an AES hardware design implemented on a SAKURA-G board. Each trace shall be averaged by four power traces with the same input. The noise of the traces usually comes from both electronic noise mainly containing power supply noise, clock generator noise, conducted emissions, and radiated emissions and algorithm noise which are the power assumption of other uncorrelated operations. For SAKURA-G is a dedicated board that may be far from being noisy, we can add the Gaussian noise of different intensity into the averaged traces to model the noise, which can be used for an initial analysis of efficiency of different detection techniques [14]. SNR (signal-to-noise ratio) is used for indicating the intensity of the noise, which is defined as follows:

$$SNR = 10 \log_{10} \frac{P_{signal}}{P_{noise}} \tag{8}$$

The comparison result is shown in Figure 5. Detection technique proposed in this paper is marked with DDVD, and correlation-enhanced method in [15] and Euclidean Distance in [17] are denoted as CE and ED, respectively. The value of SNR ranges from 0 dB to 30 dB. Each technique is done

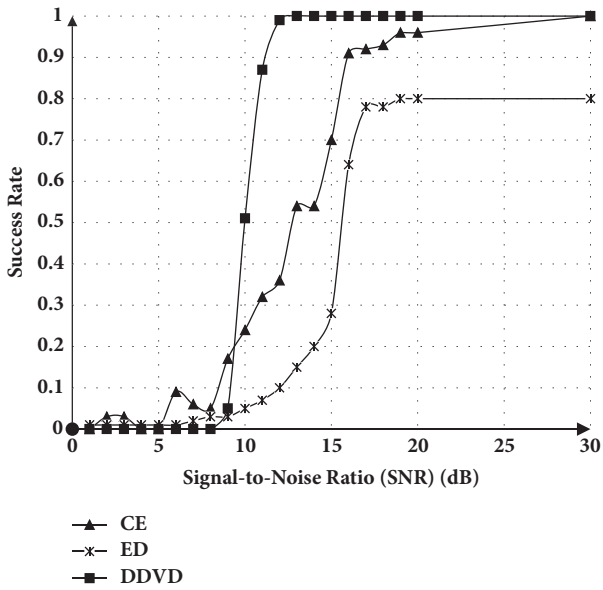


Figure 5: Comparison of detection success rate.

for 1000 times to calculate the success rate. As is shown in Figure 5, our detection technique performs better under the noise.

5.2. How Many Bits Are Best for Multiple-Bits Model. In this paper, we propose a multiple-bits collision attack model, and all the statements are based on the 4-bit model, which can be expanded to other n-bits (n ranging from 1 to 8) models. However, one question we should figure out is how many bits are best for the multiple-bits model of our attack method.

What matters in our attack strategy is the necessary number of power traces to reach a given success rate, which reflects the attack efficiency. Therefore, we will analyze the necessary number of power traces for different model both in theory and in experiment.

In Section 5.1, the performance of our detection method in noise environment is presented; thus, for a simple analysis in this section, we assume that the noise for an n-bits model only comes from the operation of other *8-n* bits and all 2^n decision making units in Figure 3 are independent. So, for the n-bits model, when preparing the power traces (Sections 3.2 and 3.3), we need h power traces of each byte to obtain the averaged traces. When we compute the Euclidean Distance between two single averaged traces whose n-bits can cause the collision (e.g., $p_{i_1}[n-1:0] \oplus p_{i_2}[n-1:0] = \Delta k_{i_1 i_2}[n-1:0]$), the probability that these two averaged traces have the least Euclidean Distance is

$$Pr = 1 - \frac{C_{2^{8-n}}^{h} \times C_{2^{8-n}-h}^{h}}{C_{2^{8-n}}^{h} \times C_{2^{8-n}}^{h}} \tag{9}$$

where letter C is denoted as combinatorial number, n equals the number of bits in the model, and h equals the number of traces for calculating the average. Due to the fact that 8 n-bits of one byte are random, there are a total of $C_{2^{8-n}}^{h} \times C_{2^{8-n}}^{h}$ kinds of choices to determine these two averaged power traces.

Table 1: The necessary number of original attacks to reach 90% success rate.

n-bits	1	2	3	**4**	5	6	7	8
Number	288	160	168	**128**	192	256	512	256

Table 2: The necessary number of improved attacks to reach 90% success rate.

n-bits	1	2	3	**4**	5	6	7	8
Number	No	128	120	**96**	140	256	256	256

Since there shall be only one corresponding plaintext of one byte which can cause a collision with each plaintext of another byte, there are a total of $C_{2^{8-n}}^{h} \times C_{2^{8-n}-h}^{h}$ kinds of choices that include no collision plaintext pair.

The probability of successful detection for the method proposed in Section 3 and the improved method presented in Section 4 is calculated separately as follows:

$$Pr\_det = 1 - \sum_{i=2^{n-1}}^{2^n} C_{2^n}^{i} \times (1 - Pr)^i \times Pr^{2^n - i} \tag{10}$$

$$Pr\_impro = 1 - \sum_{i=2^n - 2^{n-2}}^{2^n} C_{2^n}^{i} \times (1 - Pr)^i \times Pr^{2^n - i} \tag{11}$$

where Pr is equal to (9) and n is the number of bits in the model. From the illustration of Figure 3, there are 2^n decision making units for the n-bits model. According to the rules of voting detection that the value that occurs the maximum times is chosen as the final result, if more than half of the decision making units generate the wrong answer, the voting detection shall fail. Result of all 2^n decision making units can be seen as the binomial distribution, so the probability that more than half of the units generate wrong result is $\sum_{i=2^{n-1}}^{2^n} C_{2^n}^{i} \times (1 - Pr)^i \times Pr^{2^n - i}$. Analysis for the improved method in Section 4 is similar, and if more than three-quarters of the decision making units generate the wrong answer, the voting detection shall fail.

As for the necessary number of the power traces, it can be calculated as follows:

$$Trace\_Number = 2^n \times h \times \left\lceil \frac{8}{n} \right\rceil \tag{12}$$

where $\lceil n \rceil$ stands for the minimum integer that is larger than n. According to our attack strategy, for each n-bits model, we should get 2^n averaged power traces and each trace is averaged by h original power traces. Obviously, the method should be operated for $\lceil 8/n \rceil$ times.

According to (9), (10), (11), and (12), we estimate the necessary number of the power traces to reach a 90% success rate for the basic attack method in Section 3 and an improved version in Section 4, shown in Tables 1 and 2, respectively. The 1-bit model only has two possible results, so improved method makes no sense for it. It is obvious that, in theory, 4-bit model collision attack needs the least number of traces with high efficiency, which will be verified later in experiments.

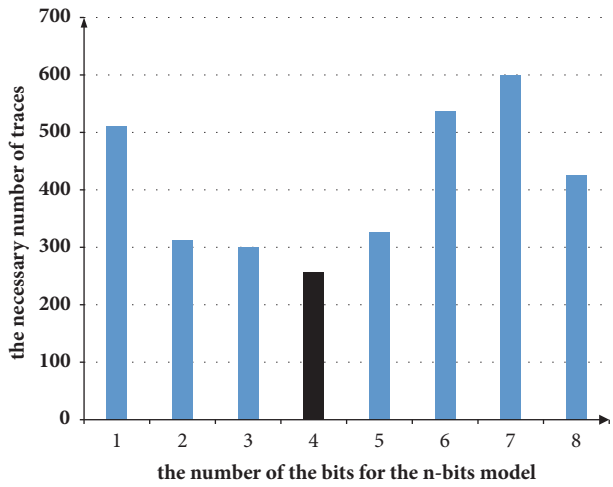


Figure 6: Necessary number of traces for MBDD to reach 90% success rate.

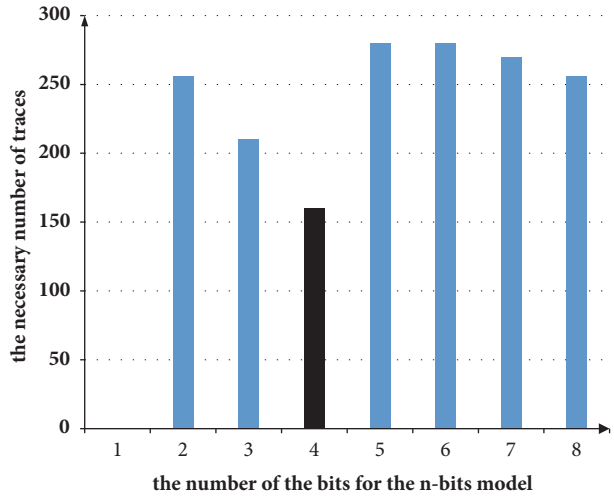


Figure 7: Necessary number of traces for MBDD-ET to reach 90% success rate.

Furthermore, the practical experiments have been carried out to find how many bits are best for the proposed multiple-bits model. Figure 6 shows the necessary number of power traces to reach a 90% success rate for the original approach presented in Section 3 (denoted as MBDD), while Figure 7 shows the he necessary number of power traces for the improved version proposed in Section 4 (denoted as MBDD-ET). As can be seen in Figures 6 and 7, it is verified that 4-bit model (in black) is the best choice when operating the proposed attack strategy.

5.3. Experiments and Results. The attack method and the improved method with error-tolerant mechanism have been performed successfully in practice against a hardware design with an 8-bit data path of AES, where 16 S-boxes are sequentially operated in every round operation. The target AES is implemented on a Xilinx SPARTAN-6 FPGA of a SAKURA-G circuit board. An Agilent MSO-X 9104A oscilloscope is employed to collect original power traces. In our case, each power trace obtained contains about 32365 points.

For a better understanding, operation on k_1^m and k_2^m corresponding to S-box 1 and S-box 2 is taken as an example to present the process of double distance voting detection. Without loss of generality, k_1^m and k_2^m are fixed as

$$k_1^m = (1101)_2\,,$$
$$k_2^m = (0110)_2\,, \tag{13}$$
$$\Delta k_{1,2}^m = (1011)_2\,.$$

Two corresponding sets of subtraces are denoted as $\{\bar{t}_1^{m\_j_1}\}_{j_1=0}^{15}$ and $\{\bar{t}_2^{m\_j_2}\}_{j_2=0}^{15}$. Figure 4 shows square of difference between each subtrace $\bar{t}_1^{m\_j_1}$ of $\{\bar{t}_1^{m\_j_1}\}_{j_1=0}^{15}$ and the trace set $\{\bar{t}_2^{m\_j_1}\}_{j_1=0}^{15}$ with 16 traces. Figures 8(a)–8(p) are like 16 decision making units in Figure 3 corresponding to j_1 ranging from 0 to 15. The accumulation of all points' square of difference between two subtraces is the value of Euclidean Distance.

We take Figure 8(a) as an example to describe its meaning. Figure 8(a) shows the result of $(\bar{t}_{1,l}^{m\_0} - \bar{t}_{2,l}^{m\_j_2})^2$ for each point l ranging from 1 to 32365 and each value of j_2 ranging from 0 to 15. The black curve represents the square of difference between each point of $\bar{t}_{1,l}^{m\_0}$ and $\bar{t}_{2,l}^{m\_0 \oplus \Delta k_{1,2}^m}$, which are two traces corresponding to a collision in theory. However, square of differences between $\bar{t}_{1,l}^{m\_0}$ and other traces in the trace set $\{\bar{t}_2^{m\_j_1}\}_{j_1=0}^{15}$ is marked by grey curves. If the black curve is lower than any other grey curves, the decision making unit will generate the right candidate. An initial and rough conclusion can be drawn that when in situations like Figure 8(a), whose black curve is close to zero, two traces corresponding to a collision may have the lowest distance, meaning that the corresponding decision making unit generates the right candidate, but in some exceptional situations such as Figure 8(p), whose black curve is higher than some grey curves, collision cannot be assured by minimum Euclidean Distance and the unit generates the wrong candidate. Therefore, voting detection works to determine the final value of $\Delta k_{1,2}^m$. As is shown in Figure 9, $(1011)_2$ occurs the maximum times, and voting detection chooses it to be the final result.

5.4. Comparison. In this section, we compare our improved attack version denoted as MBDD with correlation-enhanced collision attack [15], bitwise collision attack [19], and LDPC method with Euclidean Distance detection [16] denoted as CECA, BCA, and LDPC, respectively. Comparisons are done from three aspects, which are relation between success rate and necessary number of traces, relation between success rate and online time, and relation between offline time and online time. Each compared method was performed 1000 times for calculating an actual success rate.

In this section, t_{ave} is used for indicating the total time that the oscilloscope spends on capturing and averaging one power trace in real time, and t_s is for indicating the time that the oscilloscope spends on acquiring and saving one trace. Taking Agilent MSO-X 9104A oscilloscope that we use for

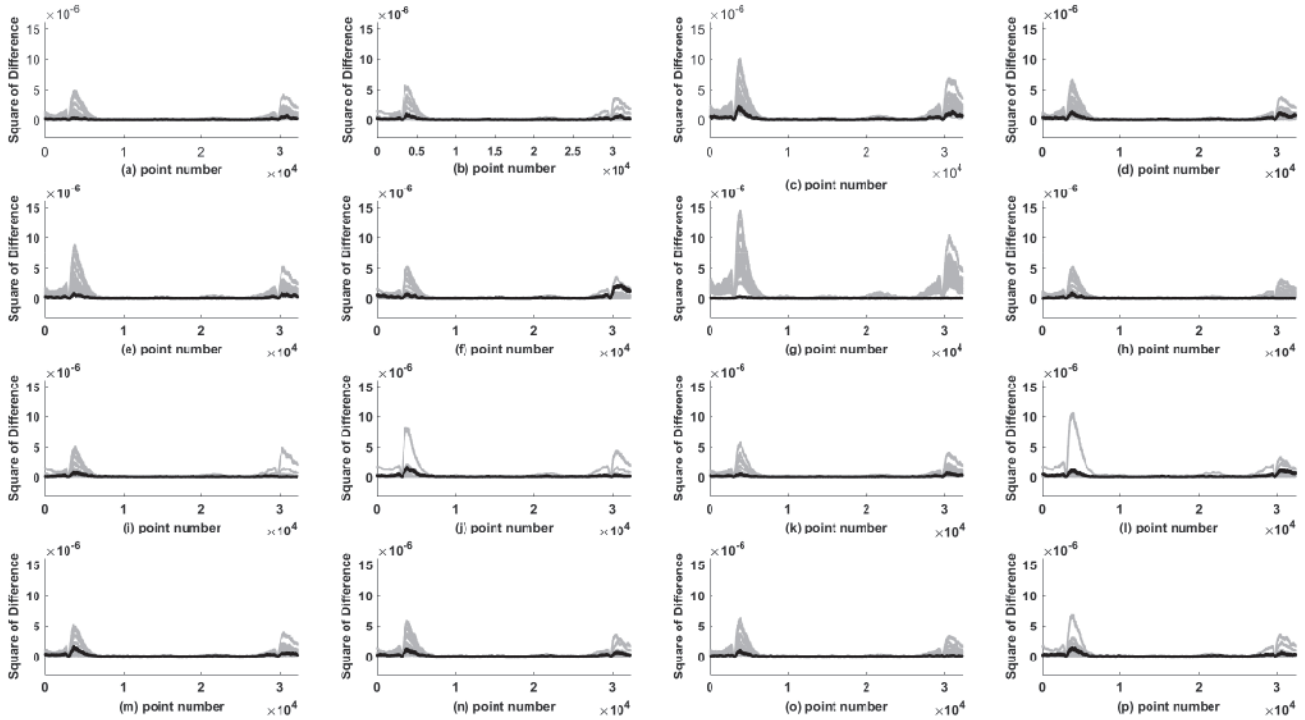


FIGURE 8: Square of difference between each subtrace in set $\{\bar{t}_1^{mj_1}\}_{j_1=0}^{15}$ and all subtraces in set $\{\bar{t}_2^{mj_1}\}_{j_1=0}^{15}$.

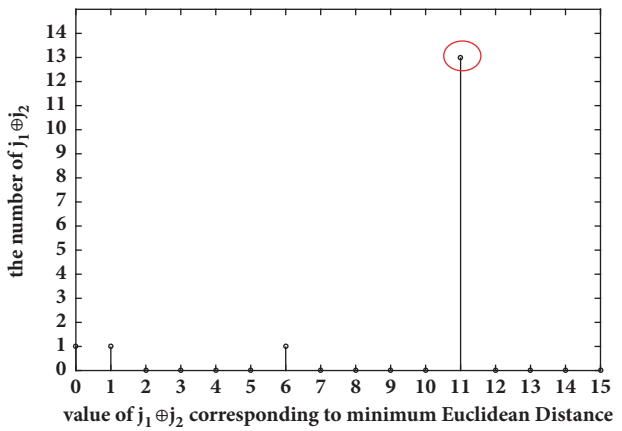


FIGURE 9: Result of the voting stage.

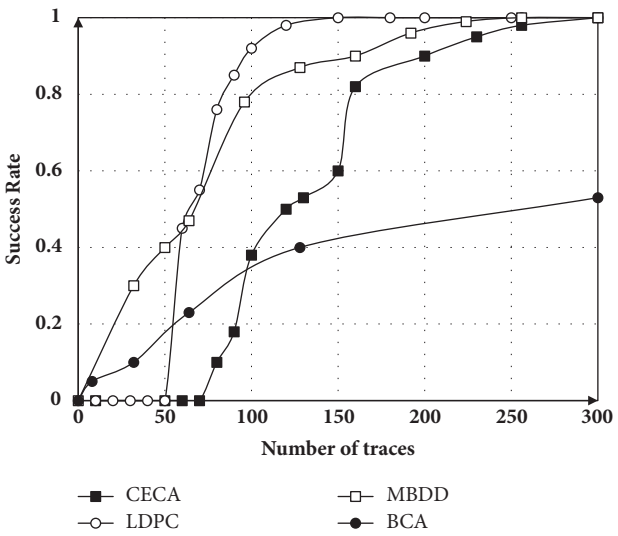


FIGURE 10: Relations between success rate and number of traces.

acquiring power traces as an example, t_s is about 50 times of t_{ave}. The number of power traces used to obtain one averaged power trace in oscilloscope is denoted as q, and the number of saved averaged power traces is n. Therefore, the online time denoted as t_{ol} can be written as

$$t_{ol} = n\left(qt_{ave} + t_s\right) = n\left(0.02q + 1\right)t_s. \tag{14}$$

And we fix $q = 6$ for this experiment, so

$$t_{ol} = 1.12nt_s. \tag{15}$$

Figure 10 presents the relations between success rate and number of traces. As can be seen from Figure 10, LDPC has a better performance. To get a high given success rate, LDPC needs less number of traces. However, in Figure 11, the success rate is as a function of the total online time rather than the number of original power traces. As is mentioned above, we can decrease the online time due to the fact that the time an oscilloscope spends on averaging one trace is much less than saving one trace. It is obvious in Figure 11 that the performance of MBDD with error-tolerant mechanism got a promotion under this setting. Due to the fact that the

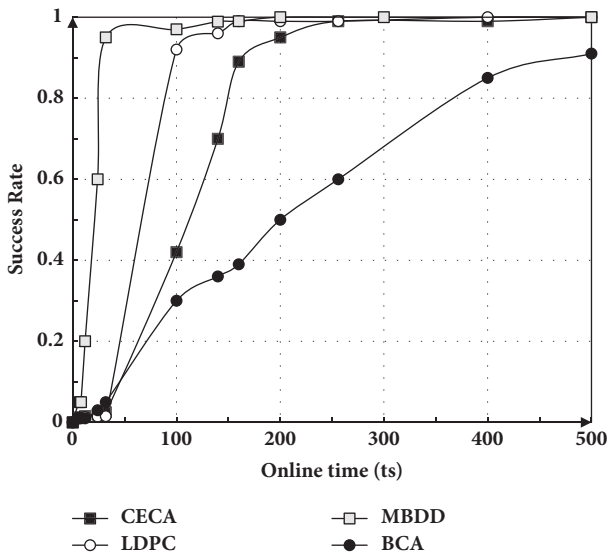


Figure 11: Relations between success rate and online time.

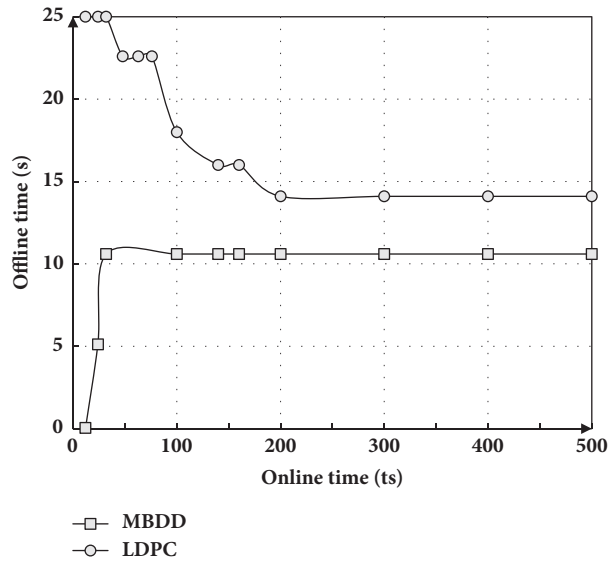


Figure 12: Relations between success rate and online time.

4-bit model of MBDD can find all 120 relations among 16 key bytes with 32 averaged power traces, the fact that the oscilloscope spends less time on averaging traces does a favor for MBDD to have a higher success rate with the same online time. Meanwhile, it seems that LDPC method does not have a remarkable promotion as MBDD with the help of averaging traces. The reason may be that the collision detection method of LDPC will need more averaged traces to detect all collisions occurring among 16 key bytes, even if traces are far from being noisy. However, the results of Figures 10 and 11 can reflect that LDPC is more tolerant to noise because the performance of LDPC in a noisy setting is almost the same as that in a less noisy setting.

Finally, we show the relation between offline time and online time for LDPC and MBDD. The offline time, which reflects the computational complexity, was estimated by MATLAB. As is shown in Figure 12, LDPC is more costly in terms of computation time than MMBD. However, the increased time overhead is slight. For LDPC, the offline time decreases as the online time increases, which indicates that the number of iterations for LDPC decoding decreases. For MBDD, the offline time increases as the online time increases, and it quickly converges to a certain value.

From these comparisons, it can be confirmed that LDPC with soft decision decoding has less trace overhead but more computation time overhead than MBDD, which can be seen as a kind of hard decision decoding procedure. In addition, the necessary number of traces for our method is 90% less than CECA and 96% less than BCA.

6. Conclusion

In this paper, we proposed a basic multiple-bits side-channel collision attack framework based on double distance voting detection. Then an improved version with modified double detection as well as error-tolerant and mechanism is presented. The 4-bit model is proven to be the optimal choice for the novel attack strategy in both theory and practice. Practical attack experiments are performed successfully on a hardware implementation of AES on SAKURA-G circuit board with Xilinx SPARTAN-6. Results show that our detection method performs steadily in noisy environment. We compare our methods with other attacking methods; our method needs less computation time but more traces than LDPC method, and to reach 90% success rate, the necessary number of traces for our method is 90% less than CECA and 96% less than BCA. The novel framework proposed in this paper can be utilized in other cryptographic symmetric algorithms.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Major Program "Core of Electronic Devices, High-End General Chips, and Basis of Software Products" of the Ministry of Industry and Information Technology of China (no. 2014ZX01032205).

References

[1] P. C. Kocher, "Cryptanalysis of Diffie-Hellman, RSA, DSS, and Other Systems Using Timing Attacks," in *Advances in Cryptology—CRYPTO '96*, Lecture Notes in Computer Science, pp. 104–113, Springer, Berlin, Germany, 1996.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 1666, pp. 388–397, 1999.

[3] E. Brier, C. Clavier, and F. Olivier, *Correlation Power Analysis with a Leakage Model International Workshop on Cryptographic*

Hardware and Embedded Systems, Springer, Berlin, Heidelberg, Germany, 2004.

[4] B. Gierlichs et al., *Mutual information analysis: A generic side-channel distinguisher*, Springer-Verlag, 2008.

[5] S. Chari, J. R. Rao, and P. Rohatgi, *Template attacks International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, Berlin, Heidelberg, Germany, 2003.

[6] S. Jin, T. Kim, H. Kim, and S. Hong, "Power Trace Selection Method in Template Profiling Phase for Improvements of Template Attack," *Journal of the Korea Institute of Information Security and Cryptology*, vol. 27, no. 1, pp. 15–23, 2017.

[7] O. Markowitch, L. Lerman, and G. Bontempi, "Side channel attack: an approach based on machine learning," in *Proceedings of the International Workshop on Constructive Side-Channel Analysis and Security Design*, vol. 2011.

[8] H. Dobbertin, "Cryptanalysis of MD4," *Journal of Cryptology*, vol. 11, no. 4, pp. 253–271, 1998.

[9] K. Schramm, T. Wollinger, and C. Paar, "A New Class of Collision Attacks and Its Application to DES," in *Fast Software Encryption*, vol. 2887 of *Lecture Notes in Computer Science*, pp. 206–222, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[10] K. Schramm, G. Leander, P. Felke, and C. Paar, "A Collision-Attack on AES," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, vol. 3156 of *Lecture Notes in Computer Science*, pp. 163–175, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[11] H. Ledig, F. Muller, and F. Valette, "Enhancing Collision Attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, vol. 3156 of *Lecture Notes in Computer Science*, pp. 176–190, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[12] A. Bogdanov, "Improved side-channel collision attacks on AES," in *Proceedings of the 14th International Workshop on Selected Areas in Cryptography*, vol. 4876 of *LNCS*, pp. 84–95, 2007.

[13] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Improved collision-correlation power analysis on first order protected AES," in *Proceedings of the 13th International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2011*, vol. 6917, pp. 49–62, Springer, October 2011.

[14] A. Bogdanov, "Multiple-differential side channel collision attacks on AES," in *Proceedings of the Cryptographic Hardware and Embedded Systems*, vol. 5154 of *LNCS*, pp. 30–40, 2008.

[15] A. Moradi, O. Mischke, and T. Eisenbarth, "Correlation-enhanced power analysis collision attack," in *Cryptographic Hardware and Embedded Systems, CHES 2010*, S. Mangard and F.-X. Standaert, Eds., vol. 6225 of *Lecture Notes in Computer Science*, pp. 125–139, Springer, Berlin, Germany, 2010.

[16] B. Gérard and F. Standaert, "Unified and Optimized Linear Collision Attacks and Their Application in a Non-profiled Setting," in *Cryptographic Hardware and Embedded Systems – CHES 2012*, vol. 7428 of *Lecture Notes in Computer Science*, pp. 175–192, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[17] A. Bogdanov and I. Kizhvatov, "Beyond the limits of DPA: combined side-channel collision attacks," *Institute of Electrical and Electronics Engineers. Transactions on Computers*, vol. 61, no. 8, pp. 1153–1164, 2012.

[18] Y. Ren, L. Wu, and A. Wang, "Double sieve collision attack based on bitwise detection," *KSII Transactions on Internet and Information Systems*, vol. 9, no. 1, pp. 296–308, 2015.

[19] D. Wang and A. Wang, "Bitwise collision attack based on second-order distance," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 3, pp. 1802–1819, 2017.

Hindawi
Security and Communication Networks
Volume 2018, Article ID 3578942, 11 pages
https://doi.org/10.1155/2018/3578942

Research Article

An Efficient Certificateless Generalized Signcryption Scheme

Bo Zhang,[1,2,3] **Zhongtian Jia**,[1,3] **and Chuan Zhao**[1]

[1]*School of Information Science and Engineering, University of Jinan, Jinan 250022, China*
[2]*School of Information Technology, Deakin University, Melbourne, VIC 3125, Australia*
[3]*Shandong Provincial Key Laboratory of Network Based Intelligent Computing, University of Jinan, Jinan 250022, China*

Correspondence should be addressed to Bo Zhang; zhangbosdu@gmail.com

Received 3 November 2017; Revised 2 March 2018; Accepted 5 April 2018; Published 15 May 2018

Academic Editor: Jiankun Hu

Generalized signcryption can adaptively work as an encryption scheme, a signature scheme, or a signcryption scheme with only one algorithm. The paper proposes an efficient certificateless generic signcryption scheme without utilizing bilinear pairing operations. It is proved to satisfy confidentiality and unforgeability against chosen ciphertext and message attacks in an adaptive manner, respectively, in the random oracle model. Due to the lower computational cost and communication overhead, the proposed scheme is suitable for low power and processor devices.

1. Introduction

In the traditional Public Key Infrastructure (PKI), a certificate authority (CA) which is a third party issues the certificates to bind the identity of a user and the corresponding public key. The certificate provides an unforgeable and trusted link by CA's digital signature. However, the problem of certificate management, including the storage, revocation, and distribution of certificates, is complex in this kind of PKI. Identity-based Public Key Cryptosystems (ID-PKC) were introduced by Shamir [1] in 1984 to simplify certificate management problem. A user's public key can be easily derived from arbitrary strings corresponding to his identity information, such as passport number, telephone number, name, and email address. A trusted third party named private key generator (PKG) computes private keys from a master secret and users' identity information and distributes these private keys to users participating in the scheme. This eliminates the need for certificates as used in a traditional PKI. ID-based systems may be a good alternative for certificate-based systems from the viewpoint of efficiency and convenience. But an inherent problem of ID-based cryptosystems is the key escrow; that is, the PKG knows the user's private key, resulting in no user privacy and authenticity. To eliminate these problems simultaneously, Al-Riyami and Paterson introduced the concept of certificateless public key cryptography (CL-PKC) in 2003 [2]. In a CL-PKC, a public/secret key pair is produced by the user himself independently without requiring the public key to be certified. Also, a partial private key is generated by a semi-trusted third party, called the key generation center (KGC), from the unique identifier information of the user. Knowing only one of them should not be able to impersonate the user and carry out any of the cryptographic operations as the user. In other words, CL-PKC can act as an intermediate between traditional PKI and ID-PKC.

The confidentiality and authenticity of messages are the basic requirement for secure communication. In 1997, Zheng proposed a cryptographic primitive signcryption [3], which simultaneously fulfils the integrated function of public encryption and digital signature with a computing and communication cost significantly smaller than that required by the signature-then-encryption method. According to the three public key authentication methods, signcryption can be divided into three types: PKI-based signcryption schemes [4–10], ID-based signcryption schemes (IBSC) [11–16], and certificateless signcryption schemes (CLSC) [17–20].

Sometimes, confidentiality and authenticity are needed separately, and sometimes, both of them are needed simultaneously. We can use encryption, signature, or signcryption to achieve the security properties, respectively. But maintaining three different primitives or components at the same time is quite a burden to a system, especially to the low

power and processor devices in low-bandwidth environments. Generalized signcryption (GSC) was proposed [21–23] to solve this problem. GSC scheme can adaptively work as encryption scheme, signature scheme, or signcryption scheme with only one algorithm. In other words, without any additional modification and computation, it provides double functions when confidentiality and authenticity are required simultaneously and the separate encryption or signature function when one of them is required. So, the GSC scheme can be viewed as a primitive with three work modes. In 2010, the first certificateless generalized signcryption (CLGSC) scheme was introduced by Ji et al. [24]. In their work, the formal definition, security model, and a concrete scheme were proposed. But Kushwah and Lai [25] noted that the scheme [24] is not existentially unforgeable against Type I adversary, and they proposed a new secure and efficient CLGSC scheme. Zhou et al. [26] proposed a more efficient CLGSC scheme based on the certificateless signcryption proposed in [17]. However, all the existing CLGSC schemes are realized with bilinear pairing operations. Compared with other operations, the bilinear pairing operation is much more complicated. Therefore, a concrete scheme without bilinear pairing is more suitable for applications. Very recently, Zhou et al. [27] introduced the key-insulated mechanism into GSC and propose a concrete scheme without bilinear pairings in the certificateless cryptosystem setting.

In this paper, we give a formal definition and the security concept of CLGSC and propose an efficient concrete scheme without utilizing bilinear pairing operations based on a certificateless signcryption-tag key encapsulation mechanism [28]. The concrete scheme is proved to satisfy confidentiality and unforgeability against chosen ciphertext and message attacks in an adaptive manner, respectively, in the random oracle model. Due to less computational cost and communication overhead, the proposed scheme is suitable for low power and processor devices.

The rest of the paper is organized as follows. The security problems, complexity assumptions, and the formal model of CLGSC scheme are introduced in Section 2. We describe a new CLGSC scheme in Section 3 and give the security proof and performance analysis of the new scheme in Sections 4 and 5, respectively. Finally, the conclusions are given in Section 6.

2. Preliminaries

2.1. Security Problems and Complexity Assumptions. Several related mathematical hard problems and security assumptions are presented here.

(i) The Elliptic Curve Discrete Log Problem (ECDLP) [29]: for group G_q which is generated by $P \in G_q$, given $Q \in_R G_q$, to find $x \in \mathbb{Z}_q^*$ such that $Q = xP$.

Definition 1 (ECDLP assumption). For group G_q which is generated by $P \in G_q$, given $Q \in_R G_q$, the successful advantage of any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ is presented as $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{ECDLP}} = \Pr[A(P, Q = xP) = x \mid x \in \mathbb{Z}_q^*]$. If there exists no PPT adversary $\mathcal{A}$ with nonnegligible advantage ϵ in solving the ECDLP problem, we say that the ECDLP assumption holds.

(ii) One-sided Gap Diffie-Hellman problem (ECDLP) [30]: for group G_q which is generated by $P \in G_q$, $Q = xP$ is a fixed point, given $R = yP$, to find xyP with the help of a one-sided decision Diffie-Hellman (ODDH) oracle. The ODDH oracle gets the tuple (P, xP, yP, cP) as the input and outputs 1 if $c = ab(\mathrm{mod}q)$ and 0 otherwise.

Definition 2 (ECDLP assumption). For group G_q which is generated by $P \in G_q$, $Q = xP$ is a fixed point, given $R = yP$. The successful advantage of any PPT adversary $\mathcal{A}$ is presented as $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{ECDLP}} = \Pr[A(P, xP, yP) = xyP \mid x, y \in \mathbb{Z}_q^*]$. If there exists no PPT adversary $\mathcal{A}$ with nonnegligible advantage ϵ by making q_o ODDH oracle queries in solving the ECDLP problem, we say that the (q_o, ϵ)-ECDLP assumption holds.

2.2. Certificateless Generic Signcryption Scheme (CLGSC)

2.2.1. Framework. Certificateless generic signcryption scheme (CLGSC) consists of the following probabilistic polynomial time algorithms.

(1) Setup. Take a security parameter l as input, KGC runs Setup algorithm to generate common parameters *params* and a master key *msk*. *params* are publicly available, whereas the *msk* is kept by the KGC secretly. Formally, we can write

$$(params, msk) \longleftarrow Setup\left(1^l\right). \tag{1}$$

(2) Set-User-Key. Take the common parameters *params* and the identity information *ID* of himself as input; each user runs Set-User-Key algorithm to generate a secure value and the corresponding public key value for himself. It returns the user's secret value x and a corresponding public value PV. Formally, we can write

$$(x, PV) \longleftarrow Set\text{-}User\text{-}Key\left(params, ID\right). \tag{2}$$

(3) Extract-Partial-Private-Key. Given the common parameters *params*, an identity *ID*, and the corresponding public value PV, KGC runs Extract-Partial-Private-Key algorithm to generate the partial private key d associated with *ID*. It distributes d to the user via a secure channel. Formally, we can write

$$d \longleftarrow Extract\text{-}partial\text{-}private\text{-}key\left(params, msk, ID, PV\right). \tag{3}$$

(4) Set-Private-Key. Given the common parameters *params*, the partial private key d, and the secret value x, the user with identity *ID* runs this algorithm to generate the full private key *SK* for himself. Formally, we can write

$$SK \longleftarrow Set\text{-}private\text{-}key\left(params, x, d\right). \tag{4}$$

(5) Set-Public-Key. Given the common parameters *params*, the partial private key d, the secret value x, and the public value PV, the user with identity *ID* runs this algorithm to

generate the full public key PK as the output. Formally, we can write

$$PK \longleftarrow \textit{Set-public-key}\left(params, x, d, PV\right). \quad (5)$$

(6) CLGSC-Signcrypt. Given the common parameters *params*, the message m, the receiver's identity ID_B, and the full public value PK_B, the user with identity ID_A and the full private key SK_A runs this algorithm to generate the ciphertext δ as the output. Note that ID_A and ID_B could be null string. Formally, we can write

$$\left(ID_A, ID_B, \delta\right) \longleftarrow \textit{CLGSC-signcrypt}\left(params, m, ID_A, SK_A, ID_B, PK_B\right). \quad (6)$$

(7) CLGSC-Unsigncrypt. Given the ciphertext δ, the sender's identity ID_A, and the public key PK_A, the receiver with identity ID_B and the full private key SK_B runs this algorithm to unsigncrypt (or decrypt) the ciphertext. It returns m or true for the valid signcryption ciphertext or signature; return $\perp$ means invalid. Note that ID_A and ID_B could be null string. Formally, we can write

$$\left(m/true/\perp\right) \longleftarrow \textit{CLGSC-unsigncrypt}\left(params, \delta, ID_A, PK_A, ID_B, SK_B\right). \quad (7)$$

2.2.2. Security Model. A CLGSC must satisfy confidentiality in encryption mode or signcryption mode and unforgeability in signcryption mode or signature mode. In a CLGSC scheme, we must consider two types of adversaries: a common user of the system and a honest-but-curious KGC. A common user cannot be in possession of the master secret key generated by KGC. But he can replace the public key of the users with valid public keys of his choice in an adaptive manner. This type of adversary is modeled by the Type I adversary. An honest-but-curious KGC knows the KGC's master secret key. But he is not able to replace the public keys of the users. This type of adversary is modeled by the Type II adversary.

An adversary $\mathcal{A}$ can access seven kinds of oracles as follows.

Set-User-Key Queries. $\mathcal{A}$ requests the secret value for a user U with ID_U. $\mathcal{C}$ uses the Set-User-Key algorithm to compute (x_u, PV_U) and sends x_U to $\mathcal{A}$. If U's public key has already been replaced, then a Type I adversary cannot submit U's identity ID_U and requests the secret value of U.

Extract-Partial-Private-Key Queries. $\mathcal{A}$ requests the partial private key for a user U with ID_U; $\mathcal{C}$ uses the Set-User-Key algorithm to compute (x_u, PV_U) and then sends a partial private key d_U generated by the Extract-Partial-Private-Key algorithm to $\mathcal{A}$.

Set-Private-Key Queries. $\mathcal{A}$ requests the private key for a user U with ID_U; $\mathcal{C}$ sends the full private key SK_U generated by the Set-User-Key algorithm and Extract-Partial-Private-Key algorithm to $\mathcal{A}$. Note that if U's public key has already been replaced, then a Type I adversary cannot submit the identity ID_U and requests the full private key of U.

Set-Public-Key Queries. $\mathcal{A}$ requests the public key for a user U with ID_U; $\mathcal{C}$ returns the public key PK_U to $\mathcal{A}$ generated by the Set-User-Key algorithm and Extract-Partial-Private-Key algorithm.

Public-Key-Replacement Queries. $\mathcal{A}$ computes a new public key PK'_U for ID_U and replaces PK_U. Note that a Type II adversary cannot access Public-Key-Replacement queries.

CLGSC-Signcrypt Queries. $\mathcal{A}$ submits (ID_A, ID_B, m) to $\mathcal{C}$, in which m is a message and ID_A and ID_B are the sender's and the receiver's identities, respectively. $\mathcal{C}$ returns the ciphertext δ to $\mathcal{A}$. Note that if the public key of the sender has been replaced, then $\mathcal{C}$ may not return the ciphertext δ. In this case, $\mathcal{A}$ must provide the secret value to $\mathcal{C}$.

CLGSC-Unsigncrypt Queries. $\mathcal{A}$ submits (ID_A, ID_B, δ) to $\mathcal{C}$, in which δ is a signature or signcryption ciphertext and ID_A and ID_B are the sender's and the receiver's identities, respectively. $\mathcal{C}$ returns the output of CLGSC-unsigncrypt to $\mathcal{A}$. Note that if the public key of the receiver is replaced, then $\mathcal{C}$ may not return the corresponding value. In this case, $\mathcal{A}$ must provide the secret value to $\mathcal{C}$.

Confidentiality

Definition 3 (IND-CLGSC-CCA2 confidentiality). A certificateless generic signcryption scheme in signcryption mode or encryption mode is semantically secure against adaptive chosen ciphertext attacks if, for all PPT adversary, the advantage is negligible in the following games. The games are played between a challenger $\mathcal{C}$ and the adversaries $\mathcal{A}_I$ and $\mathcal{A}_{II}$, respectively.

GAME 1 (IND-CLGSC-CCA2-I)

Initial. $\mathcal{C}$ generates the system parameters *params* and the master secret key *msk* by running the Setup algorithm. It keeps *msk* secret and sends *params* to $\mathcal{A}_I$.

Phase I. $\mathcal{A}_I$ performs a polynomially bounded number of the above queries.

Challenge. $\mathcal{A}_I$ outputs a tuple (m_0, m_1, ID_A, ID_B), in which m_0 and m_1 are distinct messages of equal length and ID_A and ID_B are the sender's and the receiver's identities, respectively. Here, it is to be noted that ID_B's full private key has not been extracted by $\mathcal{A}_I$ in Phase I. It is also to be noted that ID_B's partial private key has not been extracted and his public key has not been replaced simultaneously. $\mathcal{C}$ picks $\gamma \in \{0, 1\}$ randomly, runs the algorithm of CLGSC-signcrypt with (ID_A, ID_B, m_γ), and sends the output δ to $\mathcal{A}_I$.

Phase II. $\mathcal{A}_I$ asks queries adaptively again. However, the full private key for ID_B may not be extracted by $\mathcal{A}_I$ and the partial private key for ID_B may not be extracted if the public key of ID_B has been replaced in Phase I. Only after the public key

PK_A or PK_B has been replaced, CLGSC-unsigncrypt query on δ with sender ID_A and receiver ID_B is allowed.

Guess Stage. $\mathscr{A}_I$ outputs his guess γ' and if $\gamma' = \gamma$ he wins the game.

The advantage of $\mathscr{A}_I$ is $Adv_{\mathscr{A}_I}^{IND\text{-}CLGSC\text{-}CCA2\text{-}I} = 2\Pr[\gamma' = \gamma] - 1$.

GAME 2 (IND-CLGSC-CCA2-II)

Initial. $\mathscr{C}$ generates *params* and *msk* by running the Setup algorithm. It sends *params* and *msk* to $\mathscr{A}_{II}$.

Phase I. $\mathscr{A}_{II}$ performs a polynomially bounded number of queries just as $\mathscr{A}_I$ in IND-CLGSC-CCA2-*I* game. Extract-Partial-Private-Key queries are not included here, because $\mathscr{A}_{II}$ knows *msk*, and he can generate users' partial private keys by himself.

Challenge. At the end of Phase *I*, $\mathscr{A}_{II}$ outputs a tuple (m_0, m_1, ID_A, ID_B), in which m_0 and m_1 are distinct messages of equal length and ID_A and ID_B are the sender's and the receiver's identities, respectively. Here, it is to be noted that $\mathscr{A}_{II}$ must have made no Set-Private-Key queries on ID_B in Phase *I*. $\mathscr{C}$ picks $\gamma \in \{0, 1\}$ randomly, runs the algorithm of CLGSC-signcrypt with (m_γ, ID_A, ID_B), and sends the output δ to $\mathscr{A}_{II}$.

Phase II. $\mathscr{A}_{II}$ asks queries adaptively again. However, the full private key for ID_B may not be extracted and only after the public key PK_A or PK_B has been replaced, CLGSC-unsigncrypt query on δ with sender ID_A and receiver ID_B is allowed.

Guess Stage. $\mathscr{A}_{II}$ outputs his guess γ' and if $\gamma' = \gamma$ he wins the game.

The advantage of $\mathscr{A}_{II}$ is $Adv_{\mathscr{A}_{II}}^{IND\text{-}CLGSC\text{-}CCA2\text{-}II} = 2\Pr[\gamma' = \gamma] - 1$.

Note that, in the above games, only the signcryption mode and encryption mode of the CLGSC scheme must be considered. The receiver's identity ID_B cannot be vacant. If the sender's identity ID_A is not vacant, the algorithm runs in signcryption mode; otherwise it runs in encryption mode.

Unforgeability

Definition 4 (EUF-CLGSC-CMA unforgeability). A certificateless generic signcryption scheme in signature mode or signcryption mode is existentially unforgeable against adaptive chosen message attacks if, for all PPT adversary, the advantage is negligible in the following games. The games are played between a challenger $\mathscr{C}$ and the adversaries $\mathscr{F}_I$ and $\mathscr{F}_{II}$, respectively.

GAME 3 (EUF-CLGSC-CMA-I)

Initial. $\mathscr{C}$ generates *params* and *msk* by running the Setup algorithm. It keeps *msk* secret and sends *params* to $\mathscr{F}_I$.

Training Phase. Like $\mathscr{A}_I$ in Phase *I* of the IND-CLGSC-CCA2-*I* game, $\mathscr{F}_I$ may perform a series of adaptive queries.

Forgery. $\mathscr{F}_I$ outputs a tuple $(ID_A^*, ID_B^*, \delta^*)$. It must not be an output of the CLGSC-signcrypt query. The full private key of ID_A^* must not be extracted by $\mathscr{F}_I$ during the Training Phase. Moreover, $\mathscr{F}_I$ must have not replaced ID_A^*'s public key and extracted ID_A^*'s partial private key simultaneously. If the output of CLGSC-unsigncrypt is not $\perp$, $\mathscr{F}_I$ wins the game.

GAME 4 (EUF-CLGSC-CMA-II)

Initial. $\mathscr{C}$ generates *params* and *msk* by running the Setup algorithm. It sends *params* and *msk* to $\mathscr{F}_{II}$.

Training Phase. Like $\mathscr{A}_{II}$ in Phase *I* of the IND-CLGSC-CCA2-*II* game, $\mathscr{F}_{II}$ may perform a series of adaptive queries.

Forgery. $\mathscr{F}_{II}$ outputs a tuple $(ID_A^*, ID_B^*, \delta^*)$. It must not be an output of the CLGSC-signcrypt query. During the Training Phase, $\mathscr{F}_{II}$ must have made no Set-Private-Key queries and Set-User-Key queries on ID_A^*. If the output of CLGSC-unsigncrypt is not $\perp$, $\mathscr{F}_{II}$ wins the game.

Note that, in the above games, only the signcryption mode and signature mode of the CLGSC scheme must be considered. The sender's identity ID_A cannot be vacant. If the receiver's identity ID_B is not vacant, the algorithm runs in signcryption mode; otherwise it runs in signature mode.

3. The Concrete Scheme

Motivated by the pairing-free CLSC-TKEM protocol, in this section, we present a novel certificateless generalized signcryption scheme. It consists of seven algorithms.

(1) Setup. Given a security parameters $l \in \mathbb{Z}^+$, the KGC executes the following operations:

(i) It chooses a l-bits prime q and the tuple $\{F_q, E/F_q, G_q, P\}$, where G_q is generated by P.

(ii) It chooses $s \in_R \mathbb{Z}_q^*$ uniformly as the master key and computes $P_{pub} = s \cdot P$.

(iii) Let H_0, H_1, H_2 be cryptography hash functions, where $H_0 : \{0,1\}^* \times G_q^2 \to \mathbb{Z}_q^*$, $H_1 : G_q^3 \times \{0,1\}^* \to \{0,1\}^w$, $H_2 : \{0,1\}^* \times \{0,1\}^* \times G_q^3 \to \mathbb{Z}_q^*$, where w is the plaintext block length.

(iv) Define an index function $f(ID)$ as follows: $f(ID) = 0$ if $ID = \emptyset$; otherwise, $f(ID) = 1$.

(v) The public parameters and functions are presented as $params = \{F_q, E/F_q, G_q, P, P_{pub}, H_0, H_1, H_2\}$.

(2) Set-User-Key. A user **U** with the identity ID_U randomly chooses $x_U \in \mathbb{Z}_q^*$ as its secret value and computes the corresponding public value as $PV_U = x_U \cdot P$.

(3) Extract-Partial-Private-Key. **U** sends (ID_U, PV_U) to the KGC. In turn, the KGC generates and returns the partial private key of **U** as follows:

(i) It chooses $r_U \in \mathbb{Z}_q^*$ and computes $R_U = r_U \cdot P$.

(ii) It computes $d_U = r_U + s \cdot H_0(ID_U, R_U, PV_U) \bmod q$.

d_U is the partial private key of **U**. **U** can accept d_U as a valid partial private key by determining if $d_U \cdot P = R_U + H_0(ID_U, R_U, PV_U) \cdot P_{pub}$ holds.

(4) Set-Private-Key. The user **U** takes the pair (x_U, d_U) as its full private key SK_U.

(5) Set-Public-Key. The user **U** takes the pair (PV_U, R_U) as its full public key PK_U.

(6) CLGSC-Signcrypt. With the message m and the receiver's identity ID_B, the sender **A** performs as follows:

(i) It chooses $s_A \in \mathbb{Z}_q^*$ randomly and computes $V = s_A \cdot P$.

(ii) It computes $f(ID_A)$, $f(ID_B)$.

(iii) It computes $c = H_1(V, T, s_A PV_B, ID_B) \cdot f(ID_B) \oplus m$, where $T = s_A \cdot (H_0(ID_B, R_B, PV_B) \cdot P_{pub} + R_B + PV_B)$.

(iv) It computes $W = (d_A + x_A \cdot H_2(ID_A, m, V, T, PV_A) + s_A \cdot H_2(ID_A, m, V, T, R_A)) \cdot f(ID_A)$.

(v) It sets $\delta = (V, W, c)$ and returns (ID_A, ID_B, δ) as the ciphertext.

(7) CLGSC-Unsigncrypt. Given the ciphertext $(ID_A, ID_B, \delta = (V, W, c))$, the receiver ID_B decrypts and verifies the ciphertext as follows:

(i) It computes $f(ID_A)$, $f(ID_B)$.

(ii) It computes $m' = c \oplus (H_1(V, T', x_B V, ID_B) \cdot f(ID_B))$, where $T' = V \cdot (x_B + d_B)$.

(iii) It checks if $W \cdot P = f(ID_A)(R_A + H_0(ID_A, R_A, PV_A) \cdot P_{pub} + PV_A \cdot H_2(ID_A, m', V, T', PV_A) + V \cdot H_2(ID_A, m', V, T', R_A))$. If the equation does not hold, then return $\perp$ indicating the message is not valid. Otherwise, return true when $f(ID_B) = 0$ indicating it is a valid signature of user **A** or m indicating it is a valid encryption/signcryption ciphertext of the message m sent to user **B**.

Correctness of the Scheme. The correctness of the proposed concrete scheme is proved as follows.

(i) Correctness of the Encryption

$$\begin{aligned} m' &= c \oplus H_1\left(V, T', x_B V, ID_B\right) = c \oplus H_1\left(V, V\right. \\ &\cdot \left(x_B + d_B\right), x_B s_A P, ID_B\big) = c \oplus H_1\left(V, s_A\right. \\ &\cdot \left(x_B \cdot P + d_B \cdot P\right), s_A x_B P, ID_B\big) = c \oplus H_1\left(V, s_A\right. \\ &\cdot \left(PV_B + R_B + H_0\left(ID_B, R_B, PV_B\right) \cdot P_{pub}\right), s_A PV_B, \\ &ID_B\big) = c \oplus H_1\left(V, T, s_A PV_B, ID_B\right) = H_1\left(V, T,\right. \\ &\left. s_A PV_B, ID_B\right) \oplus m \oplus H_1\left(V, T, s_A PV_B, ID_B\right) = m. \end{aligned} \tag{8}$$

(ii) Correctness of the Signature

$$\begin{aligned} W \cdot P &= \left(d_A + x_A \cdot H_2\left(ID_A, m, V, T, PV_A\right) + s_A\right. \\ &\cdot H_2\left(ID_A, m, V, T, R_A\right)) \cdot P = d_A \cdot P + x_A \cdot P \\ &\cdot H_2\left(ID_A, m, V, T, PV_A\right) + s_A \cdot P \\ &\cdot H_2\left(ID_A, m, V, T, R_A\right) = r_A \cdot P + s \cdot P \\ &\cdot H_0\left(ID_A, R_A, PV_A\right) + x_A \cdot P \\ &\cdot H_2\left(ID_A, m, V, T, PV_A\right) + s_A \cdot P \\ &\cdot H_2\left(ID_A, m, V, T, R_A\right) = R_A + H_0\left(ID_A, R_A, PV_A\right) \\ &\cdot P_{pub} + PV_A \cdot H_2\left(ID_A, m, V, T, PV_A\right) + V \\ &\cdot H_2\left(ID_A, m, V, T, R_A\right). \end{aligned} \tag{9}$$

4. Security Analysis of the Proposed Scheme

In this section, the security of the proposed concrete CLGSC scheme is proved as follows.

