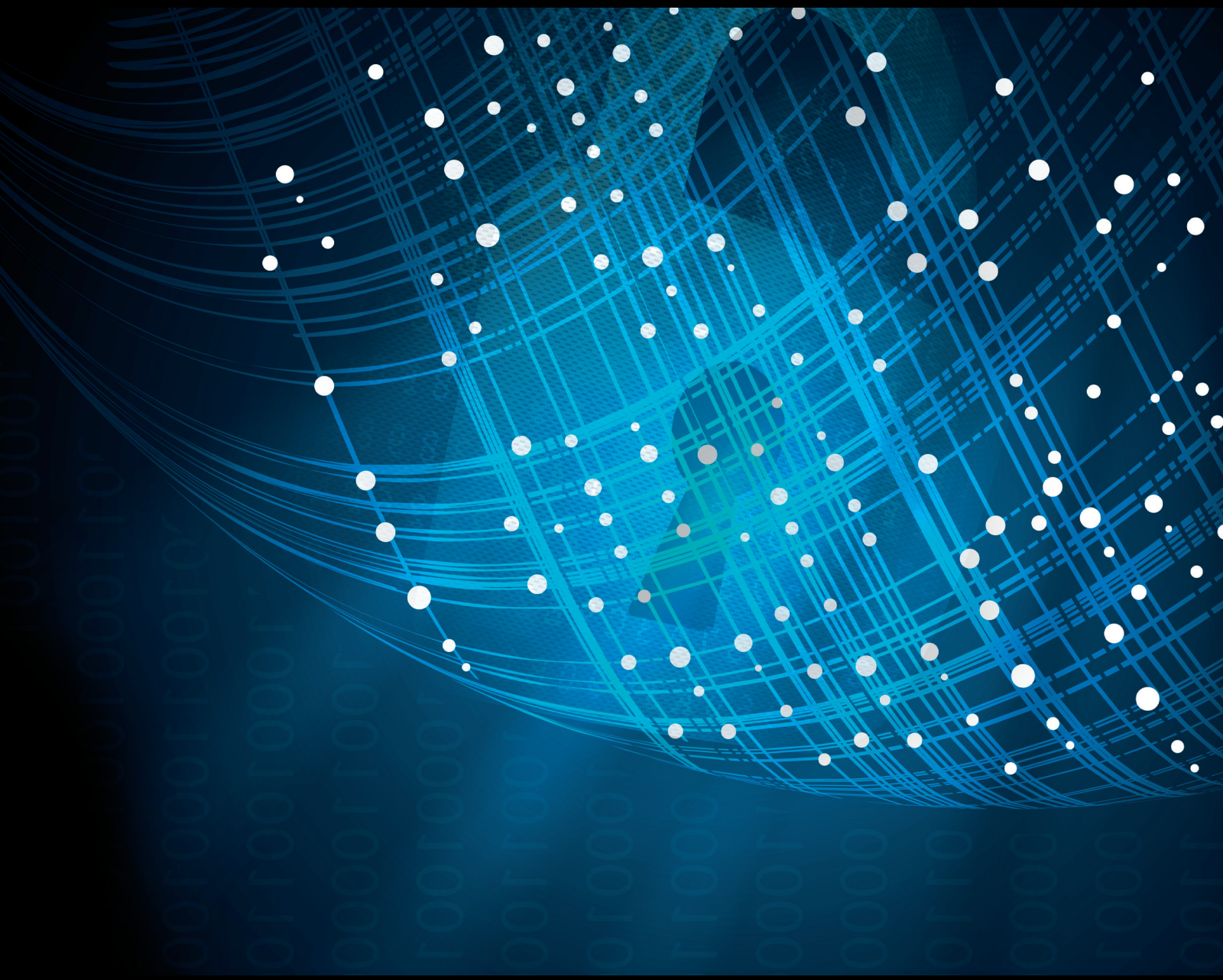
Security and Communication Networks

Cryptography and Security Tools and Techniques for Networked Embedded Systems

Lead Guest Editor: Giovanni Agosta
Guest Editors: Karine Heydemann, Nicolas Sklavos, and Leonel Sousa