4.1. Confidentiality

Theorem 5. *The CLGSC scheme is semantically secure against adaptive chosen ciphertext attacks in encryption mode or signcryption mode in the random oracle model.*

Theorem 5 is proved based on Lemmas 6 and 7.

Lemma 6. *If an adversary $\mathcal{A}_I$ has a nonnegligible advantage ε against the IND-CLGSC-CCA2-I security of our scheme and performing q_{H_i} queries to oracles H_i $(i = 0, 1, 2)$, q_{ppk} Extract-Partial-Private-Key queries, and q_{sk} Set-Private-Key queries, then there is an algorithm that solves the ECDLP problem with probability $\varepsilon \cdot (1 - q_{ppk}/q_{H_0}) \cdot (1 - q_{sk}/q_{H_0}) \cdot (1/(q_{H_0} - q_{ppk} - q_{sk})) \cdot 1/q_{H_1}$.*

Proof. Given an instance of the ECDLP problem, for group G_q generated by P and a fixed second point $Q(= aP)$ having as input $R(= bP) \in G_q$, $\mathcal{C}$ has to compute for the point $S(= cP) \in G_q$ such that $c = ab(\bmod q)$ with the help of a ODDH oracle. Suppose the IND-CLGSC-CCA2-*I* security of the CLGSC can be violated by a Type *I* adversary $\mathcal{A}_I$. $\mathcal{C}$ can utilize $\mathcal{A}_I$ to compute abP as the solution to this instance by the following interactive game.

$\mathcal{C}$ chooses $s \in_R \mathbb{Z}_q^*$ uniformly as the master key and computes $P_{pub} = s \cdot P$. It sends *params* to $\mathcal{A}_I$ and maintains lists L_i $(0 \le i \le 2)$ to keep the consistency between the responses to the hash queries and a list L_k of issued keys which are initially empty. $\mathcal{C}$ selects t randomly, where $1 \le t \le q_{H_0}$, and takes ID_t as the target identity. $\mathcal{C}$ chooses $e_t, x_t \in_R \mathbb{Z}_q^*$ and sets $H_0(ID_t, R_t, PV_t) = -e_t$, $R_t = e_t P_{pub} + aP - x_t P$, and $PV_t = x_t P$. $\mathcal{C}$ inserts $(ID_t, R_t, PV_t, -e_t)$ into the list L_0 and $(ID_t, \perp, x_t, R_t, PV_t)$ into the list L_k.

$\mathcal{C}$ answers $\mathcal{A}_I$'s queries to random oracles H_i $(0 \le i \le 2)$ as follows.

(i) H_0 *queries*: when $\mathscr{A}_I$ submits a H_0 query with (ID_i, R_i, PV_i) for some $i \in [1, q_0]$, $\mathscr{C}$ checks in L_0, and if $(ID_i, R_i, PV_i, -e_i)$ exists, $\mathscr{C}$ returns $-e_i$. Otherwise, $\mathscr{C}$ chooses $e_i \in_R \mathbb{Z}_q^*$ and returns $-e_i$ to $\mathscr{A}_I$. Then $\mathscr{C}$ inserts $(ID_i, R_i, PV_i, -e_i)$ into the list L_0.

(ii) H_1 *queries*: when $\mathscr{A}_I$ submits a H_1 query with (V_i, T_i, Y_i, ID_i) for some $i \in [1, q_1]$, $\mathscr{C}$ sets the tuple (aP, V_i, T_i) as the input of ODDH orale. If the output of ODDH oracle is 1, then $\mathscr{C}$ returns T_i as the solution abP and stops; else $\mathscr{C}$ searches in L_1, if $(V_i, *, Y_i, ID_i, l_i)$ exists, it replaces the symbol $*$ with T_i and returns l_i. Otherwise, $\mathscr{C}$ chooses $l_i \in_R \{0,1\}^n$ and returns l_i to $\mathscr{A}_I$. Then $\mathscr{C}$ inserts $(V_i, T_i, Y_i, ID_i, l_i)$ into the list L_1.

(iii) H_2 *queries*: when $\mathscr{A}_I$ submits a H_2 query with $(ID_i, m_i, V_i, T_i, PV_i/R_i)$ for some $i \in [1, q_2]$, $\mathscr{C}$ checks in L_2, and if $(ID_i, m_i, V_i, T_i, PV_i/R_i, h_i)$ exists in L_2, $\mathscr{C}$ returns h_i. Otherwise, $\mathscr{C}$ chooses $h_i \in_R \mathbb{Z}_q^*$ and returns h_i to $\mathscr{A}_I$. Then $\mathscr{C}$ inserts $(ID_i, m_i, V_i, T_i, PV_i/R_i, h_i)$ into L_2.

$\mathscr{C}$ can answer $\mathscr{A}_I$'s other queries as follows.

Phase I

(i) Set-User-Key Queries. $\mathscr{A}_I$ requests a secret value of the user with ID_i. If the public key of ID_i has not been replaced, then $\mathscr{C}$ responds with x_i by retrieving from the list L_k.

(ii) Extract-Partial-Private-Key Queries. $\mathscr{A}_I$ requests the partial private key of a user with ID_i. If $ID_i = ID_t$, $\mathscr{C}$ aborts the execution. Otherwise, $\mathscr{C}$ checks in L_k, and if $(ID_i, d_i, x_i, R_i, PV_i)$ exists, $\mathscr{C}$ returns d_i. Otherwise, $\mathscr{C}$ computes the partial private key of ID_i by using the actual Extract-Partial-Private-Key algorithm, and $\mathscr{C}$ inserts $(ID_i, d_i, x_t, R_i, PV_i)$ into the list L_k and returns d_i.

(iii) Set-Private-Key Queries. $\mathscr{A}_I$ requests a user's full private key with ID_i. $\mathscr{C}$ aborts the execution when $ID_i = ID_t$. Otherwise, $\mathscr{C}$ checks in L_k, and if $(ID_i, d_i, x_i, R_i, PV_i)$ exists, $\mathscr{C}$ returns the corresponding private key (x_i, d_i). Otherwise, $\mathscr{C}$ picks $e_i, b_i, x_i \in_R \mathbb{Z}_q^*$, then sets $H_0(ID_i, R_i, PV_i) = -e_i$, $R_i = e_i P_{pub} + b_i P$, and computes $PV_i = x_i P \cdot d_i = b_i$ and it satisfies the equation $d_i P = R_i + H_0(ID_i, R_i, PV_i) P_{pub}$. $\mathscr{C}$ returns (x_i, d_i) to $\mathscr{A}_I$ and inserts $(ID_i, R_i, PV_i, -e_i)$ into the list L_0 and $(ID_i, d_i, x_i, R_i, PV_i)$ into L_k.

(iv) Set-Public-Key Queries. $\mathscr{A}_I$ requests a user's public key with ID_i. $\mathscr{C}$ checks in L_k, and if $(ID_i, d_i, x_i, R_i, PV_i)$ exists, $\mathscr{C}$ returns the corresponding public key (R_i, PV_i). Otherwise, $\mathscr{C}$ picks $e_i, b_i, x_i \in_R \mathbb{Z}_q^*$, then sets $H_0(ID_i, R_i, PV_i) = -e_i$, $R_i = e_i P_{pub} + b_i P$, and computes the public key as $PV_i = x_i P \cdot d_i = b_i$ and it satisfies the equation $d_i P = R_i + H_0(ID_i, R_i, PV_i) P_{pub}$. $\mathscr{C}$ returns (PV_i, R_i) to $\mathscr{A}_I$ and inserts $(ID_i, R_i, PV_i, -e_i)$ into the list L_0 and $(ID_i, d_i, x_i, R_i, PV_i)$ into L_k.

(v) Public-Key-Replacement Queries. $\mathscr{A}_I$ requests to replace a user's public key (R_i, PV_i) with chosen values (ID_i, R_i', PV_i'). $\mathscr{C}$ updates corresponding tuple with $(ID_i, -, -, R_i', PV_i')$.

(vi) CLGSC-Signcrypt Queries. $\mathscr{A}_I$ sends the tuple $(ID_A, PV_A, R_A, ID_B, PV_B, R_B, m)$ to $\mathscr{C}$. For each query (ID_A, ID_B), if $ID_A \neq ID_t$, $\mathscr{C}$ executes the Set-Private-Key algorithm to compute SK_A corresponding to ID_A. Then, $\mathscr{C}$ gets the ciphertext δ by running the actual CLGSC-signcrypt algorithm. $\mathscr{C}$ sends δ to $\mathscr{A}_I$. If $ID_A = ID_t$ (and hence, $ID_B \neq ID_t$), $\mathscr{C}$ can obtain the full private key SK_B corresponding to ID_B. $\mathscr{C}$ computes $V = rP - h_t^{-1} \cdot ap$, $T = V \cdot (x_B + d_B)$, sets $H_2(ID_A, m, V, T, PV_A) = h_t'$, $H_2(ID_A, m, V, T, R_A) = h_t$, and adds the tuples $(ID_A, m, V, T, PV_A, h_t')$ and $(ID_A, m, V, T, R_A, h_t)$ to the list L_2 in which $r, h_t, h_t' \in_R \mathbb{Z}_q^*$. $\mathscr{C}$ computes $c = H_1(V, T, s_B V, ID_B) \cdot f(ID_B) \oplus m$ and $W = r \cdot h_t + x_A \cdot h_t' - x_A$. $\mathscr{C}$ outputs $(ID_A, ID_B, \delta = (V, W, c))$ as the ciphertext.

The tuple $(ID_A, ID_B, \delta = (V, W, c))$ can pass the verification as the valid ciphertext because the equality holds as follows:

$$\begin{aligned} & R_A + H_0(ID_A, R_A, PV_A) \cdot P_{pub} + PV_A \\ & \quad \cdot H_2(ID_A, m, V, T, PV_A) + V \\ & \quad \cdot H_2(ID_A, m, V, T, R_A) = e_t P_{pub} + aP - x_t P \\ & \quad + (-e_t) P_{pub} + x_t P \cdot h_t' + \left(rP - h_t^{-1} aP\right) \cdot h_t \\ & \quad = \left(r \cdot h_t + x_t h_t' - x_t\right) \cdot P = W \cdot P. \end{aligned} \tag{10}$$

(vii) CLGSC-Unsigncrypt Queries. $\mathscr{A}_I$ submits (ID_A, ID_B, δ) to $\mathscr{C}$. If $ID_B \neq ID_t$, $\mathscr{C}$ obtains the receiver's private key and returns the output of CLGSC-unsigncrypt algorithm to $\mathscr{A}_I$. Note that if the receiver's public value is replaced, $\mathscr{C}$ may not obtain the receiver's secret value. In this case, receiver's secret value is requested to be provided by $\mathscr{A}$. Otherwise, $\mathscr{C}$ searches in L_2 for $(ID_A, m, V, T, PV_A, h_i)$ and $(ID_A, m, V, T, R_A, h_i')$. If the entries exist and the equality $W \cdot P = R_A + H_0(ID_A, R_A, PV_A) \cdot P_{pub} + PV_A \cdot h_i + V \cdot h_i'$ holds, T is retrieved. If $\mathscr{C}$ can find a tuple (V, T, Y, ID_B, l) in L_1 making the ODDH oracle return 1 when queries are on (ap, V, T), then the message is $c \oplus l$.

Challenge. $\mathscr{A}_I$ submits $(m_0, m_1, ID_A^*, ID_B^*)$ in which m_0 and m_1 are distinct messages of equal length and ID_A^* and ID_B^* are the sender's and the receiver's identities, respectively. Here, it is to be noted that $\mathscr{A}_I$ must have made no Set-Private-Key queries on ID_B^* in Phase *I*. It is also to be noted that ID_B^*'s partial private key has not been extracted and his public key has not been replaced simultaneously. $\mathscr{C}$ aborts the game if $ID_B^* \neq ID_t$. Otherwise, $\mathscr{C}$ generates the challenge ciphertext as follows.

(1) It sets $V^* = bP$ and chooses $T^* \in_R G_q$.

(2) It selects randomly a bit $\gamma \in \{0,1\}$ and a random hash value h and sets $c^* = m_\gamma \oplus h$.

(3) It selects W^*, satisfies the equation $W^* P = R_A + H_0(ID_A, R_A, PV_A) \cdot P_{pub} + PV_A \cdot H_2(ID_A, m, V^*, T^*, PV_A) + V^* \cdot H_2(ID_A, m, V^*, T^*, R_A)$, and sends $\delta^* = (V^*, W^*, c^*)$ to $\mathscr{A}_I$.

Phase II. $\mathscr{A}_I$ asks queries adaptively again. In addition, the full private key for ID_B^* may not be extracted by $\mathscr{A}_I$ and the partial

private key for ID_B^* may not be extracted if the public key of ID_B^* has been replaced in Phase *I*. Only after the public key PK_A^* or PK_B^* has been replaced, CLGSC-unsigncrypt query on δ with sender ID_A^* and receiver ID_B^* is allowed.

Guess. Since $\mathcal{A}_I$ is able to break the IND-CLGSC-CCA2-*I* security of the CLGSC, a H_1 query with (V^*, T^*, Y^*, ID_B^*) should have been asked. Note that $T^* = b \cdot (H_0(ID_B, R_B, PV_B) \cdot P_{pub} + R_B + PV_B) = abP$. Therefore, one of the T's in L_1 is the ECDLP problem's solution. $\mathcal{C}$ chooses one T randomly and outputs it as the solution.

In the above challenge query, the senders ID_A and ID_A^* can be $\emptyset$ for the encryption mode; otherwise, it works as signcryption. Thus, the proof is suitable for the two modes.

Analysis. Lets E_1, E_2, and E_3 be the events when $\mathcal{C}$ aborts this game.

(i) E_1 is an event in which the target identity ID_t's partial private key is queried by $\mathcal{A}_I$. The probability of E_1 is $\Pr[E_1] = q_{ppk}/q_{H_0}$.

(ii) E_2 is an event in which the target identity ID_t's private key is queried by $\mathcal{A}_I$. The probability of E_2 is $\Pr[E_2] = q_{sk}/q_{H_0}$.

(iii) E_3 is an event in which the target identity ID_t has not been chosen as the receiver by $\mathcal{A}_I$ during the challenge phase. The probability of E_3 is $\Pr[E_3] = 1 - 1/(q_{H_0} - q_{ppk} - q_{sk})$.

Thus, $\mathcal{C}$ does not abort this game's probability of $\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3] = (1 - q_{ppk}/q_{H_0}) \cdot (1 - q_{sk}/q_{H_0}) \cdot (1/(q_{H_0} - q_{ppk} - q_{sk}))$.

$\mathcal{C}$ chooses the solution of ECDLP problem from L_1's probability of $1/q_{H_1}$. So, the successful advantage of $\mathcal{C}$ is $ADV_{\mathcal{C}}^{OGDH} = \Pr[\mathcal{C}(P, aP, bP) = abP] = \varepsilon \cdot (1 - q_{ppk}/q_{H_0}) \cdot (1 - q_{sk}/q_{H_0}) \cdot (1/(q_{H_0} - q_{ppk} - q_{sk})) \cdot 1/q_{H_1}$. □

Lemma 7. *If $\mathcal{A}_{II}$ has nonnegligible advantage ε against the IND-CLGSC-CCA2-II security of our scheme and performing q_{sv} Extract-Secret-Value queries, q_{sk} Set-Private-Key queries, and q_{H_i} queries to oracles H_i $(i = 0, 1, 2)$, then there is an algorithm that solves the ECDLP problem with probability $\varepsilon \cdot (1 - q_{sv}/q_{H_0}) \cdot (1 - q_{sk}/q_{H_0}) \cdot (1/(q_{H_0} - q_{sk} - q_{sv}))$.*

Proof. Given an instance of the ECDLP problem, for group G_q generated by P and a fixed second point $Q(= aP)$ having as input $R(= bP) \in G_q$, $\mathcal{C}$ has to compute for the point $S(= cP) \in G_q$ such that $c = ab(\mathrm{mod}\, q)$ with the help of a ODDH oracle. Suppose the IND-CLGSC-CCA2-*II* security of the CLGSC can be violated by a Type *II* adversary $\mathcal{A}_{II}$. $\mathcal{C}$ can utilize $\mathcal{A}_{II}$ to compute abP as the solution to this instance by the following interactive game.

$\mathcal{C}$ chooses $s \in_R \mathbb{Z}_q^*$ uniformly as the master key and computes $P_{pub} = s \cdot P$. $\mathcal{C}$ sends *params* and s to $\mathcal{A}_{II}$ and maintains lists L_i $(0 \le i \le 2)$ to avoid the inconsistency between the responses to the hash queries and a list L_k of issued keys which is initially empty. $\mathcal{C}$ selects t randomly, where $1 \le t \le q_{H_0}$, and fixes ID_t as the target identity. $\mathcal{C}$ chooses $a_t, l_t \in_R \mathbb{Z}_q^*$, sets $H_0(ID_t, R_t, PV_t) = l_t$, and computes $R_t = a_t P, d_t = a_t + l_t \cdot s$, and the public key as $PV_t = aP$. $\mathcal{C}$ inserts (ID_t, R_t, PV_t, l_t) into the list L_0 and $(ID_t, d_t, \perp, R_t, PV_t)$ into the list L_k.

$\mathcal{C}$ answers $\mathcal{A}_{II}$'s queries to random oracles H_i $(0 \le i \le 2)$ as follows:

(i) H_0 *queries*: when $\mathcal{A}_{II}$ submits a H_0 query with (ID_i, R_i, PV_i) for some $i \in [1, q_0]$, $\mathcal{C}$ checks if there exists a tuple (ID_i, R_i, PV_i, l_i) in L_0. If such a tuple exists, $\mathcal{C}$ answers with l_i. Otherwise, $\mathcal{C}$ chooses $l_i \in_R \mathbb{Z}_q^*$ and returns l_i as the answer. Then $\mathcal{C}$ inserts (ID_i, R_i, PV_i, l_i) into the list L_0.

(ii) H_1 *queries*: when $\mathcal{A}_{II}$ submits a H_1 query with (V_i, T_i, Y_i, ID_i), where $i \in [1, q_1]$, $\mathcal{C}$ sets the tuple (aP, V_i, Y_i) as the input of ODDH oracle. If the output of ODDH oracle is 1, then $\mathcal{C}$ outputs Y_i as the solution; else $\mathcal{C}$ searches L_1 with entries $(V_i, T_i, *, ID_i, l_i)$. If such a tuple exists, it replaces the symbol $*$ with Y_i and returns l_i. Otherwise, $\mathcal{C}$ chooses $l_i \in_R \{0, 1\}^n$, inserts $(V_i, T_i, Y_i, ID_i, l_i)$ into L_1, and returns l_i to $\mathcal{A}_{II}$.

(iii) H_2 *queries*: when $\mathcal{A}_{II}$ submits a H_2 query with $(ID_i, m_i, V_i, T_i, PV_i/R_i)$, where $i \in [1, q_2]$, $\mathcal{C}$ checks whether $(ID_i, m_i, V_i, T_i, PV_i/R_i, h_i)$ exists in L_2. If it exists, $\mathcal{C}$ returns h_i. Otherwise, $\mathcal{C}$ chooses $h_i \in_R \mathbb{Z}_q^*$, inserts $(ID_i, m_i, V_i, T_i, PV_i/R_i, h_i)$ into L_2, and returns h_i.

$\mathcal{C}$ can answer $\mathcal{A}_{II}$'s other queries as follows.

Phase I

(i) Set-User-Key Queries. $\mathcal{A}_{II}$ requests a user's secret value with ID_i. If $ID_i = ID_t$, $\mathcal{C}$ aborts. If $ID_i \neq ID_t$, $\mathcal{C}$ checks for a tuple $(ID_i, d_i, x_i, R_i, PV_i)$ in L_k. If it exists, $\mathcal{C}$ returns x_i. Otherwise, $\mathcal{C}$ chooses $a_i, x_i, l_i \in_R \mathbb{Z}_q^*$, then sets $H_0(ID_i, R_i, PV_i) = l_i$, and computes $R_i = a_i P$, $d_i = a_i + l_i \cdot s$, and the public key as $PV_i = x_i P$. $\mathcal{C}$ inserts (ID_i, R_i, PV_i, l_i) into the list L_0 and $(ID_i, d_i, x_i, R_i, PV_i)$ into the list L_k and returns x_i.

(ii) Set-Private-Key Queries. $\mathcal{A}_{II}$ produces ID_i to $\mathcal{C}$ and requests a user's private key with ID_i. If $ID_i = ID_t$, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ checks for a tuple $(ID_i, d_i, x_i, R_i, PV_i)$ in L_k. If it exists, $\mathcal{C}$ returns (x_i, d_i). Otherwise, $\mathcal{C}$ chooses $a_i, x_i, l_i \in_R \mathbb{Z}_q^*$, then sets $H_0(ID_i, R_i, PV_i) = l_i$, and computes $R_i = a_i P$, $d_i = a_i + l_i \cdot s$, and $PV_i = x_i P$. $\mathcal{C}$ inserts (ID_i, R_i, PV_i, l_i) into the list L_0 and $(ID_i, d_i, x_i, R_i, PV_i)$ into the list L_k and returns (x_i, d_i).

(iii) Set-Public-Key Queries. $\mathcal{A}_{II}$ requests a user's public key with ID_i. $\mathcal{C}$ checks for a tuple $(ID_i, d_i, x_i, R_i, PV_i)$. If it exists, $\mathcal{C}$ returns the corresponding public key (PV_i, R_i). Otherwise, $\mathcal{C}$ chooses $a_i, x_i, l_i \in_R \mathbb{Z}_q^*$, then sets $H_0(ID_i, R_i, PV_i) = l_i$, and computes $R_i = a_i P$, $d_i = a_i + l_i \cdot s$, and the public key as $PV_i = x_i P$. $\mathcal{C}$ inserts (ID_i, R_i, PV_i, l_i) into the list L_0 and $(ID_i, d_i, x_i, R_i, PV_i)$ into the list L_k and returns (PV_i, R_i).

(iv) CLGSC-Signcrypt Queries. $\mathcal{A}_{II}$ sends the tuple ID_A, $PV_A, R_A, ID_B, PV_B, R_B, m$ to $\mathcal{C}$. For each query (ID_A, ID_B), if $ID_A \neq ID_t$, $\mathcal{C}$ executes the Set-Private-Key algorithm to compute SK_A corresponding to ID_A. Then, $\mathcal{C}$ gets the ciphertext δ by running the actual CLGSC-signcrypt algorithm. $\mathcal{C}$ sends δ to $\mathcal{A}_{II}$. If $ID_A = ID_t$ (and hence, $ID_B \neq ID_t$), $\mathcal{C}$ can obtain the full private key SK_B corresponding to ID_B. $\mathcal{C}$ computes $V = rP - h_t^{-1} h_t' \cdot ap$,

$T = V \cdot (x_B + d_B)$, sets $H_2(ID_A, m, V, T, PV_A) = h_t'$, $H_2(ID_A, m, V, T, R_A) = h_t$, and adds the tuples $(ID_A, m, V, T, PV_A, h_t')$ and $(ID_A, m, V, T, R_A, h_t)$ to the list L_2 in which $r, h_t, h_t' \in_R \mathbb{Z}_q^*$. $\mathscr{C}$ computes $c = H_1(V, T, s_B V, ID_B) \cdot f(ID_B) \oplus m$ and $W = r \cdot h_t + d_t$. $\mathscr{C}$ outputs $(ID_A, ID_B, \delta = (V, W, c))$ as the ciphertext.

The tuple $(ID_A, ID_B, \delta = (V, W, c))$ can pass the verification as the valid ciphertext because the equality holds as follows:

$$\begin{aligned} & R_A + H_0(ID_A, R_A, PV_A) \cdot P_{pub} + PV_A \\ & \cdot H_2(ID_A, m, V, T, PV_A) + V \\ & \cdot H_2(ID_A, m, V, T, R_A) = a_t P + l_t P + aPh_t' \\ & + \left(rP - h_t^{-1} aP\right) \cdot h_t = \left(r \cdot h_t + a_t + sl_t\right) \cdot P \\ & = \left(r \cdot h_t + d_t\right) \cdot P = W \cdot P. \end{aligned} \tag{11}$$

(v) CLGSC-Unsigncrypt Queries. $\mathscr{A}_{II}$ submits (ID_A, ID_B, δ) to $\mathscr{C}$. If $ID_B \neq ID_t$, $\mathscr{C}$ obtains the receiver's private key, runs the CLGSC-unsigncrypt algorithm, and returns the output of CLGSC-unsigncrypt to $\mathscr{A}_{II}$. Otherwise, $\mathscr{C}$ searches in L_2 for $(ID_A, m, V, T, PV_A, h_i')$ and $(ID_A, m, V, T, R_A, h_i)$. If the entries exist and the equality $W \cdot P = R_A + H_0(ID_A, R_A, PV_A) \cdot P_{pub} + PV_A \cdot h_i' + V \cdot h_i$ holds, T is retrieved. If $\mathscr{C}$ can find a tuple (V, T, Y, ID_B, l) in L_1 making the ODDH oracle return 1 when queries are on (aP, V, Y), then the message is $c \oplus l$.

Challenge. $\mathscr{A}_{II}$ submits $(m_0, m_1, ID_A^*, ID_B^*)$ in which m_0 and m_1 are distinct messages of equal length and ID_A^* and ID_B^* are the sender's and the receiver's identities, respectively. Here, it is to be noted that $\mathscr{A}_{II}$ must have made no Set-Private-Key queries on ID_B^* in Phase *I*. $\mathscr{C}$ aborts the game if $ID_B^* \neq ID_t$. Otherwise, $\mathscr{C}$ generates the challenge ciphertext as follows.

(1) It sets $V^* = bP$, where bP is given in the instance of the ECDLP problem and computes $T^* = (x_B^* + d_B^*) \cdot V^*$.

(2) It selects randomly a bit $\gamma \in \{0, 1\}$ and h as a random hash value and sets $c^* = m_\gamma \oplus h$.

(3) It selects W^*, satisfies the equation $W^* P = R_A + H_0(ID_A, R_A, PV_A) \cdot P_{pub} + PV_A \cdot H_2(ID_A, m, V^*, T^*, PV_A) + V^* \cdot H_2(ID_A, m, V^*, T^*, R_A)$, and sends $\delta^* = (V^*, W^*, c^*)$ to $\mathscr{A}_{II}$.

Phase II. $\mathscr{A}_{II}$ asks queries adaptively again. In addition, it cannot query CLGSC-unsigncrypt on δ^*.

Guess. Since $\mathscr{A}_{II}$ is able to break the IND-CLGSC-CCA2-*II* security of the CLGSC, a H_1 query with (V^*, T^*, Y^*, ID_B^*) should have been asked. Note that $Y^* = x_B^* \cdot V^* = abP$. Therefore, one of the q_{H_1} values of Y^* in L_1 is the ECDLP problem's solution. $\mathscr{C}$ chooses one Y^* randomly and outputs it as the solution.

In the above challenge query, the senders ID_A and ID_A^* can be $\emptyset$ for the encryption mode; otherwise, it works as signcryption. Thus, the proof is suitable for the two modes.

Analysis. In order to assess the probability of success of the challenger, $\mathscr{C}$ lets E_1, E_2, and E_3 be the events in which $\mathscr{C}$ aborts the IND-CLGSC-CCA2-*II* game.

(i) E_1 is an event in which $\mathscr{A}_{II}$ asks to query the secret value of the target identity ID_t. The probability of E_1 is $\Pr[E_1] = q_{sv}/q_{H_0}$.

(ii) E_2 is an event in which $\mathscr{A}_{II}$ asks to query the private key of the target identity ID_t. The probability of E_2 is $\Pr[E_2] = q_{sk}/q_{H_0}$.

(iii) E_3 is an event in which the target identity ID_t has not been chosen as the receiver by $\mathscr{A}_{II}$ during the challenge. The probability of E_3 is $\Pr[E_3] = 1 - 1/(q_{H_0} - q_{sk} - q_{sv})$.

Thus, $\mathscr{C}$ does not abort the IND-CLGSC-CCA2-II game's probability of $\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3] = (1 - q_{sv}/q_{H_0}) \cdot (1 - q_{sk}/q_{H_0}) \cdot (1/(q_{H_0} - q_{sk} - q_{sv}))$.

So, the successful advantage of $\mathscr{C}$ is $ADV_{\mathscr{C}}^{OGDH} = \Pr[\mathscr{C}(P, aP, bP) = abP] = \varepsilon \cdot (1 - q_{sv}/q_{H_0}) \cdot (1 - q_{sk}/q_{H_0}) \cdot (1/(q_{H_0} - q_{sk} - q_{sv}))$. □

4.2. Unforgeability

Theorem 8. *The CLGSC scheme in signcryption mode or signature mode is existentially unforgeable.*

Theorem 8 is proved based on Lemmas 9 and 10.

Lemma 9. *If an adversary $\mathscr{F}_I$ has nonnegligible advantage ε against the EUF-CLGSC-CMA-I security of our scheme and performing q_{ppk} Extract-Partial-Private-Key queries, q_{sk} Set-Private-Key queries, and q_{H_i} queries to oracles H_i $(i = 0, 1, 2)$, then there is an algorithm that solves the ECDLP problem with probability $\varepsilon \cdot (1 - q_{ppk}/q_{H_0}) \cdot (1 - q_{sk}/q_{H_0}) \cdot (1/(q_{H_0} - q_{ppk} - q_{sk}))$.*

Proof. Given an instance of the ECDLP problem $(P, bP) \in G_q$, $\mathscr{C}$ must find b. Suppose the EUF-CLGSC-CMA-*I* security of the CLGSC can be violated by a forger $\mathscr{F}_I$. $\mathscr{C}$ can utilize $\mathscr{F}_I$ to compute b as the solution by the following interactive game.

$\mathscr{C}$ chooses $s \in_R \mathbb{Z}_q^*$ uniformly as the master key and computes $P_{pub} = sP$; it sends *params* to $\mathscr{F}_I$ and maintains lists L_i $(0 \leq i \leq 2)$ to keep consistency between the responses to the hash queries and a list L_k of issued keys which is initially empty.

Training Phase. $\mathscr{F}_I$ may make a series of queries and all of the queries are responded to identically as those queries in the IND-CLGSC-CCA2-*I* game.

Forgery. $\mathscr{F}_I$ returns a valid ciphertext from the sender ID_A to the receiver ID_B. If $ID_A \neq ID_t$, $\mathscr{C}$ aborts the execution of this game. We are ready to apply the forking lemma [31] that essentially says the following: consider the concrete scheme producing signatures (m, V, h, W) or signcryption ciphertexts of the form (c, V, h, W), where each of V, h, W corresponds to one of the three moves of a honest-verifier zero-knowledge protocol. If $\mathscr{F}_I$ is a sufficiently efficient forger in the above

interaction and forges a signature or signcryption ciphertext (V, W, c) in a time t with probability $\epsilon \geqslant 10(q_{sc}+1)(q_{sc}+q_h)/2^l$ (l being a security parameter so that h is uniformly taken from a set of 2^l elements) when making q_{sc} CLGSC-signcrypt queries and q_h random oracle calls and if the triples (V, h, W) can be simulated without knowing the private key, then there exists a Turing machine F' that uses $\mathscr{F}_I$ to produce two valid ciphertexts (V^*, W, c^*) and (V^*, W^*, c^*) on the same message m, in expected time $t' \leqslant 120686q_h t/\epsilon$. Thus, we can get $W \cdot P = R_A - e_t \cdot P_{pub} + h_t' \cdot PV_A + h_t \cdot V$ and $W^* \cdot P = R_A - e_t \cdot P_{pub} + h_t' \cdot PV_A + h_t^* \cdot V$. Let $V = bP$. We can obtain the following value.

$$\begin{aligned} W \cdot P - W^* \cdot P &= h_t \dot{V} - h_t^* \cdot V \\ &\Longrightarrow (W - W^*) \cdot P = (h_t - h_t^*) \cdot V \\ &\Longrightarrow (W - W^*) \cdot P = (h_t - h_t^*) \cdot bP \\ &\Longrightarrow W - W^* = (h_t - h_t^*) \cdot b \\ &\Longrightarrow (W - W^*)\,\dot{(}h_t - h_t^*)^{-1} = b. \end{aligned} \tag{12}$$

Therefore, $\mathscr{C}$ solve the ECDLP as $b = (W - W^*)\dot{(}h_t - h_t^*)^{-1}$ for the ECDLP problem.

In the above forgery query, the receiver ID_B can be $\emptyset$ for the signature mode; otherwise it works as signcryption. Thus, the proof is suitable for the two modes.

Analysis. Let E_1, E_2, and E_3 be the events when $\mathscr{C}$ aborts the game.

(i) E_1 is an event in which the target identity ID_t's partial private key is queried by $\mathscr{F}_I$. The probability of E_1 is $\Pr[E_1] = q_{ppk}/q_{H_0}$.

(ii) E_2 is an event in which the target identity ID_t's private key is queried by $\mathscr{F}_I$. The probability of E_2 is $\Pr[E_2] = q_{sk}/q_{H_0}$.

(iii) E_3 is an event in which the target identity ID_t has not been chosen as the sender by $\mathscr{F}_I$ during the forgery. The probability of E_3 is $\Pr[E_3] = 1 - 1/(q_{H_0} - q_{ppk} - q_{sk})$.

Thus, $\mathscr{C}$ does not abort the EUF-CLGSC-CMA-I game's probability of $\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3] = (1 - q_{ppk}/q_{H_0}) \cdot (1 - q_{sk}/q_{H_0}) \cdot (1/(q_{H_0} - q_{ppk} - q_{sk}))$.

So, the successful advantage of $\mathscr{C}$ is $ADV_{\mathscr{C}}^{ELDLP} = \Pr[\mathscr{C}(P, bP) = b] = \varepsilon \cdot (1 - q_{ppk}/q_{H_0}) \cdot (1 - q_{sk}/q_{H_0}) \cdot (1/(q_{H_0} - q_{ppk} - q_{sk}))$. □

Lemma 10. *If an adversary $\mathscr{F}_{II}$ has nonnegligible advantage ε against the EUF-CLGSC-CMA-II security of our scheme performing q_{sv} Extract-Secret-Value queries and q_{H_i} queries to oracles H_i $(i = 0, 1, 2)$, then there is an algorithm $\mathscr{C}$ that solves the ECDLP problem with probability $\varepsilon \cdot (1 - q_{sv}/q_{H_0}) \cdot (1 - q_{sk}/q_{H_0}) \cdot (1/(q_{H_0} - q_{sk} - q_{sv}))$.*

Proof. Given an instance of the ECDLP problem (P, bP), $\mathscr{C}$ must find b. Suppose the EUF-CLGSC-CMA-*II* security of the CLGSC can be violated by a forger $\mathscr{F}_{II}$. $\mathscr{C}$ can utilize $\mathscr{F}_{II}$ to compute b as the solution by the following interactive game.

$\mathscr{C}$ chooses $s \in_R \mathbb{Z}_q^*$ uniformly as the master key and computes $P_{pub} = sP$; it sends *params* and s to $\mathscr{F}_{II}$ and maintains lists L_i $(0 \leq i \leq 2)$ to keep consistency between the responses to the hash queries and a list L_k of issued keys which is initially empty.

Training Phase. $\mathscr{F}_{II}$ may make a series of queries and all the queries are responded to identically as those queries in the IND-CLGSC-CCA2-*II* game.

Forgery. Eventually, $\mathscr{F}_{II}$ returns a valid ciphertext from the sender ID_A to the receiver ID_B. If $ID_A \neq ID_t$, $\mathscr{C}$ aborts the execution of this game. It follows from the forking lemma [31] that if $\mathscr{F}_{II}$ is a sufficiently efficient forger in the above interaction, then we can construct another probabilistic polynomial time Turing machine $\mathscr{F}_{II}'$ that outputs two ciphertexts (V^*, W, c^*) and (V^*, W^*, c^*) on the same message m. Thus, we can get $W \cdot P = R_t + l_t \cdot P_{pub} + h_t' \cdot PV_t + h_t V$ and $W^* \cdot P = R_t + l_t \cdot P_{pub} + h_t' \cdot PV_t + h_t^* V$. Let $V = bP$. We can obtain the following value.

$$\begin{aligned} W \cdot P - W^* \cdot P &= h_t \dot{V} - h_t^* \cdot V \\ &\Longrightarrow (W - W^*) \cdot P = (h_t - h_t^*) \cdot V \\ &\Longrightarrow (W - W^*) \cdot P = (h_t - h_t^*) \cdot bP \\ &\Longrightarrow W - W^* = (h_t - h_t^*) \cdot b \\ &\Longrightarrow (W - W^*)\,\dot{(}h_t - h_t^*)^{-1} = b. \end{aligned} \tag{13}$$

Therefore, $\mathscr{C}$ solves the ECDLP as $b = (W - W^*)\dot{(}h_t - h_t^*)^{-1}$ for the ECDLP problem.

In the above forgery query, the receiver ID_B can be $\emptyset$ for the signature mode; otherwise it works as signcryption. Thus, the proof is suitable for the two modes.

Analysis. Let E_1, E_2, and E_3 be the events when $\mathscr{C}$ aborts the game.

(i) E_1 is an event in which the target identity ID_t's secret value is queried by $\mathscr{F}_{II}$. The probability of E_1 is $\Pr[E_1] = q_{sv}/q_{H_0}$.

(ii) E_2 is an event in which the target identity ID_t's private key is queried by $\mathscr{F}_{II}$. The probability of E_2 is $\Pr[E_1] = q_{sk}/q_{H_0}$.

(iii) E_3 is an event in which the target identity ID_t has not been chosen as the sender by $\mathscr{F}_{II}$ during forgery. The probability of E_3 is $\Pr[E_3] = 1 - 1/(q_{H_0} - q_{sk} - q_{sv})$.

Thus, $\mathscr{C}$ does not abort the EUF-CLGSC-CMA-II game's probability of $\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3] = (1 - q_{sv}/q_{H_0}) \cdot (1 - q_{sk}/q_{H_0}) \cdot (1/(q_{H_0} - q_{sk} - q_{sv}))$.

So, the successful advantage of $\mathscr{C}$ is $ADV_{\mathscr{C}}^{OGDH} = \Pr[\mathscr{C}(P, aP, bP) = abP] = \varepsilon \cdot (1 - q_{sv}/q_{H_0}) \cdot (1 - q_{sk}/q_{H_0}) \cdot (1/(q_{H_0} - q_{sk} - q_{sv}))$. □

5. Performance Analysis

Since computation time and ciphertext size are two important factors affecting efficiency, we compare our scheme

Table 1: Comparisons of the computational overhead and storage costs.

Schemes	Ciphertext size	CLGSC-signcrypt time	CLGSC-unsigncrypt time
[24]	$2\lvert G_1\rvert + \lvert m\rvert + \lvert ID\rvert + \lvert G_2\rvert + \lvert q\rvert$	$(3T_E + 2T_M + 4T_H) \approx 188.5\,T_{\text{Mul}}$	$(T_E + T_M + 4T_H + 2T_P) \approx 246.5\,T_{\text{Mul}}$
[25]	$2\lvert G_1\rvert + \lvert m\rvert + \lvert ID\rvert + \lvert G_2\rvert$	$(2T_E + 3T_M + 3T_H) \approx 174\,T_{\text{Mul}}$	$(T_E + 3T_M + 3T_H + 2T_P) \approx 304.5\,T_{\text{Mul}}$
[26]	$2\lvert G_1\rvert + \lvert m\rvert$	$(T_E + 4T_M + 3T_H + T_P) \approx 246.5\,T_{\text{Mul}}$	$(T_M + 3T_H + 5T_P) \approx 464\,T_{\text{Mul}}$
[27]*	$2\lvert G_1\rvert + \lvert q\rvert + \lvert m\rvert$	$7T_M + 7T_H \approx 203T_{\text{Mul}}$	$8T_M + 7T_H \approx 232\,T_{\text{Mul}}$
Ours	$2\lvert G_q\rvert + \lvert m\rvert$	$(5T_M + 4T_H) \approx 145\,T_{\text{Mul}}$	$(4T_M + 4T_H) \approx 116\,T_{\text{Mul}}$

*Note that the authors also considered the private key exposure problem in their paper.

with several existing schemes in these two terms from two aspects: CLGSC-signcrypt and CLGSC-unsigncrypt. We pay attention to operations such as bilinear pairing operations, exponentiation operations, scalar multiplication operations, and hash operations. We define the notations in the Notations section and adopt the experiment testing results from [32–34].

The comparison is shown in Table 1; $|G_1|$ denotes the size of an element in G_1, $|G_2|$ denotes the size of an element in G_2, $|m|$ denotes the length of message m, $|ID|$ denotes the length of identity ID, and $|q|$ is the size of an element in $\mathbb{Z}_q^*$.

Since the pairing and exponentiation operations require much more time than the multiplication operation, our proposed scheme is implemented without pairing and exponentiation operations. From Table 1, it shows that our proposed CLGSC scheme requires much less computational time than the other four schemes. So, our scheme has the shortest ciphertext size and is of high efficiency too.

6. Conclusion

In this paper, a concrete CLGSC scheme without utilizing bilinear pairing operations is proposed, and its security is proved in the random oracle model under the ECDLP and ECDLP assumptions, including security against both an adaptively chosen ciphertext attack and an existential forgery of Type *I* and *II* adversaries. The new scheme is computationally efficient and is suitable for low power and processor devices.

Notations

T_{Mul}: Time required for executing a modular multiplication operation
T_E: Time required for executing an exponentiation $T_E \approx 43.5T_{\text{Mul}}$
T_M: Time required for executing a scalar multiplication $T_M \approx 29T_{\text{Mul}}$
T_H: Time complexity for executing the simple hash function, which is negligible
T_P: Time required for executing a bilinear pairing operation $T_P \approx 87T_{\text{Mul}}$.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the National Nature Science Foundation of China (no. 61702218), Shandong Provincial Natural Science Foundation (no. ZR2014FL011, no. ZR2015FL023), and International Joint Training Program for Young and Middle-Aged Scholar of Shandong Province.

References

[1] A. Shamir, "Identity-based cryptosystem and signature scheme," in *Proceedings of the Cryptology-CRYPTO*, vol. 196 of *Lecture Notes in Computer Science*, pp. 120–126, 1984.

[2] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proceedings ofthe Proceedings of Cryptology-ASIACRYPT*, vol. 2894 of *Lecture Notes in Computer Science*, pp. 452–474, 2003.

[3] Y. L. Zheng, "Digital signcryption or how to achieve cost (signature & encryption) ≪ cost (Signature) + cost (encryption)," in *Proceedings of the Cryptology-CRYPTO*, vol. 1294 of *Lecture Notes in Computer Science*, pp. 165–179, 1997.

[4] Y. L. Zheng, "Signcryption and its applications in efficient public key solutions," in *Proceedings of the Information Security Workshop (ISW '97)*, vol. 1397 of *An invited lecture*, pp. 291–312, 1997.

[5] F. Bao and R. H. Deng, "A signcryption scheme with signature directly verifiable by public key," in *Proceedings of Public Key Cryptography*, vol. 1431 of *Lecture Notes in Computer Science*, pp. 55–59, 1998.

[6] C. Gamage, J. Leiwo, and Y. Zheng, "Encrypted message authentication by firewalls," in *Proceedings of the Public Key Cryptography*, vol. 1560 of *Lecture Notes in Computer Science*, pp. 69–81, 1999.

[7] J. Malone-Lee and W. Mao, "Two birds one stone: signcryption using RSA," in *Proceedings of the Cryptology-CT-RSA*, vol. 2612 of *Lecture Notes in Computer Science*, pp. 211–226, 2003.

[8] B. Libert and J. Quisquater, "Efficient Signcryption with Key Privacy from Gap Diffie- Hellman Groups," in *Proceedings of the Public Key Cryptography*, vol. 2947 of *Lecture Notes in Computer Science*, pp. 187–200, 2004.

[9] F. Li, H. Zhang, and T. Takagi, "Efficient signcryption for heterogeneous systems," *IEEE Systems Journal*, vol. 7, no. 3, pp. 420–429, 2013.

[10] F. Li, B. Liu, and J. Hong, "An efficient signcryption for data access control in cloud computing," *Computing*, vol. 99, no. 5, pp. 465–479, 2017.

[11] J. Malone-Lee, "Identity based signcryption," http://eprint.iacr.org.

[12] B. Libert and J. J. Quisquater, "A new identity based signcryption scheme from pairings," in *Proceedings of the IEEE Information Theory Workshop, ITW '03*, pp. 155–158, 2003.

[13] X. Boyen, "Multipurpose identity-based signcryption: a Swiss Army knife for identity-based cryptography," in *Proceedings of the Cryptology-Crypto*, vol. 2729 of *Lecture Notes in Computer Science*, pp. 383–399, 2003.

[14] L. Chen and J. Malone-Lee, "Improved identity-based signcryption," in *Proceedings of the Public Key Cryptography*, vol. 3386 of *Lecture Notes in Computer Science*, pp. 362–379, 2005.

[15] P. S. Barreto, B. Libert, N. McCullagh, and J. J. Quisquater, "Efficient and provably-secure identity-based signatures and signcryption from bilinear maps," in *Proceedings of the Cryptology-ASIACRYPT 2005*, vol. 3788 of *Lecture Notes in Computer Science*, pp. 515–532, 2005.

[16] H. J. Jo, J. H. Paik, and D. H. Lee, "Efficient privacy-preserving authentication in wireless mobile networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 7, pp. 1469–1481, 2014.

[17] M. Barbosa and P. Farshim, "Certificateless signcryption," in *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS '08)*, pp. 369–372, ACM, Tokyo, Japan, March 2008.

[18] F. G. Li, M. Shirase, and T. Takagi, "Certificateless hybrid signcryption," in *Proceedings of the ISPEC '09*, vol. 5451 of *Lecture Notes in Computer Science*, pp. 112–123, 2009.

[19] A. Yin and H. Liang, "Certificateless hybrid signcryption scheme for secure communication of wireless sensor networks," *Wireless Personal Communications*, vol. 80, no. 3, pp. 1049–1062, 2015.

[20] F. Li, Y. Han, and C. Jin, "Certificateless online/offline signcryption for the Internet of Things," *Wireless Networks*, vol. 23, no. 1, pp. 145–158, 2017.

[21] Y. L. Han and X. Y. Yang, "New ECDSA-verifiable generalized signcryption," *Chinese Journal of Computers*, vol. 29, no. 11, pp. 2003–2012, 2006.

[22] Y. L. Han, X. Y. Yang, P. Wei et al., "ECGSC: elliptic curve based generalized signcryption," in *Proceedings of the 3rd International Conference on Ubiquitous Intelligence and Computing- UIC '06*, vol. 4159 of *Lecture Notes in Computer Science*, pp. 956–965, 2006.

[23] Y. Han, "Generalization of signcryption for resources-constrained environments," *Wireless Communications and Mobile Computing*, vol. 7, no. 7, pp. 919–931, 2007.

[24] H. F. Ji, W. B. Han, and L. Zhao, "Certificateless generalized signcryption," Cryptology ePrint Archive, Report 2010/204, http://eprint.iacr.org.

[25] P. Kushwah and S. Lai, "Efficient generalized signcryption schemes," Cryptology ePrint Archive, Report 2010/346, http://eprint.iacr.org.

[26] C. X. Zhou, W. Zhou, and X. W. Dong, "Provable certificateless generalized signcryption scheme," *Designs, Codes and Cryptography*, vol. 71, no. 2, pp. 331–346, 2014.

[27] C. X. Zhou, Z. Zhao, W. Zhou et al., "Certificateless key-insulated generalized signcryption scheme without bilinear pairings," *Security and Communication Networks*, vol. 2017, Article ID 8405879, 17 pages, 2017.

[28] S.-H. Seo, J. Won, and E. Bertino, "pCLSC-TKEM: a pairing-free certificateless signcryption- tag key encapsulation mechanism for a privacy-preserving IoT," *Transactions on Data Privacy*, vol. 9, no. 2, pp. 101–130, 2016.

[29] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[30] N. Koblitz and A. Menezes, "Intractable problems in cryptography," in *Proceedings of the 9th International Conference on Finite Field and Their Applications, Contemporary Mathematics*, pp. 279–300, 2010.

[31] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, 2000.

[32] Y. F. Chung, K. H. Huang, F. Lai, and T. S. Chen, "ID-based digital signature scheme on the elliptic curve cryptosystem," *Computer Standards & Interfaces*, vol. 29, no. 6, pp. 601–604, 2007.

[33] S. K. H. Islam and G. P. Biswas, "A pairing-free identity-based authenticated group key agreement protocol for imbalanced mobile networks," *Annals of Telecommunications-Annales des Télécommunications*, vol. 67, no. 11-12, pp. 547–558, 2012.

[34] S. K. H. Islam and G. P. Biswas, "Provably secure and pairing-free certificateless digital signature scheme using elliptic curve cryptography," *International Journal of Computer Mathematics*, vol. 90, no. 11, pp. 2244–2258, 2013.

Hindawi
Security and Communication Networks
Volume 2018, Article ID 5967635, 12 pages
https://doi.org/10.1155/2018/5967635

Research Article

LWR-Based Fully Homomorphic Encryption, Revisited

Fucai Luo,[1,2,3] **Fuqun Wang**,[4,5] **Kunpeng Wang**,[1,2,3] **Jie Li**,[1,2,3] **and Kefei Chen**[4,5]

[1]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China*
[2]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China*
[3]*Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing, China*
[4]*College of Science, Hangzhou Normal University, Hangzhou, China*
[5]*Westone Cryptologic Research Center, Beijing, China*

Correspondence should be addressed to Fucai Luo; luofucai@iie.ac.cn

Received 29 August 2017; Revised 3 January 2018; Accepted 18 January 2018; Published 23 April 2018

Academic Editor: Amir Anees

Very recently, Costache and Smart proposed a fully homomorphic encryption (FHE) scheme based on the Learning with Rounding (LWR) problem, which removes the noise (typically, Gaussian noise) sampling needed in the previous lattices-based FHEs. But their scheme did not work, since the noise of homomorphic multiplication is complicated and large, which leads to failure of decryption. More specifically, they chose LWR instances as a public key and the private key therein as a secret key and then used the tensor product to implement homomorphic multiplication, which resulted in a tangly modulus problem. Recall that there are two moduli in the LWR instances, and then the moduli will tangle together due to the tensor product. Inspired by their work, we built the first workable LWR-based FHE scheme eliminating the tangly modulus problem by cleverly adopting the celebrated *approximate eigenvector* method proposed by Gentry et al. at Crypto 2013. Roughly speaking, we use a specific matrix multiplication to perform the homomorphic multiplication, hence no tangly modulus problem. Furthermore, we also extend the LWR-based FHE scheme to the multikey setting using the tricks used to construct LWE-based multikey FHE by Mukherjee and Wichs at Eurocrypt 2016. Our LWR-based multikey FHE construction provides an alternative to the existing multikey FHEs and can also be applied to multiparty computation with higher efficiency.

1. Introduction

Fully homomorphic encryption (FHE) is a cryptographic primitive that allows performing arbitrarily complex and efficiently computable evaluations over encrypted data without decrypting them. But the problem of how to construct a FHE scheme had been bothering cryptologists since it was initially introduced by Rivest et al. [1]. Until 2009, this conundrum was compromised due to Gentry's first plausible candidate FHE construction based on ideal lattices [2]. Since then, a series of works [3–8] have been presented with much progress mainly in security assumptions and efficiencies.

Gentry's seminal work [2] showed for the first time that FHE can be based on cryptographic assumptions and put forward a remarkable "bootstrapping" theorem to achieve full homomorphism (which needs a "circular security" assumption). However, his scheme relies on relatively stronger cryptographic assumptions on ideal lattices (ideal lattices are a special breed that we know relatively little about) and can only evaluate "low degree" polynomials homomorphically without "bootstrapping."

The LWE-based FHEs [5, 9–13] enjoy higher efficiency and stronger security compared to the previous schemes [2–4, 7] (following a similar framework to Gentry's work), due to the simple algebraic structure of the well-studied LWE [14] and classical (quantum) reduction from some apparently intractable lattice problems (e.g., GapSVP) to LWE [14, 15]. The first LWE-based FHE was proposed by Brakerski and Vaikuntanathan [5] (henceforth, BV11b), who used a novel *relinearization* technique to construct a "somewhat homomorphic" encryption scheme based on LWE problem and introduced a novel *dimension-modulus reduction* technique, without resorting to the "squashing paradigm" used in the previous schemes [2–4, 7]. The subsequent improved

works mainly refer to Brakerski et al.'s [9] BGV12 and Brakerski's [10] Bra12. In BGV12, Brakerski, Gentry, and Vaikuntanathan used the *dimension reduction* and *modulus reduction* (which are originated from BV11b) iteratively and gradually, to construct a "leveled" FHE scheme (capable of evaluating arbitrary polynomial-depth circuits). Bra12 used the *dimension reduction* without the *modulus reduction*, to build a better leveled FHE scheme, which is superior to the previous best known in simplicity, noise management, and security. This is mainly because of their noise which only grows linearly ($B \to B \cdot \mathrm{poly}(n)$) with every homomorphic multiplication, while in all previous works, the noise grows quadratically ($B \to B^2 \cdot \mathrm{poly}(n)$) without *modulus reduction*.

In Crypto 2013, Gentry et al. [16] (henceforth, GSW13) presented a LWE-based FHE scheme of GSW style (which was improved by Alperin-Sheriff and Peikert [17], henceforth, AP14), using two novel techniques of so-called *approximate eigenvector* and *flatten*, where the ciphertext is a matrix rather than vector. For the most part, its homomorphic addition and multiplication are just matrix addition and multiplication, which avoids the *key switching*, *modulus switching*, and the "evaluation" key used in previous schemes (e.g., BV11b, BGV12). It is important to note that, besides the fact that the scheme does not need an "evaluation" key, it has an interesting property of asymmetric noise growth because of its specific GSW style. Based on GSW13, a sequence of schemes was proposed, including bootstrapping schemes [11, 17], multikey schemes [6, 12, 13], and some other related schemes [18, 19] (these schemes mainly leverage the homomorphic operations of GSW13).

Motivations. The above-mentioned LWE-based FHEs and related schemes suffered the complex and time-consuming Gaussian or sub-Gaussian noise sampling, due to the fact that the corresponding LWE problem needs a noise (error) vector sampled from a distribution, typically (discrete) Gaussian or sub-Gaussian distribution [5, 9, 10, 12, 16, 17]. In particular, some schemes (e.g., [6, 18]) based on the LWE problem have to sample Gaussian noise in the encryption process, which seriously weakens the schemes' efficiencies. Moreover, it has been recently shown (e.g., [20, 21]) that the Gaussian sampling will create lots of potential side-channel vulnerabilities that result in complete leakage of the secret key. Although it is possible to design good implementations which protect against side-channel attacks, these implementations are often very complex.

As a matter of course, this raises a question: *can we cast away the Gaussian noise sampling in building a* FHE *scheme while maintaining the same (almost) security level as those based on* LWE *problem*? Indeed, this is valuable theoretically and practically and even pedagogically.

Very recently, Costache and Smart [22] showed a FHE scheme based on the ring-LWR problem (or RLWR, a variant of Learning with Rounding (LWR) problem). Their scheme removes the Gaussian noise needed in the previous LWE-based FHEs and results in slightly smaller ciphertexts. Roughly speaking, they focused their attention on BGV12 and used the techniques of *relinearization* and *modulus switching* to build a RLWR-based FHE scheme. However, the LWR (the definition of LWR will be presented in Section 2.2), mainly leveraged by a scaled rounding function [23] including two different moduli q and p, makes the tensor product $\otimes$ used by them to implement homomorphic multiplication intractable. In more detail, they chose RLWR instances as a public key and the private key therein as a secret key, and the ciphertext was computed as a vector $\mathbf{c} = (v, w) \in R_q \times R_p$ decrypting to message $\mu \in R_t$, where R_q, R_p, R_t are quotient rings. Then, they used the tensor product $\otimes$ to implement homomorphic multiplication, which results in a product of two different elements belonging to different rings (i.e., two moduli are tangled together). This makes their analysis of the multiplication noise complicated and obscure. In fact, this "tangly modulus" problem brings a large multiplication noise to their decryption equation (in terms of ciphertext after homomorphic multiplication) and thus leads to failure of the decryption. Therefore, how to construct a FHE scheme based on LWR problem, we think, is still an open problem, while we focus our attention on this problem.

Table 1: Comparison with LWE-based FHEs: GSW13 and AP14.

FHE	Modulus q	Security loss	Gaussian noise
GSW13 and AP14	$(O(n \log q))^{L+O(1)}$	$O(1)$	Yes
Our LWR-based FHE	$(O(n \log q))^{L+O(1)}$	$O(qn \log q)$	No

(1) We consider a leveled FHE scheme where the depth of circuits is polynomial L. The homomorphic evaluation capability, efficiency, and security of the FHE scheme mainly depend on modulus q, under the same dimension n. Note that the modulus q of AP14 is relatively smaller than that of GSW13, for the sub-Gaussian in AP14 results in a tighter noise growth than the Gaussian. Here, we ignore this little difference. (2) Here, the security loss is caused by the reduction between the security of the FHE scheme and the LWE problem. Since the security of our FHE scheme is directly based on the LWR problem, the security loss involves the reduction loss incurred by the reduction between LWE and LWR (see Theorem 7 in Section 2).

Our Results. In this paper, we propose a workable LWR-based FHE scheme eliminating the tangly modulus problem by cleverly adopting the celebrated *approximate eigenvector* method in GSW13. Roughly speaking, we use a specific matrix multiplication to perform the homomorphic multiplication, which avoids the tangly modulus problem, where the specific matrix multiplication involves a variant of gadget matrix (which will be described in Section 2.3). The efficiency of our scheme is almost comparable to that of GSW13 and AP14 without counting the cost of Gaussian noise sampling, for the size of modulus q is almost the same as theirs which can be seen from Table 1 (our modulus q is larger; see Section 4.2). Indeed, it is mainly our larger security loss (up to a polynomial factor) that results in the larger modulus q (up to a polynomial factor). Our scheme can be seen as an alternative to the GSW13 and AP14.

Furthermore, we also extend the LWR-based FHE scheme to multikey setting using the tricks used to construct LWE-based multikey FHE by Mukherjee and Wichs [12] at Eurocrypt 2016. Again, we leverage the specific GSW

style to avoid the tangly modulus problem. Interestingly, it seems that the method of [12] has been tailored to be employed in constructing our LWR-based multikey FHE scheme, since it helps to avoid the tangly modulus problem when expanding the valid ciphertexts needed in multikey setting (see Section 5). Compared to Mukherjee and Wichs's scheme, our scheme can also be applied to multiparty computation and is more efficient; again, this is largely for the reason that there is no Gaussian noise sampling. This can be also verified by Table 1 and by the fact that the LWE-based multikey FHE [12] is an extension of AP14.

What is more, our LWR-based FHE and its extended multikey FHE support bootstrapping procedures; this is due to the fact that the encryption and decryption processes in our schemes are very similar to those in AP14, except that there are two different moduli in our scheme (which will not tangle together because of our specific structure). We believe that it is straightforward to convert our LWR-based scheme into a RLWR-based one.

Our Techniques. In our LWR-based FHE, we choose m pairs of LWR instances as a public key and the private key therein as a secret key. The public key is assembled as a matrix

$$\mathbf{A} = \begin{bmatrix} \overline{\mathbf{A}} \\ \mathbf{b} \end{bmatrix} \in \mathbb{Z}_q^{n\times m}\,[\times]\,\mathbb{Z}_p^m \quad \text{such that } \mathbf{s}\cdot\mathbf{A} = \mathbf{e} \bmod p, \tag{1}$$

where $\overline{\mathbf{A}} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{b} \in \mathbb{Z}_p^m$, and the secret key is $\mathbf{s} = (-p/q\cdot\mathbf{s}', 1)$, where $\mathbf{s}' \in \{0,1\}^n$. Since there are two different moduli in a matrix, we use the symbol $[\times]$ (see Section 2) to differentiate from the conventional symbol $\times$.

For a message $\mu \in \{0,1\}$, the ciphertext is computed as

$$\mathbf{C} = \mathbf{A}\cdot\mathbf{R} + \mu\mathbf{G} \in \mathbb{Z}_q^{n\times N}\,[\times]\,\mathbb{Z}_p^N, \tag{2}$$

where $\mathbf{G} \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N$ is a variant of gadget matrix (see Section 2.3). To perform homomorphic multiplication (addition is very natural), do the following: given two ciphertexts $\mathbf{C}_1, \mathbf{C}_2$ decrypting to message $\mu_1, \mu_2 \in \{0,1\}$, we have

$$\mathbf{C}_{\text{mult}} \triangleq \mathbf{C}_1\cdot\mathbf{G}^{-1}(\mathbf{C}_2) \in \mathbb{Z}_q^{n\times N}\,[\times]\,\mathbb{Z}_p^N, \tag{3}$$

since matrix $\mathbf{G}^{-1}(\mathbf{C}_2) \in \{0,1\}^{N\times N}$ does not affect the structure of matrix $\mathbf{C}_1\cdot\mathbf{G}^{-1}(\mathbf{C}_2)$ (see Operations in Section 2). Hence, we assert that the structures of ciphertexts remain unchanged after L levels of multiplication, so they can be properly decrypted. Indeed, our homomorphic multiplication will not cause the tangly modulus problem existent in Costache and Smart's solution [22], because these two different modular operations (in terms of moduli q, p) can be partitioned by our matrix operation but cannot be by the tensor product used by them. We defer the details to Section 4.

For the secret key, $\mathbf{s} = (-p/q\cdot\mathbf{s}', 1)$, where p, q are public and $\mathbf{s}'$ is privately and uniformly chosen from $\{0,1\}^n$ instead of $\mathbb{Z}_q$ for initial noise and efficiency. This is reasonable and secure relying on Theorem 7 in Section 2.2 and the hardness of LWE problem with binary secret (which was proved at least as hard as the original LWE problem by Brakerski et al. [24]).

Organization. In Section 2, we give some preliminaries including the LWE problem, LWR problem, and a variant of gadget matrix. We describe a basic LWR-based encryption scheme and give its full security proof in Section 3. Section 4 goes into the core of our LWR-based FHE scheme. In Section 5, we extend our main construction to the LWR-based multikey FHE. Finally, we conclude the paper in Section 6.

2. Preliminaries

As a preliminary matter, we give some explanations for some notations and operations throughout the paper.

Notations. For an integer q, we define the set $\mathbb{Z}_q \triangleq (-q/2, q/2] \cap \mathbb{Z}$, and all logarithms on q are to base 2. All arithmetics are performed over $\mathbb{Z}$ or $\mathbb{Q}$ when division is used, and for ease of use, we let $[n] \triangleq \{1,\ldots,n\}$. We denote vectors in bold lowercase (e.g., $\mathbf{x}$) and matrices in bold uppercase (e.g., $\mathbf{A}$). Let $\mathbf{A}^t$ (resp., $\mathbf{x}^t$) be the transpose of $\mathbf{A}$ (resp., $\mathbf{x}$). For any $x \in \mathbb{Q}$, we denote by $\lfloor x\rfloor$, $\lceil x\rceil$, and $\lceil x\rfloor$ the rounding of x down, up, or to the nearest integer; these notations also apply to vector and matrix. We say that a function $\text{negl}(n)$ is negligible if there are not any polynomial fractions smaller than the $\text{negl}(n)$ for sufficiently large n. All vectors are treated as rows in the paper.

Probability. We let $x \leftarrow_r \mathcal{D}$ denote that x is sampled uniformly at random from a distribution $\mathcal{D}$ and $x \leftarrow_r \mathcal{S}$ (e.g., $\mathbb{Z}_q$) denote that x is uniform over a set $\mathcal{S}$.

Operations. For an n-dimensional vector $\mathbf{x} = \{x_1,\ldots,x_n\}$, we denote its norm by $\|\mathbf{x}\| \triangleq \max\{|x_i|\}_{i\in[n]}$, where $|x_i|$ refers to its magnitude. The multiplication between two vectors $\mathbf{x}, \mathbf{y}$ over $\mathbb{Z}_q$ is denoted by $\langle\mathbf{x},\mathbf{y}\rangle_q$ (i.e., $\langle\mathbf{x},\mathbf{y}\rangle \bmod q$). What we want to particularly explain beforehand is the multiplication between two matrices.

Given two different moduli q, p, and any integer n, m, for $(n+1)\times m$ matrix $\mathbf{A}$, by abuse of notation, we let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}[\times]\mathbb{Z}_p^m$ indicate that all row vectors of $\mathbf{A}$ are over $\mathbb{Z}_q^m$ but the last one (i.e., $(n+1)$-th) is over $\mathbb{Z}_p^m$, and we state that we should be careful about, for example, multiplying a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}[\times]\mathbb{Z}_p^m$ by a matrix $\mathbf{R} \in \{0,1\}^{m\times m}$ (or a vector $\mathbf{r} \in \{0,1\}^m$). More precisely, we should multiply every row vector of $\mathbf{A}$ by the matrix $\mathbf{R} \in \{0,1\}^{m\times m}$, that is, multiply each row vector of $\mathbf{A}$ by each column vector of $\mathbf{R} \in \{0,1\}^{m\times m}$, and then take a modular operation (take the product modulo a modulus). Note that the last row vector of $\mathbf{A}$ is over $\mathbb{Z}_p^m$, and thus the multiplications between this row vector and the matrix $\mathbf{R} \in \{0,1\}^{m\times m}$ are evaluated over $\mathbb{Z}_p^m$ (take the products modulo the modulus p), while all the other multiplications are evaluated over $\mathbb{Z}_q^m$ (take the products modulo the modulus q). (These two modular operations can be partitioned due to the matrix structure which, roughly speaking, avoids the tangly

modulus problem existent in [22].) This principle also applies to addition between two matrices over $\mathbb{Z}_q^{n\times m}[\times]\mathbb{Z}_p^m$. Indeed, this is an important difference between LWE and LWR when performing addition and multiplication operations on matrices.

In our security proof, we will use the following variant of the standard leftover hash lemma [25].

Lemma 1 (see [5]). *Let λ, n, q, p, and $m \geq n\log q + \log p + 2\lambda$ be integers, and let $\mathbf{y} \leftarrow_r \mathbb{Z}_q^n \times \mathbb{Z}_p$. For matrix $\mathbf{A} \leftarrow_r \mathbb{Z}_q^{n\times m}[\times]\mathbb{Z}_p^m$, let $\mathbf{r} \leftarrow_r \{0,1\}^m$; one has*

$$\triangle\left((\mathbf{A}, \mathbf{A}\cdot\mathbf{r}), (\mathbf{A}, \mathbf{y})\right) \leq \frac{1}{2}\sqrt{\frac{q^n \cdot p}{2^m}} < 2^{-\lambda}, \tag{4}$$

where $\triangle(X, Y)$ denotes the statistical distance between two distributions X, Y.

2.1. Learning with Errors (LWE). The well-known LWE problem has been enjoying fame for its applicability in constructions of lattice-based schemes conjectured to be secure in quantum setting, ever since Regev introduced it and showed a quantum reduction [14] from some worst-case hardness of the standard lattice problems to LWE problem (followed by classical reductions [15, 24]). It is defined as follows.

Definition 2 (see [9]). For positive integers n, $q = q(n)$, $m \geq O(n\log q)$, a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and an error distribution χ over $\mathbb{Z}$ (typically, a (discrete) Gaussian distribution with standard deviation αq for $\alpha = o(1)$). Let $\mathcal{A}_{n,q,m,\chi}$ be the distribution over $\mathbb{Z}_q^{n+1}$ obtained by choosing m independent samples $\mathbf{a} \leftarrow_r \mathbb{Z}_q^n$ and an error term $e \leftarrow_r \chi$ and outputting m pairs $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s}\rangle_q + e)$. Then, the (average-case) decision problem of the LWE, denoted by $\mathsf{DLWE}_{n,q,m,\chi}$, is to distinguish, given arbitrarily many independent samples, the uniform distribution over $\mathbb{Z}_q^{n+1}$ from $\mathcal{A}_{n,q,m,\chi}$, for a fixed $\mathbf{s} \in \mathbb{Z}_q^n$. The search problem of $\mathsf{LWE}_{n,q,m,\chi}$ is aim to find secret $\mathbf{s}$ given m independent samples from $\mathsf{LWE}_{n,q,m,\chi}(\mathbf{s})$ (for $\mathbf{s} \in \mathbb{Z}_q^n$). The $\mathsf{LWE}_{n,q,m,\chi}$ assumption is that the $\mathsf{LWE}_{n,q,m,\chi}$ problem is infeasible.

Definition 3 (B-bounded distributions [16]). A distribution ensemble $\{\chi_n\}_{n\in\mathbb{N}}$, supported over the integers, is called B-bounded if

$$Pr_{e\leftarrow_r\chi_n}\left[|e| > B\right] = \mathrm{negl}(n). \tag{5}$$

2.2. Learning with Rounding (LWR). Learning with Rounding (LWR) problem, which was used to construct lossy trapdoor functions, reusable computational extractors, and deterministic encryption, was firstly proposed by Banerjee et al. [23] for improving the efficiency of pseudorandom generator (PRG) based on the LWE problem. Indeed, the LWR problem can be seen as a deterministic alternative to LWE problem, except that the noise in LWR is deterministic which derandomizes the Gaussian noise in LWE, and the noise in LWR resulted from the scale rounding function being smaller than that in LWE. Specifically, the noise in LWE is B-bounded ($B > 2\sqrt{n}$ for security [14]), whereas it has magnitude of less than $1/2$ in LWR. We recall the scaled rounding function $\lceil\cdot\rfloor_p$ [23] which is defined as follows: for $p < q$,

$$\begin{aligned} \lceil\cdot\rfloor_p : \mathbb{Z}_q &\longrightarrow \mathbb{Z}_p \\ a &\longmapsto \left\lfloor \frac{p}{q}\cdot a \right\rfloor. \end{aligned} \tag{6}$$

Similar to the LWE problem, we get the following analogous definition for the LWR problem.

Definition 4 (see [23]). For integers n, $q > p$, and m, sample uniformly at random an n-dimensional vector $\mathbf{s}$ from $\mathbb{Z}_q^n$, and then the LWR distribution is defined as $\mathsf{LWR}_{n,q,p}(\mathbf{s}) \triangleq \{(\mathbf{a}, \lceil\langle\mathbf{a}, \mathbf{s}\rangle_q\rfloor_p) \in \mathbb{Z}_q^n \times \mathbb{Z}_p \mid \mathbf{a} \leftarrow_r \mathbb{Z}_q^n\}$, where the pair $(\mathbf{a}, \lceil\langle\mathbf{a}, \mathbf{s}\rangle_q\rfloor_p)$ is LWR sample (instance), and let $\mathsf{LWR}_{n,m,q,p}(\mathbf{s})$ be the distribution comprised of m independent samples from $\mathsf{LWR}_{n,q,p}(\mathbf{s})$. Then, the search $\mathsf{LWR}_{n,m,q,p}$ problem is defined as finding secret $\mathbf{s}$ given m independent samples from $\mathsf{LWR}_{n,q,p}(\mathbf{s})$, while the decision $\mathsf{DLWR}_{n,m,q,p}$ problem is to distinguish m independent samples (with nonnegligible advantage) chosen from the distribution $\mathsf{LWR}_{n,m,q,p}(\mathbf{s})$, for a fixed $\mathbf{s} \leftarrow_r \mathbb{Z}_q^n$, from m samples chosen from uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_p$. The $\mathsf{LWR}_{n,m,q,p}$ assumption is that the $\mathsf{DLWR}_{n,m,q,p}$ problem is infeasible.

As to the hardness of the LWR problem, Banerjee et al. [23] presented an efficient reduction from the LWE problem to the LWR problem for modulus q of superpolynomial size, followed by Alwen et al. [26], who gave a reduction that allowed for a polynomial modulus q, but restricted the number of samples and failed to apply to all values of the modulus q. In 2016, Bogdanov et al. [27] generalized the theorem of [26] by eliminating the theoretic restriction on the modulus q, but the number of samples was required to be less than $O(q/Bp)$ (weaker than that of [26]), while Alperin-Sheriff and Apon [28] showed a dimension-preserving reduction from LWE to LWR with a polynomial-sized modulus, which immediately implies improvements in parameters (i.e., security and efficiency) for all known applications of polymodulus LWR.

Note that Brakerski et al. [24] proved that the bin-LWE (implies that the secret key is uniformly chosen from $\{0,1\}^n$) is at least as hard as the original LWE problem (up to logarithmic loss). Hence, we can uniformly choose the secret $\mathbf{s}$ in LWR from $\{0,1\}^n$ under the hardness of bin-LWE and the following theorem (Theorem 5). Next, we recall the main theorem in [27] (it is sufficient for our schemes, though the number of samples was required to be less than $O(q/Bp)$). Then, by combining this theorem with Lemma 6 (it is a simple fact; here we present it as a lemma), we get our crucial Theorem 7 on which the proposed schemes' security is based. Note that Theorem 7 concerns the search problem of bin-LWE, which is harder than the decision problem of bin-LWE.

Theorem 5 (see [27]). *For every $\varepsilon > 0$, $n, m, q \geq 2mpB$, and if there is an algorithm $\mathscr{A}$ such that*

$$\left|Pr_{\mathbf{A},\mathbf{s}}\left[\mathscr{A}\left(\mathbf{A}, \lceil \mathbf{As} \rfloor_p\right) = 1\right] - Pr_{\mathbf{A},\mathbf{u}}\left[\mathscr{A}\left(\mathbf{A}, \lceil \mathbf{u} \rfloor_p\right) = 1\right]\right| \geq \varepsilon, \tag{7}$$

where $\mathbf{A} \leftarrow_r \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \leftarrow_r \{0,1\}^n$, and $\mathbf{u} \leftarrow_r \mathbb{Z}_q^m$, then there exists an efficient algorithm $\mathscr{B}$ that runs in time polynomial in n, m, the number of divisors of q, and the running time of $\mathscr{A}$ such that

$$Pr_{\mathbf{A},\mathbf{s}}\left[\mathscr{B}\left(\mathbf{A}, \mathbf{As} + \mathbf{e}\right) = \mathbf{s}\right] \geq \left(\frac{\varepsilon}{4qm} - \frac{2^n}{p^m}\right)^2 \cdot \frac{1}{\left(1 + 2Bp/q\right)^m} \tag{8}$$

for noise distribution $\mathbf{e}$ that is B-bounded and balanced in each coordinate.

Lemma 6. *If $p \mid q$, then the distribution $\lceil \mathbf{u} \rfloor_p$ is uniform over $\mathbb{Z}_p^m$ for $\mathbf{u} \leftarrow_r \mathbb{Z}_q^m$ (i.e., the distribution $\lceil \mathbf{u} \rfloor_p$ is equivalent to $\mathbb{Z}_p^m$).*

Theorem 7. *For every $\varepsilon > 0$, $n, m, q \geq 2mpB$, $p \mid q$, and if there is an algorithm $\mathscr{A}$ such that*

$$\left|Pr_{\mathbf{A},\mathbf{s}}\left[\mathscr{A}\left(\mathbf{A}, \lceil \mathbf{As} \rfloor_p\right) = 1\right] - Pr_{\mathbf{A},\mathbf{v}}\left[\mathscr{A}\left(\mathbf{A}, \mathbf{v}\right) = 1\right]\right| \geq \varepsilon, \tag{9}$$

where $\mathbf{A} \leftarrow_r \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \leftarrow_r \{0,1\}^n$, and $\mathbf{v} \leftarrow_r \mathbb{Z}_p^m$, then there exists an efficient algorithm $\mathscr{B}$ that runs in time polynomial in n, m, the number of divisors of q, and the running time of $\mathscr{A}$ such that

$$Pr_{\mathbf{A},\mathbf{s}}\left[\mathscr{B}\left(\mathbf{A}, \mathbf{As} + \mathbf{e}\right) = \mathbf{s}\right] \geq \left(\frac{\varepsilon}{4qm} - \frac{2^n}{p^m}\right)^2 \cdot \frac{1}{\left(1 + 2Bp/q\right)^m} \tag{10}$$

for noise distribution $\mathbf{e}$ that is B-bounded and balanced in each coordinate.

Proof of Sketch. Note that there is no restriction on the condition $p \mid q$ in Theorem 5. In other words, Theorem 5 holds even under the additional condition $p \mid q$. Therefore, Theorem 7 is obvious by combining Lemma 6 which implies that the vector $\lceil \mathbf{u} \rfloor_p$ is equivalent to $\mathbf{v}$. □

We remark that the term $Pr_{\mathbf{A},\mathbf{s}}[\mathscr{A}(\mathbf{A}, \lceil \mathbf{As} \rfloor_p) = 1] - Pr_{\mathbf{A},\mathbf{v}}[\mathscr{A}(\mathbf{A}, \mathbf{v}) = 1]$ in Theorem 7 can be interpreted as the decision $\mathsf{DLWR}_{n,m,q,p}$ problem, for the fixed $\mathbf{s} \leftarrow_r \{0,1\}^n$.

2.3. Variant of Gadget Matrix. The gadget matrix, which was proposed by [29], was used to build a LWE-based FHE scheme of GSW style by AP14, which extended the "flatten" skill in GSW13. Here, we construct a variant of gadget matrix which is specific to our LWR-based FHE scheme.

Since there are two moduli q, p in the LWR problem, a variant of gadget matrix $\mathbf{G}$ with invariant function is as follows:

$$\mathbf{G} \coloneqq \begin{bmatrix} \mathbf{g} & & \\ & \ddots & \\ & & \mathbf{g} \\ & & & \mathbf{g}_{n+1} \end{bmatrix} \in \mathbb{Z}_q^{n\times N}\,[\times]\,\mathbb{Z}_p^N, \tag{11}$$

where $\mathbf{g} \coloneqq (1, 2, \ldots, 2^{\lceil \log q \rceil - 1}) \in \mathbb{Z}_q^{\lceil \log q \rceil}$, $\mathbf{g_{n+1}} \coloneqq (1, 2, \ldots, 2^{\lceil \log p \rceil - 1}) \in \mathbb{Z}_p^{\lceil \log p \rceil}$, and $N = n\lceil \log q \rceil + \lceil \log p \rceil$. We also define the deterministic inversion function $\mathbf{G}^{-1} : \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N \to \{0,1\}^{N\times N}$, which is equal to bit decomposition that decomposes x into its bit representation over $\mathbb{Z}_q$ or $\mathbb{Z}_p$ and has the following property: for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N$, it holds that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$. Note that $\mathbf{G}^{-1}$ is an expensively randomized inversion function in AP14, although it leads to a tighter noise analysis for its homomorphic operations.

3. Building Block

As a warmup, in this section, we present a basic encryption scheme construction based on LWR which is adapted from the ring-LWR-based FHE scheme [22]. Then, based on the basic encryption scheme, we give our LWR-based FHE scheme in the next section. We remark that the secret key is uniformly chosen from $\{0,1\}^n$ to keep the noise growing slower and speed up encryption and decryption processes, under the same (almost) security level as GSW13 and AP14. This can be guaranteed by Theorem 7 in Section 2.2 and the hardness of bin-LWE problem.

3.1. The Basic LWR-Based Encryption Scheme. The basic encryption scheme based on LWR is constructed as follows.

(i) *Setup*(1^λ): choose parameters $n = n(\lambda)$, $\ell = \ell(\lambda)$, moduli $q = q(\lambda)$, and $p = p(\lambda)$ satisfying $p \mid q$ (this condition is needed for the security proof). Let $params = (n, \ell, q, p)$.

(ii) *KeyGen*$(params)$: choose uniformly at random a secret key $\mathbf{s}' \leftarrow_r \{0,1\}^n$. Choose uniformly at random ℓ vectors $\mathbf{a_i} \leftarrow_r \mathbb{Z}_q^n$, and then compute

$$b_i = \left\lceil \left\langle \mathbf{a_i}, \mathbf{s}' \right\rangle_q \right\rfloor_p \in \mathbb{Z}_p \tag{12}$$

for $i \in [\ell]$, which results in ℓ pairs of LWR samples $\{(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_\ell, b_\ell) \in \mathbb{Z}_q^n \times \mathbb{Z}_p\}$. Assemble a matrix $\overline{\mathbf{A}} \in \mathbb{Z}_q^{n\times\ell}$ by the ℓ column vectors $\mathbf{a_i}^t$ and set $\mathbf{b} = (b_1, \ldots, b_\ell)$, and then let $\mathbf{A} = \begin{bmatrix} \overline{\mathbf{A}} \\ \mathbf{b} \end{bmatrix} \in \mathbb{Z}_q^{n\times\ell}[\times]\mathbb{Z}_p^\ell$. The final public key is $pk \coloneqq \mathbf{A}$, and the secret key is $sk : \mathbf{s} = (-p/q \cdot \mathbf{s}', 1)$. For later use, we write $e_i = b_i - (p/q)\langle \mathbf{a_i}, \mathbf{s}' \rangle_q \in [-1/2, 1/2]$ for $i \in [\ell]$. Then, according to (15), it holds that $\mathbf{s} \cdot \mathbf{A} = \mathbf{e} \bmod p$, where the noise $\mathbf{e} = (e_1, \ldots, e_\ell) \in [-1/2, 1/2]^\ell$.

(iii) *Enc*$(params, pk, \mu)$: to encrypt a message $\mu \in \{0,1\}$, choose a random vector $\mathbf{r} \leftarrow_r \{0,1\}^\ell$, and then output

$$\mathbf{c} = \mathbf{A} \cdot \mathbf{r}^t + \left(\mathbf{0}, \mu \cdot \left\lceil \frac{p}{2} \right\rfloor\right)^t \in \mathbb{Z}_q^n \times \mathbb{Z}_p. \tag{13}$$

(iv) *Dec*$(params, sk, \mathbf{c})$: given the ciphertext $\mathbf{c} \in \mathbb{Z}_q^n \times \mathbb{Z}_p$, output

$$\mu = \left\lceil \frac{1}{\lceil p/2 \rfloor} \cdot \langle \mathbf{s}, \mathbf{c} \rangle \right\rfloor \pmod 2. \tag{14}$$

Correctness. As long as the noise $|e| \triangleq |\langle \mathbf{e}, \mathbf{r}\rangle| = |\sum_{i=1}^{\ell} r_i \cdot e_i| \leq 1/2 \cdot \ell < 1/2 \cdot \lceil p/2 \rfloor$, where $r_i \in \{0,1\}$ is the entry of the vector $\mathbf{r}$, the above Dec algorithm can rightly recover the message μ. We prove this by the following steps: For every column vector $(\mathbf{a}_i, b_i)^t$ of the matrix $\mathbf{A}$, it holds that

$$\begin{aligned} \mathbf{s} \cdot (\mathbf{a_i}, b_i)^t &= -\frac{p}{q} \left\langle \mathbf{s}', \mathbf{a_i} \right\rangle + b_i \\ &= -\frac{p}{q} \cdot \left(\left\langle \mathbf{s}', \mathbf{a_i} \right\rangle_q + kq \right) + b_i \\ &= -\frac{p}{q} \left\langle \mathbf{s}', \mathbf{a_i} \right\rangle_q + b_i - kp = e_i - kp. \end{aligned} \tag{15}$$

Then, we have $\mathbf{s} \cdot \mathbf{A} = \mathbf{e} \bmod p$, and it follows that

$$\begin{aligned} &\left\lceil \frac{1}{\lceil p/2 \rfloor} \cdot \langle \mathbf{s}, \mathbf{c} \rangle \right\rfloor \\ &= \left\lceil \frac{1}{\lceil p/2 \rfloor} \cdot \left(\langle \mathbf{e}, \mathbf{r} \rangle + \mu \cdot \left\lceil \frac{p}{2} \right\rfloor - kp \right) \right\rfloor = \mu \bmod 2, \end{aligned} \tag{16}$$

iff $|\langle \mathbf{e}, \mathbf{r}\rangle| \leq 1/2 \cdot \ell < 1/2 \cdot \lceil p/2 \rfloor$.

3.2. Security. We argue that the above basic encryption scheme is IND-CPA secure under Theorem 7. In fact, the structure of our basic encryption scheme is identical to that of [14, 16, 17], and so is the strategy of security proof. In our paper, Theorem 7 guarantees the hardness of $\mathsf{LWR}_{n,m,q,p}$ assumption (where $\mathbf{s} \leftarrow_r \{0,1\}^n$) assuming the bin-LWE problem; hence, we can base the security of our basic encryption scheme on the LWR problem. For completeness, we give a full security proof below.

Theorem 8. *The above basic* LWR*-based encryption scheme is IND-CPA secure under the* $\mathsf{LWR}_{n,m,q,p}$ *assumption.*

Proof. We prove the theorem via a series of hybrid experiments. The analysis of probability is omitted here, since it is natural and very similar to that of [4, 14].

(i) Hybrid 0: this is the real system (we set $\ell = m$, where $m = O(n \log q + \log p) \geq n \log q + \log p + 2\lambda$).

(ii) Hybrid 1: this is the same as Hybrid 0 except that we use m pairs $\{(\mathbf{a_i}, b_i)\}_{i=1,\ldots,m}$ chosen uniformly from $\mathbb{Z}_q^n \times \mathbb{Z}_p$, instead of being chosen from LWR samples in Hybrid 0, to assemble public key $\mathbf{A}$. The ciphertext $\mathbf{c}$ is thus generated by encrypting μ using the public key $\mathbf{A}$ as per the encryption procedure in Hybrid 0.

(iii) Hybrid 2: this is the same as Hybrid 1 except that the ciphertext $\mathbf{c} \in \mathbb{Z}_q^n \times \mathbb{Z}_p$ is chosen from $\mathbb{Z}_q^n \times \mathbb{Z}_p$ uniformly and independently of the public key.

Firstly, we claim that Hybrid 1 is indistinguishable from Hybrid 0, under the $\mathsf{LWR}_{n,m,q,p}$ assumption. This is shown by a simple reduction: if any hypothetical adversary $\mathscr{A}$ can distinguish these two hybrids, then we can construct an algorithm $\mathscr{B}$ to break the $\mathsf{LWR}_{n,m,q,p}$ assumption. Specifically, $\mathscr{B}$ simply collects its input samples (challenged samples) into the public key $\mathbf{A}$ and encrypts a message μ using the public key $\mathbf{A}$ to get a ciphertext $\mathbf{c}$, and then it invokes $\mathscr{A}$ on $(\mathbf{A}, \mathbf{c})$, outputting the same accept/reject decision. It is clear that $\mathscr{B}$ perfectly simulates Hybrid 0 or Hybrid 1 depending on whether its input samples are LWR samples or uniform over $\mathbb{Z}_q^n \times \mathbb{Z}_p$; that is, if the input samples are LWR samples, then the ciphertext $\mathbf{c}$ is generated as per Hybrid 0; otherwise, the ciphertext $\mathbf{c}$ is generated as per Hybrid 1. Therefore, $\mathscr{B}$ and $\mathscr{A}$ have equal distinguishing advantages. Because $\mathscr{B}$'s advantage must be negligible by hypothesis, so is $\mathscr{A}$'s.

In the second place, we claim that Hybrid 1 is statistically indistinguishable from Hybrid 2; that is, even a computationally unbounded adversary has only negligible advantage in distinguishing them. This is easy to be justified by Lemma 1 in Section 2. In more detail, by Lemma 1, the term $\mathbf{A} \cdot \mathbf{r}$ in Hybrid 1 is statistically indistinguishable from any element $\mathbf{y} \in \mathbb{Z}_q^n \times \mathbb{Z}_p$. Hence, the ciphertext in Hybrid 1 is statistically indistinguishable from that of Hybrid 2, for adding any fixed vector $(\mathbf{0}, \mu \cdot \lceil p/2 \rfloor)^t$ to $\mathbf{A} \cdot \mathbf{r}$ preserves its uniform distribution.

To sum up, we conclude that Hybrid 0 is indistinguishable from Hybrid 2, and the ciphertext in Hybrid 2 is independent of message μ (regardless of $\mu = 0$ or $\mu = 1$), which completes the proof. □

4. LWR-Based FHE Scheme

4.1. Our Construction. Our LWR-based FHE scheme is constructed as follows.

(i) *Setup*$(1^\lambda, 1^L)$: choose dimension parameter $n = n(\lambda, L)$ and moduli $q = q(\lambda, L)$ and $p = p(\lambda, L)$ satisfying $p \mid q$. Also, choose parameter $m = m(\lambda, L) = O(n \log q + \log p)$. Let $params = (n, q, p, m)$. Let $k_1 = \lceil \log q \rceil$, $k_2 = \lceil \log p \rceil$, and $N = n \cdot k_1 + k_2$.

(ii) *KeyGen*$(params)$: choose uniformly at random a secret key $\mathbf{s}' \leftarrow_r \{0,1\}^n$. As per KeyGen algorithm in Section 3, generate a matrix

$$\mathbf{A} = \begin{bmatrix} \overline{\mathbf{A}} \\ \mathbf{b} \end{bmatrix} \in \mathbb{Z}_q^{n \times m} [\times] \mathbb{Z}_p^m. \tag{17}$$

The final public key is $pk := \mathbf{A}$, and the secret key is $sk : \mathbf{s} = (-p/q \cdot \mathbf{s}', 1)$. Note that it holds that $\mathbf{s} \cdot \mathbf{A} = \mathbf{e} \bmod p$, where the noise $\mathbf{e} = (e_1, \ldots, e_m) \in [-1/2, 1/2]^m$.

(iii) *Enc*$(params, pk, \mu)$: to encrypt a message $\mu \in \{0,1\}$, choose a random matrix $\mathbf{R} \leftarrow_r \{0,1\}^{m \times N}$. Output the ciphertext

$$\mathbf{C} = \mathbf{A} \cdot \mathbf{R} + \mu \mathbf{G} \in \mathbb{Z}_q^{n \times N} [\times] \mathbb{Z}_p^N, \tag{18}$$

where the gadget matrix $\mathbf{G}$ is brought from Section 2.3. Since we should perform the above matrix multiplication as per the specific matrix operations described in Section 2, we give Algorithm 1 that makes the encryption algorithm more readable.

Input: Parameters, a public-key matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}[\times]\mathbb{Z}_p^m$ and a message $\mu \in \{0,1\}$.
Output: A ciphertext matrix $\mathbf{C} = \mathbf{A}\cdot\mathbf{R} + \mu\mathbf{G} \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N$.
(1) Generate a variant gadget matrix $\mathbf{G}$ according to the structure of gadget matrix given in Section 2.3.
(2) Choose uniformly at random a matrix $\mathbf{R} \leftarrow_r \{0,1\}^{m\times N}$.
(3) Compute $\mathbf{C} = \mathbf{A}\cdot\mathbf{R} + \mu\mathbf{G} \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N$ as per our specific matrix operations described in Section 2.

Algorithm 1: Encryption algorithm.

Input: Parameters, a secret key $sk = \mathbf{s}$ and a ciphertext matrix $\mathbf{C} \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N$.
Output: A binary value $\mu \in \{0,1\}$.
(1) Choose a vector $\mathbf{v} := (0,\ldots,0,\lceil p/2\rfloor) \in \mathbb{Z}_p^{n+1}$, and compute a vector $\mathbf{r} \in \{0,1\}^N$, such that $\mathbf{G}\cdot\mathbf{r}^t = \mathbf{v}$, i.e., $\mathbf{r}^t = \mathbf{G}^{-1}(\mathbf{v}^t)$.
(2) Compute

$$v = \mathbf{s}\cdot\mathbf{C}\cdot\mathbf{r}^t = \mathbf{e}\cdot\mathbf{R}\cdot\mathbf{r}^t + \mu\cdot\left\lceil\frac{p}{2}\right\rfloor.$$

(3) Output $\mu = \lceil 1/\lceil p/2\rfloor\cdot v\rfloor \pmod 2$.

Algorithm 2: Decryption algorithm.

Input: Two ciphertext matrices $\mathbf{C}_1, \mathbf{C}_2$ decrypting to messages μ_1, μ_2, respectively.
Output: A derived ciphertext matrix $\mathbf{C}_{\text{mult}}$ decrypting to message $\mu_1\mu_2$.
(1) Compute a matrix $\mathbf{X} \in \{0,1\}^{N\times N}$, such that $\mathbf{G}\cdot\mathbf{X} = \mathbf{C}_2$, i.e., $\mathbf{X} = \mathbf{G}^{-1}(\mathbf{C}_2)$.
(2) Output

$$\mathbf{C}_{\text{mult}} = \mathbf{C}_1\cdot\mathbf{G}^{-1}(\mathbf{C}_2) \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N.$$

Algorithm 3: Homomorphic multiplication.

(iv) $Dec(params, sk, \mathbf{C})$: given the ciphertext $\mathbf{C} \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N$, choose a vector $\mathbf{v} := (0,\ldots,0,\lceil p/2\rfloor) \in \mathbb{Z}_p^{n+1}$ and compute

$$v = \mathbf{s}\cdot\mathbf{C}\cdot\mathbf{G}^{-1}\left(\mathbf{v}^t\right) = \mathbf{e}\cdot\mathbf{R}\cdot\mathbf{G}^{-1}\left(\mathbf{v}^t\right) + \mu\cdot\left\lceil\frac{p}{2}\right\rfloor. \quad (19)$$

Output $\mu = \lceil 1/\lceil p/2\rfloor\cdot v\rfloor \pmod 2$. We also give Algorithm 2 that makes the decryption algorithm more readable.

(v) $Add(\mathbf{C}_1, \mathbf{C}_2)$: given two ciphertext matrices $\mathbf{C}_1, \mathbf{C}_2$ decrypting to messages μ_1 and μ_2, respectively, output

$$\mathbf{C}_{\text{Add}} \triangleq \mathbf{C}_1 + \mathbf{C}_2 \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N. \quad (20)$$

(vi) $Mult(\mathbf{C}_1, \mathbf{C}_2)$: for two ciphertext matrices $\mathbf{C}_1, \mathbf{C}_2$ decrypting to messages μ_1 and μ_2, respectively, the multiplication is defined as

$$\mathbf{C}_{\text{mult}} \triangleq \mathbf{C}_1\cdot\mathbf{G}^{-1}\left(\mathbf{C}_2\right) \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N, \quad (21)$$

where $\mathbf{G}^{-1}$ is brought from Section 2.3. We give Algorithm 3 only for homomorphic multiplication, because homomorphic addition is trivial, and homomorphic NAND depends on homomorphic multiplication.

(vii) $NAND(\mathbf{C}_1, \mathbf{C}_2)$: for two ciphertext matrices $\mathbf{C}_1, \mathbf{C}_2$ decrypting to messages μ_1 and μ_2, respectively, the NAND operation is defined as

$$\mathbf{C}_{\text{NAND}} \triangleq \mathbf{G} - \mathbf{C}_1\cdot\mathbf{G}^{-1}\left(\mathbf{C}_2\right) \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N. \quad (22)$$

Note that any Boolean circuit can be converted to use only NAND operation.

(viii) $Eval(f, (\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_t))$: evaluate the Boolean function $f : \{0,1\}^t \to \{0,1\}$ on t ciphertexts $\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_t$ by the NAND operation. Output a ciphertext $\mathbf{C}_f$.

The IND-CPA security of the above scheme follows immediately from the security of the basic encryption scheme in Section 3. The ciphertext matrix is just $\mu\mathbf{G}$ plus a matrix of N encryptions of 0 under the key $\mathbf{s}$ by the basic encryption in Section 3, which is pseudorandom by Theorem 8 and hence hides $\mu\mathbf{G}$.

4.2. Correctness. In this subsection, we analyze the scheme's correctness and the noise growth of each homomorphic operation. Moreover, we also give a more detailed analysis for homomorphic multiplication to illustrate that our proposed construction is practicable.

Correctness. The correctness is obvious according to the correctness of the basic encryption scheme in Section 3, iff the

magnitude of noise $\mathbf{e}^* \triangleq \mathbf{e} \cdot \mathbf{R} \cdot \mathbf{G}^{-1}(\mathbf{v}^t)$ is less than $1/2 \cdot \lceil p/2 \rfloor$. In fact, according to the Dec, we have

$$\mathbf{s} \cdot \mathbf{C} = \mu \mathbf{s} \cdot \mathbf{G} + \mathbf{e} \cdot \mathbf{R} \bmod p, \tag{23}$$

and then

$$\mathbf{s} \cdot \mathbf{C} \cdot \mathbf{G}^{-1}\left(\mathbf{v}^t\right) = \mu \cdot \left\lceil \frac{p}{2} \right\rfloor + \mathbf{e} \cdot \mathbf{R} \cdot \mathbf{G}^{-1}\left(\mathbf{v}^t\right) \bmod p, \tag{24}$$

where $\|\mathbf{e}^*\| \triangleq \|\mathbf{e} \cdot \mathbf{R} \cdot \mathbf{G}^{-1}(\mathbf{v}^t)\| < (1/2)m \cdot \lceil \log p \rfloor$. Hence, the correctness holds iff $(1/2)m \cdot \lceil \log p \rfloor < (1/2) \cdot \lceil p/2 \rfloor$.

Since the homomorphic addition is obvious, we mainly analyze homomorphic multiplication and NAND operation.

Homomorphic Multiplication. To multiply two ciphertext matrices $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{n \times N}[\times]\mathbb{Z}_p^N$ designated for messages μ_1, $\mu_2 \in \{0, 1\}$, we have

$$\begin{aligned} \mathbf{s} \cdot \mathsf{Mult}\left(\mathbf{C}_1, \mathbf{C}_2\right) &= \mathbf{s} \cdot \mathbf{C}_1 \cdot \mathbf{G}^{-1}\left(\mathbf{C}_2\right) \\ &= \left(\mathbf{e}_1 + \mu_1 \mathbf{s} \cdot \mathbf{G}\right) \cdot \mathbf{G}^{-1}\left(\mathbf{C}_2\right) \\ &= \mathbf{e}_1 \cdot \mathbf{G}^{-1}\left(\mathbf{C}_2\right) + \mu_1 \left(\mathbf{e}_2 + \mu_2 \mathbf{s} \cdot \mathbf{G}\right) \\ &= \left(\mathbf{e}_1 \cdot \mathbf{G}^{-1}\left(\mathbf{C}_2\right) + \mu_1 \mathbf{e}_2\right) + \mu_1 \mu_2 \mathbf{s} \\ &\quad \cdot \mathbf{G}, \end{aligned} \tag{25}$$

where $\mathbf{G}^{-1}(\mathbf{C}_2) \in \{0, 1\}^{N \times N}$ and $\mathbf{e}_1 \cdot \mathbf{G}^{-1}(\mathbf{C}_2) + \mu_1 \mathbf{e}_2$ is the total noise which is of magnitude

$$\left\|\mathbf{e}_1 \cdot \mathbf{G}^{-1}\left(\mathbf{C}_2\right) + \mu_1 \mathbf{e}_2\right\| \leq \frac{1}{2} m (N + 1), \tag{26}$$

due to $\|\mathbf{e}_i\|_{i \in [2]} \leq (1/2)m$ by the correctness of our basic encryption scheme. Therefore, it is clear that the noise growth factor is $N + 1$, and after L levels of homomorphic multiplication, the noise grows from initial magnitude of $(1/2)m$ to $(1/2)m(N + 1)^L$. By comparison, in GSW13, the noise at L level is near $m' B (N' + 1)^L$, where $m' = O(n \log q)$, $N' = (n + 1) \cdot \lceil \log q \rceil, B \geq \omega(\log n) \cdot \sqrt{n}$.

Homomorphic NAND. To perform NAND operation on two ciphertext matrices $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{n \times N}[\times]\mathbb{Z}_p^N$ designated for messages $\mu_1, \mu_2 \in \{0, 1\}$, we have

$$\begin{aligned} \mathbf{s} \cdot \mathsf{NAND}\left(\mathbf{C}_1, \mathbf{C}_2\right) &= \mathbf{s} \cdot \left(\mathbf{G} - \mathbf{C}_1 \cdot \mathbf{G}^{-1}\left(\mathbf{C}_2\right)\right) \\ &= \left(\mathbf{e}_1 \cdot \mathbf{G}^{-1}\left(\mathbf{C}_2\right) + \mu_1 \mathbf{e}_2\right) \\ &\quad + \left(1 - \mu_1 \mu_2\right) \mathbf{s} \cdot \mathbf{G}, \end{aligned} \tag{27}$$

where $\mathbf{e}_1 \cdot \mathbf{G}^{-1}(\mathbf{C}_2) + \mu_1 \mathbf{e}_2$ is the noise of homomorphic NAND, which is identical to that of homomorphic multiplication. Therefore, the noise is also increased by a factor of at most $N + 1$.

Analysis for Homomorphic Multiplication. Given two ciphertext matrices $\mathbf{C} = \mathbf{A} \cdot \mathbf{R} + \mu \mathbf{G}, \mathbf{C}' = \mathbf{A} \cdot \mathbf{R}' + \mu' \mathbf{G} \in \mathbb{Z}_q^{n \times N}[\times]\mathbb{Z}_p^N$ designated for messages $\mu, \mu' \in \{0, 1\}$, we let $\mathbf{C} = \begin{bmatrix} \mathbf{C}_q \\ \mathbf{c}_p \end{bmatrix}$, $\mathbf{C}' = \begin{bmatrix} \mathbf{C}'_q \\ \mathbf{c}'_p \end{bmatrix} \in \mathbb{Z}_q^{n \times N}[\times]\mathbb{Z}_p^N$. Recall that

$$\begin{aligned} &\mathbf{G} \\ &= \begin{bmatrix} 1, \ldots, 2^{\lceil \log q \rceil - 1} & & & \\ & \ddots & & \\ & & 1, \ldots, 2^{\lceil \log q \rceil - 1} & \\ & & & 1, \ldots, 2^{\lceil \log p \rceil - 1} \end{bmatrix} \\ &\in \mathbb{Z}_q^{n \times N}[\times]\mathbb{Z}_p^N. \end{aligned} \tag{28}$$

Compute $\mathbf{G}^{-1}(\mathbf{C}')$ and let $\mathbf{X} = \mathbf{G}^{-1}(\mathbf{C}')$, that is, $\mathbf{G} \cdot \mathbf{X} = \mathbf{C}'$, and then we have $\mathbf{X} = \begin{bmatrix} \mathbf{X}_q \\ \mathbf{X}_p \end{bmatrix} \in \{0, 1\}^{N \times N}$, where $\mathbf{X}_q \in \{0, 1\}^{(N - \lceil \log p \rceil) \times N}$ and $\mathbf{X}_p \in \{0, 1\}^{\lceil \log p \rceil \times N}$. Then, we have

$$\begin{aligned} \mathsf{Mult}\left(\mathbf{C}, \mathbf{C}'\right) &= \mathbf{C} \cdot \mathbf{G}^{-1}\left(\mathbf{C}'\right) = \left(\mathbf{A} \cdot \mathbf{R} + \mu \mathbf{G}\right) \cdot \begin{bmatrix} \mathbf{X}_q \\ \mathbf{X}_p \end{bmatrix} \\ &= \mathbf{A} \cdot \mathbf{R} \cdot \begin{bmatrix} \mathbf{X}_q \\ \mathbf{X}_p \end{bmatrix} + \mu \mathbf{C}' \\ &= \mathbf{A} \cdot \mathbf{R} \cdot \begin{bmatrix} \mathbf{X}_q \\ \mathbf{X}_p \end{bmatrix} + \mathbf{A} \cdot \mu \mathbf{R}' + \mu \mu' \mathbf{G} \\ &= \mathbf{A} \cdot \left(\mathbf{R} \cdot \mathbf{X} + \mu \mathbf{R}'\right) + \mu \mu' \mathbf{G}, \end{aligned} \tag{29}$$

where, as per the specific matrix multiplication described in Section 2, we can compute $\mathbf{A} \cdot \mathbf{R} \cdot \mathbf{X} \in \mathbb{Z}_q^{n \times N}[\times]\mathbb{Z}_p^N$. Since $\mathbf{s} \cdot \mathbf{A} = \mathbf{e} \bmod p$, $\mathbf{X} = \begin{bmatrix} \mathbf{X}_q \\ \mathbf{X}_p \end{bmatrix}$ does not affect the correctness of $\mathbf{s} \cdot (\mathbf{A} \cdot \mathbf{R} \cdot \mathbf{X}) = \mathbf{e} \cdot \mathbf{R} \cdot \mathbf{X} \bmod p$ (recall that $\mathbf{s} = (-p/q \cdot \mathbf{s}', 1)$). Therefore, (25) holds.