Cryptography and Security Tools and Techniques for Networked Embedded Systems

Security and Communication Networks

Cryptography and Security Tools and Techniques for Networked Embedded Systems

Lead Guest Editor: Giovanni Agosta
Guest Editors: Karine Heydemann, Nicolas Sklavos, and Leonel Sousa

Contents

Algebraic Degree Estimation of ACORN v3 Using Numeric Mapping
Lin Ding, Lei Wang, Dawu Gu, Chenhui Jin, and Jie Guan
Research Article (5 pages), Article ID 7429320, Volume 2019 (2019)

Analysis of DES Plaintext Recovery Based on BP Neural Network
Sijie Fan and Yaqun Zhao
Research Article (5 pages), Article ID 9580862, Volume 2019 (2019)

An Indistinguishably Secure Function Encryption Scheme
Ping Zhang, Yamin Li, and Muhua Liu
Research Article (10 pages), Article ID 8567534, Volume 2019 (2019)

A Novel Construction of Constrained Verifiable Random Functions
Muhua Liu, Ping Zhang, and Qingtao Wu
Research Article (15 pages), Article ID 4187892, Volume 2019 (2019)

A Highly Effective Data Preprocessing in Side-Channel Attack Using Empirical Mode Decomposition
ShuaiWei Zhang, XiaoYuan Yang, Lin Chen, and Weidong Zhong
Research Article (10 pages), Article ID 6124165, Volume 2019 (2019)

Hindawi
Security and Communication Networks
Volume 2019, Article ID 7429320, 5 pages
https://doi.org/10.1155/2019/7429320

Research Article

Algebraic Degree Estimation of ACORN v3 Using Numeric Mapping

Lin Ding,[1,2] **Lei Wang**,[1,3] **Dawu Gu**,[1] **Chenhui Jin**,[2] **and Jie Guan**[2]

[1]*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*
[2]*PLA SSF Information Engineering University, Zhengzhou 450001, China*
[3]*Westone Cryptologic Research Center, Beijing 100000, China*

Correspondence should be addressed to Lin Ding; dinglin_cipher@163.com

Received 6 May 2019; Accepted 24 October 2019; Published 20 November 2019

Guest Editor: Leonel Sousa

ACORN v3 is a lightweight authenticated encryption cipher, which was selected as one of the seven finalists of CAESAR competition in March 2018. It is intended for lightweight applications (resource-constrained environments). By using the technique numeric mapping proposed at CRYPTO 2017, an efficient algorithm for algebraic degree estimation of ACORN v3 is proposed. As a result, new distinguishing attacks on 647, 649, 670, 704, and 721 initialization rounds of ACORN v3 are obtained, respectively. So far, as we know, all of our distinguishing attacks on ACORN v3 are the best. The effectiveness and accuracy of our algorithm is confirmed by the experimental results.

1. Introduction

ACORN, which is known as ACORN v1 [1], is a lightweight authenticated encryption cipher which had been submitted to the CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness) competition [2] in 2014. The structure is based on nonlinear feedback shift register. Later, with minor modifications, it was updated as ACORN v2 [3] and ACORN v3 [4] by enhancing the security. In March 2018, ACORN v3 was selected as one of seven finalists of CAESAR competition. In February 2019, ACORN v3 was listed into the final CAESAR portfolio and recommended for the use case of lightweight applications (resource constrained environments). The state size of ACORN v3 is 293 bits. It uses a 128-bit key and a 128-bit initialization vector. The initialization of ACORN v3 consists of loading the key and IV into the state and running the cipher for 1792 steps.