4.3. Parameters and Comparisons. Compared to GSW13 and AP14, we conclude, from the above analysis, that our noise growth factor is also $N + 1$ (identical to theirs) and the size of parameter q is almost the same as theirs. Hence, we claim that the efficiency of our scheme is almost comparable to theirs without considering the cost of the Gaussian noise sampling, since our encryption and decryption processes are identical to theirs. Concretely, we assume the depth of NAND operation is L, and after L levels of homomorphic NAND operation, the noise is near $(1/2)m(N + 1)^L$, and it holds that $(1/2)m(N + 1)^L < 1/2 \cdot \lceil p/2 \rfloor$ for decryption. Then, combining with the requirements of $q \geq 2mpB$ in Theorem 7 in Section 2, and $m = O(n \log q + \log p)$, $N = n\lceil \log q \rceil + \lceil \log p \rceil$ in our scheme, we can set $q = (O(n \log q))^{L + O(1)}$, $p = (O(n \log p))^{L + O(1)}$ satisfying $q \geq 2mpB$. Compared to the parameter q of $q = (O(n \log q))^{L + O(1)}$ (since $q \geq 8m' B(N' + 1)^L$) in GSW13 and AP14, our parameter q is larger (up to a polynomial factor); this is caused by the condition $q \geq 2mpB$ needed for achieving the same (almost) security level as GSW13 and AP14. Therefore, in consideration of the cost (e.g., running time, memory) of Gaussian or sub-Gaussian

noise sampling in GSW13 and AP14, our scheme has some advantages over them and thus can be seen as an alternative to them. Actually, with the development of reduction between LWE and LWR, the moduli can be reduced.

Bootstrapping involves homomorphically evaluating the decryption function. Because our decryption process is identical to AP14, bootstrapping works on our proposed scheme as well. In more detail, similar to AP14, we can also directly evaluate the decryption function in an elementary and efficient arithmetic form, using just basic facts about cyclic groups, so as to achieve bootstrapping. In our scheme, the bootstrapping method also just results in polynomial noise, which allows the security to be based on the LWR problem with inverse-polynomial noise rates. Since the reduction from LWE to LWR incurs a polynomial loss ($O(qn\log q)$) according to Theorem 7 in Section 2, the security can be based on LWE problem with inverse-polynomial noise rates and hence on worst-case lattice problems (e.g., GapSVP) with polynomial approximation factors.

5. LWR-Based Multikey FHE Scheme

FHE has been studied extensively also due to its versatility in cryptography and industrial application, such as the problem of outsourcing computation to a third trust party (a remote server) without compromising its privacy [5] and secure multiparty computation problem [12, 30]. In Eurocrypt 2016, Mukherjee and Wichs [12] extended AP14 to multikey case and presented a multikey FHE scheme for implementing two round multiparty computations, which enabled performing homomorphic computation over encrypted data under different keys and satisfied threshold decryption. Their solution, however, relies on the LWE problem, the efficiency of which is retarded by the sub-Gaussian noise sampling. But the tricks in [12] inspire us to construct a multikey FHE scheme based on LWR. In fact, our construction also satisfies threshold decryption; we will describe it in Section 5.3. Here, we just present some necessary tools and our main construction, and we refer interested readers to [12] for the definitions of multikey FHE and multiparty computation, the security definition of multiparty computation protocol, and some other related concepts.

5.1. Overview of the Techniques. In this subsection, we will describe how to construct a LWR-based multikey FHE scheme (adapted from [12]); this helps readers understand our construction more easily. For simplicity, we consider a case of two keys, for it is natural to generalize to the case of any polynomial number of keys. According to (23) in Section 4.2, given two ciphertexts $\mathbf{C}_1$ and $\mathbf{C}_2$ decrypting to messages μ_1 and μ_2, we can perform homomorphic operations on these two ciphertexts, iff they satisfy the equation $\mathbf{s}\cdot\mathbf{C} = \mu_i\mathbf{s}\cdot\mathbf{G} + \mathbf{e}_i$ for small noise $\mathbf{e}_i$, where $i \in [2]$. Hence, for the "combined secret key" $\widehat{\mathbf{s}} = (\mathbf{s}_1, \mathbf{s}_2)$, where $\mathbf{s}_1 = (-p/q\cdot\mathbf{s}_1', 1)$ and $\mathbf{s}_2 = (-p/q\cdot\mathbf{s}_2', 1)$ which correspond to two different users, if we can construct two expanded ciphertexts $\widehat{\mathbf{C}}_1$ and $\widehat{\mathbf{C}}_2$, such that $\widehat{\mathbf{s}}\cdot\widehat{\mathbf{C}}_1 \approx \mu_1\widehat{\mathbf{s}}\cdot\widehat{\mathbf{G}}$ and $\widehat{\mathbf{s}}\cdot\widehat{\mathbf{C}}_2 \approx \mu_2\widehat{\mathbf{s}}\cdot\widehat{\mathbf{G}}$ (henceforth, $\approx$ is used for omitting the noise term $\mathbf{e}$), where $\widehat{\mathbf{G}}$ is the expansion of $\mathbf{G}$, then we can perform homomorphic operations on these two ciphertexts $\widehat{\mathbf{C}}_1$ and $\widehat{\mathbf{C}}_2$.

Exactly as Mukherjee and Wichs [12] showed, we can create the expanded ciphertext as follows:

$$\widehat{\mathbf{C}}_1 \coloneqq \begin{bmatrix} \mathbf{C}_1 & \mathbf{X}_1 \\ \mathbf{0} & \mathbf{C}_1 \end{bmatrix}, \tag{30}$$

such that

$$\widehat{\mathbf{s}}\cdot\widehat{\mathbf{C}}_1 \approx \mu_1\widehat{\mathbf{s}}\widehat{\mathbf{G}}, \tag{31}$$

iff

$$\mathbf{s}_1\cdot\mathbf{X}_1 + \mathbf{s}_2\cdot\mathbf{C}_1 \approx \mu_1\mathbf{s}_2\cdot\mathbf{G}, \tag{32}$$

where $\widehat{\mathbf{G}} \coloneqq \left[\begin{smallmatrix} \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{G} \end{smallmatrix}\right]$. Similarly, we can also generate the ciphertext $\widehat{\mathbf{C}}_2$. Here, we have made a big step forward; next, we proceed to analyze how to construct this $\mathbf{X}$ without revealing the secret key $\mathbf{s}_2$.

Assume we are given a user-specific public key $\mathbf{A}_1 = \left[\begin{smallmatrix} \overline{\mathbf{A}} \\ \mathbf{b}_1 \end{smallmatrix}\right]$, a secret key $\mathbf{s}_1 = (-p/q\cdot\mathbf{s}_1', 1)$, and a ciphertext

$$\mathbf{C}_1 = \mathbf{A}_1\cdot\mathbf{R}_1 + \mu\mathbf{G} = \begin{bmatrix} \overline{\mathbf{A}}\cdot\mathbf{R}_1 \\ \mathbf{b}_1\cdot\mathbf{R}_1 \end{bmatrix} + \mu\mathbf{G}, \tag{33}$$

which is generated as per Enc algorithm in Section 4. Assume we are also given another user-specific public key $\mathbf{A}_2 = \left[\begin{smallmatrix} \overline{\mathbf{A}} \\ \mathbf{b}_2 \end{smallmatrix}\right]$ and secret key $\mathbf{s}_2 = (-p/q\cdot\mathbf{s}_2', 1)$ (assume $\overline{\mathbf{A}}$ is the public shared parameter), such that $\mathbf{s}_2\cdot\mathbf{A}_2 = \mathbf{e} \bmod p$; then, we have

$$\begin{aligned} \mathbf{s}_2\cdot\mathbf{C}_1 &= -\frac{p}{q}\cdot\mathbf{s}_2'\cdot\overline{\mathbf{A}}\cdot\mathbf{R}_1 + \mathbf{b}_1\cdot\mathbf{R}_1 + \mu_1\mathbf{s}_2\cdot\mathbf{G} \\ &\approx (\mathbf{b}_1 - \mathbf{b}_2)\cdot\mathbf{R}_1 + \mu_1\mathbf{s}_2\cdot\mathbf{G}. \end{aligned} \tag{34}$$

Subtracting (34) from (32), we get $\mathbf{s}_1\cdot\mathbf{X}_1 \approx (\mathbf{b}_2 - \mathbf{b}_1)\cdot\mathbf{R}_1$ over $\mathbb{Z}_p$. We proceed to use an important tool brought from [12] to construct such matrix $\mathbf{X}$.

Linear Combination. For a matrix $\mathbf{R} \in \{0,1\}^{m\times N}$ with mN entries $\{\mathbf{R}[i,j]\}$ for $i \in [m]$, $j \in [N]$, let $\mathbf{C}^{(i,j)} \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N$ be LWR-based FHE encryption of $\mathbf{R}[i,j]$ under a secret key $\mathbf{s} = (-p/q\cdot\mathbf{s}', 1)$, where $\mathbf{s}' \leftarrow_r \{0,1\}^n$. Let $\mathbf{b} \in \mathbb{Z}_p^m$ be a vector, where $\mathbf{b}[i]$ is its ith entry. Then, there is a polynomial time deterministic algorithm

$$\mathbf{C}^{\mathrm{lc}} = \mathsf{LComb}\left(\left(\mathbf{C}^{(1,1)}, \ldots, \mathbf{C}^{(m,N)}\right), \mathbf{b}\right), \tag{35}$$

which outputs $\mathbf{C}^{\mathrm{lc}} \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N$ such that $\mathbf{s}\cdot\mathbf{C}^{\mathrm{lc}} = \mathbf{b}\cdot\mathbf{R} + \mathbf{e}$, where $\|\mathbf{e}\|_\infty \le (1/2)(mN)^2$.

The algorithm $\mathsf{LComb}((\mathbf{C}^{(1,1)}, \ldots, \mathbf{C}^{(m,N)}), \mathbf{b})$ is implemented as follows:

(1) For each $i \in [m]$, $j \in [N]$, define a matrix $\mathbf{Z}_{i,j} \in \mathbb{Z}_p^{m\times N}$ as follows:

$$\mathbf{Z}_{i,j}[a,b] = \begin{cases} \mathbf{b}[i] & \text{when } a = n,\ b = j, \\ 0 & \text{otherwise.} \end{cases} \tag{36}$$

In other words, $\mathbf{Z}_{i,j}$ will be 0 everywhere except the nth row and jth column where it has the value $\mathbf{b}[i]$.

(2) Output $\mathbf{C}^{\mathrm{lc}} \in \mathbb{Z}_q^{n\times N}[\times]\mathbb{Z}_p^N$, where $\mathbf{C}^{\mathrm{lc}} = \sum_{i=1,j=1}^{m,N} \mathbf{C}^{(i,j)} \cdot \mathbf{G}^{-1}(\mathbf{Z}_{i,j})$.

Then, we analyze its correctness as follows:

$$\begin{aligned}\mathbf{s}\cdot\mathbf{C}^{\mathrm{lc}} &= \mathbf{s}\cdot\sum_{i,j}^{m,N}\mathbf{C}^{(i,j)}\cdot\mathbf{G}^{-1}\left(\mathbf{Z}_{i,j}\right)\\ &= \sum_{i,j}^{m,N}\left(\mathbf{R}\left[i,j\right]\cdot\mathbf{s}\cdot\mathbf{G}+\mathbf{e}_{i,j}\right)\cdot\mathbf{G}^{-1}\left(\mathbf{Z}_{i,j}\right)\\ &= \sum_{i,j}^{m,N}\left(\mathbf{R}\left[i,j\right]\cdot\mathbf{s}\cdot\mathbf{Z}_{i,j}+\mathbf{e}'_{i,j}\right)\\ &= \mathbf{s}\cdot\sum_{i,j}^{m,N}\mathbf{R}\left[i,j\right]\cdot\mathbf{Z}_{i,j}+\sum_{i,j}^{m,N}\mathbf{e}'_{i,j}\\ &= \left(-\frac{p}{q}\cdot\mathbf{s}',1\right)\cdot\begin{bmatrix}\mathbf{0}\\ \mathbf{b}\cdot\mathbf{R}\end{bmatrix}+\mathbf{e}'' = \mathbf{b}\cdot\mathbf{R}+\mathbf{e}'',\end{aligned} \tag{37}$$

where $\mathbf{e}_{i,j}$ is the noise contained in $\mathbf{C}^{(i,j)}$ with magnitude of $\|\mathbf{e}_{i,j}\|_\infty \le (1/2)m$ (according to Section 3), and then $\mathbf{e}'_{i,j} = \mathbf{e}_{i,j}\cdot\mathbf{G}^{-1}(\mathbf{Z}_{i,j})$ has magnitude $\|\mathbf{e}'_{i,j}\|_\infty \le (1/2)mN$, and finally $\mathbf{e}'' = \sum_{i,j}^{m,N}\mathbf{e}'_{i,j}$ has magnitude $\|\mathbf{e}''\|_\infty \le (1/2)(mN)^2$.

5.2. Construction of Two-Key FHE Scheme from LWR. For ease of description, here we just give a construction of two-key FHE scheme based on LWR, for everything extends naturally to any polynomial number of keys.

Construction. Now, we describe the two-key FHE construction.

(i) *TFHE.Setup*$(1^\lambda, 1^L)$: run the Setup and KeyGen algorithms in Section 4 to generate the parameters:

$$params := \left(n,q,p,m,\overline{\mathbf{A}}\right). \tag{38}$$

(ii) *TFHE.KeyGen*(*params*): run the KeyGen algorithms in Section 4 to get

$$\begin{aligned}sk:\mathbf{s}_i &= \left(-\frac{p}{q}\cdot\mathbf{s}'_i,1\right),\\ pk:\mathbf{A}_i &= \begin{bmatrix}\overline{\mathbf{A}}\\ \mathbf{b}_i\end{bmatrix}\in\mathbb{Z}_q^{n\times m}[\times]\,\mathbb{Z}_p^m,\\ &\qquad\text{for } i\in[2].\end{aligned} \tag{39}$$

(iii) *TFHE.Enc*$(params, pk_i, \mu_i)$: for $i \in [2]$, run the Enc algorithm in Section 4 to get a ciphertext $\mathbf{C}_i$ and the corresponding random matrix $\mathbf{R}_i \in \{0,1\}^{m\times N}$. For all entries $\{\mathbf{R}_i[i,j]\}_{i\in[m],j\in[N]}$ of the uniform matrix $\mathbf{R}_i \in \{0,1\}^{m\times N}$, we use the same secret key $\mathbf{s}_i$ to encrypt them, which leads to a helper set $\mathscr{H}_i = \{\mathbf{C}_i^{(i,j)}\}_{i\in[m],j\in[N]}$. Output the ciphertext $\mathbf{C}_i$ and the helper set $\mathscr{H}_i$.

(iv) *TFHE.Expand*$(params, (pk_1, pk_2), \mathbf{C}_i, \mathscr{H}_i)$: given $\mathbf{C}_i$ and $\mathscr{H}_i$, when $i = 1$, run the linear combination algorithm to generate a matrix $\mathbf{X}_1 = \mathsf{LComb}((\mathbf{C}_1^{(1,1)}, \ldots, \mathbf{C}_1^{(m,N)}), \mathbf{b}_2 - \mathbf{b}_1)$, such that

$$\mathbf{s}_1\cdot\mathbf{X}_1 = \left(\mathbf{b}_2-\mathbf{b}_1\right)\cdot\mathbf{R}_1+\mathbf{e}_{\mathbf{X}1}. \tag{40}$$

Finally, output an expanded ciphertext:

$$\widehat{\mathbf{C}}_1 := \begin{bmatrix}\mathbf{C}_1 & \mathbf{X}_1\\ \mathbf{0} & \mathbf{C}_1\end{bmatrix}, \tag{41}$$

where ciphertext $\mathbf{C}_1$ corresponds to the message μ_1. Similarly, when $i = 2$, output an expanded ciphertext $\widehat{\mathbf{C}}_2$.

(v) *TFHE.Eval*$(params, f, (\widehat{\mathbf{C}}_1, \widehat{\mathbf{C}}_2))$: on inputting two expanded ciphertexts, run Eval algorithm in Section 4, albeit with expanded dimensions $n' = 2n+2$ and $N' = 2N$, and the gadget matrix $\widehat{\mathbf{G}} := \left[\begin{smallmatrix}\mathbf{G} & \mathbf{0}\\ \mathbf{0} & \mathbf{G}\end{smallmatrix}\right]$. Output the ciphertext $\widehat{\mathbf{C}}_f$.

(vi) *TFHE.Dec*$(params, (sk_1, sk_2), \widehat{\mathbf{C}}_f)$: on inputting a ciphertext $\widehat{\mathbf{C}}_f$ and two secret keys sk_1 and sk_2, parse $sk_1 = \mathbf{s}_1$ and $sk_2 = \mathbf{s}_2$, and then obtain a new secret key $\widehat{\mathbf{s}} = (\mathbf{s}_1, \mathbf{s}_2)$. Finally, run the Dec algorithm in Section 4 to get a plaintext $f(\mu_1, \mu_2)$.

Correctness. It is sufficient to verify that $\widehat{\mathbf{s}}\cdot\widehat{\mathbf{C}}_1 \approx \mu_1\widehat{\mathbf{s}}\cdot\widehat{\mathbf{G}}$ and $\widehat{\mathbf{s}}\cdot\widehat{\mathbf{C}}_2 \approx \mu_2\widehat{\mathbf{s}}\cdot\widehat{\mathbf{G}}$. Adding (34) and (40) up, that is,

$$\mathbf{s}_2\cdot\mathbf{C}_1 \approx \left(\mathbf{b}_1-\mathbf{b}_2\right)\cdot\mathbf{R}_1+\mu_1\mathbf{s}_2\cdot\mathbf{G} \tag{42}$$

plus

$$\mathbf{s}_1\cdot\mathbf{X}_1 = \left(\mathbf{b}_2-\mathbf{b}_1\right)\cdot\mathbf{R}_1+\mathbf{e}_{\mathbf{X}1}, \tag{43}$$

results in

$$\begin{aligned}\widehat{\mathbf{s}}\cdot\widehat{\mathbf{C}}_1 &= \left(\mathbf{s}_1\cdot\mathbf{C}_1, \mathbf{s}_1\cdot\mathbf{X}_1+\mathbf{s}_2\cdot\mathbf{C}_1\right)\\ &\approx \left(\mu_1\mathbf{s}_1\cdot\mathbf{G}, \mu_1\mathbf{s}_2\cdot\mathbf{G}\right) = \mu_1\widehat{\mathbf{s}}\cdot\widehat{\mathbf{G}}.\end{aligned} \tag{44}$$

This also applies to verify $\widehat{\mathbf{s}}\cdot\widehat{\mathbf{C}}_2 \approx \mu_2\widehat{\mathbf{s}}\cdot\widehat{\mathbf{G}}$.

Parameters and Security. Though our LWR-based multikey FHE scheme does not need the sub-Gaussian noise sampling compared to Mukherjee and Wichs's [12], the structure of our construction is identical to theirs. Indeed, we just cleverly use the scaled rounding function instead of their sub-Gaussian noise, to hide the message, and this is done thanks to our specific structure which helps us circumvent the tangly modulus problem that exists in [22]. Since the LWE-based multikey FHE [12] is an extension of AP14, while the

parameter q in our LWR-based FHE is almost identical to that of AP14 according to Section 4.2, hence the LWR-based multikey FHE which is the extension of LWR-based FHE has almost the same parameter q as that of [12].

As to the security, we remark that the main difference between multikey setting in this section and single key setting in Section 4 is the number of ciphertexts. In multikey setting, there is an additional helper set $\mathcal{H}$ which involves mN ciphertexts, but the mN ciphertexts are encryptions of entries of the random matrix $\mathbf{R} \in \{0,1\}^{m\times N}$, which are independent of other ciphertexts. In other words, the helper set $\mathcal{H}$ does not affect scheme's security; therefore, our LWR-based multikey FHE scheme has the same security level as the LWR-based FHE scheme in Section 4.

5.3. Threshold Decryption for Two-Key FHE Scheme. In [12], the main idea of designing a multiparty computation protocol is to leverage the property of "threshold decryption" of multikey FHE and rely on the "smudging lemma [31]." In the secure multiparty computation distributed protocol, the parties run a secure distributed protocol using their own secret key to decrypt the output ciphertext and finally recover the plaintext y. Specifically, each party will obtain partial information by operating their own secret key on the common ciphertext $\widehat{\mathbf{C}}$, and then they broadcast the information they get added by some medium-sized smudging noise from a uniform distribution (adding the noise is needed to "smudge out" any information about the noise contained in ciphertext $\widehat{\mathbf{C}}$ and is needed for security proof); finally, each party can sum up all the other partial information and get the final decryption y. We can achieve the construction of multiparty computation protocol as well, for our LWR-based multikey FHE scheme also satisfies the threshold decryption. Compared to [12], our threshold decryption also needs the smudging noise, but the whole computation overhead is lower in that our LWR-based multikey FHE scheme does not need the sub-Gaussian noise sampling. The threshold decryption follows. For simplicity, we again cover the case of two-key FHE.

(i) $\mathit{TFHE.PartDec}(\widehat{\mathbf{C}}, (pk_1, pk_2), (sk_1, sk_2))$: given the output (common) ciphertext $\widehat{\mathbf{C}}$ under two public keys pk_1, pk_2 and the secret keys $sk_1 = \mathbf{s}_1, sk_2 = \mathbf{s}_2$, do the following:

(1) Parse the ciphertext $\widehat{\mathbf{C}}$ consisting of two submatrices $\widehat{\mathbf{C}}^{(i)}$ such that $\widehat{\mathbf{C}} = \left[\begin{smallmatrix}\widehat{\mathbf{C}}^{(1)}\\ \widehat{\mathbf{C}}^{(2)}\end{smallmatrix}\right]$.

(2) Define a vector $\widehat{\mathbf{v}} \in \mathbb{Z}_p^{2n+2}$ as $\widehat{\mathbf{v}} = (0, \ldots, 0, \lceil p/2 \rfloor)$.

(3) For $i \in [2]$, compute $\phi_i = \mathbf{s}_i\widehat{\mathbf{C}}^{(i)} \cdot \widehat{\mathbf{G}}^{-1}(\widehat{\mathbf{v}}^t) \in \mathbb{Z}_p$, and then compute $\tau_i = \phi_i + e_i^* \in \mathbb{Z}_p$, where $e_i^* \leftarrow_r [-B_{\mathrm{smdg}}^{\mathrm{dec}}, B_{\mathrm{smdg}}^{\mathrm{dec}}]$ is random "smudging noise." We can set $B_{\mathrm{smdg}}^{\mathrm{dec}} = 2^{L\lambda\log\lambda}$. Note that the parameter $B_{\mathrm{smdg}}^{\mathrm{dec}} = B_\chi \cdot 2^{L\lambda\log\lambda}$ in [12], where $B_\chi > \omega(\log n) \cdot \sqrt{n}$.

(ii) $\mathit{TFHE.FinDec}(\tau_1, \tau_2)$: given τ_1, τ_2, compute the sum $\tau = \sum_{i=1}^2 \tau_i$. Finally, output $\mu' = |\lceil (1/\lceil p/2 \rfloor) \cdot \tau \rfloor|$.

Correctness. Given the common ciphertext $\widehat{\mathbf{C}}$ designated for the message $\mu \in \{0,1\}$, parse the parties' keys as $\widehat{\mathbf{s}} = (\mathbf{s}_1, \mathbf{s}_2)$. By the correctness of LWR-based two-key FHE scheme, it holds that

$$\widehat{\mathbf{s}} \cdot \widehat{\mathbf{C}} = \sum_{i=1}^{2} \mathbf{s}_i \cdot \widehat{\mathbf{C}}^{(i)} = \mu\widehat{\mathbf{s}} \cdot \widehat{\mathbf{G}} + \widehat{\mathbf{e}} \bmod p. \tag{45}$$

Then, we have

$$\begin{aligned}\tau &= \sum_{i=1}^{2}\tau_i = \sum_{i=1}^{2}\phi_i + \sum_{i=1}^{2}e_i^* = \sum_{i=1}^{2}\mathbf{s}_i \cdot \mathbf{C}^{(i)} \cdot \widehat{\mathbf{G}}^{-1}\left(\widehat{\mathbf{v}}^t\right) + e^* \\ &= \left(\mu\widehat{\mathbf{s}} \cdot \widehat{\mathbf{G}} + \widehat{\mathbf{e}}\right) \cdot \widehat{\mathbf{G}}^{-1}\left(\widehat{\mathbf{v}}^t\right) + e^* \\ &= \mu \cdot \left\lceil \frac{p}{2} \right\rfloor + e' + e^* \bmod p,\end{aligned} \tag{46}$$

where $|e' + e^*| < 1/2 \cdot \lceil p/2 \rfloor$ holds under the appropriate parameters; for example, we can set $p = 2^{\omega(L\lambda\log\lambda)}$. Therefore, the correctness holds.

6. Conclusions

We present the first workable LWR-based FHE scheme. Our FHE scheme erases the expensive Gaussian noise sampling and thus can be seen as an alternative to the existing LWE-based FHEs. Furthermore, based on our main construction, we give the first LWR-based multikey FHE scheme, which is an alternative to the existing multikey FHEs that can also be applied to multiparty computation. However, neither our LWR-based multikey FHE scheme nor Mukherjee and Wichs's can be used to construct multiparty computation protocol without the smudging lemma (which means that the corresponding modulus has to be set exponentially in security parameter). Therefore, it remains an open problem to construct multiparty computation protocol via multikey FHEs without the smudging lemma.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is supported in part by the National Basic Research Program of China (973 Project, no. 2014CB340603), the National Key Research and Development Program of China (2017YFB0802000), the National Natural Science Foundation of China (nos. 61672030, 61272040, and U1705264), the Research Foundation of Hangzhou Normal University (no. 2017QDL002), and the Scientific Research Fund of Zhejiang Provincial Education Department (no. Y201737292).

References

[1] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[2] C. Gentry, *A Fully Homomorphic Encryption Scheme*, Stanford University, 2009.

[3] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 6110 of *Lecture Notes in Computer Science*, pp. 24–43, Springer, Berlin, Germany, 2010.

[4] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Annual Cryptology Conference*, vol. 6841 of *Lecture Notes in Computer Science*, pp. 505–524, Springer, Heidelberg, Berlin, Germany, 2011.

[5] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pp. 97–106, IEEE Computer Society, 2011.

[6] M. Clear and C. McGoldrick, "Multi-identity and multi-key leveled fhe from learning with errors," in *Proceedings of the Annual Cryptology Conference*, pp. 630–656, Springer, Heidelberg, Berlin, Germany, 2015.

[7] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *Public Key Cryptography*, vol. 6056 of *Lecture Notes in Computer Science*, pp. 420–443, Springer, Berlin, Germany, 2010.

[8] L. Ducas and D. Micciancio, "Fhew: Bootstrapping homomorphic encryption in less than a second," *EUROCRYPT*, vol. 9056, pp. 617–640, 2015.

[9] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Conference on Innovations in Theoretical Computer Science, ITCS 2012*, pp. 309–325, USA, January 2012.

[10] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *CRYPTO*, vol. 7417, pp. 868–886, Springer, 2012.

[11] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," in *Proceedings of the 22nd International Conference on the Theory and Application of Cryptology and Information Security*, Lecture Notes in Computer Science, pp. 3–33, Springer, Hanoi, Vietnam, 2016.

[12] P. Mukherjee and D. Wichs, "Two round multiparty computation via multi-key fhe," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 9666 of *Lecture Notes in Computer Science*, pp. 735–763, Springer, Berlin, Germany, 2016.

[13] C. Peikert and S. Shiehian, "Multi-key fhe from lwe, revisited," in *Theory of Cryptography Conference*, vol. 9986 of *Lecture Notes in Computer Science*, pp. 217–238, Springer, Berlin, Germany, 2016.

[14] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, Article ID 1568324, 34 pages, 2009.

[15] C. Peikert, "Public-key cryptosystems from the worst-case shortest vector problem," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC '09*, pp. 333–342, ACM, USA, June 2009.

[16] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology–CRYPTO 2013*, vol. 8042, pp. 75–92, Springer, 2013.

[17] J. Alperin-Sheriff and C. Peikert, "Faster bootstrapping with polynomial error," in *Proceedings of the International Cryptology Conference*, pp. 297–314, Springer, Berlin, Germany, 2014.

[18] Z. Brakerski, D. Cash, R. Tsabary, and H. Wee, "Targeted homomorphic attribute-based encryption," in *Proceedings of the Theory of Cryptography Conference*, pp. 330–360, Springer, Berlin, Germany, 2016.

[19] S. Gorbunov, V. Vaikuntanathan, and D. Wichs, "Leveled fully homomorphic signatures from standard lattices," in *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, pp. 469–477, Portland, Oregon, USA, June 2015.

[20] P. Pessl, "Analyzing the shuffling side-channel countermeasure for lattice-based signatures," in *Proceedings of the 17th International Conference on Cryptology in India*, vol. 10095, pp. 153–170, Springer, Kolkata, India, 2016.

[21] L. G. Bruinderink, A. Hülsing, T. Lange, and Y. Yarom, "Flush, gauss, and reload–a cache attack on the bliss lattice-based signature scheme," in *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*, pp. 323–345, Springer, 2016.

[22] A. Costache and N. P. Smart, "Homomorphic encryption without gaussian noise," *IACR Cryptology ePrint Archive*, vol. 163, 2017.

[23] A. Banerjee, C. Peikert, and A. Rosen, "Pseudorandom functions and lattices," *Advances in Cryptology–EUROCRYPT 2012*, pp. 719–737, 2012.

[24] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, "Classical hardness of learning with errors (extended abstract)," in *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, pp. 575–584, ACM, New York, NY, USA, 2013.

[25] J. HÅstad, R. Impagliazzo, L. A. Levin, and M. Luby, "A pseudorandom generator from any one-way function," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1364–1396, 1999.

[26] J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs, "Learning with rounding, revisited: New reduction, properties and applications," in *Advances in Cryptology–CRYPTO 2013*, pp. 57–74, Springer, 2013.

[27] A. Bogdanov, S. Guo, D. Masny, S. Richelson, and A. Rosen, "On the hardness of learning with rounding over small modulus," in *Proceedings of the Theory of Cryptography Conference*, vol. 9562, pp. 209–224, Springer, Berlin, Germany, 2016.

[28] J. Alperin-Sheriff and D. Apon, "Dimension-preserving reductions from lwe to lwr," *IACR Cryptology ePrint Archive*, vol. 2016, no. 589, 2016.

[29] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *EuroCrypt*, vol. 7237 of *Lecture Notes in Computer Science*, pp. 700–718, Springer, Heidelberg, Berlin, Germany, 2012.

[30] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, pp. 1219–1234, ACM, New York, NY, USA, 2012.

[31] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs, "Multiparty computation with low communication, computation and interaction via threshold FHE," in *Proceedings of the 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Advances in Cryptology - EUROCRYPT 2012, pp. 483–501, Cambridge, UK, April 15-19, 2012.

Hindawi
Security and Communication Networks
Volume 2018, Article ID 5823230, 8 pages
https://doi.org/10.1155/2018/5823230

Research Article

To Study the Effect of the Generating Polynomial on the Quality of Nonlinear Components in Block Ciphers

Shahid Mahmood[1], **Shabieh Farwa**[2], **Muhammad Rafiq**[2], **Syed Muhammad Jawwad Riaz**[2], **Tariq Shah**[3] and **Sajjad Shaukat Jamal**[3]

[1]*Department of Mechanical Engineering, Sarhad University of Science and Information Technology, Peshawar, Pakistan*
[2]*Department of Mathematics, COMSATS Institute of Information Technology, Wah Cantt, Pakistan*
[3]*Department of Mathematics, Quaid-i-Azam University, Islamabad, Pakistan*

Correspondence should be addressed to Shabieh Farwa; drsfarwa@gmail.com

Received 1 November 2017; Revised 1 March 2018; Accepted 5 March 2018; Published 10 April 2018

Academic Editor: Amir Anees

Substitution box (*S*-box), being the only nonlinear component, contributes to the confusion creating capability of a cryptosystem. Keeping in view the predominant role of *S*-box, many design algorithms to synthesize cryptographically stronger *S*-boxes have gained pivotal attention. A quick review of these algorithms shows that all these ideas mainly concentrate on the choice of bijective Boolean functions, with nonobservance to the irreducible polynomial that generates the Galois field. In this paper, we propose that the selection of irreducible polynomial has a deep influence on the highly desirable features of an *S*-box such as nonlinearity, strict avalanche, bit independence, linear approximation probability, and differential approximation probability. We underpin our claim by investigating a detailed model, which deploys the same algorithm but different polynomials and produces unusual changes in the results regarding the performance parameters of *S*-box.

1. Introduction

Electronic exchange of data has undoubtedly revolutionized the communication in recent years but, on the other hand, the secure transfer of confidential material over Internet has become the biggest challenge nowadays. It definitely demands seriously high level of security. The main problem is to avoid unauthorised access to the secret data. To achieve the desired level of security, many techniques such as cryptography, watermarking, and steganography have been the major focus of research for past few years [1–5]. In this paper, we deal with cryptography.

Cryptography is categorized into two types, *symmetric key cryptography* and *asymmetric key cryptography*. The symmetric key cryptography can be further split into two types: block ciphers and the stream ciphers. Advanced Encryption Standard is an example of block cipher that was officially adopted by the US government as the Federal Information Processing Standard (FIPS) in May 2002. AES algorithm [6] is based on four steps: round key addition, byte substitution, shift row, and mix column, but the most influential of all these is the byte substitution step. This step relies on a substitution box (*S*-box), which serves as the only nonlinear component in any *substitution-permutation network* (SPN).

It has been established that the substitution box (*S*-box) is a standout in different block ciphers and is a widely used mechanism in any substitution-permutation network as a source to produce nonlinearity [6]. It renders an absolutely complex, unforeseeable layout to requite various blocks of bits in output data. To extend high resistance against unexpected surveillance, *S*-box structure is required to fulfil certain standards. The indispensable involvement of *S*-box to induce complexity and nonlinearity motivates studying the properties and algorithms for safer and more reliable *S*-boxes. In this regard, many advanced structural developments are witnessed in literature. Khan et al. [7] proposed a technique for *S*-box construction based on chaotic Lorenz systems. Hussain et al. presented *S*-box algorithms using generalized Bakers map [8] and projective general linear group [9]. Algebraic, analytical, and chaotic approaches for *S*-box are

studied in [10–13]. Özkaynak et al. [14] applied fractional-order chaotic Chen system, to develop S-box. Tian and Lu [15] structured dynamic chaos-based S-box in conjunction with DNA sequence operation. Some other more efficient algorithms could be reviewed in [6, 16–20]. In addition to this, applications of S-boxes in digital image encryption, steganography, and watermarking have become quite popular and influential in recent years [4, 11, 13, 21].

The study of innovation in design algorithms for S-boxes witnesses that the change of model and the selection of Boolean function contribute little to the performance indices of an S-box. We, in this paper, propose that the performance of an S-box is highly related to the background Galois field. The fact that finite fields of the same order are isomorphic is definitely of worth but the scrambling effect of a nonlinear Boolean function applied on two different fields of the same order might vary. Since in cryptography, an S-box is the salient component used to produce confusion in the data, it is worth studying that the confusion creating ability is associated with the choice of the irreducible polynomial used to form the background Galois field.

In [9], Hussain et al. presented an algorithm for generating S-box through the application of a linear fractional transformation on the Galois field GF(2^8), structured by the polynomial $X^8 + X^4 + X^3 + X^2 + 1$. We in the proposed work show that the same algorithm used for a different polynomial exhibits highly improved values of nonlinearity, strict avalanche criterion (SAC), bit independent criterion (BIC), linear approximation probability (LAP), and differential approximation probability (DAP). By comparing the numerical results of these tests, we prove that different polynomials produce significantly different results. This observation leads to revising the existing models by choosing different background polynomials as it could be more influential in improvement of ideas rather changing the whole scheme.

We organize the contents of this paper as follows. In Section 2, we discuss the properties of the background Galois field GF(2^8). The detailed algorithm for the design of the S-box is presented in Section 3. Section 4 deals with the analyses of S-boxes against several common attacks and the comparison of respective results. We further compare the cryptographic standing of both of the newly synthesized S-boxes with the state-of-the-art AES S-box. Conclusion is presented in Section 5.

2. Generating Polynomial and the Galois Field

For any prime p, Galois field GF(p^n) is expressed as the factor ring $\mathbb{F}_p[X]/(\mu(x))$ where $\mu(x) \in \mathbb{F}_p[X]$ is an irreducible polynomial of degree n. For GF(2^8) we choose an irreducible polynomial of degree 8 that generates the maximal ideal of the principal ideal domain $\mathbb{F}_2[X]$. We know that the multiplicative group of the resultant field GF(2^8) is cyclic and hence each nonzero element of the field can be expressed as a power of the generator $\alpha = 00000010$.

In order to support our claim regarding the effect of polynomial, we choose two irreducible primitive polynomials μ_1 and μ_2 of degree 8, to construct Galois fields $\mathbb{F}_1$ and $\mathbb{F}_2$, respectively, where $\mu_1 = X^8 + X^6 + X^5 + X^4 + 1$ and $\mu_2 = X^8 + X^4 + X^3 + X^2 + 1$, as used in [9]. We may choose other polynomials as well to compare our calculations but the selected pair beautifully serves for the purpose. Let G_i represents the multiplicative group of the Galois field, $\mathbb{F}_i$. The exponential form of elements of the multiplicative group G_1, along with their inverses, is represented in Table 1; however the elements of G_2 are presented in Table 2 of [9]. In the next section, we use these calculations to develop the corresponding S-boxes.

3. Algorithm for S-Box

An $n \times n$ S-box is defined by a *vector Boolean function* $\mathcal{S}_n$: GF(2^n) $\rightarrow$ GF(2^n), defined as

$$\mathcal{S}_n(v) = \left(s_1(v), s_2(v), \ldots, s_n(v)\right), \tag{1}$$

where $v = (v_1, v_2, \ldots, v_{2^n}) \in \mathrm{GF}(2^n)$ and each of s_i's is regarded as a component Boolean function.

For a field $\mathbb{F}$, the general linear group GL($n, \mathbb{F}$) is a group formed by all $n \times n$ invertible matrices. A projective general linear group of degree n over a field $\mathbb{F}$ is defined to be the quotient group of GL($n, \mathbb{F}$) by its center. For this paper, we form the 8×8 S-box by considering the action of the Galois field GF(2^8) on the projective linear group PGL(2, GF(2^8)); that is, we take a function σ : PGL(2, GF(2^8)) $\times$ GF(2^8) $\rightarrow$ GF(2^8) defined as follows:

$$\sigma(v) = \frac{\alpha v + \beta}{\gamma v + \delta}. \tag{2}$$

In (2), σ is known as a linear fractional transformation (LFT) with α, β, γ and $\delta \in$ GF(2^8) satisfying the nondegeneracy condition $\alpha\delta - \beta\gamma \neq 0$. The ease of implementation, lesser computational labour, and high algebraic complexity of an LFT are the prime features that give incentive to employ this map for byte substitution. We may choose any values for LFT parameters that satisfy the aforementioned condition but, for the presented calculations, we, in particular, choose the same values as in [9], so that a comparison could be set easily. We consider $\alpha = 35$, $\beta = 15$, $\gamma = 9$ and $\delta = 5$. The images of the map σ, when applied on $\mathbb{F}_1$ and $\mathbb{F}_2$, produce our S-boxes S_1 and S_2, respectively, as shown in Tables 2 and 3.

4. Performance Analysis of S-Boxes

The cryptographic strength if the S-boxes, generated in the foregoing section, are examined through the most widely used analysis techniques such as nonlinearity, bit independence, strict avalanche, and linear and differential approximation probabilities. In the following subsections we present all these performance indices one by one and compare the performance of S_1 and S_2 with one another, as well as, with the ever-prevailing algorithm AES.

4.1. Nonlinearity. Nonlinearity analysis measures the distance of the reference function from all of the affine functions.

Table 1: Exponential representation and the multiplicative inverses of elements of G_1.

$x \in G_1$	α^n	x^{-1}
1	α^{255}	1
2	α^{1}	184
3	α^{231}	208
4	α^{2}	92
5	α^{207}	159
6	α^{232}	104
7	α^{59}	134
8	α^{3}	46
9	α^{35}	173
10	α^{208}	247
11	α^{154}	139
12	α^{233}	52
13	α^{20}	48
14	α^{60}	67
15	α^{183}	117
16	α^{4}	23
17	α^{159}	252
18	α^{36}	238
19	α^{66}	83
20	α^{209}	195
21	α^{118}	204
22	α^{155}	253
23	α^{251}	16
24	α^{234}	26
25	α^{245}	181
26	α^{21}	24
27	α^{11}	180
28	α^{61}	153
29	α^{130}	121
30	α^{184}	130
31	α^{146}	73
32	α^{5}	179
33	α^{122}	232
34	α^{160}	126
35	α^{79}	141
36	α^{37}	119
37	α^{113}	220
38	α^{67}	145
39	α^{106}	248
40	α^{210}	217
41	α^{224}	175
42	α^{119}	102
43	α^{221}	188
44	α^{156}	198
45	α^{242}	108
46	α^{252}	8
47	α^{32}	172
48	α^{235}	13
49	α^{213}	53
50	α^{246}	226
51	α^{135}	84
52	α^{22}	12
53	α^{42}	49
54	α^{12}	90

Table 1: Continued.

$x \in G_1$	α^n	x^{-1}
55	α^{140}	148
56	α^{62}	244
57	α^{227}	223
58	α^{131}	132
59	α^{75}	203
60	α^{185}	65
61	α^{191}	224
62	α^{147}	156
63	α^{94}	68
64	α^{6}	225
65	α^{70}	60
66	α^{123}	116
67	α^{195}	14
68	α^{161}	63
69	α^{53}	157
70	α^{80}	254
71	α^{167}	143
72	α^{38}	131
73	α^{109}	31
74	α^{114}	110
75	α^{203}	154
76	α^{68}	240
77	α^{51}	150
78	α^{107}	124
79	α^{49}	186
80	α^{211}	212
81	α^{40}	196
82	α^{225}	239
83	α^{189}	19
84	α^{120}	51
85	α^{111}	227
86	α^{222}	94
87	α^{240}	193
88	α^{157}	99
89	α^{116}	163
90	α^{243}	54
91	α^{128}	149
92	α^{253}	4
93	α^{205}	158
94	α^{33}	86
95	α^{18}	192
96	α^{236}	190
97	α^{163}	235
98	α^{214}	162
99	α^{98}	88
100	α^{247}	113
101	α^{55}	123
102	α^{136}	42
103	α^{102}	189
104	α^{23}	6
105	α^{82}	135
106	α^{43}	160
107	α^{177}	169

Table 1: Continued.

$x \in G_1$	α^n	x^{-1}
108	α^{13}	45
109	α^{169}	199
110	α^{141}	74
111	α^{89}	155
112	α^{63}	122
113	α^{8}	100
114	α^{228}	215
115	α^{151}	237
116	α^{132}	66
117	α^{72}	15
118	α^{76}	221
119	α^{218}	36
120	α^{186}	152
121	α^{125}	29
122	α^{192}	112
123	α^{200}	101
124	α^{148}	78
125	α^{197}	187
126	α^{95}	34
127	α^{174}	140
128	α^{7}	200
129	α^{150}	171
130	α^{71}	30
131	α^{217}	72
132	α^{124}	58
133	α^{199}	202
134	α^{196}	7
135	α^{173}	105
136	α^{162}	167
137	α^{97}	176
138	α^{54}	246
139	α^{101}	11
140	α^{81}	127
141	α^{176}	35
142	α^{168}	255
143	α^{88}	71
144	α^{39}	249
145	α^{188}	38
146	α^{110}	183
147	α^{239}	243
148	α^{115}	55
149	α^{127}	91
150	α^{204}	77
151	α^{17}	241
152	α^{69}	120
153	α^{194}	28
154	α^{52}	75
155	α^{166}	111
156	α^{108}	62
157	α^{202}	69
158	α^{50}	93
159	α^{48}	5
160	α^{212}	106
161	α^{134}	168
162	α^{41}	98

Table 1: Continued.

$x \in G_1$	α^n	x^{-1}
163	α^{139}	89
164	α^{226}	207
165	α^{74}	231
166	α^{190}	177
167	α^{93}	136
168	α^{121}	161
169	α^{78}	107
170	α^{112}	201
171	α^{105}	129
172	α^{223}	47
173	α^{220}	9
174	α^{241}	216
175	α^{31}	41
176	α^{158}	137
177	α^{65}	166
178	α^{117}	233
179	α^{250}	32
180	α^{244}	27
181	α^{10}	25
182	α^{129}	242
183	α^{145}	146
184	α^{254}	2
185	α^{230}	209
186	α^{206}	79
187	α^{58}	125
188	α^{34}	43
189	α^{153}	103
190	α^{19}	96
191	α^{182}	234
192	α^{237}	95
193	α^{15}	87
194	α^{164}	205
195	α^{46}	20
196	α^{215}	81
197	α^{171}	213
198	α^{99}	44
199	α^{86}	109
200	α^{248}	128
201	α^{143}	170
202	α^{56}	133
203	α^{180}	59
204	α^{137}	21
205	α^{91}	194
206	α^{103}	230
207	α^{29}	164
208	α^{24}	3
209	α^{25}	185
210	α^{83}	251
211	α^{26}	228
212	α^{44}	80
213	α^{84}	197
214	α^{178}	236
215	α^{27}	114
216	α^{14}	174
217	α^{45}	40

Table 1: Continued.

$x \in G_1$	α^n	x^{-1}
218	α^{170}	219
219	α^{85}	218
220	α^{142}	37
221	α^{179}	118
222	α^{90}	245
223	α^{28}	57
224	α^{64}	61
225	α^{249}	64
226	α^{9}	50
227	α^{144}	85
228	α^{229}	211
229	α^{57}	250
230	α^{152}	206
231	α^{181}	165
232	α^{133}	33
233	α^{138}	178
234	α^{73}	191
235	α^{92}	97
236	α^{77}	214
237	α^{104}	115
238	α^{219}	18
239	α^{30}	82
240	α^{187}	76
241	α^{238}	151
242	α^{126}	182
243	α^{16}	147
244	α^{193}	56
245	α^{165}	222
246	α^{201}	138
247	α^{47}	10
248	α^{149}	39
249	α^{216}	144
250	α^{198}	229
251	α^{172}	210
252	α^{96}	17
253	α^{100}	22
254	α^{175}	70
255	α^{87}	142

Nonlinearity criterion outlines the total number of bits that must be altered in the truth table of a Boolean function to get close to the nearby affine function [22].

Table 4 shows that, for S_1, the average nonlinearity measure is 112., which is the highest figure attained by the AES S-box. Figure 1 shows the comparison which clearly depicts outstanding performance of S_1 as compared to S_2.

4.2. Linear Approximation Probability. The measure of unevenness of an event is determined by linear approximation probability. This analysis is used to evaluate the maximum imbalance of the outcome. Mathematically, the linear approximation probability for a given S-box is defined as follows:

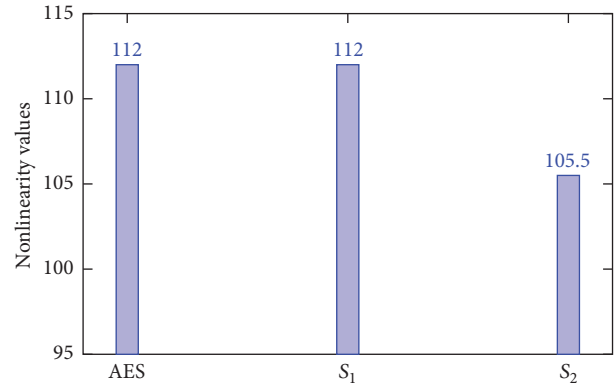


Figure 1: Nonlinearity of different S-boxes.

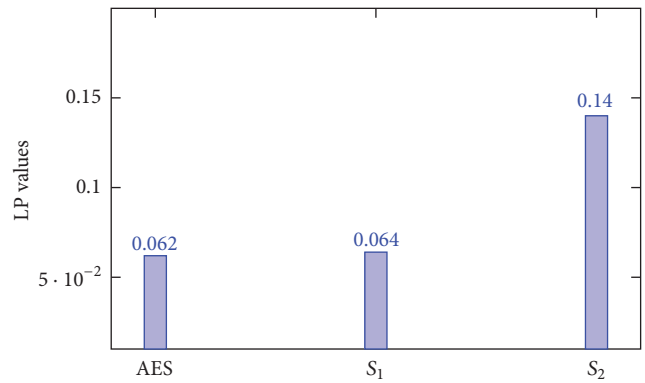


Figure 2: LP of different S-boxes.

$$\mathrm{LP} = \max_{\Gamma_x, \Gamma_y \neq 0} \left| \frac{\#\left\{x \mid x \cdot \Gamma_x = S(x) \cdot \Gamma_y\right\}}{2^n} - \frac{1}{2} \right|, \tag{3}$$

where X represents the set of all possible inputs and Γ_x and Γ_y are the input and output masks, respectively. Numerical results presented in Table 5 and compared in Figure 2 show that the linear approximation probability of S_1 is much better than S_2.

4.3. Differential Approximation Probability. For further analysis, we use the differential approximation probability, which determines the differential uniformity demonstrated by an S-box. The mathematical expression for DP is given by the following:

$$\mathrm{DP} = \left[\frac{\#\left\{x \in X \mid S(x) \oplus S(x \oplus \Delta x) = \Delta y\right\}}{2^n} \right]. \tag{4}$$

In the above-mentioned expression input and output differentials are represented by Δx and Δy, respectively. The smaller the differential uniformity, the stronger the S-box. It is evident from Table 5 and Figure 3 that in terms of the differential approximation probability S_1 is much stronger than S_2.

4.4. Strict Avalanche Criterion. This criterion examines the changes in the output bits caused as a result of single input

Table 2: S-box S_1.

3	214	37	74	126	4	18	26	219	62	45	226	50	136	104	148
154	38	200	199	185	228	170	245	177	114	137	231	139	35	8	134
158	27	223	232	17	157	217	49	83	141	171	42	47	206	64	194
106	29	30	110	14	40	72	236	105	221	202	87	241	41	11	71
186	28	253	175	67	31	23	33	66	189	117	118	94	149	135	252
235	43	124	125	90	229	204	215	218	100	101	249	243	54	173	166
138	234	244	201	167	44	250	25	16	187	207	246	107	103	161	1
183	99	179	240	129	123	188	193	20	143	155	174	7	220	213	239
108	84	113	184	5	57	208	153	75	112	223	178	180	150	65	24
224	248	102	89	70	111	59	172	95	131	198	163	93	164	55	209
86	132	6	225	51	79	53	34	97	48	197	142	182	210	91	247
195	15	10	144	85	63	168	238	196	162	98	32	251	254	203	156
80	152	237	24	127	78	165	12	52	222	122	58	211	36	140	191
216	146	109	96	147	73	116	190	128	68	56	77	115	160	19	92
69	2	192	121	145	21	76	61	60	181	133	151	159	0	88	205
13	230	22	82	39	119	9	255	120	46	81	212	227	130	176	169

Table 3: S-box S_2.

198	214	241	163	130	165	217	127	179	123	111	197	43	141	237	3
168	201	17	121	142	101	232	174	11	249	16	156	10	50	183	65
72	184	200	132	58	47	27	159	231	189	8	18	206	194	177	31
193	92	122	192	85	137	243	49	178	170	36	135	230	95	100	128
13	109	227	0	224	144	208	78	173	32	139	234	107	82	172	81
51	233	12	154	94	161	244	55	7	34	251	225	153	93	254	138
102	240	115	242	110	134	124	79	157	160	90	238	73	53	169	250
136	118	112	48	40	114	22	246	46	131	23	69	52	235	248	2
116	91	117	26	166	25	219	59	54	229	120	245	89	185	99	226
105	45	60	199	164	191	228	202	37	104	143	209	220	147	44	186
145	125	203	29	38	41	215	108	64	88	119	74	213	96	211	83
218	146	196	205	67	152	129	175	84	158	207	176	80	62	150	86
57	155	195	216	75	19	1	87	33	68	71	236	239	255	35	212
148	188	133	15	204	187	42	182	97	56	24	221	252	30	77	181
4	247	167	21	9	222	180	190	151	140	39	171	14	126	66	253
103	223	70	98	28	20	63	162	61	113	149	210	106	5	6	76

bit change. This is one of the most desirable features of any cryptographic design that when we change a single input bit, changes must occur in half of the output bits. In other words an S-box, $\mathcal{S}_n : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is said to satisfy SAC if, for a change in an input bit, the probability of change in the output bit is 1/2. The results are shown in Table 5 and Figure 4.

4.5. Bit Independence Criterion. The independent behavior of the pair of variables and the variations of input bits are considered as important factors of bit independence criterion. In bit independence criterion, input bits are transformed exclusively, and then output results are scrutinized for their independency [23]. Bit independence has great worth in cryptographic structures. The goal of reaching the maximum complexity and perplexity in a system can be achieved through this property of increasing independence between the bits. In cryptographic systems, the increased independence between bits is an essential requirement as it makes harder to understand and forecast the design of the system.

The numerical results of BIC when applied to the proposed S-box are given in Table 5 and are compared in Figure 5. It can be observed that according to these results our S-box S_1 is pretty similar to the AES S-box and is much better than S_2.

One can observe that overall performance of S_1 is much better than that of S_2. The performance parameters for S_1 seem to be pretty close to that of AES S-box. The algorithm used for both S_1 and S_2 is the same but the primitive polynomial selected to generate the Galois field is different, which really contributes to the outputs.

5. Conclusion

The kernel of the presented work lies in the fact that the choice of the background Galois field and its generating primitive

Table 4: Performance Indices for new S-box.

Analysis	Max.	Min.	Average	Square deviation	DP	LP
Nonlinearity	113	111	**112**			
SAC	0.546875	0.429688	**0.498291**	0.0157537		
BIC		111	**111.751**	0.6227		
DP					**0.015625**	
LP						**0.064063**

Table 5: Comparison of performance indices for different S-boxes.

S-box	Nonlinearity	SAC	BIC	DP	LP
AES	112	0.5058	112.0	0.0156	0.062
S_1	**112**	**0.498291**	**111.751**	**0.015625**	**0.064063**
S_2	**105.5**	**0.507**	**106**	**0.0242**	**0.140**

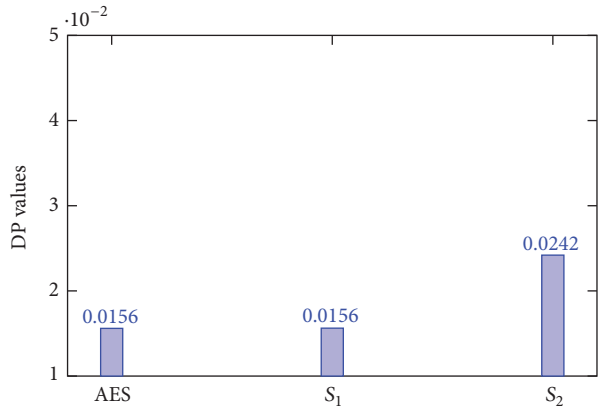


Figure 3: DP of different S-boxes.

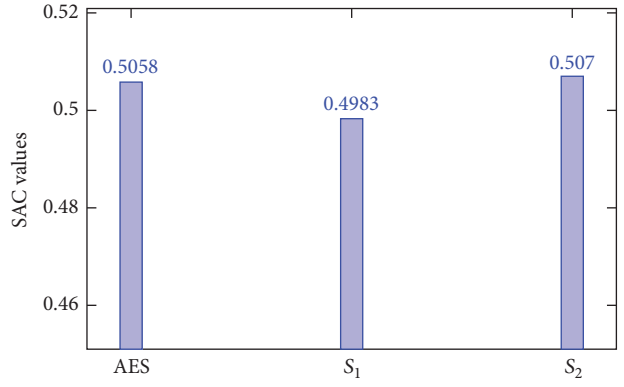


Figure 4: SAC of different S-boxes.

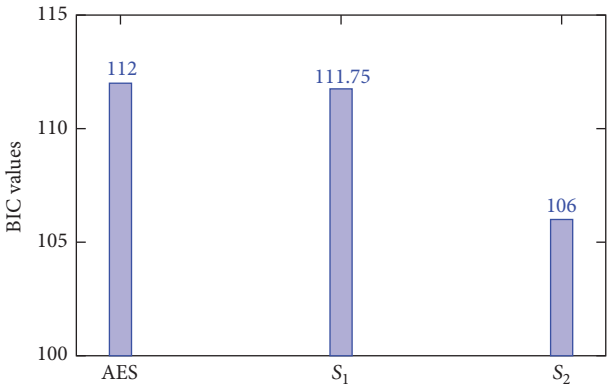


Figure 5: BIC of different S-boxes.

polynomial matters to the function and performance of the substitution boxes. This fact leads to the fascinating idea that, rather than the development of new algorithms, the improvement of the existing algorithms is worth studying as its least laborious but most effective. We propose, on the basis of the example discussed, that the effect of the choice of generating polynomial may lead to an intensive research in future to modify the design models of S-boxes. It will definitely affect the applications of S-boxes in other branches of the digital communication, such as steganography, watermarking, and image encryption.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors are grateful to the Sarhad University of Science and Information Technology (Pakistan), for providing partial funding for this research work.

References

[1] C. E. Shannon, "Communication theory of secrecy systems," *Bell Labs Technical Journal*, vol. 28, pp. 656–715, 1949.

[2] H. Feistel, "Cryptography and computer privacy," *Scientific American*, vol. 228, no. 5, pp. 15–23, 1973.

[3] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, "Digital image steganography: survey and analysis of current methods," *Signal Processing*, vol. 90, no. 3, pp. 727–752, 2010.

[4] S. S. Jamal, T. Shah, S. Farwa, and M. U. Khan, "A new technique of frequency domain watermarking based on a local ring," *Wireless Networks*.

[5] S. S. Jamal, T. Shah, and I. Hussain, "An efficient scheme for digital watermarking using chaotic map," *Nonlinear Dynamics*, vol. 73, no. 3, pp. 1469–1474, 2013.

[6] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-The Advanced Encryption Standard*, Springer, Berlin, Germany, 2002.

[7] M. Khan, T. Shah, H. Mahmood, M. A. Gondal, and I. Hussain, "A novel technique for the construction of strong S-boxes based on chaotic Lorenz systems," *Nonlinear Dynamics*, vol. 70, no. 3, pp. 2303–2311, 2012.

[8] I. Hussain, T. Shah, M. A. Gondal, and H. Mahmood, "Efficient method for designing chaotic S-boxes based on generalized Baker's map and TDERC chaotic sequence," *Nonlinear Dynamics*, vol. 74, no. 1-2, pp. 271–275, 2013.

[9] I. Hussain, T. Shah, H. Mahmood, and M. A. Gondal, "A projective general linear group based algorithm for the construction of substitution box for block ciphers," *Neural Computing and Applications*, vol. 22, no. 6, pp. 1085–1093, 2013.

[10] S. Farwa, N. Muhammad, T. Shah, and S. Ahmad, "A Novel Image Encryption Based on Algebraic S-box and Arnold Transform," *3D Research*, vol. 8, no. 3, article no. 26, 2017.

[11] S. Farwa, T. Shah, and L. Idrees, "A highly nonlinear S-box based on a fractional linear transformation," *SpringerPlus*, vol. 5, no. 1, article no. 1658, 2016.

[12] S. V. Radhakrishnan and S. Subramanian, "An analytical approach to s-box generation," *Computers and Electrical Engineering*, vol. 39, no. 3, pp. 1006–1015, 2013.

[13] S. Farwa, T. Shah, N. Muhammad, N. Bibi, A. Jahangir, and S. Arshad, "An image encryption technique based on chaotic S-box and Arnold transform," *International Journal of Advanced Computer Science & Applications*, vol. 8, no. 6, pp. 360–364, 2017.

[14] F. Özkaynak, V. Çelik, and A. B. Özer, "A new S-box construction method based on the fractional-order chaotic Chen system," *Signal, Image and Video Processing*, vol. 11, no. 4, pp. 659–664, 2017.

[15] Y. Tian and Z. Lu, "Novel permutation-diffusion image encryption algorithm with chaotic dynamic S-box and DNA sequence operation," *AIP Advances*, vol. 7, no. 8, Article ID 085008, 2017.

[16] X.-M. Zhang, Y. Zheng, and H. Imai, "Relating differential distribution tables to other properties of substitution boxes," *Designs, Codes and Cryptography. An International Journal*, vol. 19, no. 1, pp. 45–63, 2000.

[17] X. Y. Shi, X. You, and K. Y. Lam, "A method for obtaining cryptographically strong 8×8 S-boxes," *Int Conf Infor Network Appl*, vol. 2, no. 3, pp. 14–20, 2002.

[18] M.-T. Tran, D.-K. Bui, and A.-D. Duong, "Gray S-box for advanced encryption standard," *International Conference on Computational Intelligence and Security*, pp. 253–258, 2008.

[19] M. A. Gondal, Abdul Raheem, and I. Hussain, "A Scheme for Obtaining Secure S-Boxes Based on Chaotic Baker's Map," *3D Research*, vol. 5, no. 3, 2014.

[20] M. Khan, T. Shah, H. Mahmood, and M. A. Gondal, "An efficient method for the construction of block cipher with multi-chaotic systems," *Nonlinear Dynamics*, vol. 71, no. 3, pp. 489–492, 2013.

[21] A. U. Rehman, J. S. Khan, J. Ahmad, and S. O. Hwang, "A New Image Encryption Scheme Based on Dynamic S-Boxes and Chaotic Maps," *3D Research*, vol. 7, no. 1, article no. 7, pp. 1–8, 2016.

[22] K. Nyberg, *Perfect nonlinear S-boxes, Advances in Cryptology - EUROCRYPT'91*, vol. 547 of *Lecture Notes in Comput. Sci.*, Springer, Berlin, Germany, 1991.

[23] A. F. Webster and S. E. Tavares, "On the design of S-boxes, Advances in Cryptology," in *Proceedings of CRYPTO'85, Springer-Verlag*, pp. 523–534, Berlin, Germany, 1986.

Hindawi
Security and Communication Networks
Volume 2018, Article ID 9701756, 14 pages
https://doi.org/10.1155/2018/9701756

Research Article

Side-Channel Attacks and Countermeasures for Identity-Based Cryptographic Algorithm SM9

Qi Zhang[1,2] **An Wang**[1] **Yongchuan Niu**[3] **Ning Shang**[1] **Rixin Xu**[1] **Guoshuang Zhang**[4] **and Liehuang Zhu**[1]

[1]*School of Computer Science, Beijing Institute of Technology, Beijing 100081, China*
[2]*State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China*
[3]*Data Communication Science and Technology Research Institute, Beijing 100191, China*
[4]*Science and Technology on Information Assurance Laboratory, Beijing 100072, China*

Correspondence should be addressed to An Wang; wanganl@bit.edu.cn and Liehuang Zhu; liehuangz@bit.edu.cn

Received 2 November 2017; Revised 8 February 2018; Accepted 21 February 2018; Published 5 April 2018

Academic Editor: Amir Anees

Identity-based cryptographic algorithm SM9, which has become the main part of the ISO/IEC 14888-3/AMD1 standard in November 2017, employs the identities of users to generate public-private key pairs. Without the support of digital certificate, it has been applied for cloud computing, cyber-physical system, Internet of Things, and so on. In this paper, the implementation of SM9 algorithm and its Simple Power Attack (SPA) are discussed. Then, we present template attack and fault attack on SPA-resistant SM9. Our experiments have proved that if attackers try the template attack on an 8-bit microcontrol unit, the secret key can be revealed by enabling the device to execute one time. Fault attack even allows the attackers to obtain the 256-bit key of SM9 by performing the algorithm twice and analyzing the two different results. Accordingly, some countermeasures to resist the three kinds of attacks above are given.

1. Introduction

With the development of integrated circuit and communication technology, smart devices are not only widely spread in our daily life with the proliferation of Internet of things, but also extensively used in the global IT environments and critical infrastructures. Security becomes a critical issue since attacks on these devices may directly harm the consumers. Several papers [1–4] have studied related security and wireless issues.

Identity-Based Cryptography (IBC) which applies user identity as the public key was proposed by Shamir in 1984 [5] to reduce the complexity of key and certificate management. Developed by the Commercial Cryptography Administration of China in 2016, SM9 [6] has become the most typical identity-based cryptographic algorithm in China. Compared with traditional cryptographic algorithms, SM9 not only omits the exchange of digital certificates and public key processes, but also simplifies the deployment and management of the security systems. Because of its usability and simplicity, SM9 has been employed as the standard for commercial cryptography in China. Its digital signature algorithm has become an international standard as the main part of the ISO/IEC 14888-3/AMD1 in November 2017 [7] too. It is also adopted to secure various systems and scenarios like E-mail [8], cloud storage, intelligent devices [9], industrial control, online communications, mobile payment, and so on.

As described in [10] by Kocher et al. in 1999, it has been proved that even though mathematical characteristics can guarantee the security of cryptographic algorithms in theory, their implementation may suffer from Side-Channel Attack (SCA). SCA allows attackers to reveal secrets by analyzing the side information of an attacked device which is running a cryptographic algorithm, such as power consumption, electromagnetic radiation, and execution time. Because of the low cost and high efficiency, SCA has successfully cracked lots of devices which run DES [11], AES [12], RSA [13], and ECC

[14]. Despite SM9 algorithm being secure in cryptography theory, whether it is against SCA is still a matter of concern.

At present, the three main SCA techniques are Simple Power Attack (SPA), template attack, and fault attack. Due to the versatility and operability, they have been studied in depth and used to crack various cryptographic algorithms. SPA [15] exploits one trace to reconstruct the sequence of operations during the secret computation and derive information about the secrets from this sequence. As a special power analysis, template attack [16] makes a better use of all information present in each sample. And it is hence the strongest form of SCA possible in an information theoretic sense given the few samples that are available. Fault attack, as another main branch of SCA, often injects errors into cryptographic computation processes and identifies the secret key by analyzing the mathematical and statistical properties of wrong calculation results. Proposed by Biham and Shamir in [17], the fault attack on RSA has become a milestone for the security of public key cryptographic devices.

In this paper, we show that SCA does have a practical threat to the implementation of SM9. We propose the above three kinds of SCA attacks on SM9 algorithm. After this, some corresponding countermeasures are also introduced. The main contributions of this paper are as follows.

(1) A SPA attack on SM9 algorithm is proposed. And we also introduce some countermeasures to resist SPA.

(2) Different from general Elliptic Curve Digital Signature Algorithm (ECDSA), the key is a point on elliptic curves rather than a scalar in scalar multiplication for SM9 algorithm. According to this feature, a template attack is presented for SPA-resistant SM9 implementation and several countermeasures are provided.

(3) We propose a fault attack and conduct experiments to prove that software implementation of SM9 algorithm is vulnerable to this scheme. And then, some corresponding countermeasures are also presented.

This paper is organized as follows. In Section 2, the summarization of SM9 algorithm and its implementation are introduced. We give the basic idea of SPA on SM9 in detail and put forward some countermeasures in Section 3. Then, in Section 4, a template attack is provided to attack the protected SM9 which can resist SPA and the corresponding countermeasures are also given. In Section 5, a fault attack for SPA-resistant SM9 algorithm is presented and several countermeasures are introduced against this scheme. Finally, we conclude this paper in Section 6.

2. The Preliminaries

2.1. SM9 Digital Signature Generation Algorithm. SM9 digital signature algorithm usually assumes a scenario where Alice communicates with Bob. Alice generates the signature (h, S) of message M by SM9 digital signature generation algorithm for authentication and sends them to Bob. Bob validates the received message M' and its signature (h', S') with signature verification algorithm to ensure the authenticity and integrity of this digital signature.

In order to express clearly, we give the meanings of letters as follows. The signature private key of Alice denoted as ds_A is provided by Key Generation Center (KGC). Group G_1 and group G_2 are addition cyclic groups of order N and their generators are denoted as P_1 and P_2, respectively. Group G_T is a multiplicative cyclic group with order N. Let e denote the bilinear pairs mapping function from $G_1 \cdot G_2$ to G_T. KGC generates a random number ks as the signature master private key and computes $P_{\text{pub-}s} = [ks]P_2$ as the master public key. $P_{\text{pub-}s}$ is well-known, and ks is kept secretly by KGC. A cryptographic hash function is denoted as $H_2(m, N)$.

Algorithm 1 shows SM9 digital signature generation algorithm. At the beginning of the signature process, system parameters are provided as a part of inputs to make this algorithm work.

2.2. The Implementation of SM9 Digital Signature Algorithm

2.2.1. Scalar Multiplication. Scalar multiplication is the most important part in elliptic curve cryptography algorithm and its fast implementation is an inevitable demand of practical applications. As shown in Algorithm 1, scalar multiplication is directly related to the signature private key ds_A in SM9 digital signature generation algorithm.

The operation of adding a point P to itself for k times is called scalar multiplication and is denoted as $Q = kP$. For decades, many methods have been proposed to implement this operation and the most common is binary algorithm. There are two ways to implement scalar multiplication, left-to-right and right-to-left. And the former is shown in Algorithm 2 and the special point O is called the point at infinity. In the following sections, we assume that scalar multiplication is executed with the left-to-right binary algorithm.

In Algorithm 2, point doubling is executed n times. In probability, the count of 1 in a scalar integer k is close to $n/2$, so point addition is executed nearly $n/2$ times. Let A represent point addition and D represent point doubling; the operation quantity of binary method is approximately $(n/2) \cdot A + n \cdot D$.

2.2.2. Point Addition and Point Doubling. An elliptic curve is a set of points (x, y) in which $x, y \in F$. F denotes a finite field. The set of points on an elliptic curve, together with a special point O called the point at infinity, can be equipped with an Abelian group structure by addition operation. An elliptic curve over F can be expressed as the form of Weierstrass equation:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{1}$$

where $a_i \in F$. If prime number $p \neq 2$ and 3, the Weierstrass equation can be transformed to

$$y^2 = x^3 + ax + b, \tag{2}$$

with $a, b \in F$.

For prime number $p \neq 2$ and 3, let $P = (x_1, y_1) \neq O$ be a point. The inverse of P is $-P = (x_1, -y_1)$ and $Q = (x_2, y_2) \neq O$ is a second point with $Q \neq -P$. Point addition $P+Q = (x_3, y_3)$ can be calculated as

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1, \end{aligned} \tag{3}$$

Input: system parameters, signature master public key $P_{\text{pub-}s}$, message M, signature private key ds_A.
Output: M, (h, S).
(1) Compute $g = e(P_1, P_{\text{pub-}s})$ in G_T.
(2) Generate a random number $r \in [1, N-1]$.
(3) Compute $w = g^r$ in G_T, and convert the data type of w to bit stream.
(4) Compute $h = H_2(M \parallel w, N)$.
(5) Compute $l = (r - h) \bmod N$.
(6) **if** $l == 0$ **then**
(7) **goto** (2).
(8) Compute $S = [l]ds_A$ in G_1.
(9) Convert the data type of h and S to byte stream.
(10) **return** M, (h, S)

Algorithm 1: SM9 digital signature generation algorithm.

Input: point P, n-bit integer $k = \sum_{i=0}^{n-1} k_i 2^i$, $k_i \in \{0, 1\}$.
Output: $Q = kP$.
(1) $Q = O$.
(2) **for** $i = n - 1$ to 0 **do**
(3) $Q = 2Q$.
(4) **if** $k_i == 1$ **then**
(5) $Q = Q + P$.
(6) **end for**
(7) **return** Q

Algorithm 2: Left-to-right binary algorithm.

with

$$\lambda = \begin{cases} \dfrac{(y_2 - y_1)}{(x_2 - x_1)}, & P \neq Q \\ \dfrac{\left(3x_1^2 + a\right)}{(2y_1)}, & P = Q. \end{cases} \tag{4}$$

The operation of $P + Q$ is called point addition if $P \neq Q$. And it is called point doubling if $P = Q$. Obviously, point addition differs from point doubling in the form of formula.

2.2.3. Montgomery Modular Multiplication. Given a modulus N and two integers X and Y of size N in base b, with $\gcd(N, b) = 1$ and $r = b^{\lceil \log_b(N) \rceil}$, Montgomery modular multiplication algorithm [18] computes:

$$\text{MontMul}(X, Y, N) = X \cdot Y \cdot r^{-1} \bmod N. \tag{5}$$

Montgomery modular multiplication algorithm is shown in Algorithm 3. Its essence is to combine $X \cdot Y$ and $U \cdot N$ by traditional multiplication method. As described in line 2 to line 8, U is unknown at first. With obtaining u_i calculated by $x_i \cdot y_0$, each $u_i \cdot N$ can be calculated too. Finally, U can be derived. MontMul(X, Y, N) can be computed by alternating all $x_i \cdot Y$ and $u_i \cdot N$ and adding up to A. Figure 1 illustrates the intuitive graphical representation of CIOS modular multiplication [19, 20].

2.3. Power Analysis Attack. In SPA [15], attackers directly observe power consumption for a single execution of target operation without any statistical methods. As a special power analysis, template attack [16] generally consists of the following three phases. The first is template building phase, and attackers build templates to characterize devices by executing a sequence of instructions on fixed data. Next, it allows attackers to match the templates to the power consumption traces of devices in template matching phase. Finally, attackers can do some analysis and derive secret information during offline searching phase.

Hamming weight model proposed in [10, 21] analyzes the correlation between power consumption and the register switching from one state to the other. It is generally assumed that power consumption depends on the number of bits switching from 0 to 1 or 1 to 0 within the corresponding time. For n-bit register, binary data $D = [d_{n-1} d_{n-2} \cdots d_1 d_0]$ is coded as $D = \sum_{i=0}^{n-1} d_i 2^i$ with $d_i = 0$ or 1; its Hamming weight $\text{HW}(D) = \sum_{i=0}^{n-1} d_i$ is the number of bits set to 1. Considering a chip as a large set of elementary electrical components, its power consumption contains not only the state changes but also other variables' consumption, such as offsets, time dependent components, and noise. Therefore, the basic model for the data dependency can be described as $W = a\text{HW}(D) + b$, where a is a constant and b indicates the other consumption.

2.4. Fault Attack. Fault attack [17] allows attackers to disturb cryptographic devices by physical methods to make them

Input: $X = (x_{n-1} \cdots x_1 x_0)_b$, $Y = (y_{n-1} \cdots y_1 y_0)_b$, $N = (n_{n-1} \cdots n_1 n_0)_b$, $n' = -N^{-1} \bmod b$, $\gcd(N, b) = 1$, $r = b^{\lceil \log_b(N) \rceil}$.
Output: $X \cdot Y \cdot r^{-1} \bmod N$.
(1) $A = (a_{n-1} \cdots a_1 a_0)_b \leftarrow 0$.
(2) **for** $i = 0$ to $n - 1$ **do**
(3) $A \leftarrow A + x_i \cdot Y$.
(4) $u_i \leftarrow a_0 \cdot n' \bmod b$.
(5) $A \leftarrow (A + u_i \cdot N)/b$.
(6) **if** $A \geq N$ **then**
(7) $A \leftarrow A - N$.
(8) **end for**
(9) **return** A

Algorithm 3: Montgomery modular multiplication algorithm.

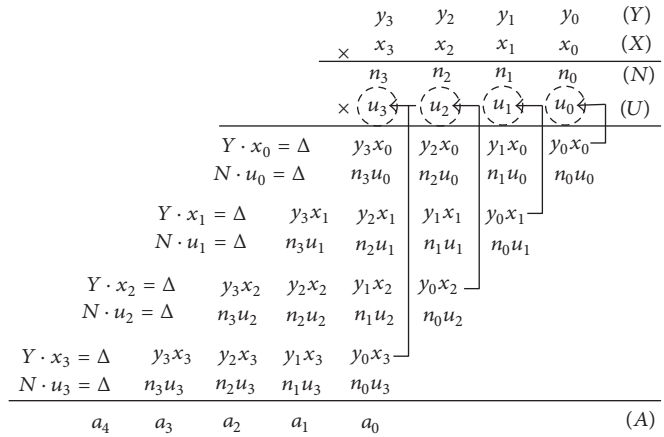


Figure 1: CIOS modular multiplication.

run in wrong states. Due to the injected fault, the devices perform some operations in modified environment and produce incorrect results. Combining with the algorithm in the devices, some knowledge related to the secret key could be gained from the results. Because of the lower cost, simple operation, and obvious effect, fault attack has become one of the most concerned SCA techniques.

Faults in devices can be made for a variety of reasons. In general, variations in normal working conditions can be injecting faults into a system effectively. For example, changing supply voltage or clock frequency can disrupt execution process and cause the processor to skip some instructions or change its output. Exposing devices in the temperatures outside its operational range usually makes random modifications to the memory. It is also possible to inject faults more accurately by using the inherent photoelectric effects of electric circuits. Under the exposure of photons, devices can produce induced currents and disrupt normal operations. In fact, lasers can make faults more precise in terms of target area and injection time. Also, faults can be injected in packaged circuits without removing the packaging by X-rays and ion beams.

2.5. Problem Formulation. Different from the general ECDSA and SM2 algorithm, the secret key in scalar multiplication is the point on elliptic curves rather than the scalar for SM9 algorithm. The scalar of the classical ECC and SM2 is a secret and the point is known, while the scalar and the point are both unknown in SM9. Therefore, the attack methods of the two are fundamentally different. In this paper, we focus on this issue to present template attack described in Section 4 and fault attack described in Section 5 which are only applied to SM9.

3. Simple Power Attack and Countermeasures on SM9

3.1. Simple Power Attack on SM9. In this paper, we focus the computation of scalar multiplication $S = [l]ds_A$ to apply our SPA attack which is described in line 8 of Algorithm 1.

Our SPA attack against SM9 recovers l by observing the differences in power consumption caused by the difference operations for bit 1 and bit 0. As described in Algorithm 2, it always performs a point doubling operation whether the bit is 0 or 1. And an extra point addition will be performed if the bit is 1. Because of the differences between side-channel pattern of doubling and that of addition, attackers can easily reveal l from a single power trace. As N and S are both known, ds_A can be calculated by the formula $ds_A = l^{-1}S$.

Attackers can also perform SPA attack on $w = g^r$ as shown in line 3 of Algorithm 1. The SPA attack on modular exponentiation is similar to that of scalar multiplication. Employing the different power consumption by manipulating 1 and 0 can derive the exponent r. The lengths of r, l, and N are 256 bits so that l can be obtained by the formula $l = (r - h) \bmod N$. According to the scheme described above, attackers can also restore the secret key ds_A.

3.2. Countermeasures against Simple Power Attack. Based on our SPA attack, we can draw a conclusion that r and l are equally important in the security of SM9 digital signature algorithm. It is necessary to deploy some countermeasures on r and l against SPA attack.

There are five ways [22, 23] against SPA scheme for l in SM9 digital signature algorithm. In addition, countermeasures to protect the exponent r should be implemented. We would not repeat the descriptions about the methods for r here as they are similar to that of l. We also can refer to the SPA countermeasures of RSA [13] for r against SPA attack.

In conclusion, the five ways are as follows.

(1) Indistinguishable Point Operation Formulae. Indistinguishable Point Operation Formulae (IPOF) try to eliminate the difference between point addition and point doubling. The usage of unified formulae for point doubling and addition is a special case of IPOF. However, even when unified formulae are in use, the implementation of the underlying arithmetic, especially the operations with conditional instructions, may still reveal the type of the point operation (addition or doubling).

(2) Double-and-Add-Always Algorithm. The double-and-add-always algorithm ensures that the sequence of operations during a scalar multiplication is independent of the scalar by inserting dummy point additions. However, due to the use of dummy operations, it makes the time complexity doubled and may cause safe-error fault attack.

(3) Atomic Block Algorithm. Instead of making the group operations indistinguishable, one can rewrite them as sequences of side-channel atomic blocks that are indistinguishable for SPA attack. If dummy atomic blocks are added, then this countermeasure may enable safe-error attack.

(4) Montgomery Ladder Method. Because the intermediate values are stored in registers randomly in Montgomery ladder method, the Hamming weight of secret information would not leak out to attackers.

(5) Random Splitting l and r. There are two different ways to split l and r. Here we take l as an example and r also should be protected by the same principle. One is to transform l to $l+kN$ where k is a random integer and do scalar multiplication by the formula:

$$\begin{aligned} S &= [l]\, ds_A \longrightarrow \\ S &= [l + kN]\, ds_A. \end{aligned} \tag{6}$$

Another is to convert l to $l + k$ and k where k is a 256-bit random value. The formula is

$$\begin{aligned} S &= [l]\, ds_A \longrightarrow \\ S &= [l + k]\, ds_A. \end{aligned} \tag{7}$$

4. Template Attack and Countermeasures on SM9

4.1. Template Attack on SM9. The template attack proposed in this paper reveals ds_A in the case that both l and r are unknown. As shown in Section 2, the first step of scalar multiplication performs a point doubling operation. And λ needs to be calculated by

$$\lambda = \frac{3x^2 + a}{2y}, \tag{8}$$

where x and y represent the x-coordinate and y-coordinate of ds_A. We focus this computation x^2 to perform our template attack. For ease of description, we use X (256-bit) to replace x in Formula (8) and x_i ($0 \le i < 32$) denotes one byte of X.

Assume that SM9 digital signature algorithm is executed on an 8-bit microcontroller, and the power consumption of intermediate values in the calculation process of CIOS modular multiplication algorithm can be acquired. We give the letters meanings as follows. $X = (x_{31}, \ldots, x_1, x_0)$ is the x-coordinate of $ds_A(X, Y)$. And $t_h(x_i x_j)$ and $t_l(x_i x_j)$ are the power consumption traces with the high 8-bit and low 8-bit of the intermediate value $x_i x_j$ ($0 \le i < 32$, $0 \le j < 32$). The template with Hamming weight from 0 to 8 is denoted as $T_\tau = \{\tau_0, \tau_1, \ldots, \tau_8\}$ and $\mathrm{Match}(t, T_\tau)$ is a method to reflect the degree of t and T_τ. $\mathrm{HW}_h(x_i x_j)$ and $\mathrm{HW}_l(x_i x_j)$ denote the Hamming weight of high 8-bit and low 8-bit of $x_i x_j$, respectively.

As shown in Figure 2, there are three phases in our template attack. Firstly, in the template building phase, we focus on the operation $x_i x_j$ ($0 \le i < 32$, $0 \le j < 32$) and calculate the Hamming weight of high 8-bit and low 8-bit of $x_i x_j$ as the target of template. We build templates of Hamming weight $T_\tau = \{\tau_0, \tau_1, \ldots, \tau_8\}$ to characterize devices. Next, we match the templates to the power consumption traces of devices with the match function $\mathrm{Match}(t, T_\tau)$ to obtain $\mathrm{HW}_h(x_i x_j)$ and $\mathrm{HW}_l(x_i x_j)$ in template matching phase. Finally, two searching operations are carried out during offline searching phase and the secret key ds_A can be derived by analysis.

Algorithm 4 shows our template attack. There are two searching operations used to narrow the range of candidates in offline searching phase. The first is shown in offline searching phase from the line 1 to the line 6. For each x_i ($0 \le i < 32$), we traverse j from 0 to 255, and add j satisfying $\mathrm{HW}_h(j^2) == \mathrm{HW}_h(x_i x_i)$ and $\mathrm{HW}_l(j^2) == \mathrm{HW}_l(x_i x_i)$ to set $U_{1,i}$ where candidates of x_i are stored. The second searching is based on $U_{1,i}$, as demonstrated in line 7 to the line 14. For each element a in $U_{1,i}$ ($0 \le i < 32$) and each element b in $U_{1,j}$ ($i < j < 32$), we calculate the high 8-bit and low 8-bit Hamming weight of ab. Selecting a and b with the conditions

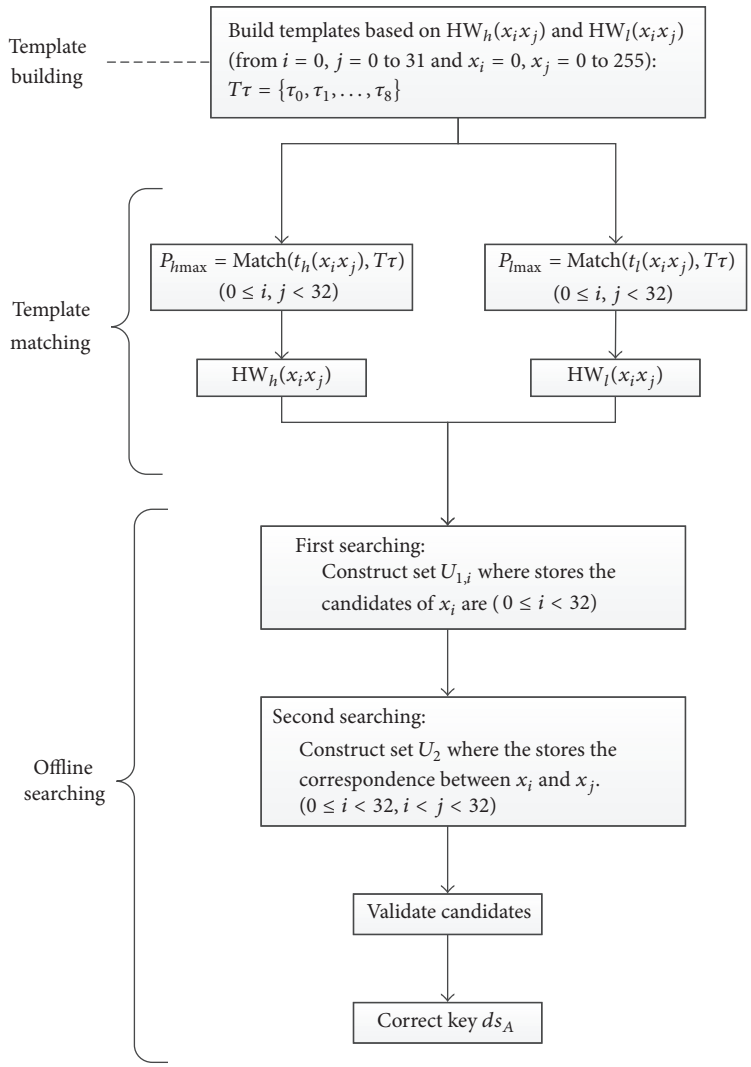


Figure 2: The flowchart of template attack on SM9.

of $HW_h(ab) == HW_h(x_ix_j)$ and $HW_l(ab) == HW_l(x_ix_j)$ and adding them in set U_2, U_2 consists of many pairs (i, j, a, b) and represents the correspondence between x_i and x_j. The pair (i, j, a, b) means that if $x_i == a$ so $x_j == b$. Next, the possible values of $X = (x_{31}, \ldots, x_1, x_0)$ can be obtained. It is necessary to validate whether they are the points of the elliptic curve. Finally, the secret key ds_A can be recovered.

4.2. Template Attack Experiments on SM9. We present concrete experiments on side-channel traces captured from a real device. We implemented the Montgomery modular multiplication algorithm and focused on a single precision multiplication power consumption on AT89S52 8-bit microcontroller. Traces were acquired on a Lecroy WaveRunner oscilloscope with a sampling rate of 10 GS/s. In our experiments, the parameters X and Y of ds_A are shown in Table 1.

The templates of Hamming weight from 0 to 8 built in our attack are illustrated in Figures 3 and 4.

Table 1: Template attack experiment parameters.

Parameter	Value (hexadecimal)
X	93DE051D 62BF718F F5ED0704 487D01D6
	E1E40869 09DC3280 E8C4E481 7C66DDDD
Y	21FE8DDA 4F21E607 63106512 5C395BBC
	1C1C00CB FA602435 0C464CD7 0A3EA616

During our template attack, the power consumption of intermediate values in the calculation process of CIOS modular multiplication algorithm is acquired. Figure 5 shows the $t_h(x_0x_0)$ (black-line) and $t_l(x_0x_0)$ (dark-gray-line) which are the power consumption traces with the high 8-bit and low 8-bit of x_0x_0. The $\text{Match}(t, T_\tau)$ applied in our attack is least square method to reflect the distance of t and T_τ. Hence, $HW_h(x_0x_0)$ and $HW_l(x_0x_0)$ were revealed which were equal to 6 and 4, respectively. For each i from 0 to 31, $HW_h(x_ix_i)$ and $HW_l(x_ix_i)$ can be recovered by the steps described above.

Input: $t_h(x_i x_j), t_l(x_i x_j)$ $(0 \le i < 32, 0 \le j < 32)$.
Output: $ds_A(X, Y)$.
Template Building Phase:
(1) Build power consumption templates $T_\tau = \{\tau_0, \tau_1, \ldots, \tau_8\}$ based on $\mathrm{HW}_h(x_i x_j)$ and $\mathrm{HW}_l(x_i x_j)$ where i, j from 0 to 31 and x_i, x_j from 0 to 255.
Template Matching Phase:
(1) **for** $i = 0$ to 31 **do**
(2) **for** $j = 0$ to 31 **do**
(3) calculate $p_{h\max} = \mathrm{Match}(t_h(x_i x_j), T_\tau)$ to recover $\mathrm{HW}_h(x_i x_j)$.
(4) calculate $p_{l\max} = \mathrm{Match}(t_l(x_i x_j), T_\tau)$ to recover $\mathrm{HW}_l(x_i x_j)$.
(5) **end for**
(6) **end for**
Off-line Searching Phase:
(1) **for** $i = 0$ to 31 **do**
(2) **for** $j = 0$ to 255 **do**
(3) **If** $\mathrm{HW}_h(j^2) == \mathrm{HW}_h(x_i x_i)$ && $\mathrm{HW}_l(j^2) == \mathrm{HW}_l(x_i x_i)$ **then**
(4) Add j to set $U_{1,i}$ where stores candidates of x_i.
(5) **end for**
(6) **end for**
(7) **for** $i = 0$ to 31 **do**
(8) **for** $j = i + 1$ to 31 **do**
(9) **for** each a in $U_{1,i}$ && each b in $U_{1,j}$ **do**
(10) **If** $\mathrm{HW}_h(ab) == \mathrm{HW}_h(x_i x_j)$ && $\mathrm{HW}_l(ab) == \mathrm{HW}_l(x_i x_j)$ **then**
(11) Add the pair (i, j, a, b) to set U_2.
(12) **end for**
(13) **end for**
(14) **end for**
(15) Search $X = (x_{31}, ..., x_1, x_0)$ in U_2.
(16) **for** each X in U_2 **do**
(17) Compute the corresponding Y and verify X and Y.
(18) **return** $ds_A(X, Y)$

Algorithm 4: Template attack on SM9.

Table 2: The sets of candidates of x_i.

Parameter	$U_{1,i}$	The set of candidates of x_i (hexadecimal)
x_0	$U_{1,0}$	DB, DD, ED
x_1	$U_{1,1}$	DB, DD, ED
x_2	$U_{1,2}$	1D, 1F, 26, 29, 31, 32, 33, 43, 4F, 61, 66, 9D, B6
...	...	...
...	...	...
...	...	...
x_{29}	$U_{1,29}$	05, 07, 09, 0A, 0E
x_{30}	$U_{1,30}$	1E, 34, 5E, 6F, DE
x_{31}	$U_{1,31}$	2B, 37, 3B, 47, 55, 56, 67, 71, 93, AA, B9, E7

Then, the first searching phase is performed. For each i from 0 to 31, we traverse j from 0 to 255 and calculate high 8-bit and low 8-bit Hamming weight of j^2. Selecting j with the conditions of $\mathrm{HW}_h(j^2) == \mathrm{HW}_h(x_i x_i)$ and $\mathrm{HW}_l(j^2) == \mathrm{HW}_l(x_i x_i)$ and adding j to set $U_{1,i}$, the set $U_{1,i}$ where candidates of x_i are stored is given as shown in Table 2.

And then, the second searching phase is carried out. For each element a in $U_{1,i}$ $(0 \le i < 32)$ and each element b in $U_{1,j}$ $(i < j < 32)$, we calculate the high 8-bit and low 8-bit Hamming weight of ab. Select a and b with the conditions of $\mathrm{HW}_h(ab) == \mathrm{HW}_h(x_i x_j)$ and $\mathrm{HW}_l(ab) == \mathrm{HW}_l(x_i x_j)$ and adding them in set U_2. U_2 consists of many pairs (i, j, a, b) and represents the correspondence between x_i and x_j. Next, the possible values of $X = (x_{31}, \ldots, x_1, x_0)$ can be obtained. The result of offline searching phase is shown in Figures 6 and 7. Finally, the value of X is 93DE051D 62BF718F F5ED0704 487D01D6 E1E40869 09DC3280 E8C4E481 7C66DDDD and it is validated to be correct. And ds_A can be revealed

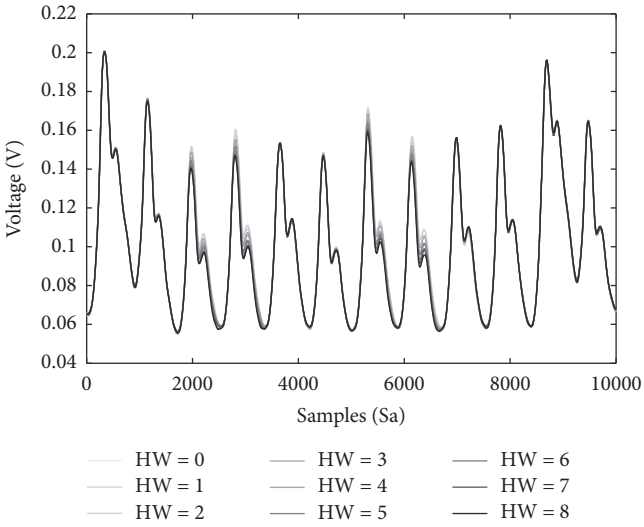


Figure 3: The templates of Hamming weight from 0 to 8.

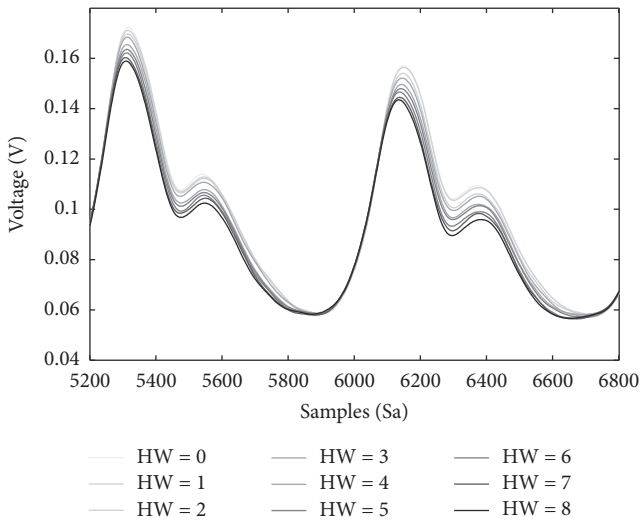


Figure 4: The templates of Hamming weight from 0 to 8 (zoom in).

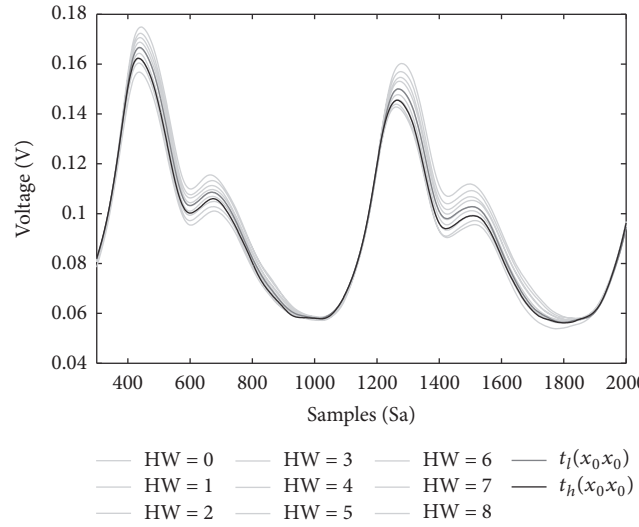


Figure 5: Template matching results of $t_h(x_0x_0)$ and $t_l(x_0x_0)$.

successfully. The experiments have proved that our attack works well. If attackers employ the template attack on an 8-bit microcontrol unit, the key often can be obtained just by analyzing algorithm procedure of one message.

4.3. Countermeasures Against Template Attack. There are four ways [23, 24] to resist the template attack presented above.

(1) Base Point Blinding. This method is to select a random point R and convert $S = [l]ds_A$ to $S = [l](R + ds_A) - [l]R$. The value of $[l](R + ds_A)$ is computed first and the known value $[l]R$ is subtracted at the end of scalar multiplication to ensure its correctness. The $[l]R$ and R are stored secretly in the cryptographic devices and updated at each iteration.

(2) Random Projective Coordinates. Let $ds'_A(X', Y', Z)$ denote the mapping value of $ds_A(X, Y)$ in the Jacobian coordinates where $X = X'/Z^2$ and $Y = Y'/Z^3$. Point $ds'_A(X', Y', Z)$ is considered to be the same as point $Q(r^2X', r^3Y', rZ)$ where r is a random number. For different r, ds_A is not the same. The random variable r can be updated in every execution or after each doubling or addition. Attackers could not accurately gain the values of ds_A and $2ds_A$, so this method does resist the attack.

(3) Random EC Isomorphism. Based on the isomorphism of elliptic curves, we transform scalar multiplication algorithm to another stochastic mapping domain. A random isomorphism curve $E'(F_p)$ is generated and ds_A is mapped to ds'_A which is on $E'(F_p)$. The calculation of S is on $E'(F_p)$ rather than $E(F_p)$. Finally, we should map ds'_A to ds_A and get S. As ds'_A can not be estimated, this method can resist the template attack.

(4) Random Field Isomorphism. This method makes use of isomorphisms between fields. To compute $S = [l]ds_A$, it first randomly chooses a field K' to K through isomorphism ψ and then computes

$$Q = \psi^{-1}\left(k\left(\psi\left(P\right)\right)\right). \tag{9}$$

It means that $Q \in E'(K')$ is used to represent $ds_A \in E(\mathrm{K})$ and $S' = [l]Q$ is calculated on $E'(K')$. Finally we need to transform S' to $S \in E(K)$. The value of ds_A is hidden in this method so as to resist the above attack.

Figures 8 and 9 show the templates of Hamming weight from 0 to 8 after applying some countermeasures.

5. Fault Attack and Countermeasures on SM9

5.1. Fault Attack on SM9. Base point is one of the system parameters in scalar multiplication, and it is determined by protocols. So we inject a single-bit fault on base point to recover the secret key. Two assumptions are made before

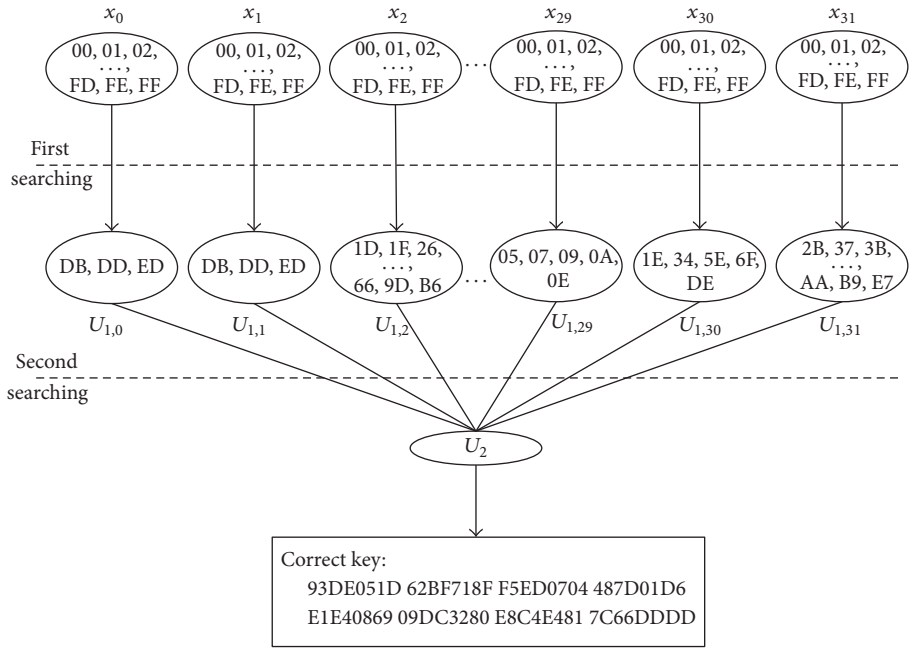


Figure 6: The result of offline searching phase.

the fault attack on SM9 is performed. One hypothesis is that attackers can inject a fault on $ds_A(X, Y)$ at an unknown location during the moments of error detection and signature computation. Let $ds'_A(X', Y')$ denote the fault key. Another assumption is that the injected fault causes a bit value to flip from 0 to 1.

Figure 10 illustrates the flowchart of fault attack on SM9. As an example, we performed the fault attack on $E : y^2 = x^3 + b$. The correct signature for message M is calculated by the formula $S(X_S, Y_S) = [l]ds_A(X, Y)$. Then, a single-bit fault is injected to the x-coordinate of $ds_A(X, Y)$ and the fault key is denoted as $ds'_A(X', Y')$. The wrong signature for message M is calculated by the formula $S'(X'_S, Y'_S) = [l']ds'_A(X', Y')$. Obviously, $S(X_S, Y_S)$ and $ds_A(X, Y)$ are on $E : y^2 = x^3 + b$, while $S'(X'_S, Y'_S)$ and $ds'_A(X', Y')$ are on $E' : y'^2 = x'^3 + b'$. As $S(X_S, Y_S)$ and $S'(X'_S, Y'_S)$ are both known, b and b' can be calculated. So we have

$$\begin{aligned} Y^2 &= X^3 + b, \\ Y'^2 &= X'^3 + b', \\ Y &= Y'. \end{aligned} \tag{10}$$

Then, (11) can be gained:

$$X^3 - X'^3 + b - b' = 0 \pmod{p}. \tag{11}$$

Let $\alpha = b - b'$; (11) is equal to

$$X^3 - X'^3 + \alpha = 0 \pmod{p}. \tag{12}$$

Because only one bit is different between X and X', (12) can be expressed as

$$X^3 - \left(X + 2^n\right)^3 + \alpha = 0 \pmod{p}, \tag{13}$$

where $n \in \{0, 1, \ldots, 255\}$. And n represents the location of the different bit.

Convert (13) into the following form:

$$3 \cdot 2^n \cdot X^2 + 3 \cdot 2^{2n} \cdot X + 2^{3n} - \alpha = 0 \pmod{p}. \tag{14}$$

For each n from 0 to 255, its corresponding equation like (14) can be obtained. All values of X satisfying this equation are calculated by mathematical methods and added to set U_X. If U_X is an empty set, that means the fault position n corresponding to this equation is incorrect. Otherwise, the fault position n is correct and we need to validate each $X \in U_X$ for getting the real X. It is worth noting that as long as n is correct, the equation must have solutions. Once the X is derived, the value of $ds_A(X, Y)$ can be calculated correctly.

Obviously, if the fault is injected into the y-coordinate of $ds_A(X, Y)$, the above fault attack is still performed validly. And our fault attack also has the ability to break the SPA-resistant SM9 implementation.

To sum up the above, the fault attack algorithm is shown in Algorithm 5.