1.1. Previous Attacks on ACORN. In 2014, Wu had submitted an authenticated encryption cipher, known as ACORN v1 to CAESAR competition. After then, some attacks on ACORN v1 and its tweaked version ACORN v2 were presented in [5–11]. Besides these attacks, a cube attack on 477 rounds of ACORN v2 was proposed in [12] to recover the 128-bit key with a total attack complexity of 2^{35}, and when the goal is to recover one bit of the secret key, 503 rounds of ACORN v2 were attacked. Later, the authenticated encryption cipher was updated as ACORN v3 with minor modifications by enhancing the security.

Until now, several attacks on ACORN v3 have been published in [13–16]. However, there are no attacks better than exhaustive key search on ACORN v3 so far. Based on cube testers and d-monomial test, Ghafari and Hu proposed a new attack framework in [17, 18] and presented a practical distinguishing attack on 676 rounds of ACORN v3 with time complexity of 200×2^{33}. This has been the best-known distinguishing attack on the round reduced variants of ACORN v3 so far. Recently, some key recovery attacks on ACORN v3 had been proposed. At CRYPTO 2017, Todo et al. [19] proposed possible key recovery attacks on 647, 649, and 704 rounds of ACORN v3, where no more than one bit of the secret key can be recovered with unknown probability in around 2^{78}, 2^{109}, and 2^{122}, respectively. The attack was improved by Wang et al. in [20, 21].

1.2. Numeric Mapping. At CRYPTO 2017, Liu [22] exploited a new technique, called *numeric mapping*, to iteratively estimate the upper bound on the algebraic degree of the internal states of an NFSR. Based on this new technique, he developed an algorithm for estimating the algebraic degree of NFSR-based cryptosystems and gave distinguishing attacks on Trivium-like ciphers, including Trivium, Kreyvium, and TriviA-SC as applications.

1.3. Our Contributions. In this paper, we focus on proposing an efficient algorithm for algebraic degree estimation of ACORN v3. By applying our algorithm, we investigate the mixing efficiency of ACORN v3. When taking all the key and IV bits as initial input variables, the result shows that the lower bound on the maximum number of initialization rounds of ACORN v3 such that the generated keystream bit does not achieve maximum algebraic degree is 669 (out of 1792). When taking all the IV bits as input variables, the result shows that the lower bound on the maximum number of initialization rounds of ACORN v3 such that the generated keystream bit does not achieve maximum algebraic degree is 708 (out of 1792). When taking a subset of all the IV bits as initial input variables, we apply our algorithm to ACORN v3 to exploit new distinguishing attacks. Some distinguishing attacks on round reduced variants of ACORN v3 we have obtained are listed in Table 1, and comparisons with previous works are made. As shown in Table 1, our results are the best-known distinguishing attacks on the cipher so far. Note that three key recovery attacks on the cipher in [19–21] are also listed in Table 1. In these attacks, the recovered secret variables are generally smaller than 1 bit, while the time complexities are significantly high. Because of the high time complexities, these attacks are impractical and cannot be verified by experiments, and the success probabilities of key recovery are difficult to estimate as they are based on some assumptions. Compared with them, our attacks have significantly better time complexities. Meanwhile, our attacks are deterministic rather than statistical, that is, our attacks hold with probability 1.

To verify these cryptanalytic results, we make an amount of experiments on round reduced variants of ACORN v3. The experimental results show that our distinguishing attacks are always consistent with our evaluated results. They are strong evidences of high accuracy of our algorithm.

This paper is organized as follows. Some notations are defined and the technique numeric mapping is introduced in Section 2. In Section 3, algebraic degree estimation of ACORN v3 is presented. The paper is concluded in Section 4.