Remark 1. In the above attack, we assume that the injected fault causes a bit to flip from 0 to 1. As we all know, common faults are single-bit flipping fault, single-bit constant fault, and multibits fault. In fact, no matter which fault is injected, as long as $\alpha \neq 0$, the above attack can be performed. If $\alpha \neq 0$, for single-bit constant fault, we can think of it as a single-bit flipping fault. For multibits fault, the only difference

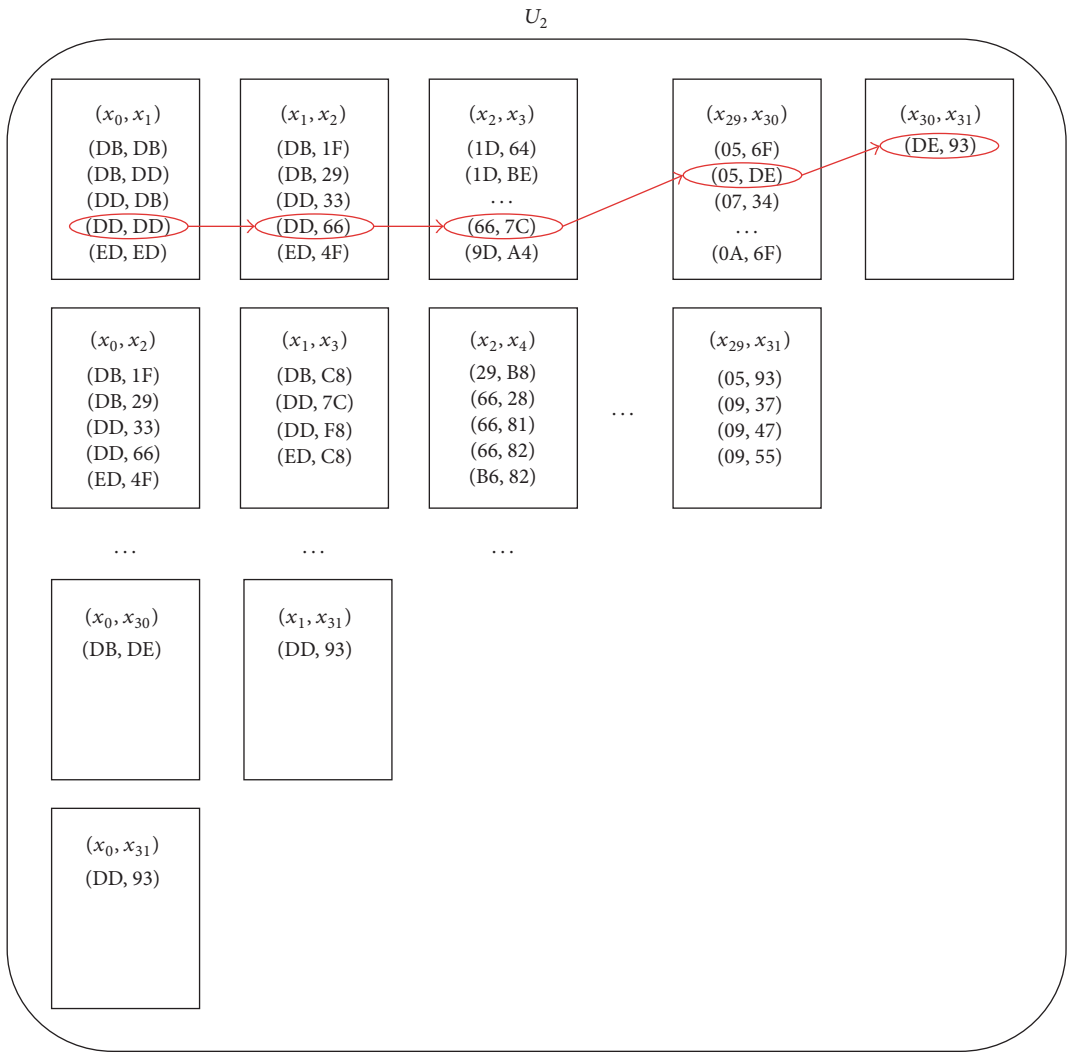


Figure 7: The set U_2 after offline searching phase.

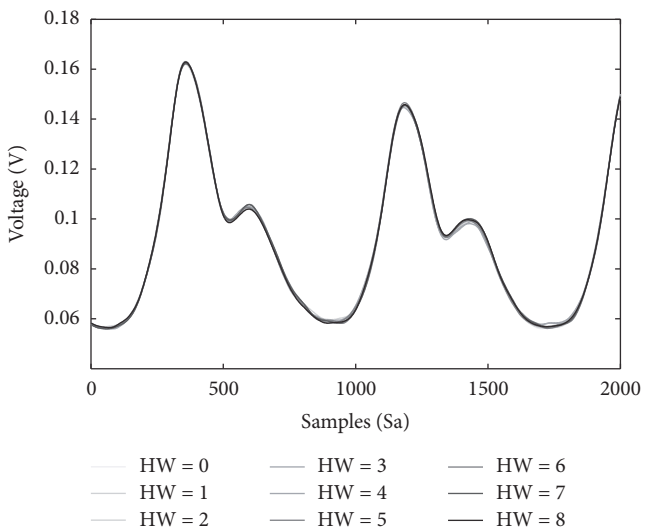


Figure 8: The protected templates of Hamming weight.

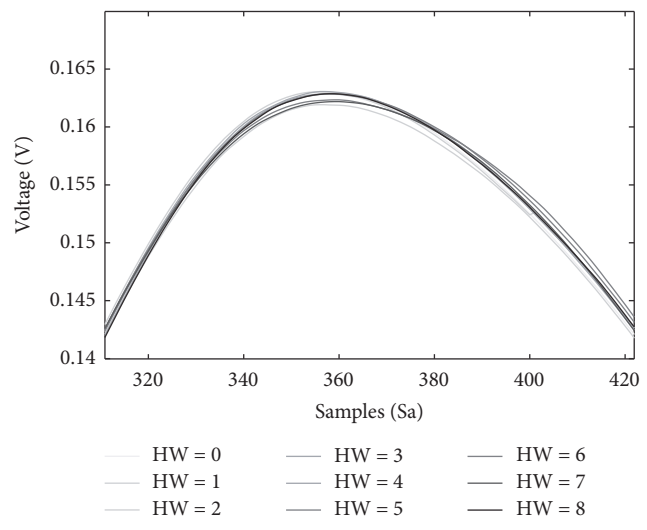


Figure 9: The protected templates of Hamming weight (zoom in).

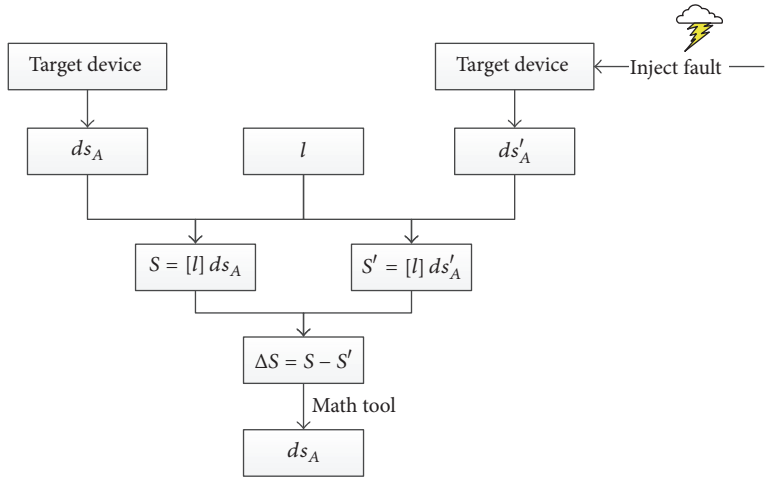


Figure 10: The flowchart of fault attack on SM9.

Input: SM9 system parameters, message M.
Output: $ds_A(X, Y)$.
(1) Calculate the correct signature value $S(X_S, Y_S) = [l]ds_A(X, Y)$.
(2) Inject fault on $ds_A(X, Y)$ and denote as $ds'_A(X', Y')$.
(3) Calculate the wrong signature value $S'(X'_S, Y'_S) = [l']ds'_A(X', Y')$.
(4) Calculate $b = Y_S^2 - X_S^3$.
(5) Calculate $b' = Y_S'^2 - X_S'^3$.
(6) Establish equations of X and X'.
(7) **for** $n = 0$ to 255 **do**
(8) Compute X by $3 \cdot 2^n \cdot X^2 + 3 \cdot 2^{2n} \cdot X + 2^{3n} - \alpha = 0 \pmod p$.
(9) **end for**
(10) Calculate Y by $y^2 = x^3 + b$ and verify X and Y.
(11) **return** $ds_A(X, Y)$

Algorithm 5: Fault attack on SM9.

is that multi-n should be searched. We believe that using mathematical methods to solve this issue is not a difficult task.

5.2. Fault Attack Experiments on SM9. Assume that the elliptic curve used in our experiments is $E : y^2 = x^3 + b$. The q is a prime number, and N means the order of group G_1. The $ds_A(X, Y)$ denotes a point of G_1, and the scalar is denoted as l. $S(X_S, Y_S)$ is computed by $S(X_S, Y_S) = [l]ds_A(X, Y)$. Let $ds'_A(X', Y')$ denote the fault point, and the wrong signature of message M is computed by the formula $S'(X'_S, Y'_S) = [l']ds'_A(X', Y')$. The experiment parameters are listed in Table 3. We can see that the fault location is equal to 1.

We conduct some experiments with the attack in Algorithm 5, and the intermediate values of SM9 digital signature algorithm are shown in Table 4. When $n = 1$, the equation has two solutions that may be the correct X. Let X_1 and X_2 denote the two solutions, and Y_{11}, Y_{12}, Y_{21}, and Y_{22} denote the corresponding Y calculated by $y^2 = x^3 + b$. Comparing with Table 3, X_1 and Y_{12} are the correct x-coordinate and y-coordinate of ds_A, so the fault attack also shows its feasibility.

Table 3: Fault attack experiment parameters.

Parameter	Value (hexadecimal)
q	B6400000 02A3A6F1 D603AB4F F58EC745
	21F2934B 1A7AEEDB E56F9B27 E351457D
N	B6400000 02A3A6F1 D603AB4F F58EC744
	49F2934B 18EA8BEE E56EE19C D69ECF25
X	93DE051D 62BF718F F5ED0704 487D01D6
	E1E40869 09DC3280 E8C4E481 7C66DDD**D**
X'	93DE051D 62BF718F F5ED0704 487D01D6
	E1E40869 09DC3280 E8C4E481 7C66DDD**F**
Y	21FE8DDA 4F21E607 63106512 5C395BBC
	1C1C00CB FA602435 0C464CD7 0A3EA616
l	291FE3CA C8F58AD2 DC462C8D 4D578A94
	DAFD5624 DDC28E32 8D293668 8A86CF1A

Usually, attackers can reveal the 256-bit ds_A by enabling the device to execute twice and comparing the two different results by our fault attack.

Table 4: The intermediate values of fault attack.

Intermediate values	Value (hexadecimal)
X_S	A5702F05 CF131530 5E2D6EB6 4B0DEB92
	3DB1A0BC F0CAFF90 523AC875 4AA69820
Y_S	78559A84 4411F982 5C109F5E E3F52D72
	0DD01785 392A727B B1556952 B2B013D3
b	05
X'_S	48F78AA4 9A9443DE 0DC72D8C A91BB73C
	1B2D6C4F 1FB2DA61 2E9EFACA 45D87116
Y'_S	1FC550A9 CE2DCEAD 83CF500D AEB690A4
	4401D3E6 11DC9F02 B3D5FE40 482C72E6
b'	66B99EE0 079E3769 9982B5B7 280386ED
	D0DA26F4 BB2CE030 3F07C542 841F257C
α	66B99EE0 079E3769 9982B5B7 280386ED
	D0DA26F4 BB2CE030 3F07C542 841F2577
n	01
X_1	**93DE051D 62BF718F F5ED0704 487D01D6**
	E1E40869 09DC3280 E8C4E481 7C66DDDD
Y_{11}	94417225 B381C0EA 72F3463D 99556B89
	05D6927F 201ACAA6 D9294E50 D9129F67
Y_{12}	**21FE8DDA 4F21E607 63106512 5C395BBC**
	C1C00CB FA602435 0C464CD7 0A3EA616
X_2	2261FAE2 9FE43561 E016A44B AD11C56E
	400E8AE2 109EBC5A FCAAB6A6 66EA679E
X_{21}	9A426326 D3E2BCF1 AFA21657 5CF85BA8
	B347DB94 ECF14C97 104FCE07 67C23979
Y_{22}	1BFD9CD9 2EC0EA00 266194F8 98966B9C
	6EAAB7B6 2D89A244 D51FCD20 7B8F0C04

5.3. Countermeasures against Fault Attack. We introduce three countermeasures against the above fault attack [25–27]; they are as follows.

(1) Point Validation. This method verifies if a point lies on the specified curve or not. It should be performed before and after scalar multiplication. If the point ds_A or result does not belong to the original curve, no output should be given. It is an effective countermeasure against our fault attack.

(2) Curve Integrity Check. The curve integrity check is to detect faults on curve parameters. Before starting an SM9 algorithm the curve parameters are read from the memory and verified using an error detecting code (i.e., cyclic redundancy check) before the algorithm execution. It is an effective method to prevent the fault attack above.

(3) Coherence Check. A coherence check verifies the intermediate or final results with respect to a valid pattern. Randomly coding the intermediate variables such as scalar, base point, and curve parameters is a most common operation.

(4) Combined Curve Check. This method uses a reference curve to detect faults. It makes use of two curves: a reference curve $E := E(F_p)$ and a combined curve E' that is defined over the ring Z'. In order to compute $S = [l]ds_A$ on curve E, it first generates a combined point ds'_A from ds_A and $ds_A \in E(F_p)$ (with prime order). Two scalar multiplications are then performed: $S' = [l]ds'_A$ on E' and $S = [l]ds_A$ on E. If no error occurred, S and S' (mod p) will be equal. Otherwise, the one of the results is faulty and the results should be aborted. It is also an effective countermeasure against the fault attack presented in this paper.

(5) Security Curve Selection. Using the NIST curves [28] with the fragile twin curves should be avoided.

6. Conclusion

In this paper, we propose SPA attack, template attack, and fault attack on SM9 algorithm. After this, some corresponding countermeasures are also introduced. We also conduct some experiments to prove the validity of these attacks. Although this paper mainly studies SCA attacks of SM9 digital signature algorithm, we have reason to believe that these schemes are equally effective for SM9 encryption algorithm.

Overall, a security SM9 digital signature algorithm implementation should pay attention to these points shown in Table 5. And we also provide the overhead to deploy these countermeasures as a reference guidance for the SM9 algorithm security implementation [27].

Table 5: Countermeasures and overhead. Cost estimation: negligible (<10%), low (10%–50%), and high (>50%).

Countermeasures	Target attacks	Computation overhead
Indistinguishable point operation	SPA	Low
Double-and-add-always	SPA	Low
Atomic block	SPA	Negligible
Montgomery ladder	SPA	Low
Random splitting	SPA	High
Base point blinding	Template attack	Negligible
Random projective coordinates	Template attack	Negligible
Random EC isomorphism	Template attack	Low
Random field isomorphism	Template attack	Low
Point validation	Fault attack	Negligible
Curve integrity check	Fault attack	Negligible
Coherence check	Fault attack	Low
Combined curve check	Fault attack	Low
Security curve selection	Fault attack	-

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by Beijing Natural Science Foundation (no. 4162053), the National Cryptography Development Fund (no. MMJJ20170201), the Foundation of Science and Technology on Information Assurance Laboratory, and Beijing Institute of Technology Research Fund Program for Young Scholars.

References

[1] X. Hei, X. Du, J. Wu, and F. Hu, "Defending resource depletion attacks on implantable medical devices," in *Proceedings of the of IEEE GLOBECOM*, MIA, FL, USA, 2010.

[2] X. Du, Y. Xiao, M. Guizani, and H.-H. Chen, *An Effective Key Management Scheme for Heterogeneous Sensor Networks, Ad Hoc Networks*, vol. 5, Elsevier, 2007.

[3] X. J. Du, M. Guizani, Y. Xiao, and H.-H. Chen, "Transactions papers a routing-driven Elliptic Curve Cryptography based key management scheme for Heterogeneous Sensor Networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1223–1229, 2009.

[4] X. Du, M. Guizani, Y. Xiao, and H.-H. Chen, "Secure and efficient time synchronization in heterogeneous sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 4, pp. 2387–2394, 2008.

[5] A. Shamir, *Identity-based cryptosystems and signature schemes Crypto*, vol. 84, 1984.

[6] F. Yuan and Z. Cheng, "Overview on SM9 Identity-Based Cryptographic Algorithm," *Journal of Information Security Research*, 2016.

[7] *ISO/IEC 14888-3:2016 DAmd 1, Information technology - Security techniques - Digital signatures with appendix*, Part 3: Discrete logarithm based mechanisms, https://www.iso.org/standard/70631.html.

[8] Z. Cheng, *E-mail security system*, 2016, https://www.olymtech.net/products/application/mail-client.html.

[9] M. Zhang, X. Du, and K. Nygard, "Improving Coverage Performance in Sensor Networks by Using Mobile Sensors," in *Proceedings of the IEEE MILCOM*, Atlantic City, NJ, USA, 2005.

[10] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology*, pp. 388–397, Springer, 1999.

[11] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of data encryption standard Test Conference," in *Proceedings of the ITC International IEEE*, pp. 339–344, USA, October 2004.

[12] E. Oswald, *A side-channel analysis resistant description of the AES S-box International Workshop on Fast Software Encryption*, Springe, Berlin, Heidelberg, Germany, 2005.

[13] Y. Yarom and K. Falkner, *FLUSH+ RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack USENIX Security Symposium*, 2014.

[14] T. Izu and T. Takagi, *A fast parallel elliptic curve multiplication resistant against side channel attacks Public Key Cryptography*, 2002.

[15] S. Mangard, *A simple power-analysis (SPA) attack on implementations of the AES key expansion International Conference on Information Security and Cryptology*, vol. 2587 of *Springer*, Berlin, Heidelberg, Germany, 2003.

[16] S. Chari, J. R. Rao, and P. Rohatgi, *Template attacks International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, Berlin, Heidelberg, Germany, 2003.

[17] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in CryptologyCRYPTO'97*, pp. 513–525, 1997.

[18] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519–521, 1985.

[19] Ç. K. Koç, T. Acar, and B. S. Kaliski, "Analyzing and comparing montgomery multiplication algorithms," *IEEE Micro*, vol. 16, no. 3, pp. 26–33, 1996.

[20] M. McLoone, C. McIvor, and J. V. McCanny, "Coarsely integrated operand scanning (CIOS) architecture for high-speed Montgomery modular multiplication Field-Programmable Technology," in *Proceedings of the IEEE International Conference*, pp. 185–191, December 2004.

[21] E. Brier, C. Clavier, and F. Olivier, *Correlation Power Analysis with a Leakage Model International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, Berlin, Heidelberg, Germany, 2004.

[22] L. Goubin and J. Patarin, *DES and Differential Power Analysis the Duplication Method Cryptographic Hardware and Embedded Systems*, Springer, Berlin, Heidelberg, Germany, 1999.

[23] P.-Y. Liardet and N. P. Smart, *Preventing SPA/DPA in ECC systems using the Jacobi form International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, Berlin, Heidelberg, Germany, 2001.

[24] J. Cheol Ha and S. Jae Moon, *Randomized Signed-Scalar Multiplication of ECC to Resist Power Attacks CHES*, 2003.

[25] J.-W. Lee, S.-C. Chung, H.-C. Chang, and C.-Y. Lee, *An Efficient Countermeasure against Correlation Power-Analysis Attacks with Randomized Montgomery Operations for DF-ECC Processor CHES*, 2012.

[26] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[27] J. Fan and I. Verbauwhede, "An Updated Survey on Secure ECC Implementations: Attacks, Countermeasures and Cost," in *Cryptography and security*, vol. 6805, pp. 265–282, 2012.

[28] M. Brown, D. Hankerson, J. Lpez, and A. Menezes, "Software implementation of the NIST elliptic curves over prime fields," *Topics in CryptologylCT-RSA*, vol. 2001, pp. 250–265, 2001.

Hindawi
Security and Communication Networks
Volume 2018, Article ID 7206835, 8 pages
https://doi.org/10.1155/2018/7206835

Research Article

Analysis of Software Implemented Low Entropy Masking Schemes

Dan Li,[1,2] Jiazhe Chen,[2] An Wang,[3] and Xiaoyun Wang[1,4]

[1] *Institute for Advanced Study, Tsinghua University, Beijing 100084, China*
[2] *China Information Technology Security Evaluation Center, Beijing 100085, China*
[3] *School of Computer Science, Beijing Institute of Technology, Beijing 100081, China*
[4] *Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China*

Correspondence should be addressed to Jiazhe Chen; jiazhechen@gmail.com
and Xiaoyun Wang; xiaoyunwang@mail.tsinghua.edu.cn

Received 31 October 2017; Accepted 16 January 2018; Published 26 March 2018

Academic Editor: Emanuele Maiorana

Low Entropy Masking Schemes (LEMS) are countermeasure techniques to mitigate the high performance overhead of masked hardware and software implementations of symmetric block ciphers by reducing the entropy of the mask sets. The security of LEMS depends on the choice of the mask sets. Previous research mainly focused on searching balanced mask sets for hardware implementations. In this paper, we find that those balanced mask sets may have vulnerabilities in terms of absolute difference when applied in software implemented LEMS. The experiments verify that such vulnerabilities certainly make the software LEMS implementations insecure. To fix the vulnerabilities, we present a selection criterion to choose the mask sets. When some feasible mask sets are already picked out by certain searching algorithms, our selection criterion could be a reference factor to help decide on a more secure one for software LEMS.

1. Introduction

First introduced by Kocher [1], side channel attacks (SCA) can be used to evaluate the implementation security of cryptographic ciphers by analyzing the time, the electromagnetic radiation, the power consumption, and so on [2–6].

To resist SCA, several valid countermeasures have been proposed [7–10]. Among those countermeasures, masking schemes are most popular and widely applied. The main idea of masking schemes is to make the side channel information independent of the sensitive data by randomizing the intermediate values. In general first-order masking scheme, any sensitive intermediate variable denoted by Z will be split into two shares so that $Z = S_0 \oplus S_1$, where the randomly drawn variable S_0 is called the mask. All the computations of the cryptographic algorithm are performed on the shared values independently. At the same time, the sensitive data must be recovered by recombining the two shares. For this purpose, every computation function f of cryptographic algorithms should be designed to satisfy $f(Z) = S_0' \oplus S_1'$, where S_0' and S_1' are the new shares after the operation f. If f is a linear operation with respect to XOR, then $S_0' = f(S_0)$ and $S_1' = f(S_1)$. When f is the substitution box (S-Box), some adjustment is necessary to make up for its nonlinear property. The adjusted S-Box function changes along with the value of the mask, which makes it hard to compute canceling the sensitive intermediate value analytically. Therefore, precomputing and caching the required masked S-Boxes are more relevant and efficient. However, if the mask is drawn randomly from 2^n possible masks, too much memory is required to keep all the possible masked S-Boxes. To offer a reasonable solution to balance the security protection and the performance of implementations, Low Entropy Masking Schemes (LEMS) [10, 11] are designed by limiting the amount of mask entropy.

LEMS use the masks drawn from the limited mask set $\mathcal{M} = \{m_1, m_2, \ldots, m_s\} \subset \mathbb{F}_2^n$ whose mask entropy is $\log_2(s)$. The security of LEMS implementations should be guaranteed

in two aspects. In the architecture aspect, cryptographic algorithms should carefully be implemented to avoid first-order leakage [12]. Some countermeasure techniques such as shuffling [13] can also be combined to help defeat certain bivariate and higher order attacks [14–17]. Another aspect is the chosen mask set which plays significant roles in security. Some research studied how to select them for hardware implemented LEMS [11, 18]. The selection criterion of the mask sets considered finding secure mask sets under two important assumptions [19]. The first one is that the attackers could only exploit the leakage of the masked value $Z \oplus M$. The second one is that the deterministic part of the leakage function $l_{Z\oplus M}$ is linear in the bits of masked variable $Z \oplus M$, such as Hamming weight function. Under those two conditions, the main goal of selecting mask sets for LEMS is to find balanced mask sets resistant to high order univariate CPA (following the definition of [20], the attack combining n different time instances is called n-variate attack and the n_{th} order attack is the one with n_{th} order statistical moments). Therefore, making $\mathsf{E}((l_{Z\oplus M})^{\alpha} \mid Z)$ independent of intermediate Z is the selection criterion of the mask sets for the designer of the hardware countermeasures. However, we find it is not enough for software implemented LEMS. The absolute difference $|l_{z\oplus m} - l_{z'\oplus m'}|$ may bring the unbalance to the intermediate pair (z, z'), which allows attackers to get the information of (z, z') when only the leakages corresponding to the masked values are available.

Our Contributions. In this paper, we study the unbalance in terms of absolute difference on software Low Entropy Masking Schemes (LEMS) implementations and make selection criterion for their mask sets.

(i) We find that the mask sets selected according to selection criteria in [11, 18] have the vulnerabilities based on the absolute difference measurements on software LEMS. Such vulnerabilities make the software LEMS implementations insecure when the leakages corresponding to the masked values could be exploited.

(ii) To fix the vulnerabilities and make software LEMS implementations resistant to high order univariate attacks, we further extend the selection criterion of balanced mask sets. Moreover, we prove the perfect balanced mask sets should not be linear, and their cardinalities should satisfy certain conditions.

(iii) When some feasible mask sets are already picked out by searching algorithms like those in [11], our selection criterion could be a reference factor to help decide on a more secure one from them.

Organization. The rest of the paper is organized as follows. In Section 2, we introduce the notations and some related background knowledge. Section 3 presents vulnerabilities that make the software LEMS insecure. Section 4 proves the necessary conditions that the balanced mask sets should satisfy and discusses the selection methods of mask sets. Finally, Section 5 concludes the paper.

2. Preliminaries

In this paper, sets are denoted with calligraphic letters (e.g., $\mathcal{M}$). We use capital letters (e.g., M) and lowercase ones (e.g., m) for random variables and their realizations, respectively. Throughout the paper, Z and Z' are independent and uniformly distributed random variables representing intermediates. M and M' are two independent random variables drawn from the uniform distribution in the mask set $\mathcal{M}$.

Let l_ω be the value of leakage measurements corresponding to the intermediate value ω, $\omega \in \mathbb{F}_2^n$. To match with realistic leakage functions in practice, the widely applied Hamming weight leakage model is used during the choice of the mask sets in this paper. Thus, in software environments, $l_\omega = \epsilon \mathsf{HW}[\omega] + \delta$, where ϵ is an unknown constant and δ is the Gaussian distributed ($\mathcal{N}(0, \sigma^2)$) noise. In hardware environments, $l_\omega = \epsilon \mathsf{HW}[\omega]$ (to describe the theories in [11, 18] more clearly, we use the same no noise model here). We further denote the absolute difference of two measurements corresponding to the values ω_1 and ω_2 by $|l_{\omega_1} - l_{\omega_2}|$.

Mean and variance are denoted by E and Var, respectively. Let X_1 and X_2 be two independent random variables and f be a certain function. X_1 is randomly drawn from $\mathcal{X}$. $\mathsf{E}(f(X_1, X_2) \mid X_1 = x_1)$ is the conditional expectation when $X_1 = x_1$. The variance among those conditional expectations is

$$\mathsf{Var}\left(\mathsf{E}\left(f\left(X_1, X_2\right) \mid X_1\right)\right) = \frac{1}{|\mathcal{X}|} \cdot \sum_{x_1 \in \mathcal{X}} \left(\mathsf{E}\left(f\left(X_1, X_2\right) \mid X_1 = x_1\right) - \mathsf{E}\left(f\left(X_1, X_2\right)\right)\right)^2 \quad (1)$$

which can measure the dispersion degree of $\mathsf{E}(f(X_1, X_2) \mid X_1)$. Obviously, when $\mathsf{Var}(\mathsf{E}(f(X_1, X_2) \mid X_1)) = 0$, the specific value of X_1 cannot be recognized according to $\mathsf{E}(f(X_1, X_2) \mid X_1)$. This property was mainly applied by some works [11, 18] studying the selection criterion of mask sets for hardware LEMS. Their theories are as follows.

To defeat high order univariate CPA, the value of intermediate Z should be independent of the statistic values of $l_{Z\oplus M} = \epsilon \mathsf{HW}[Z \oplus M]$. Usually, those statistics indicate αth moments denoted by $\mathsf{E}((\epsilon \mathsf{HW}[Z \oplus M])^{\alpha})$. Hence, $\mathsf{Var}(\mathsf{E}((\epsilon \mathsf{HW}[Z \oplus M])^{\alpha} \mid Z)) = 0$ is the selection criterion. The mask set is said to resist univariate dth-order attacks if $\forall\ 1 \le \alpha \le d, \alpha \in \mathbb{N}$, $\mathsf{Var}(\mathsf{E}((\epsilon \mathsf{HW}[Z \oplus M])^{\alpha} \mid Z)) = 0$.

The work in [11] proved that only 12 mask values are sufficient for $d = 2$ when $n = 8$, ($\mathcal{M}_{12} = \{03, 18, 3F, 55, 60, 6E, 8C, A5, B2, CB, D6, F9\}$). The work in [18] further studied the linear code mask sets for different d and n. For example, in [8, 4, 4] linear code mask set can reach the standard of $d = 3$ with 16 mask values when $n = 8$ (like $\mathcal{M}_{16} = \{00, 0F, 36, 39, 53, 5C, 65, 6A, 95, 9A, A3, AC, C6, C9, F0, FF\}$ used in DPA Contest v4). The linear mask set $\mathcal{M}$ has the property that $m_i \oplus m_j \in \mathcal{M}, m_i, m_j \in \mathcal{M}$ [21]. We will discuss and use the property in the following sections.

3. Vulnerabilities on Software LEMS

As stated in Section 2, the selection of the mask sets for hardware LEMS considers the balance between the intermediate

values Z and the leakage measurements $l_{Z\oplus M}$ to avoid leaking the information of Z. Nonetheless, the unbalance of absolute difference measurements $|l_{Z\oplus M} - l_{Z'\oplus M'}|$ may leak the information of intermediate pair (Z, Z') in software LEMS. In this section, we will study (α represents the order with respect to the absolute difference; indeed, the absolute difference itself is not first order according to Taylor expansion [22]; hence, the order with respect to the original leakage measurement here is higher than α) $\mathsf{E}^{(\alpha)}_{(Z,Z')} = \mathsf{E}(|l_{Z\oplus M} - l_{Z'\oplus M'}|^{\alpha} \mid Z, Z')$, $\alpha = 1, 2$. The proofs will show that $\mathsf{E}^{(2)}_{(Z,Z')}$ is independent of (Z, Z') if the mask set satisfies the hardware selection criterion: $\mathsf{Var}(\mathsf{E}(\mathsf{HW}[Z \oplus M]^{\alpha} \mid Z)) = 0$, $\alpha = 1, 2$. And it is uncertain for $\mathsf{E}^{(1)}_{(Z,Z')}$. The unbalanced $\mathsf{E}^{(1)}_{(Z,Z')}$ leads to the unbalanced variance and coefficient of variation (coefficient of variation is the ratio of standard deviation to mean), which can also help identify the intermediate pair (Z, Z') in attacks. The results of experiments show that the unbalance of $\mathsf{E}^{(1)}_{(Z,Z')}$ makes the implementations insecure. Those vulnerabilities are the properties of mask sets and cannot be fixed by the architectures of specific implementations like shuffling. So finding the balanced mask sets in terms of absolute difference is necessary for software LEMS, which will be discussed in the next section.

As $l_{z\oplus m} - l_{z'\oplus m'} \sim \mathcal{N}(\epsilon\mathsf{HW}[z \oplus m] - \epsilon\mathsf{HW}[z' \oplus m'], 2\sigma^2)$, $\mathsf{E}((l_{z\oplus m} - l_{z'\oplus m'})^2) = (\epsilon\mathsf{HW}[z \oplus m] - \epsilon\mathsf{HW}[z' \oplus m'])^2 + 2\sigma^2$ and $\mathsf{E}(|l_{z\oplus m} - l_{z'\oplus m'}|) = f(\epsilon(\mathsf{HW}[z \oplus m] - \mathsf{HW}[z' \oplus m']), \sqrt{2}\sigma)$ according to Appendix A. We deduce that

$$\mathsf{E}^{(1)}_{(z,z')} = \mathsf{E}\left(\left|l_{Z\oplus M} - l_{Z'\oplus M'}\right| \mid Z = z,\ Z' = z'\right) \tag{2}$$

$$= \frac{1}{|\mathscr{M}|^2} \sum_{m,m' \in \mathscr{M}} \mathsf{E}\left(\left|l_{z\oplus m} - l_{z'\oplus m'}\right|\right) \tag{3}$$

$$= \frac{1}{|\mathscr{M}|^2} \sum_{m,m' \in \mathscr{M}} f\left(\epsilon\left(\mathsf{HW}\left[z \oplus m\right] - \mathsf{HW}\left[z' \oplus m'\right]\right), \sqrt{2}\sigma\right), \tag{4}$$

$$\mathsf{E}^{(2)}_{(z,z')} = \mathsf{E}\left(\left(l_{Z\oplus M} - l_{Z'\oplus M'}\right)^2 \mid Z = z,\ Z' = z'\right) \tag{5}$$

$$= \frac{1}{|\mathscr{M}|^2} \sum_{m,m' \in \mathscr{M}} \mathsf{E}\left(\left(l_{z\oplus m} - l_{z'\oplus m'}\right)^2\right) \tag{6}$$

$$= \frac{1}{|\mathscr{M}|^2} \sum_{m,m' \in \mathscr{M}} \left(\left(\epsilon\mathsf{HW}\left[z \oplus m\right] - \epsilon\mathsf{HW}\left[z' \oplus m'\right]\right)^2 + 2\sigma^2\right) \tag{7}$$

$$= \mathsf{E}\left((\epsilon\mathsf{HW}\left[Z \oplus M\right])^2 \mid Z = z\right) + \mathsf{E}\left((\epsilon\mathsf{HW}\left[Z \oplus M\right])^2 \mid Z = z'\right) - 2\mathsf{E}\left(\epsilon\mathsf{HW}\left[Z \oplus M\right] \mid Z = z\right) \mathsf{E}\left(\epsilon\mathsf{HW}\left[Z \oplus M\right] \mid Z = z'\right) + 2\sigma^2. \tag{8}$$

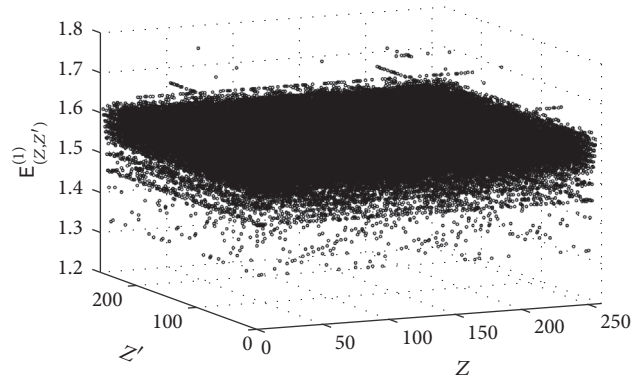


Figure 1: $\mathsf{E}^{(1)}_{(Z,Z')}$ over $\mathscr{M}_{12}$.

Obviously, for the mask set $\mathscr{M}$ which satisfies the hardware selection criterion ($\mathsf{Var}(\mathsf{E}((\epsilon\mathsf{HW}[Z \oplus M])^{\alpha} \mid Z)) = 0$, $\alpha = 1, 2$), $\mathsf{E}^{(2)}_{(Z,Z')}$ is independent of (Z, Z').

$\mathsf{E}^{(1)}_{(Z,Z')}$ is associated with the noise. For certain value σ, the value of $\mathsf{E}^{(1)}_{(Z,Z')}$ converges from $2\sigma/\sqrt{\pi}$ to $(\epsilon/|\mathscr{M}|^2)\sum_{m,m' \in \mathscr{M}} |\mathsf{HW}[z \oplus m] - \mathsf{HW}[z' \oplus m']|$ along with $\epsilon/\sigma = 0 \to \infty$. Hence, we can evaluate the unbalance of $\mathsf{E}^{(1)}_{(Z,Z')}$ for a certain mask set with $(1/|\mathscr{M}|^2)\sum_{m,m' \in \mathscr{M}} |\mathsf{HW}[z \oplus m] - \mathsf{HW}[z' \oplus m']|$. We take the mask set $\mathscr{M}_{12} \subset \mathbb{F}_2^8$ mentioned in Section 2 as an example and draw values of $\mathsf{E}^{(1)}_{(Z,Z')}$ for 2^{2*8} intermediate pairs (Z, Z') in Figure 1 which shows that $\mathscr{M}_{12}$ has vulnerabilities in terms of the absolute difference. Univariate attacks using these vulnerabilities can be performed on one S-Box.

The results of experiments in Appendix B verify that such vulnerabilities we highlighted can really threaten the security of software LEMS implementations. To make software LEMS implementations resistant to high order univariate attacks (CPA and also attacks based on the vulnerabilities above), specific implementations like shuffling are not enough and selecting the balanced mask sets in terms of the absolute difference is necessary.

4. Selection of Balanced Mask Sets

In this section, we will modify the selection criterion to find the balanced mask sets. The proofs give two conditions that the balanced mask sets should satisfy, which considerably narrow down the search for the mask sets.

The selection of the mask sets should first satisfy the criteria for hardware selections: $\mathsf{Var}(\mathsf{E}(\mathsf{HW}[Z\oplus M]^{\alpha}) \mid Z) = 0$ at least for $\alpha = 1, 2$. In such a condition, $\mathsf{E}^{(2)}_{(Z,Z')}$ is balanced as analyzed in Section 3. Hence, if $\mathsf{Var}(\mathsf{E}^{(1)}_{(Z,Z')} \mid Z, Z') = 0$, $\mathsf{E}^{(1)}$, $\mathsf{Var}_{(Z,Z')}$ and $\mathsf{CV}_{(Z,Z')}$ will also be balanced. According to (4), $\mathsf{E}^{(1)}_{(Z,Z')}$ can further be denoted by $\sigma f_e(\epsilon/\sigma, z, z')$. We can deduce that

$$\mathsf{Var}\left(\mathsf{E}^{(1)}_{(Z,Z')} \mid Z, Z'\right) = \mathsf{E}\left(\left(\mathsf{E}^{(1)}_{(Z,Z')}\right)^2\right) - \left(\mathsf{E}\left(\mathsf{E}^{(1)}_{(Z,Z')}\right)\right)^2$$

$$= \frac{\sigma^2}{2^{2n}} \sum_{z,z' \in \mathbb{F}_2^n} \left(f_e\left(\frac{\epsilon}{\sigma}, z, z'\right)\right)^2 - \left(\frac{\sigma}{2^{2n}} \sum_{z,z' \in \mathbb{F}_2^n} f_e\left(\frac{\epsilon}{\sigma}, z, z'\right)\right)^2. \tag{9}$$

$\mathsf{Var}(\mathsf{E}^{(1)}_{(Z,Z')} \mid Z, Z')$ will converge from 0 to $\gamma = \mathsf{Var}(\mathsf{E}(|\epsilon \mathsf{HW}[Z \oplus M] - \epsilon \mathsf{HW}[Z' \oplus M']| \mid Z, Z'))$ when $\epsilon/\sigma = 0 \to \infty$ for any fixed value σ.

The value of γ is an intrinsic property of the mask set. Thus, $\gamma = 0$ is the selection criterion. In this case, $\mathsf{E}^{(1)}_{(Z,Z')}$ will be balanced for any ϵ/σ. Aiming at the selection criterion, we can deduce the following conclusions to help select mask sets.

$\gamma = 0$ indicates $\mathsf{E}(|\mathsf{HW}[Z \oplus M] - \mathsf{HW}[Z' \oplus M']| \mid Z, Z')$ is a constant, the value of which is $\mathsf{E}(|\mathsf{HW}[Z \oplus M] - \mathsf{HW}[Z' \oplus M']|)$. We have the following.

Lemma 1. $\mathsf{E}(|\mathsf{HW}[Z \oplus M] - \mathsf{HW}[Z' \oplus M']|) = (2n-1)!!/2^n(n-1)!$.

Proof. Let $\mathsf{W}_a = \mathsf{E}(|\mathsf{HW}[Z] - \mathsf{HW}[Z']| \mid \mathsf{HW}[Z \oplus Z'] = a)$. $\mathsf{W}_0 = 0$, obviously. For $k \in \mathbb{N}$, we can deduce that

$$\begin{aligned} \mathsf{W}_{2k+1} &= \frac{1}{2^{2k+1}} \sum_{i=0}^{2k+1} \binom{2k+1}{i} |2k+1-2i| \\ &= \frac{1}{2^{2k+1}} \sum_{i=0}^{k} 2 \binom{2k+1}{i} (2k+1-2i) \\ &= \frac{1}{2^{2k}} \left((2k+1)\, 2^{2k} - 2(2k+1) \sum_{i=0}^{k-1} \binom{2k}{i} \right) \\ &= \frac{(2k+1)\binom{2k}{k}}{2^{2k}} = \frac{(2k+1)(2k)!}{2^{2k}(k!)^2} \\ &= \frac{(2k+1)!!}{(2k)!!}. \end{aligned} \tag{10}$$

The second equality uses $\binom{2k+1}{i} = \binom{2k+1}{2k+1-i}$.

The third one is according to $\sum_{i=0}^{k} \binom{2k+1}{i} = 2^{2k}$ and $i\binom{2k+1}{i} = (2k+1)\binom{2k}{i-1}$.

Similarly, $\mathsf{W}_{2k+2} = (2k+1)!!/(2k)!! = \mathsf{W}_{2k+1} = ((2k+1)/2k)\mathsf{W}_{2k}$. Hence

$$\begin{aligned} &\mathsf{E}\left(\left|\mathsf{HW}[Z \oplus M] - \mathsf{HW}\left[Z' \oplus M'\right]\right|\right) \\ &= \frac{1}{|\mathscr{M}|^2 4^n} \sum_{z,z' \in \mathbb{F}_2^n} \sum_{m,m' \in \mathscr{M}} \left|\mathsf{HW}[z \oplus m] - \mathsf{HW}\left[z' \oplus m'\right]\right| \\ &= \frac{1}{|\mathscr{M}|^2 4^n} \sum_{m,m' \in \mathscr{M}} \sum_{z,z' \in \mathbb{F}_2^n} \left|\mathsf{HW}[z] - \mathsf{HW}\left[z'\right]\right| \\ &= \frac{1}{4^n} \sum_{z,z' \in \mathbb{F}_2^n} \left|\mathsf{HW}[z] - \mathsf{HW}\left[z'\right]\right| \\ &= \frac{1}{2^n} \sum_{i=0}^{n} \binom{n}{i} \mathsf{W}_i. \end{aligned} \tag{11}$$

We will use mathematical induction to prove $\mathsf{A}_n = \sum_{i=0}^{n} \binom{n}{i} \mathsf{W}_i = (2n-1)!!/(n-1)!$.

When $n = 1$, $\mathsf{A}_1 = \sum_{i=0}^{1} \binom{1}{i} \mathsf{W}_i = 1 = (2-1)!!/(0)!$.

Suppose $\mathsf{A}_n = (2n-1)!!/(n-1)!$. If n is odd, we have

$$\begin{aligned} \mathsf{A}_{n+1} &= \sum_{i=0}^{n+1} \binom{n+1}{i} \mathsf{W}_i \\ &= \sum_{i=1}^{n} \left(\binom{n}{i} + \binom{n}{i-1} \right) \mathsf{W}_i + \mathsf{W}_{n+1} \\ &= \mathsf{A}_n + \sum_{i=0}^{n} \binom{n}{i} \mathsf{W}_{i+1} \\ &= \mathsf{A}_n + \sum_{i=0}^{(n-1)/2} \left(\binom{n}{2i} + \binom{n}{2i+1} \right) \mathsf{W}_{2i+1} \\ &= \mathsf{A}_n + \sum_{i=0}^{(n-1)/2} \left(\binom{n}{2i} \frac{n+1}{n} \mathsf{W}_{2i} \right. \\ &\quad \left. + \left(\binom{n}{2i} \left(1 - \frac{(n+1)(2i)}{n(2i+1)}\right) + \binom{n}{2i+1} \right) \mathsf{W}_{2i+1} \right) \\ &= \mathsf{A}_n + \frac{n+1}{n} \sum_{i=0}^{(n-1)/2} \left(\binom{n}{2i} \mathsf{W}_{2i} + \binom{n}{2i+1} \mathsf{W}_{2i+1} \right) \\ &= \frac{2n+1}{n} \mathsf{A}_n = \frac{(2n+1)!!}{(n)!}. \end{aligned} \tag{12}$$

The second equality is based on $\binom{n+1}{i} = \binom{n}{i} + \binom{n}{i-1}$. The fourth one follows $\mathsf{W}_{2i+1} = \mathsf{W}_{2i+2}$, and the fifth one uses $\mathsf{W}_{2i} = (2i/(2i+1))\mathsf{W}_{2i+1}$.

The situation when n is even can be proved similarly.

Thus, $\mathsf{E}(|\mathsf{HW}[Z \oplus M] - \mathsf{HW}[Z' \oplus M']|) = \mathsf{A}_n/2^n = (2n-1)!!/2^n(n-1)!$. □

As stated above, $\mathsf{E}(|\mathsf{HW}[Z \oplus M] - \mathsf{HW}[Z' \oplus M']| \mid Z, Z')$ for any pair (Z, Z') and the means of their combinations such as $\mathsf{E}_{\mathscr{M}} = \mathsf{E}(\mathsf{E}(|\mathsf{HW}[Z \oplus M] - \mathsf{HW}[Z \oplus M']| \mid Z))$ should be equal to the constant value $(2n-1)!!/2^n(n-1)!$. We can prove two necessary conditions for balanced mask set $\mathscr{M}$ by analyzing $\mathsf{E}_{\mathscr{M}} = (2n-1)!!/2^n(n-1)!$.

Theorem 2. *One necessary condition for $\gamma = 0$ is $|\mathscr{M}| = k2^{\lfloor n/2 \rfloor + 1}$, $k \in \mathbb{N}$.*

Proof. We deduce that

$$\mathsf{E}_{\mathscr{M}} = \mathsf{E}\left(\mathsf{E}\left(\left|\mathsf{HW}\left[Z \oplus M\right] - \mathsf{HW}\left[Z \oplus M'\right]\right| \mid Z\right)\right) \tag{13}$$

$$= \frac{1}{|\mathscr{M}|^2\, 2^n} \sum_{z \in \mathbb{F}_2^n} \sum_{m,m' \in \mathscr{M}} \left|\mathsf{HW}\left[z \oplus m\right] - \mathsf{HW}\left[z \oplus m'\right]\right| \tag{14}$$

$$= \frac{1}{|\mathscr{M}|^2} \sum_{m,m' \in \mathscr{M}} \frac{1}{2^n} \sum_{z \in \mathbb{F}_2^n} \left|\mathsf{HW}\left[z \oplus m\right] - \mathsf{HW}\left[z \oplus m'\right]\right| \tag{15}$$

$$= \frac{1}{|\mathscr{M}|^2} \sum_{m,m' \in \mathscr{M}} \mathsf{E}\left(\left|\mathsf{HW}\left[Z\right] - \mathsf{HW}\left[Z'\right]\right| \mid Z \oplus Z' = m \oplus m'\right). \tag{16}$$

Let $\mathsf{C}_i = \sum_{m,m' \in \mathscr{M}} \varrho_i(m \oplus m')$, where

$$\varrho_i(x) = \begin{cases} 1, & \mathsf{HW}\left[x\right] = i, \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

As $m \oplus m' = m' \oplus m$, C_i, $i > 0$, is even. Let $\mathsf{C}_i = 2\mathsf{C}'_i$, $i > 0$. As $\mathsf{W}_0 = 0$, $\mathsf{E}_{\mathscr{M}} = (1/|\mathscr{M}|^2) \sum_{i=0}^{n} \mathsf{C}_i \mathsf{W}_i = (2/|\mathscr{M}|^2) \sum_{i=1}^{n} \mathsf{C}'_i \mathsf{W}_i$. If $\mathsf{E}_{\mathscr{M}} = (2n-1)!!/2^n(n-1)!$, we can deduce

$$\frac{2}{|\mathscr{M}|^2} \sum_{i=1}^{n} \mathsf{C}'_i \mathsf{W}_i = \frac{(2n-1)!!}{2^n\,(n-1)!} \Longrightarrow$$

$$\frac{2^{n+1}}{|\mathscr{M}|^2} \sum_{i=1}^{n} \mathsf{C}'_i \left((n-1)!\mathsf{W}_i\right) = (2n-1)!! \Longrightarrow \tag{18}$$

$$2^{n+1} \mid |\mathscr{M}|^2.$$

The reason of the second arrow is as follows: Recall $\mathsf{W}_{2k+2} = \mathsf{W}_{2k+1} = (2k+1)!!/(2k)!!$ in Lemma 1. $\forall n \geq 2k+1$, $(n-1)!\mathsf{W}_{2k+2} = (n-1)!\mathsf{W}_{2k+1} = ((2k-1)!!)^2(2k+1)\prod_{i=2k+1}^{n-1} i \in \mathbb{N}$. In other words, $\forall i \leq n$, $(n-1)!\mathsf{W}_i \in \mathbb{N}$. $2^{n+1} \sum_{i=1}^{n} \mathsf{C}'_i((n-1)!\mathsf{W}_i) \in \mathbb{N}$ and $(2n-1)!!$ is odd. Therefore, $|\mathscr{M}|^2$ must be divisible by 2^{n+1}.

Hence, $|\mathscr{M}| = k2^{\lfloor n/2 \rfloor + 1}$, $k \in \mathbb{N}$. □

Theorem 3. *$\forall |\mathscr{M}| < 2^n \in \mathbb{N}$, if $\mathscr{M}$ is a linear mask set, $\gamma \neq 0$.*

Proof. If $\mathscr{M}$ is linear, $m \oplus m' \in \mathscr{M}$, $m, m' \in \mathscr{M}$. Let $\mathscr{D}_m = \{m \oplus m' \mid m' \in \mathscr{M}\}$. Obviously, $\forall m \in \mathscr{M}$, $\mathscr{D}_m = \mathscr{M}$. And (16) will further be

$$\mathsf{E}_{\mathscr{M}} = \frac{1}{|\mathscr{M}|^2} \sum_{m,m' \in \mathscr{M}} \mathsf{E}\left(\left|\mathsf{HW}\left[Z\right] - \mathsf{HW}\left[Z'\right]\right| \mid Z \oplus Z' = m \oplus m'\right)$$

$$= \frac{1}{|\mathscr{M}|^2} \sum_{m \in \mathscr{M}} \sum_{m' \in \mathscr{M}} \mathsf{E}\left(\left|\mathsf{HW}\left[Z\right] - \mathsf{HW}\left[Z'\right]\right| \mid Z \oplus Z' = m'\right)$$

$$= \frac{1}{|\mathscr{M}|} \sum_{i=0}^{n} \mathsf{C}_i \mathsf{W}_i, \tag{19}$$

where $\mathsf{C}_i = \sum_{m \in \mathscr{M}} \varrho_i(m)$. $\varrho_i(\cdot)$ is defined by (17).