2. Preliminaries

2.1. Notations. Let $\mathbb{F}_2 = \{0, 1\}$ be the finite field with two elements. Denote $\mathbb{F}_2^n$ the n-dimension vector space over the binary field $\mathbb{F}_2$. Let $\mathbb{B}_n$ be the set of all n-variable Boolean functions mapping from $\mathbb{F}_2^n$ into $\mathbb{F}_2$, and let $f \in \mathbb{B}_n$. The algebraic normal form (ANF) of the given Boolean function f over n variables $x_1, x_2, \ldots, x_n$ can be uniquely expressed as

$$f(x_1, x_2, \ldots, x_n) = \bigoplus_{c=(c_1,c_2,\ldots,c_n)\in\mathbb{F}_2^n} a_c \prod_{i=1}^{n} x_i^{c_i}, \quad (1)$$

where the coefficient a_c is a constant in $\mathbb{F}_2$ and c_i denotes the i-th digit of the binary encoding of c (and so the sum spans all monomials in $x_1, x_2, \ldots, x_n$). The algebraic degree of f, denoted by $\deg(f)$, is defined as $\max\{\mathrm{wt}(c) \mid a_c = 1\}$, where $\mathrm{wt}(c)$ is the Hamming weight of c. Thus, for a multivariate Boolean function, the degree of a term is the sum of the exponents of the variables in the term, and then the algebraic degree of the multivariate Boolean function is the maximum of the degrees of all terms in the Boolean function.

2.2. Cube Attack and Cube Tester. Almost any cryptographic scheme can be described by tweakable polynomials over the binary field $\mathbb{F}_2$, which contain both secret variables (e.g., key bits) and public variables (e.g., IV bits). Cube attack, proposed by Dinur and Shamir [23] at EUROCRYPT 2009, is one of general and powerful cryptanalytic techniques against symmetric-key cryptosystems. It treats the output bit of a stream cipher as an unknown Boolean polynomial $f(k_0, \ldots, k_{n-1}, v_0, \ldots, v_{m-1})$, where $k_0, \ldots, k_{n-1}$ are secret key variables and $v_0, \ldots, v_{m-1}$ are public IV variables. Given any monomial t_I which is the product of variables in $I = \{i_1, \ldots, i_d\} \subseteq \{0, 1, \ldots, m-1\}$, f can be represented as the sum of terms which are supersets of I and terms that miss at least one variable from I:

$$\begin{aligned} &f(k_0, \ldots, k_{n-1}, v_0, \ldots, v_{m-1}) \\ &\quad = t_I \cdot p_{S(I)} + q(k_0, \ldots, k_{n-1}, v_0, \ldots, v_{m-1}), \end{aligned} \quad (2)$$

where $p_{S(I)}$ is called the *superpoly* of I in f and the set $\{v_{i_1}, \ldots, v_{i_d}\}$ is called a *cube*. The idea behind cube attacks is that the sum of the Boolean polynomial $f(k_0, \ldots, k_{n-1}, v_0, \ldots, v_{m-1})$ over the cube which contains all possible values for the cube variables is exactly $p_{S(I)}$, while this is a random function for a random polynomial. In cube attacks, low-degree superpolys in secret variables are exploited to recover the key, while cube testers [24] work by distinguishing $p_{S(I)}$ from a random function. Especially, the superpoly $p_{S(I)}$ is equal to a zero constant, if the algebraic degree of f in the variables from I is smaller than the size of I. Thus, from the perspective of cube tester, estimation on algebraic degree of NFSR-based cryptosystems is an efficient way of constructing distinguishing attacks.

2.3. Numeric Mapping. At CRYPTO 2017, Liu [22] exploited a new technique, called *numeric mapping*, to iteratively estimate the upper bound on the algebraic degree of the internal states of an NFSR. Based on this new technique, he developed an algorithm for estimating the algebraic degree of NFSR-based cryptosystems. Let $f(x_1, x_2, \ldots, x_n) = \bigoplus_{c=(c_1,c_2,\ldots,c_n)\in\mathbb{F}_2^n} a_c \prod_{i=1}^{n} x_i^{c_i}$. The numeric mapping, denoted by **DEG**, is defined as

Table 1: Attacks on ACORN v3.