If $\mathsf{E}_{\mathscr{M}} = (2n-1)!!/2^n(n-1)!$, we can deduce

$$\frac{1}{|\mathscr{M}|} \sum_{i=1}^{n} \mathsf{C}_i \mathsf{W}_i = \frac{(2n-1)!!}{2^n\,(n-1)!} \Longrightarrow$$

$$\frac{2^n}{|\mathscr{M}|} \sum_{i=1}^{n} \mathsf{C}_i \left((n-1)!\mathsf{W}_i\right) = (2n-1)!! \Longrightarrow \tag{20}$$

$$2^n \mid |\mathscr{M}|$$

which contradicts $|\mathscr{M}| < 2^n$. Thus, $\mathsf{E}_{\mathscr{M}} \neq (2n-1)!!/2^n(n-1)!$, which indicates $\mathsf{Var}(\mathsf{E}(|\mathsf{HW}[Z \oplus M] - \mathsf{HW}[Z' \oplus M']| \mid Z, Z')) \neq 0$. □

Theorem 2 indicates that the search should be among mask sets satisfying $|\mathscr{M}| = k2^{\lfloor n/2 \rfloor + 1}$, $k \in \mathbb{N}$, to find the perfect balanced mask set with $\gamma = 0$. However, in consideration of the effect of the noise, $\gamma = 0$ could not be necessary. According to Theorem 3 and the results in Appendix B, the linear mask sets will be more vulnerable because of their linear property. Hence, one can first use the searching algorithms like those in [11] to get some nonlinear mask sets and use our selection criterion as a reference factor to select the one with smaller γ.

5. Conclusion

In this paper, we analyzed the vulnerabilities on the mask sets of software Low Entropy Masking Schemes implementations. We found that satisfying the conditions in [11, 18] was not enough for mask sets used in software LEMS implementations. The experiments verified that such vulnerabilities certainly made the software LEMS implementations insecure. To fix the vulnerabilities, we further gave a selection criterion. Moreover, two theorems were proved, and our selection criterion could be a reference factor when selecting the mask sets picked out by searching algorithms like those in [11].

For future work, there remain two research directions. The first direction is the proof of the existence of such perfect balanced mask sets. The second one is designing more feasible search algorithms and giving the masking values selection rules based on those conditions.

Appendix

A. The Proof of $f(\mu,\sigma)$

$f(\mu,\sigma) = \mathsf{E}(|X|)$, where random variable $X \sim N(\mu,\sigma^2)$. We can deduce that

$$\begin{aligned}\mathsf{E}(|X|) &= \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{\infty}|x|\,e^{-(x-\mu)^2/2\sigma^2}\,\mathrm{d}x \\ &= \frac{1}{\sqrt{2\pi}\sigma}\left(\int_0^{\infty} xe^{-(x-\mu)^2/2\sigma^2}\right. \\ &\quad \left.-\int_{-\infty}^{0} xe^{-(x-\mu)^2/2\sigma^2}\right)\mathrm{d}x.\end{aligned} \tag{A.1}$$

Here

$$\begin{aligned}\int_0^{\infty} xe^{-(x-\mu)^2/2\sigma^2}\,\mathrm{d}x &= \int_{-\mu}^{\infty}(y+\mu)\,e^{-y^2/2\sigma^2}\,\mathrm{d}y \\ &= \int_{-\mu}^{\infty} ye^{-y^2/2\sigma^2}\,\mathrm{d}y \\ &\quad + \int_{-\mu}^{\infty}\mu e^{-y^2/2\sigma^2}\,\mathrm{d}y\end{aligned} \tag{A.2}$$

$$\begin{aligned}&= \sigma^2 e^{-\mu^2/2\sigma^2} \\ &\quad + \sqrt{2\pi}\sigma\mu\left(1-\phi\left(\frac{-\mu}{\sigma}\right)\right),\end{aligned} \tag{A.3}$$

$$\begin{aligned}\int_{-\infty}^{0} xe^{-(x-\mu)^2/2\sigma^2}\,\mathrm{d}x &= -\sigma^2 e^{-\mu^2/2\sigma^2} \\ &\quad + \sqrt{2\pi}\sigma\mu\phi\left(\frac{-\mu}{\sigma}\right),\end{aligned} \tag{A.4}$$

where $\phi(x) = \int_{-\infty}^{x}(1/\sqrt{2\pi})e^{-y^2/2}\mathrm{d}y$ can be checked on the normal distribution table.

Therefore, using (A.3) and (A.4)

$$\begin{aligned}f(\mu,\sigma) &= \mathsf{E}(|X|) \\ &= \sqrt{\frac{2}{\pi}}\sigma e^{-\mu^2/2\sigma^2} + \mu\left(1-2\phi\left(\frac{-\mu}{\sigma}\right)\right).\end{aligned} \tag{A.5}$$

B. Results of Experiments

We take a typical [8, 4, 4] linear code mask set $\mathscr{M}_{16}$ mentioned in Section 2 and its variant $\mathscr{M}'_{16} = \{m \oplus 0x03 \mid m \in \mathscr{M}_{16}\}$, which are, respectively, used in the RSM (Rotating S-Box Masking (RSM) [10] is a realization of LEMS.) implementations of DPA Contest v4 and DPA Contest v4.2 [15], as examples to analyze the security in different SNR environment in practice. The software implementation of AES-256 in DPAcv4 is protected by basic RSM countermeasure, and the traces are collected from an ATMega-163 smart card. Our attacks are performed on the leakage of the outputs of S-Boxes in first-round AES. As the implementation of AES-128 in DPAcv4.2 is protected by enhanced RSM countermeasure using shuffling techniques, we carry out the attacks on the leakage of the ShiftRow in the first round where the noise is bigger.

Aiming at the vulnerabilities of unbalanced $\mathsf{E}^{(1)}_{(Z,Z')}$, lots of distinguishers can be designed. Here, we will present examples combined with the linear property of the mask set $\mathscr{M}$: $\forall m, m' \in \mathscr{M}, m \oplus m' \in \mathscr{M}$.

Such property results in the following: for any intermediate z, $\mathscr{M}^z = \{z \oplus m \mid m \in \mathscr{M}\}$ is the same as that of $z_i = z \oplus m_i$ [21]. The reason is, $\forall m \in \mathscr{M}, z_i \oplus m = z \oplus (m_i \oplus m) \in \mathscr{M}^z$, which means $\mathscr{M}^{z_i} \subset \mathscr{M}^z$. Moreover, $|\mathscr{M}^{z_i}| = |\mathscr{M}^z| = |\mathscr{M}|$. Hence, $\mathscr{M}^{z_i} = \mathscr{M}^z$. We further find the variants of the linear mask set $\{m \oplus \mathsf{C} \mid m \in \mathscr{M}\}$, where C is a constant also having the same properties. Gathering the intermediates with the same masked values together, $\mathbb{F}_2^n$ is divided into several sets $\mathscr{I}_i$, $i = 1, 2, \ldots, c$ ($z, z' \in \mathscr{I}$, if $\mathscr{M}^z = \mathscr{M}^{z'}$).

Let $\mathscr{O}$ be the set of all the measurements. $\mathscr{O}_i^k$ represents the set of measurements whose corresponding plaintext p satisfies $\psi(p,k) \in \mathscr{I}_i$, where $\psi(\cdot,\cdot)$ is the function of sensitive intermediate. The distinguisher could be

$$\mathsf{D}(k) = \frac{\mathsf{E}\left(\widehat{\theta}\left(\mathscr{O}_i^k, \mathscr{O}_i^k\right)\right)}{\widehat{\theta}(\mathscr{O},\mathscr{O})}, \tag{B.1}$$

where $\widehat{\theta}(\cdot,\cdot)$ is the estimated statistic value of absolute difference values between two measurements sets. When k is wrong, the classification $\mathscr{O}_i^k$ will be wrong and random, which makes the values of numerator and denominator approximate. When k is the correct key, the value of numerator will differ from that of denominator (Theorem 3 in Section 4 will prove this). $k^* = \arg\max_k\{\mathsf{D}(k)\}$ or $k^* = \arg\min_k\{\mathsf{D}(k)\}$.

$\widehat{\theta}$ can be $\mathsf{E}^{(1)}$, obviously. As $\mathsf{E}^{(2)}$ is independent of (Z, Z') and $\mathsf{Var}(X) = \mathsf{E}(X^2) - (\mathsf{E}(X))^2$, $\mathsf{CV}(X) = \sqrt{\mathsf{Var}(X)}/\mathsf{E}(X)$, we can also use Var and CV as $\widehat{\theta}$. We name those distinguishers for different statistics as r_v, r_{cv}, and $r_m^{(1)}$, respectively.

Using the traces in DPAcv4, we obtain 256 r_v, r_{cv}, and $r_m^{(1)}$ curves and show the time samples around the output of one S-Box in Figure 2(a). The correct key's r_v and r_{cv} curves have apparent peaks with 1000 traces. Furthermore, we generate $r_m^{(1)}$, r_v, and r_{cv} curves over the number of traces at the peak time sample and show the results in Figure 2(b). The black and 255 grey curves represent the cases of the correct key and wrong key hypotheses, respectively. The results show that all those distinguishers can recover the key with enough traces.

We then do the second experiment using traces in DPAcv4.2 at the ShiftRow in the first round where the weaker information is leaked. The three distinguishers succeed with

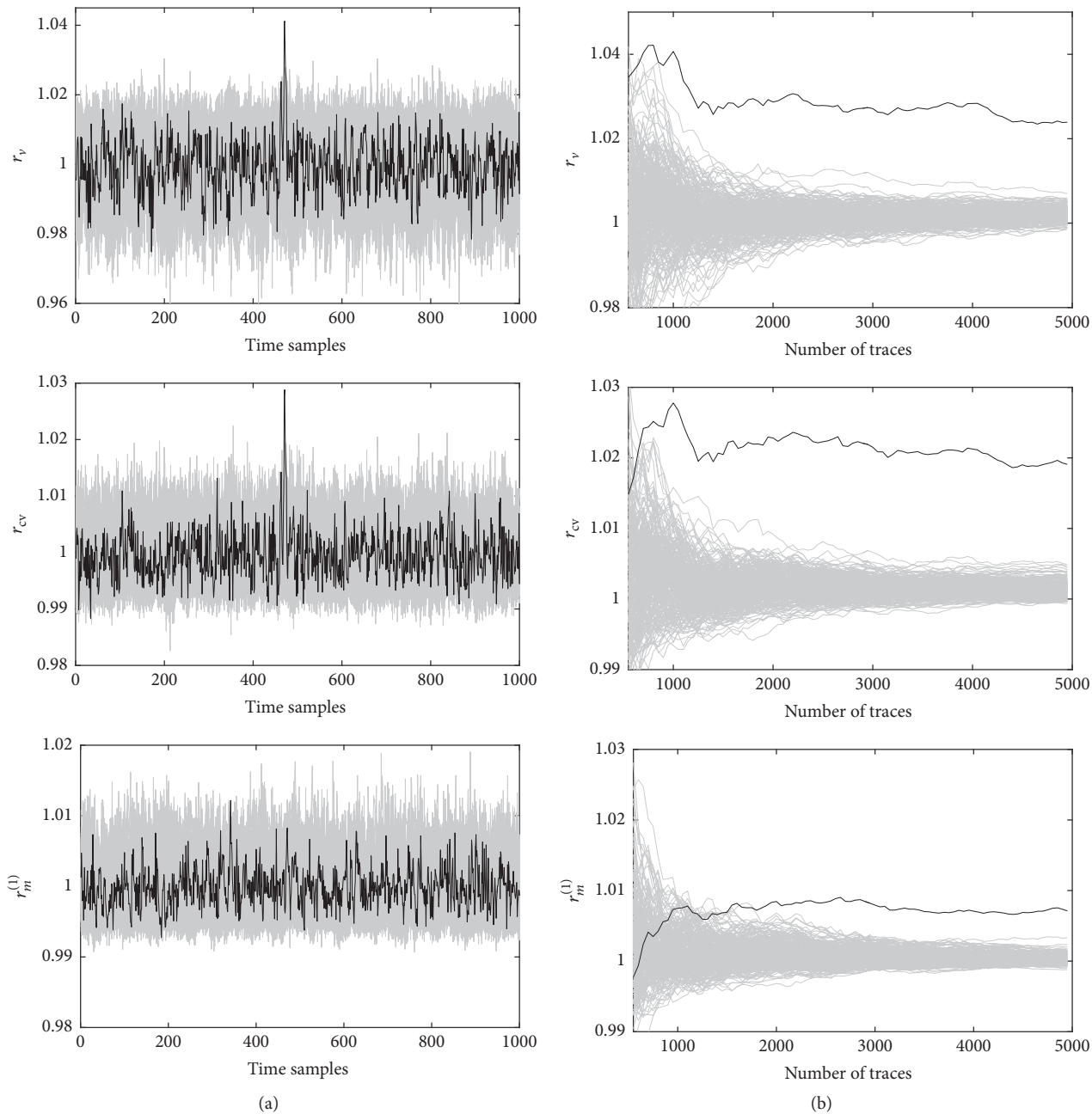


FIGURE 2: r_v, r_{cv}, and $r_m^{(1)}$ over (a) time samples using 1000 traces (b) and number of traces at the peak location.

about 6000 traces because of the lower SNR. We omit similar figures here.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by National Key Research and Development Program of China (Grant no. 2017YFA0303903), National Natural Science Foundation of China (Grant nos. 61402536 and 61402252), Beijing Natural Science Foundation (Grant no. 4162053), National Cryptography Development Fund (Grant no. MMJJ20170201), and 973 Program (Grant no. 2013CB834205).

References

[1] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proceedings of the 16th Annual International Cryptology Conference, CRYPTO '96*, Lecture Notes in Computer Science, pp. 104–113, Springer, August 1996.

[2] S. Bhasin, J.-L. Danger, S. Guilley, and Z. Najm, "Side-channel leakage and trace compression using normalized inter-class variance," in *Proceedings of the 3rd International Workshop on Hardware and Architectural Support for Security and Privacy, HASP 2014*, pp. 7:1–7:9, ACM, USA, June 2014.

[3] G. Dabosville, J. Doget, and E. Prouff, "A new second-order side channel attack based on linear regression," *IEEE Transactions on Computers*, vol. 62, no. 8, pp. 1629–1640, 2013.

[4] M. Kayaalp, N. Abu-Ghazaleh, D. Ponomarev, and A. Jaleel, "A high-resolution side-channel attack on last-level cache," in *Proceedings of the 53rd Annual ACM IEEE Design Automation Conference, DAC 2016*, USA, June 2016.

[5] A. A. Pammu, K.-S. Chong, W.-G. Ho, and B.-H. Gwee, "Interceptive side channel attack on AES-128 wireless communications for IoT applications," in *Proceedings of the 2016 IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2016*, pp. 650–653, Republic of Korea, October 2016.

[6] Y. Li, M. Chen, and J. Wang, "Introduction to side-channel attacks and fault attacks," in *Proceedings of the 7th Asia-Pacific International Symposium on Electromagnetic Compatibility, APEMC 2016*, pp. 573–575, May 2016.

[7] R. Lumbiarres-Lopez, M. Lopez-Garcia, and E. Canto-Navarro, "Hardware architecture implemented on FPGA for protecting cryptographic keys against side-channel attacks," *IEEE Transactions on Dependable and Secure Computing*, 2016.

[8] T. Backenstrass, M. Blot, S. Pontié, and R. Leveugle, "Protection of ECC computations against side-channel attacks for lightweight implementations," in *Proceedings of the 1st IEEE International Verification and Security Workshop, IVSW 2016*, pp. 1–6, July 2016.

[9] M.-L. Akkar and C. Giraud, "An implementation of DES and AES, secure against some attacks," in *Proceedings of the third International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2001*, vol. 2162 of *Lecture Notes in Computer Science*, pp. 309–318, Springer, May 2001.

[10] M. Nassar, Y. Souissi, S. Guilley, and J.-L. Danger, "RSM: a small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs," in *Proceedings of the 2012 Design, Automation & Test in Europe Conference & Exhibition, DATE 2012*, pp. 1173–1178, Dresden, Germany, March 2012.

[11] M. Nassar, S. Guilley, and J.-L. Danger, "Formal analysis of the entropy/security trade-off in first-order masking countermeasures against side-channel attacks," in *Proceedings of the 12th International Conference on Cryptology, INDOCRYPT 2011*, vol. 7107 of *Lecture Notes in Computer Science*, pp. 22–39, Springer, December 2011.

[12] A. Moradi, S. Guilley, and A. Heuser, "Detecting Hidden Leakages," in *Proceedings of the 12th International Conference on Applied Cryptography and Network Security, ACNS 2014*, vol. 8479 of *Lecture Notes in Computer Science*, pp. 324–342, Springer International Publishing, June 2014.

[13] C. Herbst, E. Oswald, and S. Mangard, "An AES smart card implementation resistant to power analysis attacks," in *Proceedings of the 4th International Conference on Applied Cryptography and Network Security, ACNS 2006*, vol. 3989 of *Lecture Notes in Computer Science*, pp. 239–252, Springer, June 2006.

[14] P. Belgarric, S. Bhasin, N. Bruneau et al., "Time-Frequency Analysis for Second-Order Attacks," in *Smart Card Research and Advanced Applications*, vol. 8419 of *Lecture Notes in Computer Science*, pp. 108–122, Springer International Publishing, Cham, 2014.

[15] S. Bhasin, N. Bruneau, J.-L. Danger, S. Guilley, and Z. Najm, "Analysis and improvements of the DPA contest v4 implementation," in *Proceedings of the 4th International Conference on Security, Privacy, and Applied Cryptography Engineering, SPACE 2014*, vol. 8804 of *Lecture Notes in Computer Science*, pp. 201–218, Springer, October 2014.

[16] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Improved collision-correlation power analysis on first order protected AES," in *Proceedings of the 13th International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2011*, vol. 6917, pp. 49–62, Springer, October 2011.

[17] X. Ye and T. Eisenbarth, "On the Vulnerability of Low Entropy Masking Schemes," in *Proceedings of the 12th International Conference on Smart Card Research and Advanced Applications, CARDIS 2013*, vol. 8419 of *Lecture Notes in Computer Science*, pp. 44–60, Springer International Publishing, November 2014.

[18] S. Bhasin, C. Carlet, and S. Guilley, "Theory of masking with codewords in hardware: low-weight dth-order correlation-immune boolean functions," *Cryptology ePrint Archive, IACR*, vol. 2013, p. 303, 2013.

[19] V. Grosso, F.-X. Standaert, and E. Prouff, "Low entropy masking schemes, revisited," in *Proceedings of the 12th International Conference on Smart Card Research and Advanced Applications, CARDIS 2013*, vol. 8419 of *Lecture Notes in Computer Science*, pp. 33–43, Springer, November 2014.

[20] A. Moradi and O. Mischke, "How far should theory be from practice? - evaluation of a countermeasure," in *Proceedings of the 14th International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2012*, vol. 7428 of *Lecture Notes in Computer Science*, pp. 92–106, Springer, September 2012.

[21] B. Ege, T. Eisenbarth, and L. Batina, "Near collision side channel attacks," in *Proceedings of the 22nd International Conference on Selected Areas in Cryptography, SAC 2015*, vol. 9566 of *Lecture Notes in Computer Science*, pp. 277–292, Springer, August 2015.

[22] http://functions.wolfram.com/ComplexComponents/Abs/06/ShowAll.html.

Hindawi
Security and Communication Networks
Volume 2018, Article ID 2567463, 10 pages
https://doi.org/10.1155/2018/2567463

Research Article

Understanding Keystroke Dynamics for Smartphone Users Authentication and Keystroke Dynamics on Smartphones Built-In Motion Sensors

Hyungu Lee,[1] **Jung Yeon Hwang**,[2] **Dong In Kim**,[1] **Shincheol Lee**,[1] **Sung-Hoon Lee**,[3] **and Ji Sun Shin**[1]

[1]*Department of Computer and Information Security, Sejong University, Seoul 05006, Republic of Korea*
[2]*Electronics and Telecommunications Research Institute, Daejeon 34113, Republic of Korea*
[3]*Information Security Engineering, University of Science and Technology, Daejeon 34113, Republic of Korea*

Correspondence should be addressed to Ji Sun Shin; jsshin.sejong@gmail.com

Received 3 November 2017; Revised 19 January 2018; Accepted 13 February 2018; Published 14 March 2018

Academic Editor: Amir Anees

Personal Identification Numbers (PINs) and pattern drawing have been used as common authentication methods especially on smartphones. Such methods, however, are very vulnerable to the shoulder surfing attack. Thus, keystroke dynamics that authenticate legitimate users based on their typing manner have been studied for years. However, many of the studies have focused on PC keyboard keystrokes. More studies on mobile and smartphones keystroke dynamics are warranted; as smartphones make progress in both hardware and software, features from smartphones have been diversified. In this paper, using various features including keystroke data such as time interval and motion data such as accelerometers and rotation values, we evaluate features with motion data and without motion data. We also compare 5 formulas for motion data, respectively. We also demonstrate that opposite gender match between a legitimate user and impostors has influence on authenticating by our experiment results.

1. Introduction

As we live in the smart era, the number of smartphone users grows every year [1, 2], whereas security measures to authenticate for an owner are standstill. Pattern drawing and PIN entering are most often used, and nowadays fingerprint scanning or other biometric data scanning is also often adopted as an authentication method [3, 4]. The latter are known to be safer than the former, since simple patterns and PINs can be leaked via shoulder surfing attacks. However, users often prefer using patterns and PINs rather than fingerprint scanning because fingerprint scanning sometimes fails and should be repeated. Therefore, to provide a moderate usability, devices providing biometric data-based authentication also provide backup authentication methods such as PIN or patterns.

Keystroke dynamics has appeared to complement such problems by checking not just the numbers or patterns but also how a user types (the time speed, touch size, and so on), the so-called keystroke dynamics. By combining PIN (or pattern) and keystroke dynamics as a multifactor authentication, keystroke dynamics strengthens user authentication. Clearly, biometric data can be multifactored with PIN and pattern as a multifactor authentication. However, the biggest problem with using biometric data is that the device should securely keep the biometric data in private: leakage of biometric data of a user invalidates lifetime use of the user's biometric data as a private key. On the other hand, keystroke dynamics change as PIN or secret pattern changes. Thus, using keystroke dynamics as an authentication factor is less risky upon compromise.

In this paper, we look into keystroke dynamics focusing on smartphones that have a touchscreen and on-board motion sensors. The keystroke dynamics authentication has been substantially researched particularly on personal computer keyboard keystroke dynamics. Keystroke dynamics on smartphones can have more diverse features since a smartphone has a touchscreen. Furthermore, since 2010, many smartphones are equipped with motion sensors such as

accelerometer and gyroscope. Since 2002, researches on mobile keystroke dynamics have been studied; however, still more researches are necessary to improve performances (error rates), characterize undiscovered important features, or find better classification methods. Therefore, in this paper, we focus on smartphone keystroke dynamics with 6-digit PIN with distance-based classification. We develop application, collect user keystroke data, experiment classifications, and show our results.

Our contributions are the following. We collected user data samples and experimented keystroke dynamics using the most simple classification algorithm, distance-based algorithm. We experimented with both of Euclidean distance and Manhattan distance and obtained better performance with Manhattan distance, 7.89% EER (equal error rate). Compared to the state of the art, ours give considerably good performance. There are three previous studies that performed better than ours. However, they are not with 6-digit PIN: one is with 4-digit or 8-digit PIN [5], one is with thumbnails (images) [6], and the other is with 300 characters [7]. Our experiment and results are most relevant to real-world applications of 6-digit PIN and keystroke dynamics multifactor authentications since to certain level of security and usability, 6-digit PIN is popularly used and no method other than the PIN itself (images or characters) is required. We also discovered an interesting aspect of keystroke dynamics. Our experiment result shows that keystroke dynamics are more effective in opposite gender imposters: FAR reduces when imposters are opposite gender compared to when imposters are the same gender. We also investigate which features have influence on reducing FAR and analyze feature characteristics. Our result can be useful to applications where gender authenticity is very important, for instance, online dating or online same gender competition exam/game. This result is not about gender classification, but one observation from keystroke dynamics based user authentication study. The contribution of this result is providing further understanding of keystroke dynamics characteristics.

The rest of the paper is organized as follows. In Section 2, we discuss related work and compare former studies. In Section 3, we discuss distance-based algorithms which are used to classify a legitimate user and how the classification works. In Section 4, we explain what features were extracted and the background of our experiment. We analyze our result in detail in Section 5. Finally, we conclude our study and propose future work in Section 6.

2. Related Work

Keystroke dynamics was proposed as a user authentication first in 1975 [19] and it was started from typing rhythms of users on the computer keyboard. Since then, two studies [20, 21] also showed the possibility with good results from experiment with few subjects in 1977 and 1980, respectively.

Keystroke dynamics experiment using the computer keyboard were conducted first by Umphress and Williams in 1985 [22]. In their study, they make a reference profile of a legitimate user using mean keystroke latency and mean time interval of consecutive characters so that it distinguishes the legitimate user from others. The result the study showed was 11.7% FAR (false acceptance rate) and 5.8% FRR (false rejection rate).

From 2002 to 2006, studies about keystroke dynamics on the mobile were studied first [8, 9, 23]. Latency between pressing and releasing a key and between pressing the first key and the last key (hold time) was used as features to authenticate [9].

In 2009, Saevanee and Bhattarakosol [24] suggested keystroke dynamics using finger pressure on the touchscreen first. It showed 99% accuracy with the Probabilistic Neural Network (PNN). As Android 1.6, called "Donut," was released on September 15, 2009, more various types of features such as a size of fingertip, orientation of a device, and angle of a device were available and, thus, more studies have made progress in reducing EER (equal error rate) using them [25].

Since December 6, 2010, Android 2.3 provided data from gyroscope, rotation vector, linear accelerometer, and gravity. Thus, more features can be extracted from them. Cai and Chen [26] first made use of orientation of a device as a feature of keystroke dynamics, and they guessed key numbers by angles of x-axis (azimuth), y-axis (pitch), and z-axis (roll) and showed 71.5% accuracy. In addition, studies using both orientation and accelerometers have been researched. Xu et al. [27] guessed the enter keys and showed 88.7% accuracy which is higher than the former study, and Wu and Chen [14] showed 0.556% EER using these two features and time, pressure, and size features.

Table 1 shows the mobile keystroke dynamics studies. Mobile keystroke dynamics compares results using PIN (Personal Identification Number) or characters in experiments. At the beginning in mobile keystroke dynamics study, EER for 4-digit PIN was 11.3% [8] and 8.5% [9]. Since then, studies have been conducted on various PINs such as 6-digit, 8-digit, 10-digit, and 16-digit PINs beside 4-digit PIN. Chang et al. [13] showed EER of 23%, 21%, and 16% for 6-digit, 8-digit, and 10-digit PINs. And also Teh et al. [18] showed EER of 7.57% for 4-digit PIN and 5.49% for 16-digit PIN. Experiments using characters as well as PIN have been conducted. Starting with EER 6.9% of the study using 6 characters [9], experiment using 6 to 8 characters showed EER of 21.02% [15], using 10 characters showed EER of 0.806% [16], and using 34 characters showed EER of 9.3% [17], respectively. Later, there was a new type of study that extracts pressure and time interval from image instead of keys [6].

Classifying users, there were different approaches using distance-based classifier beside the statistical and the neural network classifier. Among experiments on 4-digit PIN, the Euclidean distance classifier showed 20% of EER [10] and the nearest neighbor distance classifier showed 3.65% of EER [5]. In one study using characters instead of PIN, it utilized various distance-based classifiers and obtained 2.2% FAR and 4.6% FRR using both the weighted Euclidean distance classifier and the array disorder method [7]. However, their result is based on 300 characters. Study comparing various distance-based classifiers, respectively, evaluated that kNN Manhattan weighted distance and kNN Manhattan scaled weighted distance were the best as 8% EER [5]. Other study comparing three distance-based classifiers, Manhattan,

Table 1: Comparison of studies for keystroke dynamics. Motion data column indicates whether features from motion data are used or not.

Authors	Year	Methodology	Motion data	Number of subjects	Number of training samples	Classifier	EER (%)
Clarke et al. [8]	2003	4-digit PIN	X	30	30	Statistical	11.3
Clarke and Furnell [9]	2007	4-digit PIN	X	30	30	Neural network	8.5
		6 alphabetic characters				Neural network	15.2
Chang et al. [6]	2012	3–6 thumbnails	X	100	5	Statistical	6.9
De Mendizabal-Vázquez et al. [10]	2014	4-digit PIN	O	80	3–9	Euclidean distance	20
Zheng et al. [5]	2014	4-digit PIN/ 8-digit PIN	O	80	80	Nearest neighbor distance	3.65/ 4.45
Samura et al. [7]	2014	300 characters (approximately)	X	43	5	Weighted Euclidean distance + array disorder	2.2 FAR, 4.6 FRR
Giuffrida et al. [11]	2014	8-9 characters	O	20	40 (approximately)	kNN ($k = 1$) Manhattan weighted, kNN ($k = 1$) Manhattan scaled weighted	8
Antal and Szabó [12]	2015	10 characters	X	42	2/3 of data	Manhattan distance	12.9
Chang et al. [13]	2016	6-digit PIN 8-digit PIN 10-digit PIN	X	100	100	Statistical	23 21 16
Wu and Chen [14]	2015	8-digit PIN	O	100	500	SVM	0.556
Buschek et al. [15]	2015	6–8 characters	X	28	-	Probabilistic modeling	21.02
Dhage et al. [16]	2015	10 characters	X	15	10	Statistical	0.806
Bond and Awad [17]	2015	34 characters	X	25	-	Neural network	9.3
Teh et al. [18]	2016	4-digit PIN 16-digit PIN	X	50/150	7	Gaussian estimation, z-score matching function, standard deviation drift	7.57 5.49

Mahalanobis, and Euclidean distance, showed that the Manhattan distance classifier was the best being 12.9 EER and it was better than others by about 3% [12].

3. Distance-Based Classification Algorithms

As a behavioral biometric authentication, keystroke dynamics authentications make use of unique rhythms and behavior when a person types keys or characters on a keyboard. For authentication, first, a template of a legitimate user is created by feeding feature data into the template, and it is used to distinguish a legitimate user from imposters. Various classifiers are employed to decide the result such as support vector machine, multilayer perceptron, K-nearest neighbor, and distance-based classifiers.

In this paper, we use distance-based classifiers and classified if it is a legitimate user or not by computing a distance between a sample and mean point of user samples.

First of all, data samples are scaled to reduce the influences from different feature scales. For distance-based classifications, we need to define distance metric to use. In our experiment, we choose Euclidean distance and Manhattan distance. We review each definition in the next.

3.1. Scaling (Preprocessing). Since there are various on-board sensors in devices, units are different depending on a sensor

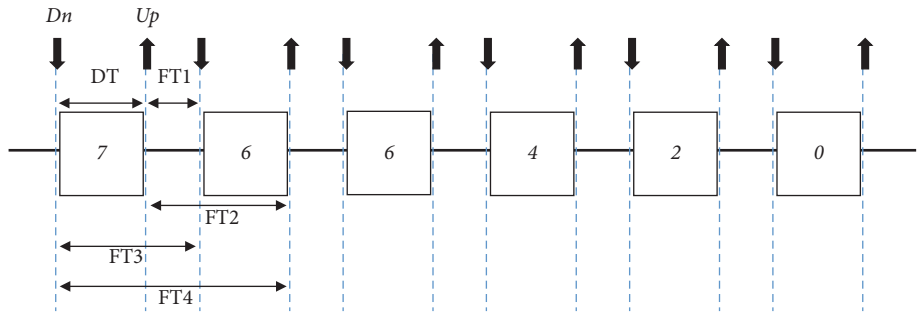


Figure 1: Feature configuration.

type and thus needed to be scaled within the same or fixed range to use as features; different units can be unexpected weighted values and cause different results. To scale various units, we use two scaling methods: the MinMax scaling and the standard scaling.

3.1.1. MinMax Scaling. The MinMax scaling is a scaling that scales data to a fixed range, 0 to 1, and it is calculated by the following equation:

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{1}$$

where X is a set of data, X_{min} is the minimum value of the data, and X_{max} is the maximum of the data. X_{sc} represents the result, scaled X.

3.1.2. Standard Scaling. The standard scaling scales data, x, where the mean is 0 and the standard deviation is 1 so that the data are scaled around 0 with the standard deviation value, 1. The formula is given by

$$z = \frac{x - \mu}{\sigma} \quad (\text{where } \mu = 0,\ \sigma = 1), \tag{2}$$

where μ is the mean of x and σ is the standard deviation of x.

3.2. Distance Metrics

3.2.1. Euclidean Distance. The Euclidean distance is calculating the distance between two n-dimension vectors, $p(p_1, p_2, \ldots, p_n)$ and $q(q_1, q_2, \ldots, q_n)$, as a straight line and the formula is given by

$$\begin{aligned} d(p,q) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \end{aligned} \tag{3}$$

3.2.2. Manhattan Distance. The Manhattan distance calculates the distance between two n-dimension vectors, $p(p_1, p_2, \ldots, p_n)$ and $q(q_1, q_2, \ldots, q_n)$, by subtracting the values and then summing the absolute of them as follows:

$$\begin{aligned} d(p,q) &= |q_1 - p_1| + |q_2 - p_2| + \cdots + |q_n - p_n| \\ &= \sum_{i=1}^{n} |q_i - p_i|. \end{aligned} \tag{4}$$

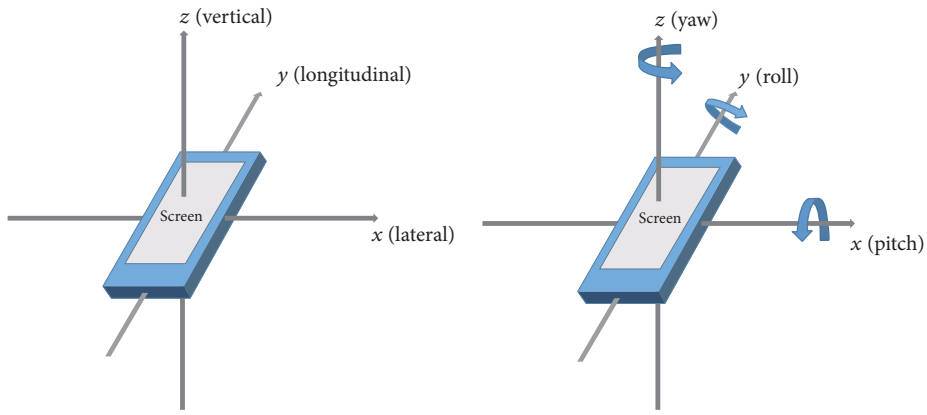
4. Features and Data Collection

Possible data extracted from smartphone are divided into two groups, keystroke data and motion data, in large. The keystroke data measured by gesture APIs that perceives touch inputs from keystroke data are "time," "size," "coordinate," and so on; "time" returns the time when events happen, "size" returns a size of a fingertip that pressed the touchscreen, "coordinate" returns coordinates of a point where a user touch. From motion sensor data, features related to movements can be extracted: accelerometer, gravity, rotation, and atmospheric pressure. In the following sections, we explain each feature.

4.1. Keystroke Data

4.1.1. Time. In "time," there are 4 types of down-time (DT) and a flight-time (FT). As Figure 1 shows, a DT is difference in time from the moment a user presses (or touches) a key (Dn) to the moment the user releases the key (Up). A FT is difference in time between pressing or releasing a key and another key; time interval between releasing a key and pressing the next key is called FT1; time interval between releasing a key and releasing the next key is called FT2; time interval between pressing a key and pressing the next key is called FT3; time interval between pressing a key and releasing the next key is called FT4. Time data, therefore, captured per 1 sample, 6-digit PIN, consist of 30-raw data in "time": 1 DT and 4 FT per 1 key.

4.1.2. Size. "Size" extracts sizes of user's fingertip each time pressing and releasing happen. Two data are captured per 1 key: one is size when pressing a key (sizeDn) and the other is size when releasing the key (sizeUp). There are thus 12-raw data in "size" for 1 sample, 6-digit PIN.



(a) Accelerometer (acc) (b) Gyroscope (gyr)

Figure 2: Axes of the motion sensors, (a) accelerometer and (b) gyroscope.

4.1.3. Coordinate. "Coordinate" extracts coordinate values for horizontal x-axis and vertical y-axis (x, y), where a key is pressed and released on the touch screen of a device. Coordinate values from pressing (xyDn) and releasing (xyUp) a key are 4-raw data so that 24-raw data of "coordinate" are extracted per 1 sample, 6-digit PIN.

4.2. Motion Sensor

4.2.1. Accelerometer (ACC). "Accelerometer" calculates device's accelerometer (m/s^2) of 3 axes, lateral x-axis, longitudinal y-axis, and vertical z-axis as Figure 2(a), by taking gravity values into account. Numbers of raw data of "accelerometer," therefore, are uneven by samples and are needed to be reshaped as regular form. Formulas and grouping the data to reshape are dealt with in Section 4.2.4 with the following "grot" and "gyr."

4.2.2. Game-Rotation (GROT). "Rotation" calculates device's angles with the geomagnetic field so that the values are influenced by the north. "Game-rotation," however, calculates the angles without any influence of the north and it means that "game-rotation" values are more accurate to measure the relative rotation. The "game-rotation" is more suitable than the "rotation" to tell person's behavior pattern, and it also returns values by 3 axes: x-axis, y-axis, and z-axis.

4.2.3. Gyroscope (GYR). "Gyroscope" measures the rate of rotation (rad/s) of a device by 3 axes, x-axis (pitch), y-axis (roll), and z-axis (yaw) as Figure 2(b). Filtering or corrections for any noise or drift are not applied in "gyroscope" data.

4.2.4. Formulas for Motion Data. As mentioned, the 3 types of motion data, "acc," "grot," and "gyr," are uneven and contain overfull data. They, therefore, need to be reshaped by some formulas. First of all, we group them by an interval between pressing and releasing a key and discard the rest of them. Then there are 5 formulas for the grouped data: average value (mean), root mean square (RMS), sum of positive values (pos), sum of positive values (neg), and standard deviation (std) [11].

In case of X array of n values, $X = \{x_1, x_2, \ldots, x_n\}$, mean ($x_{\text{mean}}$), RMS ($x_{\text{RMS}}$), pos ($x_{\text{pos}}$), neg ($x_{\text{neg}}$), and std ($x_{\text{std}}$) are defined as follows in sequence:

$$
\begin{aligned}
x_{\text{mean}} &= \frac{x_1 + x_2 + \cdots + x_n}{n}, \\
x_{\text{rms}} &= \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)}, \\
x_{\text{pos}} &= \sum_{i=i}^{n} x_i \quad (\text{where } x_i > 0), \\
x_{\text{neg}} &= \sum_{i=i}^{n} x_i \quad (\text{where } x_i < 0), \\
X_{\text{std}} &= \sqrt{\frac{\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2}{n}} \quad (\text{where } \overline{x} \text{ is mean}).
\end{aligned}
\tag{5}
$$

4.3. Raw Data Collection and Extracted Features. As shown in Figure 3, we developed an Android application that collects raw data by entering 6-digit PIN, "766420"; the PINs were generated by considering various positions of numbers on the touchscreen following [18, 26]. We installed the app on Nexus 5X and 22 subjects (users) participated in the data collection. 100 samples were collected for each user where one sample is one time input of 6-digit PIN. As Table 2 shows, each sample consists of 6 sets of 20 features per one key; 5 "time" features (DT, FT1, FT2, FT3, and FT4), 2 "size" features (sizeDn and sizeUp), 2 "coordinate" features (XyDn and XyUp), 3 "acc" features (x, y, and z), 3 "grot" features (x, y, and z), and 3 "gyr" features (x, y, and z); thus, one sample has 120 features in total.

4.4. Error Rate. Let us first define error rates. In user authentication, there can be two types of errors, false acceptance error, and false rejection error. False acceptance error (FAR) indicates error rate of accepting an imposter user as a legitimate user. False reject error (FRR) means error rate of rejecting a legitimate user as considering him/her as an imposter.

Table 2: Types and numbers of features from the PIN, "766420."

Raw data 1 key																		
Type	Keystroke									Motion								
Feature	Time					Size		Coordinate		Acc			Grot			Gyr		
	DT	FT1	FT2	FT3	FT4	sizeDn	sizeUp	*Xy*Dn	*Xy*Up	*x*	*y*	*z*	*x*	*y*	*z*	*x*	*y*	*z*
Each #	1	1	1	1	1	1	1	2	2	Formula								
			5			2		4		1	1	1	1	1	1	1	1	1
	11 features									9 features								
Total #	20 features per 1 key 1 sample = 20 ∗ 6 = 120 features																	

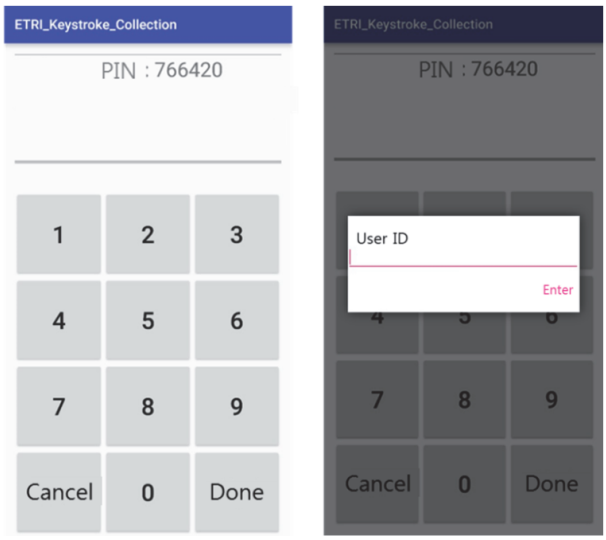


Figure 3: Application to collect user data. After entering PINs, "766420," users enter their ID on the pop-up box.

FAR tells the soundness of the system, whether it is a secure authentication mechanism or not. If FAR is high, imposter users including attackers can easily go through the authentication system. On the other hand, FRR tells the completeness of the system whether it is usable or not. If FRR is high, a user can fail in the authentication and has to retry the authentication procedure again and again. In a fuzzy data-based authentication system, there are no perfect completeness and perfect soundness since authentication factor has noise. In real applications, achieving good soundness is more important completeness because most users bear with 2-3 retrials as long as the authentication system provides expected level of security against imposters.

Depending on the threshold value distinguishing legitimate users from imposters, FAR and FRR move. Generally, reducing FAR increases FRR and vice versa. When they are equalized, we say it is EER (Equal Error Rate). EER is often used as a performance measure to show research results of fuzzy data identification/authentication. When FAR and FRR are close to EER, they approximately satisfy the relation (FAR + FRR) = 2 ∗ EER. Therefore, once EER is known, you can reduce a wanted error rate (for instance, FAR) and expect the other error rate from the relation (for instance, FRR ~ = 2 ∗ EER − FAR).

5. Experiment Results

In this section, we show our experimental results, one with distance-based classification and another with OCSVM. We present each result and their comparisons.

5.1. Distance-Based Classification. In our experiment, we use two scaling methods, MinMax scaling and standard scaling, and use two distance metrics, Euclidean distance and Manhattan distance (details are in Section 3). Our experiments are done with all four combinations of two scaling methods and two distance metrics and here we presented the best results.

We have two main results. First, our experimental result says that adding features from motion data in addition to features from keystroke data gives better performance (i.e., smaller EER). Second, our experimental result gives that keystroke authentication is more effective against opposite gender imposters. We further discuss two results in the next.

5.1.1. Adding Features from Motion Data. As discussed in Section 4.2.4, motion data are uneven and contain overfull data, while other keystroke data such as time, size, and coordinate are atomic and easily transformed into a feature. Thus, to shape motion data into a feature, there are five formula ways: "mean," "root mean square," "positive sum," "negative sum," and "standard deviation." First, we experiment with 5 different formulas featuring motion data. We all tried 4 combinations of MinMax scaling or standard scaling, and Euclidean distance or Manhattan distance. We obtained the best result with standard scaling and Manhattan distance and present the detailed result next.

Figure 4 shows the result of the averaged EER for 5 formulas, respectively, and EER of the "mean" formula, 12.63%, is the lowest rate. We also experimented with inclusion of all 5 formulas. Still, the experiment with including only "mean" formula gave the best result.

As seen in Figure 5, when "mean" formula of motion data is added, the error rate improved compared to the experiment using features only from keystroke data (time, size, and coordinate). Using the keystroke data only, we obtained 8.94% EER and 7.89% EER by adding the motion

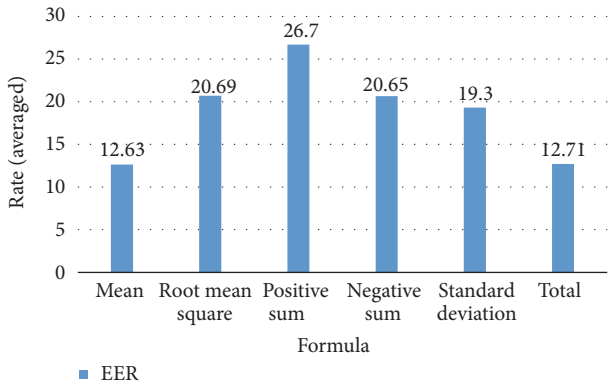


Figure 4: Comparison of formulas for motion data only. "Total" means using all of the 5 formulas as features.

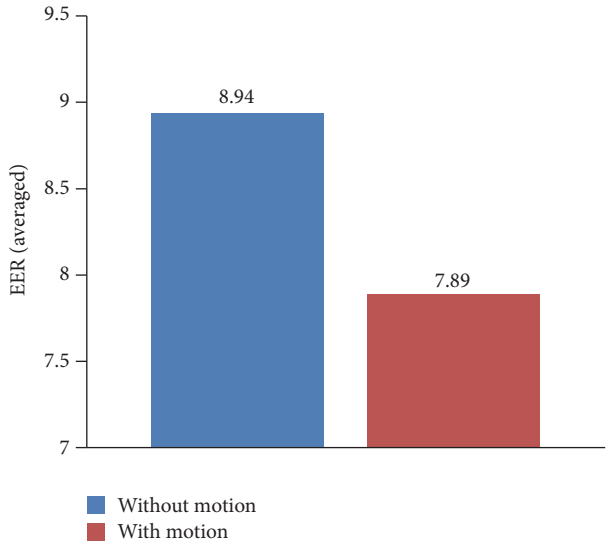


Figure 5: Rate change by adding motion data. Motion data are calculated by "mean" formula.

data. The EER decreased by 1.05% when the motion data were added.

5.1.2. FAR by Gender Match. Our experimental result shows that opposite gender's match influences the result, especially on FAR. If gender of a legitimate user and gender of impostors are different, the experiment result gives lower FAR.

In Figure 6, we draw a line between male and female by a legitimate user's gender and compare three cases, same gender, opposite gender, and total case, which is irrelevant to gender. In Figure 6, dotted bars show the test results in case of the keystroke data only and solid bars show the test results when motion data is also used. Regardless of the case, the result shows lower FAR if the gender between a legitimate user and imposters is not matched. FAR decrease by 4.07% and 0.64% in case of opposite gender for male legitimate users and female legitimate users, respectively. Figure 6 and the rest of the results, Figures 7 and 8, are based on the results from experiments with standard scaling and Manhattan distance, which gave the best result.

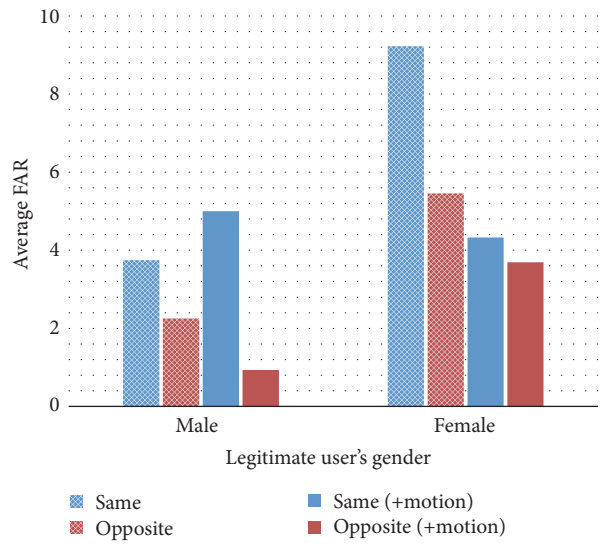


Figure 6: Average FAR by gender match. By gender match, opposite or the same, we compare keystroke data and keystroke data with motion data.

Figure 7 shows how much each feature influences FAR by gender's match. For better vision, we also provide Figure 8 showing differences of error rate between the cases with the same gender imposters and opposite gender imposters. From Figure 8, we can see that the feature "grot" ("game-rotation" that returns rotation values without the geomagnetic field) and the feature "sizeDn" (a size when pressing a key) reduce FAR the most upon standard scaling. The biggest decrease on FAR is 22.73% from "grot" and "sizeDn" and "acc" comes next as 7.08% (7.28%) and 4.98% (5.16%). This result determining features strong against opposite gender imposters can be very useful in applications where gender authenticity is critical, for instance, online dating, or online same gender competition exam/game.

5.2. One-Class Support Vector Machine (OCSVM). Support vector machines (SVMs) are one of the machine learning algorithms which is supervised and used as classification method, regression method, or outliers detection. Using labeled training data, SVMs find optimal hyperplane and it is used to differentiate unlabeled data. In SVMs, a kernel transforming data into higher dimensional data is used if data are not linearly separate. There are various kernels such as linear, polynomial, radial basis function, and sigmoid in kernel function [28].

Among SVMs, there are one-class support vector machines (OCSVMs) which are unsupervised learning and determine whether new data is outlier or not. Using the OCSVM, we compare with the same conditions that one is using keystroke data only and the other is adding motion data. As previous result with distance-based classifier shows, EER decreases by about 1.24% using the OCSVM as well as when the motion data were added as Figure 9 shows. Repeating the same experiment about gender match on the OCSVM, FAR decreasing is more obvious as shown in Figure 10. In case of male legitimate users, FARs decline by

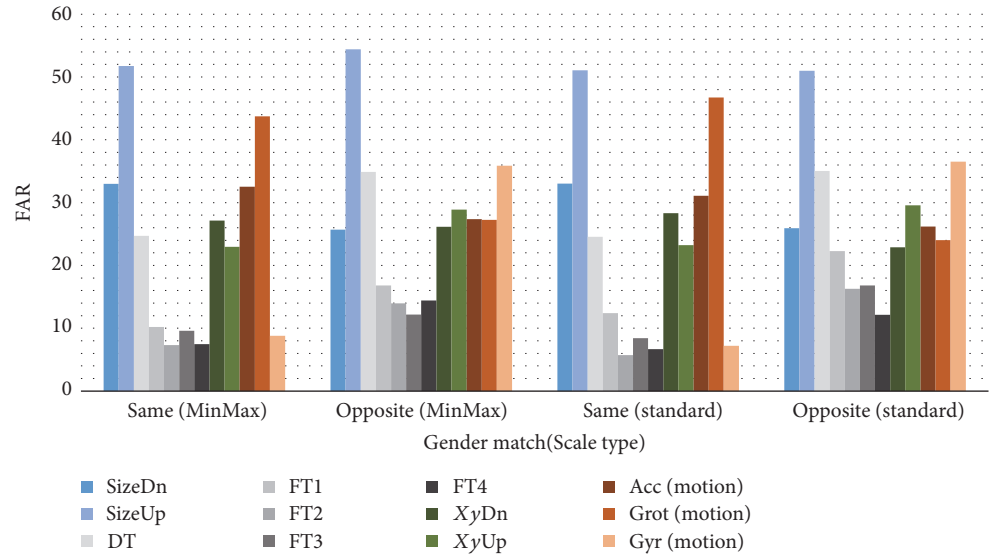


Figure 7: FAR change of single feature by gender match.

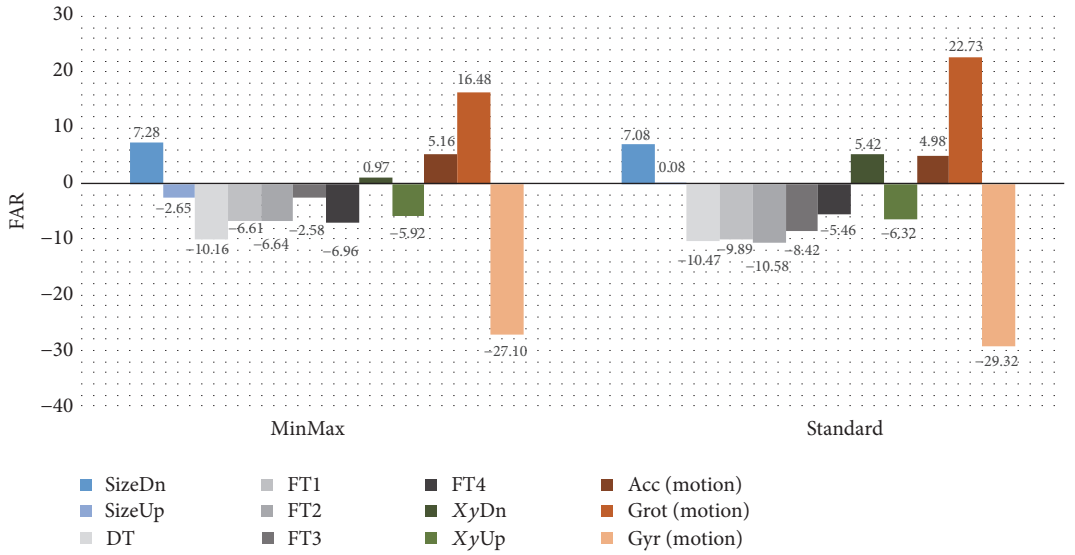


Figure 8: Visualization of differences between FAR upon experiment with the same gender and FAR upon experiment with the opposite gender imposters (i.e., the former minus the latter).

about 8% and the other case's FARs show about a 6% drop irrespective of data types.

6. Conclusion

Recently, new authentication methods on smartphones using biometrics information such as iris scan and face recognition are rising. However, the existing methods including PINs are still often used. Many studies for the keystroke dynamics are in progress to strengthen PIN-based authentication. In our study, we include features from motion data to see how they are effective in keystroke dynamics authentication. We show which formula is better to handle motion data and that keystroke authentication improves when features from motion data are added. We obtained the best result with motion data using "mean" formula and obtained 7.89% EER, which is the best performance so far upon 6-digit with distance-based classification. We also showed an interesting result that gender's match has influenced FAR and found features, "grot" ("game-rotation" that returns rotation values without the geomagnetic field) and "sizeDn" (a size when pressing a key) that influence the most. We believe our results will contribute to better understanding of keystroke dynamics authentication and to future study and development of 6-digit distance-based keystroke dynamics. In particular, our result by gender match can be adopted to applications where gender authenticity is crucial, for instance, online dating, or online same gender competition exam/game.

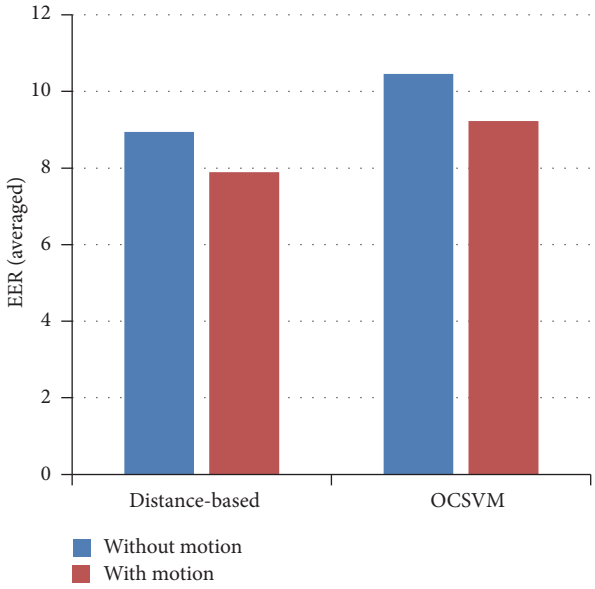


Figure 9: Rate change by adding motion data on SVMs.

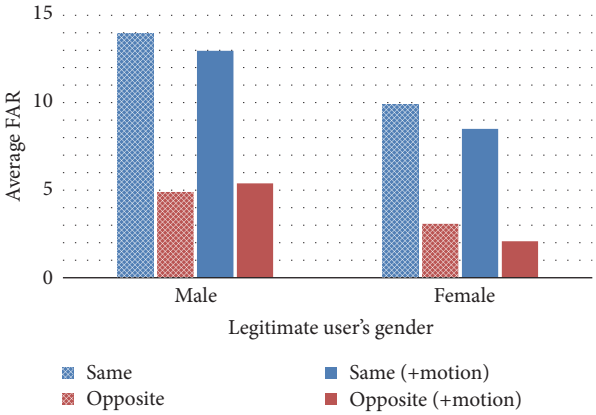


Figure 10: Average FAR by gender match. By gender match, opposite or the same, we compare keystroke data and keystroke data with motion data.

For the future work, studies for additional authentication step focusing on specific features that are strengthened to opposite gender are needed to be investigated. Also, we will continue keystroke dynamics research with different preprocessing approaches and classification methods and also to find more appropriate combinations of features reducing ERR.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean government (MSIT) (no. 2015-0-00168, Development of Universal Authentication Platform Technology with Context-Aware Multifactor Authentication and Digital Signature, and no. 2016-0-00097, Development of Biometrics-Based Key Infrastructure Technology for Online Identification).

References

[1] Statista, "Number of smartphone users in the U.S. 2010-2022," https://www.statista.com/statistics/201182/forecast-of-smartphone-users-in-the-us/.

[2] Statista, Number of smartphone users in South Korea from 2015 to 2022, https://www.statista.com/statistics/467171/forecast-of-smartphone-users-in-south-korea/.

[3] N. Ben-Asher, N. Kirschnick, H. Sieger, J. Meyer, A. Ben-Oved, and S. Möller, "On the need for different security methods on mobile phones," in *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pp. 465–473, ACM, Stockholm, Sweden, September 2011.

[4] P. K. Sari, G. S. Ratnasari, and A. Prasetio, "An evaluation of authentication methods for smartphone based on users' preferences," in *Proceedings of the IOP Conference Series: Materials Science and Engineering*, vol. 128, IOP Publishing, Bristol, UK, 2016.

[5] N. Zheng, K. Bai, H. Huang, and H. Wang, "You are how you touch: user verification on smartphones via tapping behaviors," in *Proceedings of the IEEE 22nd International Conference on Network Protocols (ICNP '14)*, pp. 221–232, IEEE, North Carolina, NC, USA, October 2014.

[6] T.-Y. Chang, C.-J. Tsai, and J.-H. Lin, "A graphical-based password keystroke dynamic authentication system for touch screen handheld mobile devices," *The Journal of Systems and Software*, vol. 85, no. 5, pp. 1157–1165, 2012.

[7] T. Samura, M. Izumi, and H. Nishimura, "Flick input authentication in Japanese free text entry on smartphones," in *Proceedings of the 53rd Annual Conference of the Society of Instrument and Control Engineers of Japan, SICE '14*, pp. 1348–1353, 2014.

[8] N. L. Clarke, S. M. Furnell, B. M. Lines, and P. L. Reynolds, "Keystroke dynamics on a mobile handset: a feasibility study," *Information Management and Computer Security*, vol. 11, no. 4, pp. 161–166, 2003.

[9] N. L. Clarke and S. M. Furnell, "Authenticating mobile phone users using keystroke analysis," *International Journal of Information Security*, vol. 6, no. 1, pp. 1–14, 2007.

[10] I. De Mendizabal-Vázquez, D. De Santos-Sierra, J. Guerra-Casanova, and C. Sánchez-Ávila, "Supervised classification methods applied to keystroke dynamics through mobile devices," in *Proceedings of the 48th Annual IEEE International Carnahan Conference on Security Technology, ICCST '14*, pp. 1–6, October 2014.

[11] C. Giuffrida, K. Majdanik, M. Conti, and H. Bos, "I sensed it was you: authenticating mobile users with sensor-enhanced keystroke dynamics," in *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, vol. 8550, pp. 92–111, Springer, Cham, Switzerland, 2014.

[12] M. Antal and L. Z. Szabó, "Keystroke Dynamics on Android Platform," *Procedia Technology*, vol. 19, pp. 820–826, 2015.

[13] T.-Y. Chang, C.-J. Tsai, W.-J. Tsai, C.-C. Peng, and H.-S. Wu, "A changeable personal identification number-based keystroke

dynamics authentication system on smart phones," *Security and Communication Networks*, vol. 9, no. 15, pp. 2674–2685, 2016.

[14] J. Wu and Z. Chen, "An implicit identity authentication system considering changes of gesture based on keystroke behaviors," *International Journal of Distributed Sensor Networks*, vol. 11, no. 6, Article ID 470274, 2015.

[15] D. Buschek, A. De Luca, and F. Alt, "Improving accuracy, applicability and usability of keystroke biometrics on mobile touchscreen devices," in *Proceedings of the 33rd Annual CHI Conference on Human Factors in Computing Systems, CHI '15*, pp. 1393–1402, April 2015.

[16] S. Dhage, P. Kundra, A. Kanchan, and P. Kap, "Mobile authentication using keystroke dynamics," in *Proceedings of the International Conference on Communication, Information and Computing Technology, ICCICT '15*, IEEE, Mumbai, India, January 2015.

[17] W. Bond and A. E. A. Awad, "Touch-based static authentication using a virtual grid," in *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security*, pp. 129–134, 2015.

[18] P. S. Teh, N. Zhang, A. B. J. Teoh, and K. Chen, "TDAS: a touch dynamics based multi-factor authentication solution for mobile devices," *International Journal of Pervasive Computing and Communications*, vol. 12, no. 1, pp. 127–153, 2016.

[19] R. Spillane, "Keyboard apparatus for personal identification," IBM Technical Disclosure Bulletin, vol. 17, no. 3346, 1975.

[20] G. E. Forsen, M. R. Nelson, and R. J. Staron, "Personal attributes authentication techniques," Tech. Rep. RADC-TR-77-333, Rome Air Development Center, New York, NY, USA, 1977.

[21] R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro, "Authentication by keystroke timing: some preliminary results," Tech. Rep. R-2526-NSF, RAND Corporation, California, Calif, USA, 1980.

[22] D. Umphress and G. Williams, "Identity verification through keyboard characteristics," *International Journal of Man-Machine Studies*, vol. 23, no. 3, pp. 263–273, 1985.

[23] N. L. Clarke, S. M. Furnell, B. M. Lines, and P. L. Reynolds, "Subscriber authentication for mobile phones using keystroke dynamics," in *Proceedings of the 3rd International Network Conference (INC '02)*, 2002.

[24] H. Saevanee and P. Bhattarakosol, "Authenticating user using keystroke dynamics and finger pressure," in *Proceedings of the 6th IEEE Consumer Communications and Networking Conference, CCNC '09*, IEEE, Nevada, Nev, USA, January 2009.

[25] M. Trojahn and F. Ortmeier, "Biometric authentication through a virtual keyboard for smartphones," *International Journal of Computer Science and Information Technology*, vol. 4, no. 5, 2012.

[26] S. Zahid, M. Shahzad, S. A. Khayam, and M. Farooq, *Keystroke-Based User Identification on Smart Phones*, Springer, Berlin, Germany, 2009.

[27] Z. Xu, K. Bai, and S. Zhu, "TapLogger: inferring user inputs on smartphone touchscreens using on-board motion sensors," in *Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 113–124, ACM, Arizona, Ariz, USA, April 2012.

[28] Wikipedia, Support vector machine, https://en.wikipedia.org/wiki/Support_vector_machine.

Hindawi
Security and Communication Networks
Volume 2018, Article ID 2369507, 9 pages
https://doi.org/10.1155/2018/2369507

Research Article

Under Quantum Computer Attack: Is Rainbow a Replacement of RSA and Elliptic Curves on Hardware?

Haibo Yi

School of Computer Engineering, Shenzhen Polytechnic, Shenzhen 518055, China

Correspondence should be addressed to Haibo Yi; haiboyi@126.com

Received 26 October 2017; Accepted 15 January 2018; Published 11 February 2018

Academic Editor: Umar M. Khokhar

Among cryptographic systems, multivariate signature is one of the most popular candidates since it has the potential to resist quantum computer attacks. Rainbow belongs to the multivariate signature, which can be viewed as a multilayer unbalanced Oil-Vinegar system. In this paper, we present techniques to exploit Rainbow signature on hardware meeting the requirements of efficient high-performance applications. We propose a general architecture for efficient hardware implementations of Rainbow and enhance our design in three directions. First, we present a fast inversion based on binary trees. Second, we present an efficient multiplication based on compact construction in composite fields. Third, we present a parallel solving system of linear equations based on Gauss-Jordan elimination. Via further other minor optimizations and by integrating the major improvement above, we implement our design in composite fields on standard cell CMOS Application Specific Integrated Circuits (ASICs). The experimental results show that our implementation takes 4.9 us and 242 clock cycles to generate a Rainbow signature with the frequency of 50 MHz. Comparison results show that our design is more efficient than the RSA and ECC implementations.

1. Introduction

The idea of public key cryptography was introduced by Diffie and Hellman. Their method for key exchange came to be known as Diffie-Hellman key exchange [1]. This was the first published practical method for establishing a shared secret key over an authenticated communications channel without using a prior shared secret. Then a public key cryptographic scheme was invented by Rivest et al. [2]. This scheme came to be known as RSA, from their initials. RSA uses exponentiation modulo, a product of two very large primes, to encrypt and decrypt, performing both public key encryption and public key digital signature. The introduction of elliptic curve cryptography by Koblitz [3] and Miller [4] in the mid-1980s has yielded new public key algorithms based on the discrete logarithm problem. Elliptic curves provide smaller key sizes and faster operations for approximately equivalent estimated security. Since then, various schemes of encryption and signature generation have been developed in the field of public key cryptography.

Efficient implementations of these schemes have played a crucial role in numerous real-world security applications, such as confidentiality, authentication, integrity, and nonrepudiation. Since software implementations even on multicore processors can often not provide the performance level needed, hardware implementations are thus the only option, which appear to be a promising solution to inherent performance issues of public key cryptographic systems and provide greater resistance to tampering. Among hardware implementations of public key cryptographic systems, RSA and elliptic curves systems are the most widely adopted candidates [5–14]. Their security lies in the difficulty of factorizing large integers and the discrete logarithm problem, respectively. Shor algorithm was invented by Shor which could solve the problems of the prime factors of large numbers and elliptic curve discrete logarithm in polynomial time [15]. Such cryptographic schemes have potential weakness under quantum computer attacks.

Multivariate cryptography is one of the most popular postquantum cryptography since it has the potential to

resist quantum computer attacks [16]. The main strength of multivariate cryptography is that its underlying mathematical problem is to solve a set of Multivariate Quadratic (MQ) polynomial equations in a finite field, which is proven to be an NP-hard problem [17]. During the past thirty years, various multivariate cryptographic schemes have been proposed, like Unbalanced Oil-Vinegar Signature (UOV) [18], Rainbow [19, 20], Tame Transformation Signature (TTS) [21, 22], and others [23–25]. Their implementations have been one of the subjects of a lot of researches and continue to be a topic of interest in many areas, for example, efficient multivariate systems on Field Programmable Gate Arrays (FPGAs) [26], small multivariate processors on FPGAs [27], high speed Rainbow on FPGAs [28], and minimized multivariate PKC on Application Specific Integrated Circuits (ASICs) [29].

Among the existing multivariate cryptographic schemes, Rainbow belongs to Oil-Vinegar family, which can be viewed as a multilayer unbalanced Oil-Vinegar system. Compared with RSA and elliptic curves, the security of Rainbow is based on solving a set of MQ polynomial equations, which has the potential to resist quantum computer attacks.

Our Contributions. In this paper, we present techniques to exploit Rainbow signature on hardware meeting the requirements of efficient high-performance applications. We propose a general architecture for efficient hardware implementations of Rainbow and enhance our design in three directions. First, we present a fast inversion in $GF((2^4)^2)$ based on binary trees, which is the extension of the work in [30]. Second, we present an efficient multiplication in $GF((2^4)^2)$ based on compact construction, which is the extension of the work in [27]. Third, we present a parallel solving system of linear equations in $GF((2^4)^2)$ based on Gauss-Jordan elimination, which is based on the work in [28]. Via further other minor optimizations and by integrating the major improvement above, our design is implemented on ASICs and provides significant reductions in time-area product. The comparisons with other public key cryptographic systems show that Rainbow has a good performance on hardware and is a better candidate than RSA and elliptic curves under quantum computer attacks.

Moreover, our design can be generalized with minor modifications that also support FPGAs. Besides, Rainbow implementations on hardware must be protected against a wide range of attacks, including side channel attacks. Side channel attack belongs to physical attack, which is any attack based on information gained from the physical implementation of cryptographic systems, rather than brute force or theoretical weaknesses in cryptographic algorithms. Therefore, we discuss defending against a possible differential power analysis for Rainbow and we present countermeasures against fault analysis and differential power analysis attack.

Organization. The rest of this paper is organized as follows: Section 2 introduces Rainbow signature schemes. Section 3 presents building blocks for Rainbow schemes. Section 4 presents efficient implementations of Rainbow on ASICs. Section 5 compares our design with other public key cryptographic systems. Section 6 discusses defending against a possible differential power analysis for Rainbow. Section 7 summarizes our design.

2. Preliminary

Among multivariate signatures, Rainbow belongs to Oil-Vinegar family, which can be viewed as a multilayer unbalanced Oil-Vinegar system. The construction of Rainbow includes affine transformation L_1, central map transformation F, and affine transformation L_2; that is,

$$F \circ L_2 (x_0, x_1, \ldots, x_{n-1}) = L_1^{-1} (y_0, y_1, \ldots, y_{m-1}). \quad (1)$$

The hash value of the message of Rainbow is $y(y_0, y_1, \ldots, y_{m-1})$ and its size is m, where $y_0, y_1, \ldots, y_{m-1}$ are elements in a finite field. We also suppose that the signature is $x(x_0, x_1, \ldots, x_{n-1})$ and its size is n, where $x_0, x_1, \ldots, x_{n-1}$ are elements in a finite field. The private keys of Rainbow are L_1, F, and L_2.

Among the existing Rainbow schemes, Rainbow(17, 13, 13) is commonly believed to provide a security level of 2^{80} [20], which works with 17 first-layer Vinegar variables and 13 first-layer and 13 second-layer Oil variables in $GF(256)$. This scheme is depicted in Table 1 and is introduced as follows.

We suppose that the hash value of the message is $y(y_0, y_1, \ldots, y_{25})$ and its size is 26, where $y_0, y_1, \ldots, y_{25}$ are field elements. We also suppose that the signature is $x(x_0, x_1, \ldots, x_{42})$ and its size is 43, where $x_0, x_1, \ldots, x_{42}$ are field elements.

In order to sign a message, we need to solve the equation

$$F \circ L_2 (x_0, x_1, \ldots, x_{42}) = L_1^{-1} (y_0, y_1, \ldots, y_{25}). \quad (2)$$

To do this, we first solve

$$\overline{y} = L_1^{-1} (y_0, y_1, \ldots, y_{25}). \quad (3)$$

L_1^{-1} is an affine transformation:

$$\overline{y} = Ay + B, \quad (4)$$

where A is a matrix with the size of 26×26 and B is a vector with the size of 26. A and B are parts of private keys.

Second, we solve

$$\overline{x} = F^{-1} (\overline{y}_0, \overline{y}_1, \ldots, \overline{y}_{25}), \quad (5)$$

where the construction depends on a map

$$F (\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{42}) = (f_0, f_1, \ldots, f_{25}). \quad (6)$$

F is a two-layer construction; namely, $(f_0, f_1, \ldots, f_{25})$ are divided into two layers:

$$\begin{aligned} &0 : f_i \mid i = 0, 1, \ldots, 12 \\ &1 : f_i \mid i = 13, 14, \ldots, 25. \end{aligned} \quad (7)$$

Similarly, $\overline{x}(\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{42})$ are divided into two layers. $\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{16}$ and $\overline{x}_{17}, \overline{x}_{18}, \ldots, \overline{x}_{29}$ are Vinegar variables and Oil variables of the first layer, respectively; $\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{42}$ and $\overline{x}_{30}, \overline{x}_{31}, \ldots, \overline{x}_{42}$ are Vinegar variables and Oil variables of the second layer, respectively.

Table 1: Parameters of Rainbow signature schemes.

Finite field	Message size	Signature size	Private key	Public key	Number of layers	LSE size
$GF((2^4)^2)$	26 bytes	43 bytes	L_1, L_2, F	$L_1 \circ F \circ L_2$	2	13×13

MQ polynomials f are defined by

$$f(O_0, O_1, \ldots, O_{o-1}) = \sum \alpha_{ij} O_i V_j + \sum \beta_{ij} V_i V_j + \sum \gamma_i V_i + \sum \delta_i O_i + \eta, \quad (8)$$

where $O_i, V_i/V_j$ are Oil and Vinegar variables on this layer and the coefficients α_{ij}, β_{ij}, γ_i, δ_i, and η are parts of private keys.

We randomly choose $\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{16}$ and evaluate $f_0, f_1, \ldots, f_{12}$. Then we solve the systems of linear equations on $\overline{x}_{17}, \overline{x}_{18}, \ldots, \overline{x}_{29}$. Then we evaluate $f_{13}, f_{14}, \ldots, f_{25}$ and solve the systems of linear equations on $\overline{x}_{30}, \overline{x}_{31}, \ldots, \overline{x}_{42}$.

Last, we solve

$$x = L_2^{-1}(\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{42}). \quad (9)$$

L_2^{-1} is an affine transformation

$$x = C\overline{x} + D, \quad (10)$$

where C is a matrix with the size of 43×43 and D is a vector with the size of 43. C and D are parts of private keys.

Then x is the signature of y.

3. Building Blocks for Rainbow Schemes

Considering Section 2, we see that, in order to generate a Rainbow signature, the following operations are required:

(1) Computing affine transformations, that is, $\overline{y} = Ay + b$, where A is a matrix and b is a vector

(2) Computing central map transformation, that is, evaluating multivariate polynomials and solving systems of linear equations.

Computing these operations requires multiplications, inversions, and solving systems of linear equations in a finite field, which are presented in the following.

3.1. A Fast Inversion Based on Binary Trees. We suppose that $a(x) = a_h x + a_l$ and $b(x) = b_h x + b_l$ are the elements in $GF((2^4)^2)$ and $b(x)$ is the inverse of $a(x)$, where the subfield is $GF(2^4)$ and a_h, a_l, b_h, and b_l are elements in $GF(2^4)$. The irreducible polynomials in $GF((2^4)^2)$ are $q(x) = x^2 + x + 9$. Then the inversion is computed as follows:

$$b_l = (a_h + a_l) \times \left(9 \times {a_h}^2 + a_l \times a_h + {a_l}^2\right)^{-1},$$
$$b_h = a_h \times \left(9 \times {a_h}^2 + a_l \times a_h + {a_l}^2\right)^{-1}. \quad (11)$$

We adopt a pipelined architecture in $GF(2^4)$, which is the extension of the work in [30]. We use two binary trees for computing squares and inversions in $GF(2^4)$, which are illustrated as follows:

(1) Each binary tree has four layers; root nodes are on the third layer.

(2) Each node has at most two child nodes, left node represents value of zero, and right node represents value of one.

(3) Each child must either be a leaf or be the root of another tree; each node has a father node when it is not a root node.

(4) Each element in a finite field has a unique traversal from root to leaf.

(5) Each leaf (most) is linked to another leaf.

Figure 1 is the architecture based on binary trees for computing squares and inversions in $GF(2^4)$. We use two architectures in our design, that is, square-trees for squares and inversion-trees for inversions.

Square-trees: we suppose that traversal from root (e_0) to leaf (e_3) includes tree nodes e_0, e_1, e_2, and e_3, which represents the element $a(x)$ in $GF(2^4)$. If traversal from root (e_4) to leaf (e_7) represents the element $b(x)$ in $GF(2^4)$, which is the square of $a(x)$, then e_3 is linked to e_7. When we are required to compute the square of $a(x)$, it is very convenient to find its square via traversing the square-trees.

Inversion-trees: we suppose that traversal from root (e_0) to leaf (e_3) includes tree nodes e_0, e_1, e_2, and e_3, which represents the element $a(x)$ in $GF(2^4)$. If traversal from root (e_4) to leaf (e_7) represents the element $b(x)$ in $GF(2^4)$, which is the inverse of $a(x)$, then e_3 is linked to e_7. When we are required to compute the inverse of $a(x)$, it is very convenient to find its inverse via traversing the inversion-trees.

Since square-trees and inversion-trees have four layers, we can use them to compute squares and inversions with pipelining. The computation of $b(x) = a(x)^{-1}$ is presented as follows:

(1) Via using square-trees, we can compute ${a_h}^2$ and ${a_l}^2$ with pipelining.

(2) Via using a multiplier, we can compute $a_l \times a_h$.

(3) Via using a multiplier and an adder, we compute $9 \times {a_h}^2$ and $a_l \times a_h + {a_l}^2$.

(4) Via using inversion-trees, we can compute $(9 \times {a_h}^2 + a_l \times a_h + {a_l}^2)^{-1}$.

(5) Via using a multiplier and an adder, we compute $(a_h + a_l) \times (9 \times {a_h}^2 + a_l \times a_h + {a_l}^2)^{-1}$ and $a_h \times (9 \times {a_h}^2 + a_l \times a_h + {a_l}^2)^{-1}$.

(6) The inversion has been computed.

3.2. An Efficient Multiplication Based on Compact Construction. We suppose that $a(x) = a_h x + a_l$ and $b(x) = b_h x + b_l$

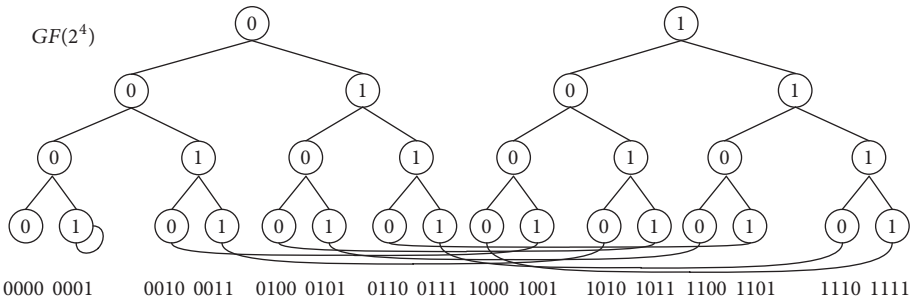


Figure 1: A pipelined architecture based on binary trees for computing squares and inversions in $GF(2^4)$.

are the elements in $GF((2^4)^2)$, where a_h, a_l, b_h, and b_l are elements in $GF(2^4)$. We also suppose that $c(x) = c_h x + c_l$ is the multiplication result of $a(x)$ and $b(x)$, where $a(x)$ is an element in $GF((2^4)^2)$ and c_h, c_l are elements in $GF(2^4)$. The irreducible polynomials in $GF((2^4)^2)$ are $q(x) = x^2 + x + 9$. Then the multiplication is computed as follows:

$$\begin{aligned} a(x) \times b(x) &= (a_h x + a_l)(b_h x + b_l) \\ &= \left(a_h b_h x^2 + (a_h b_l + a_l b_h) x + a_l b_l\right) \bmod q(x). \end{aligned} \tag{12}$$

By substituting $q(x) = x^2 + x + 9$ into (12), we have

$$\begin{aligned} c_h &= (a_h + a_l) \times (b_h + b_l) + a_l \times b_l, \\ c_l &= a_l \times b_l + 9 \times a_h \times b_h. \end{aligned} \tag{13}$$

The computations of c_h and c_l use a compact construction, which is the extension of the work in [27].

We design components *SubfieldAdder*, *SubfieldMultiplier*, and *RegshiftA*.

SubfieldAdder. It computes $a_h + a_l$ and $sa1 = b_h + b_l$ in $GF(2^4)$, where a_h, a_l, b_h, and b_l are elements in $GF(2^4)$.

SubfieldMultiplier. It computes $sa0 * sa1$, $a_h * b_h$, and $a_l * b_l$ in $GF(2^4)$, where a_h, a_l, b_h, b_l, $sa0$, and $sa1$ are elements in $GF(2^4)$.

RegshiftA. It performs the computation of right shift and a bit addition.

We adapt four *SubfieldAdder*s, three *SubfieldMultiplier*s, and *RegshiftA*, where *SubfieldAdder* and *SubfieldMultiplier* compute additions and multiplications in $GF(2^4)$, respectively.

*SubfieldAdder*0 and *SubfieldAdder*1 are used to compute

$$\begin{aligned} sa0 &= a_h + a_l, \\ sa1 &= b_h + b_l, \end{aligned} \tag{14}$$

respectively.

*SubfieldMultiplier*0, *SubfieldMultiplier*1, and *SubfieldMultiplier*2 are used to compute

$$\begin{aligned} sm0 &= sa0 * sa1, \\ sm1 &= a_h * b_h, \\ sm2 &= a_l * b_l, \end{aligned} \tag{15}$$

respectively.

RegshiftA is used to compute a right shift and a bit addition:

$$\begin{aligned} &rsa(sm1_3, sm1_2, sm1_1, sm1_0) \longrightarrow \\ &rsa(sm1_0, sm1_3, sm1_2, sm1_1), \\ &rsa(sm1_0, sm1_3, sm1_2, sm1_1) \longrightarrow \\ &rsa(sm1_0, sm1_3, sm1_2, sm1_1 + sm1_0). \end{aligned} \tag{16}$$

*SubfieldAdder*2 and *SubfieldAdder*3 are used to compute

$$\begin{aligned} c_h &= sm0 + sm2, \\ c_l &= rsa + sm2, \end{aligned} \tag{17}$$

respectively.

The multiplication has been computed.

3.3. A Parallel Solving System of Linear Equations Based on Gauss-Jordan Eliminations. We propose a parallel solving system of linear equations based on Gauss-Jordan eliminations, which is the extension of the work in [28]. We give a straightforward description of the proposed algorithm of the parallel variant of Gauss-Jordan elimination in Algorithm 1, where *operation*(i) stands for operation performed in the ith iteration, and $i = 0, 1, \ldots, 12$. The optimized Gauss-Jordan elimination with 13 iterations consists of pivoting, inversion, normalization, and elimination in each iteration.

We enhance the algorithm in four directions. First, multiplication is computed by invoking efficient multipliers designed in Section 3.2. Second, we adopt fast inverter described in Section 3.1. Third, inversion, normalization, and elimination are designed to perform simultaneously. Fourth, during the elimination in the ith iteration, we simultaneously choose the right pivot for the next iteration; namely, if

```
(1)  var
(2)    i: Integer;
(3)  begin
(4)    i := 0;
(5)    Pivoting(i = 0);
(6)    repeat
(7)      inversion(i), Normalization(i), Elimination(i);
(8)      Pivoting(i + 1);
(9)      i := i + 1;
(10)   until i = 13
(11) end.
```

Algorithm 1: Solving a system of linear equations $Ax = b$ with 13 iterations, where A is a 13×13 matrix.

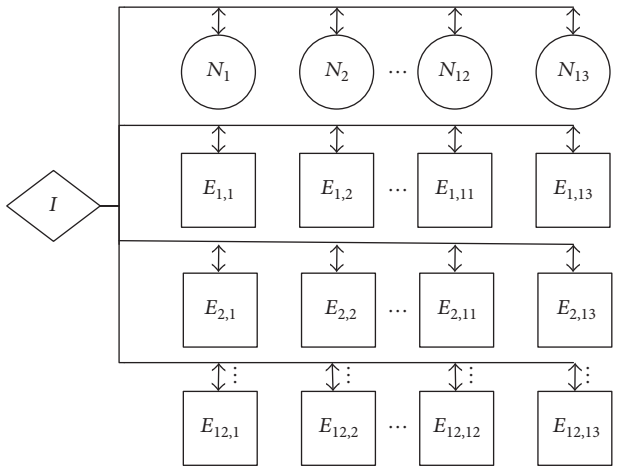


Figure 2: The proposed architecture for parallel solving system of linear equations with matrix size 13×13.

element $a_{i+1,i+1}$ of the next iteration is zero, we swap the $(i+1)$th row with another jth row with the nonzero element a_{ji}, where $i, j = 0, 1, \ldots, 12$. The difference from usual Gauss-Jordan elimination is that the usual Gauss-Jordan elimination chooses the pivot after the elimination, while we perform the pivoting during the elimination. In other words, at the end of each iteration, by judging the computational results in this iteration, we can decide the right pivoting for the next iteration. By integrating these optimizations, it takes only one clock cycle to perform one iteration.

The architecture for solving systems of linear equations in $GF((2^4)^2)$ is depicted in Figure 2 with matrix size 13×13. There exist three kinds of cells in the architecture, namely, I, N_l, and E_{kl}, where $k = 1, 2, \ldots, 12$ and $l = 1, 2, \ldots, 13$. The I cell is for fast inversion. As described in Section 3.1, two binary trees are included in the I cell for computed inversion. The N_l cells are for normalization. And the E_{kl} cells are for elimination. The architecture consists of one I cell, 13 N_l cells, and 156 E_{lk} cells.

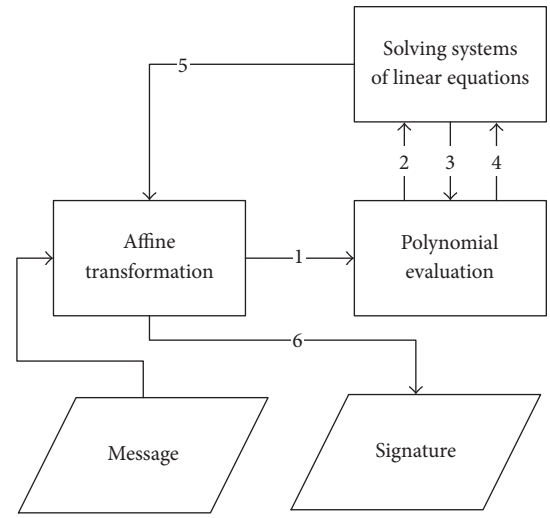


Figure 3: The flowchart of implementations of Rainbow scheme.

4. Efficient Implementation and Performance Evaluation

Rainbow(17, 13, 13) is computed via invoking affine transformation, polynomial evaluation, and solving systems of linear equations in $GF((2^4)^2)$. We depict the flowchart of implementations of Rainbow(17, 13, 13) in Figure 3:

(1) Compute the first affine transformation L_1 via invoking matrix-vector multiplication and vector addition.

(2) Evaluate the first 13 multivariate polynomials $f_0, f_1, \ldots, f_{12}$ on the first layer of central map transformation F.

(3) Solve the first systems of linear equations with matrix size 13×13 of central map transformation F.

(4) Evaluate the second 13 multivariate polynomials $f_{13}, f_{14}, \ldots, f_{25}$ on the second layer of central map transformation F.

(5) Solve the second systems of linear equations with matrix size 13×13 of central map transformation F.

(6) Compute the second affine transformation L_2 via invoking matrix-vector multiplication and vector addition.

In order to prove that the designs of Rainbow(17, 13, 13) are efficient on hardware, Hardware Description Language (Verilog HDL) code for modeling the designs has been implemented on ASICs. We implement our design in $GF((2^4)^2)$ on TSMC-0.18 μm standard cell CMOS ASICs. We use Synopsys Design Vision, which is a GUI for Synopsys Design Compiler tools. The map effort is set to medium. We present the experimental results in Tables 2 and 3, which are extracted after place and route.

Tables 2 and 3 show that Rainbow implementation includes two affine transformations with matrix sizes 26×26 and 43×43, respectively, and 26 MQ polynomial evaluations and solving two systems of linear equations with matrix size

Table 2: Implementation Results of rainbow scheme.

Signature scheme	Message size	Signature size	Time frequency	Clock cycle	Executing time	Gate equivalents
Rainbow(17, 13, 13)	26 bytes	43 bytes	50 MHz	242	4.9 us	30000

Table 3: Executing time of the implementation in clock cycles.

Steps	Components	Clock cycles
(1)	L_1^{-1} transformation	28
(2)	The first round of 13 polynomial evaluations	65
(3)	The first round of solving system of linear equations	13
(4)	The second round of 13 polynomial evaluations	78
(5)	The second round of solving system of linear equations	13
(6)	L_2^{-1} transformation	45
	Total	242

Table 4: Comparison on public key cryptographic systems.

Signature scheme	Clock cycle	Executing time (us)	Gate equivalents	Time-area product*
RSA [5]	813	1790	107000	12.00
ECC [6]	1699	9.5	138000	32.30
UOV [26]	2300	27.7	56700	17.97
amTTS [26]	312	3.9	61600	2.65
enTTS [27]	10267	170	1000	1.42
Rainbow [28]	198	3.96	150000	4.10
SFLASH [29]	3200	1600	25900	11.42
This work	242	4.84	30000	1

*The time-area (clock cycle-gate equivalent) product of our implementations is normalized to 1.

13×13. Table 3 summarizes the performance of our implementation of Rainbow signature measured in clock cycles, which shows that our design takes only 242 clock cycles to generate a Rainbow signature. In other words, our implementation takes 4840 ns to generate a Rainbow signature with the frequency of 50 MHz. Among all of the operations, MQ polynomial evaluation occupies most of the executing time.

5. Comparisons with Other Implementations

The works in [5, 6, 26–29] are believed to be the latest RSA, ECC, and multivariate public key cryptographic systems on hardware, respectively. We compare our design with these systems, which is depicted in Table 4. Comparison results show that our design is more efficient than the related implementations.

Besides, Rainbow implementation of the work in [28] is believed to be the fastest multivariate implementation, and Rainbow implementation of the work in [27] is believed to be the smallest multivariate implementation. Thus, the implementations of the work in [28], the work in [27], and this work show that Rainbow has a good performance on hardware and is a better candidate than RSA and elliptic curves under quantum computer attacks.

6. Side Channel Attack Considerations

Cryptographic systems must be protected against a wide range of attacks, including side channel attacks. Side channel attack belongs to physical attack, which is any attack based on information gained from the physical implementation of cryptographic systems, rather than brute force or theoretical weaknesses in cryptographic algorithms. The underlying principle of side channel attack is that side channel information such as power consumption, electromagnetic leaks, timing information, or even sound can provide extra sources of information about secrets in cryptographic systems, for example, cryptographic keys, partial state information, full or partial plain texts, which can be exploited to break the cryptographic systems. General classes of side channel attack include timing analysis [31], power analysis [32], electromagnetic analysis [33], fault analysis [34], acoustic cryptanalysis [35], data remanence analysis [36], and row hammer analysis attacks [37].

Fault analysis attacks intend to manipulate the environmental conditions of cryptographic systems, such as voltage, clock, temperature, radiation, light, and eddy current, to generate faults during secret-related computations, for example, multiplications and inversions in a finite field, and

observe the related behavior, which may help a cryptanalyst break the cryptographic systems. Fault analysis attacks can be engineered by simply illuminating a transistor with a laser beam, which causes some bits to assume wrong values. The notion of using a fault induced during a secret-related computation to guess the secret key has been practically observed in implementations of the RSA that use the Chinese remainder theorem [38, 39]. A general fault analysis attack on schemes of MPKC is proposed in [40]. The work in [40] has attacked partial secret keys from affine transformations of the multivariate public key cryptographic schemes.

Power analysis attack can provide detailed information by observing the power consumption of cryptographic systems, which is roughly categorized into Simple Power Analysis (SPA) [41] and Differential Power Analysis (DPA) [32]. In the family of power analysis attacks, DPA is of particular interest and is a statistical test which examines a large number of power consumption signals to retrieve secret keys. A differential power analysis attack on SFLASH is proposed in [42]. The work in [42] has attacked secret keys from SHA-1 module of the SFLASH schemes. A side channel attack to enTTS has been proposed in [43], which uses differential power analysis and fault analysis to attack two affine transformations and central map transformation. The method in [43] shows that it can obtain all secret keys of enTTS.

Since the construction of Rainbow includes two affine transformations and central map transformation, such methods in [40, 42, 43] have the potential to obtain its secret keys. Thus, we discuss defending against a possible side channel attack for Rainbow and the countermeasure is described in the following:

(1) We suppose that $y(y_0, y_1, \ldots, y_{25})$ is the message and each element of y is in $GF((2^4)^2)$.

(2) We take a random vector $y'(y'_0, y'_1, \ldots, y'_{25})$; the elements of y' are in $GF((2^4)^2)$.

(3) We compute $y'' = y' + y$.

(4) We compute $\overline{y}' = Ay' + b$ and $\overline{y}'' = Ay''$, where A is a 26×26 matrix and b is a vector with size 26.

(5) We compute $\overline{y} = \overline{y}' + \overline{y}''$, which is equivalent to $\overline{y} = Ay + b$.

(6) The first affine transformation L_1 has been computed; then we take random bytes for Vinegar variables.

(7) We double check the random bytes to protect against fault analysis attacks.

(8) We compute the multivariate polynomial evaluations and solving systems of linear equations until the central map transformation is completed.

(9) $\overline{x}(\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{42})$ is the result of central map transformation; then we take two random vectors $\overline{x}'$ and $\overline{x}''$, where $\overline{x} = \overline{x}' + \overline{x}''$ and the elements are in $GF((2^4)^2)$.

(10) We compute $x' = C\overline{x}'$ and $x'' = C\overline{x}'' + d$, where C is a 43×43 matrix and b is a vector with size 43.

(11) We compute $x = x' + x''$, which is equivalent to $x = C\overline{x} + d$.

(12) $x(x_0, x_1, \ldots, x_{42})$ is the Rainbow signature of $y(y_0, y_1, \ldots, y_{25})$.

The work in [40] uses fault analysis to attack the random bytes in central map transformations; thus we double check the random bytes to protect against fault analysis attacks. The work in [42] uses differential power analysis to attack SHA-1 module; thus we take a method to protect affine transformations. However, the countermeasure mentioned above is theoretical; we should be able to implement and verify it on hardware.

7. Conclusions

In this paper, we present techniques to exploit Rainbow signature cryptographic systems on hardware meeting the requirements of efficient high-performance applications. We propose a general architecture for efficient hardware implementations of Rainbow and enhance our design in three directions. First, we present a fast inversion in $GF((2^4)^2)$ based on binary trees. Second, we present an efficient multiplication in $GF((2^4)^2)$ based on compact construction. Third, we present a parallel solving system of linear equations in $GF((2^4)^2)$ based on Gauss-Jordan elimination. Via further other minor optimizations and by integrating the major improvement above, we implement our design in $GF((2^4)^2)$ on TSMC-0.18 μm standard cell CMOS ASICs. We use Synopsys Design Vision and the map effort is set to medium. Our design can be generalized with minor modifications that also support FPGAs.

The experimental results show that Rainbow implementation includes two affine transformations with matrix sizes 26×26 and 43×43, respectively, and 26 MQ polynomial evaluations and solving two systems of linear equations with matrix size 13×13. Our implementation takes 4840 ns and 242 clock cycles to generate a Rainbow signature with the frequency of 50 MHz. Among all of the operations, MQ polynomial evaluation occupies most of the executing time. Comparison results show that our design is more efficient than the related implementations.

Moreover, the implementations of a fast Rainbow, a small Rainbow, and this work show that Rainbow has a good performance on hardware and is a better candidate than RSA and elliptic curves under quantum computer attacks.

Besides, Rainbow implementations must be protected against a wide range of attacks, including side channel attacks. We discuss defending against a possible side channel attack for Rainbow and we present countermeasures against fault analysis and differential power analysis attack.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

Acknowledgments

The author acknowledges Shenzhen Science and Technology Program under Grants no. JCYJ20170306144219159 and no.

JCYJ20160428092427867; Science and Technology Program of Shenzhen Polytechnic (no. 601722K20018); and Special Funds for Shenzhen Strategic Emerging Industries and Future Industrial Development (no. 20170502142224600).

References

[1] W. Diffie, W. Diffie, and M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 26, no. 1, pp. 96–99, 1983.

[3] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[4] V. S. Miller, "Use of elliptic curves in cryptography," in *Proceedings of the International Cryptology Conference (CRYPTO 85)*, 426, 417 pages, Springer-Verlag, Berlin, Germany, 1985.

[5] G. D. Sutter, J.-P. Deschamps, and J. L. Imana, "Modular multiplication and exponentiation architectures for fast RSA cryptosystem based on digit serial computation," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 7, pp. 3101–3109, 2011.

[6] G. D. Sutter, J.-P. Deschamps, and J. L. Imana, "Efficient elliptic curve point multiplication using digit-serial binary field operations," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 1, pp. 217–225, 2013.

[7] A. Cilardo, A. Mazzeo, L. Romano, and G. P. Saggese, "Exploring the design-space for FPGA-based implementation of RSA," *Microprocessors and Microsystems*, vol. 28, no. 4, pp. 183–191, 2004.

[8] O. Nibouche, M. Nibouche, A. Bouridane, and A. Belatreche, "Fast architectures for FPGA-based implementation of RSA encryption algorithm," in *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT 2005)*, pp. 271–278, Washington, DC, USA, December 2004.

[9] Pund S. M., "Implementation of RSA algorithm using mersenne prime," *International Journal of Networking and Parallel Computing*, vol. 1, pp. 33–41, 2014.

[10] Q. A. Al-Haija, M. Smadi, M. Al-Ja'fari, and A. Al-Shua'ibi, "Efficient FPGA implementation of RSA coprocessor using scalable modules," in *Proceedings of the International Symposium on Emerging Inter-networks, Communication and Mobility (EICM 2014)*, pp. 647–654, Elsevier, Amsterdam, Netherlands, 2014.

[11] K. C. C. Loi and S.-B. Ko, "High performance scalable elliptic curve cryptosystem processor for Koblitz curves," *Microprocessors and Microsystems*, vol. 37, no. 4-5, pp. 394–406, 2013.

[12] K. Sakiyama, N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede, "Reconfigurable modular arithmetic logic unit supporting high-performance RSA and ECC over GF(p)," *International Journal of Electronics*, vol. 94, no. 5, pp. 501–514, 2007.

[13] M. N. Hassan and M. Benaissa, "Small footprint implementations of scalable ECC point multiplication on FPGA," in *Proceedings of the 2010 IEEE International Conference on Communications, ICC 2010*, pp. 1–4, Washington, DC, USA, May 2010.

[14] M. Varchola, T. Güneysu, and O. Mischke, "MicroECC: A lightweight reconfigurable elliptic curve crypto-processor," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (Reconfig 2011)*, pp. 204–210, Washington, DC, USA, 2013.

[15] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[16] J. Ding, J. E. Gower, and D. S. Schmidt, *Multivariate Public Key Cryptosystems*, Springer, Berlin, Germany, 2006.

[17] D. S. Johnson, "The NP-completeness column: an ongoing guide," *Journal of Algorithms*, vol. 4, no. 1, pp. 87–100, 1983.

[18] A. Kipnis, J. Patarin, and L. Goubin, "Unbalanced oil and vinegar signature schemes," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt 99)*, vol. 1999, pp. 206–222, Springer, Berlin, Germany.

[19] J. Ding and D. Schmidt, "Rainbow, a new multivariable polynomial signature scheme," in *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS 2005)*, pp. 164–175, Springer, Berlin, Germany, 2005.

[20] A. Petzoldt, S. Bulygin, and J. Buchmann, "Selecting parameters for the rainbow signature scheme," in *Post-quantum cryptography*, vol. 6061 of *Lecture Notes in Comput. Sci.*, pp. 218–240, Springer, Berlin, 2010.

[21] B. Y. Yang and J. M. Chen, "Building secure tame-like multivariate public-key cryptosystems: the new TTS," in *Proceedings of the Australasian Conference on Information Security and Privacy (ACISP 2005)*, pp. 518–531, Springer, Berlin, Germany, 2005.

[22] E. Thomae and C. Wolf, "Cryptanalysis of enhanced TTS, STS and all its variants, or: why cross-terms are important," in *Proceedings of the International Conference on Cryptology in Africa (Africacrypt 2012)*, pp. 188–202, Springer, Berlin, Germany, 2012.

[23] T. Matsumoto and H. Imai, "Public quadratic polynomial-tuples for efficient signature-verification and message-encryption," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 88)*, pp. 419–453, Springer, Berlin, Germany, 1988.

[24] E. Thomae and C. Wolf, "Solving underdetermined systems of multivariate quadratic equations revisited," in *Proceedings of the International Conference on Practice and Theory of Public-Key Cryptography (PKC 2012)*, 171, pp. Berlin, Germany–156, Springer, 2012.

[25] T. Moh, "A public key system with signature and master key functions," *Communications in Algebra*, vol. 27, no. 5, pp. 2207–2222, 1999.

[26] A. Bogdanov, T. Eisenbarth, A. Rupp et al., "Time-area optimized public-key engines: MQ-cryptosystems as replacement for elliptic curves?" in *Proceedings of the Conference on Cryptographic Hardware and Embedded Systems (CHES 2008)*, pp. 45–61, Springer, Berlin, Germany, 2008.

[27] H. Yi and S. Tang, "Very small FPGA processor for multivariate signatures," *The Computer Journal*, vol. 59, no. 7, pp. 1091–1101, 2016.

[28] S. Tang, H. Yi, J. Ding, H. Chen, and G. Chen, "High-speed hardware implementation of rainbow signature on FPGAs," in *Proceedings of the International Workshop on Post-Quantum Cryptography (PQCrypto 2011)*, pp. 228–243, Springer, Berlin, Germany, 2011.

[29] B. Yang, C. Cheng, B. Chen, and J. Chen, "Implementing minimized multivariate PKC on low-resource embedded systems," in *Proceedings of the Security in Pervasive Computing, 3rd International Conference (SPC 2006)*, vol. 3934, pp. 73–88, Springer, Berlin, Germany, 2006.

[30] H. Yi, S. Tang, and R. Vemuri, "Fast inversions in small finite fields by using binary trees," *The Computer Journal*, vol. 59, no. 7, pp. 1102–1112, 2016.

[31] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems," in *Proceedings of the International Cryptology Conference on Advances in Cryptology (CRYPTO 96)*, pp. 104–113, Springer, Berlin, Germany, 1996.

[32] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the International Cryptology Conference on Advances in Cryptology (CRYPTO 99)*, pp. 388–397, Springer, Berlin , Germany, 1999.

[33] J. J. Quisquater and D. Samyde, "ElectroMagnetic analysis: measures and countermeasures for smart cards," in *Proceedings of the International Conference on Research in Smart Cards (E-Smart 2001)*, pp. 200–210, Springer, Berlin, Germany, 2001.

[34] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," in *Proceedings of the Conference on Cryptographic Hardware and Embedded Systems (CHES 2002)*, pp. 2–12, Springer, Berlin, Germany, 2003.

[35] D. Genkin, A. Shamir, and E. Tromer, "RSA key extraction via low-bandwidth acoustic cryptanalysis," in *Proceedings of the International Cryptology Conference (CRYPTO 2014)*, pp. 17–21, Springer, Berlin, Germany, 2014.

[36] S. Skorobogatov, "Data remanence in flash memory devices," in *Proceedings of the Conference on Cryptographic Hardware and Embedded Systems (CHES 2005)*, pp. 339–353, Springer, Berlin, Germany, 2005.

[37] D.-H. Kim, P. J. Nair, and M. K. Qureshi, "Architectural support for mitigating row hammering in DRAM memories," *Computer Architecture Letters*, vol. 14, pp. 9–12, 2015.

[38] M. Joye, A. K. Lenstra, and J.-J. Quisquater, "Chinese remaindering based cryptosystems in the presence of faults," *Journal of Cryptology*, vol. 12, no. 4, pp. 241–245, 1999.

[39] D. M. D. Boneh and R. J. Lipton, "On the importance of eliminating errors in cryptographic computations," *Journal of Cryptology*, vol. 14, pp. 101–119, 1999.

[40] Y. Hashimoto, T. Takagi, and K. Sakurai, "General fault attacks on multivariate public key cryptosystems," in *Post-quantum cryptography*, vol. 7071 of *Lecture Notes in Comput. Sci.*, pp. 1–18, Springer, Heidelberg, 2011.

[41] R. Mayer-Sommer, "Smartly analyzing the simplicity and the power of simple power analysis on smartcards," in *Proceedings of the Conference on Cryptographic Hardware and Embedded Systems (CHES 2000)*, pp. 78–92, pringer, Berlin, Germany, 2000.

[42] K. Okeya, T. Takagi, and C. Vuillaume, "On the importance of protecting δ in SFLASH against side channel attacks," in *Proceedings of the International Conference on Coding and Computing (ITCC 2004)*, pp. 560–568, Washington, DC, USA, 2004.

[43] H. Yi and W. Li, "On the Importance of Checking Multivariate Public Key Cryptography for Side-Channel Attacks: The Case of enTTS Scheme," *The Computer Journal*, pp. 1–13, 2017.