Cipher	# rounds	Attack	Time complexity	Reference
ACORN v3	647	Key recovery attack	2^{78}	[19]
	647	**Distinguishing attack**	2^{21}	**Sect. 4.3**
	649	Key recovery attack	2^{109}	[19]
	649	**Distinguishing attack**	2^{24}	**Sect. 4.3**
	676	Distinguishing attack	$200 \times 2^{33} \approx 2^{40.64}$	[17]
	676	**Distinguishing attack**	2^{36}	**Sect. 4.3**
	704	Key recovery attack	2^{122}	[19]
	704	Key recovery attack	$2^{77.88}$	[20]
	704	**Distinguishing attack**	2^{61}	**Sect. 4.3**
	721	**Distinguishing attack**	2^{95}	**Sect. 4.3**
	750	Key recovery attack	$2^{125.71}$	[21]
	750	Key recovery attack	$2^{120.92}$	[20]

$$\mathbf{DEG} : \mathbb{B}_n \times \mathbb{Z}^n \longrightarrow \mathbb{Z},$$
$$(f, D) \longmapsto \max_{a_c \neq 0} \left\{ \sum_{i=1}^{n} c_i d_i \right\}, \tag{3}$$

where $D = (d_1, d_2, \ldots, d_n)$, a_c's are coefficients of algebraic normal form of f as defined previously, and denote $\mathbb{Z}^n$ the n-dimension vector space over the integer field $\mathbb{Z}$. Let $g_i\,(1 \le i \le m)$ be Boolean functions on n variables and denote $\deg(G) = (\deg(g_1), \deg(g_2), \ldots, \deg(g_m))$ for $G = (g_1, g_2, \ldots, g_m)$. We call $\mathbf{DEG}(f, D)$ a *numeric degree* of h if $d_i \ge \deg(g_i)$ for all $1 \le i \le n$, where $D = (d_1, d_2, \ldots, d_n)$. The algebraic degree of h is always less than or equal to the numeric degree of h. The algebraic degrees of the output bits with respect to the internal states can be estimated iteratively for NFSR-based cryptosystems by using numeric mapping.

3. Algebraic Degree Estimation of ACORN v3

In this section, we first briefly give a description of ACORN v3 and then propose an efficient algorithm for algebraic degree estimation of ACORN v3 to exploit new distinguishing attacks on it.

3.1. Brief Description of ACORN v3. This section presents a brief description of the authenticated encryption cipher ACORN v3. The structure of ACORN v3 is shown in Figure 1. The state size of ACORN v3 is 293 bits, denoted by $S^{(t)} = (s_0^{(t)}, s_1^{(t)}, \ldots, s_{292}^{(t)})$ at t-th clock. It is constructed by using 6 LFSRs of different lengths 61, 46, 47, 39, 37, and 59 and one additional register of length 4. It supports a 128-bit key and a 128-bit initialization vector. As an authenticated encryption scheme, ACORN v3 passes through 4 procedures: initialization, processing the associated data, encryption, and finalization. In this paper, we only focus on the process of initialization, since the number of rounds we can attack is smaller than the 1792 initialization rounds. For more details about ACORN v3, we refer to [4].

The initialization of the authenticated encryption cipher ACORN v3 consists of loading the 128-bit key $(k_0, k_1, \ldots, k_{127})$ and 128-bit IV $(iv_0, iv_1, \ldots, iv_{127})$ into the state and running the cipher for 1792 steps.

(1) Initialize the state S_{-1792} to 0

(2) Let $m_{(-1792+t)} = k_t$ for $t = 0$ to 127
Let $m_{(-1792+128+t)} = iv_t$ for $t = 0$ to 127
Let $m_{(-1792+256)} = k_{(t \bmod 128)} \oplus 1$ for $t = 0$
Let $m_{(-1792+256+t)} = k_{(t \bmod 128)}$ for $t = 1$ to 1535

(3) For $t = -1792$ to $t = -1$, $S^{(t+1)} = \text{StateUpdate}\,128\,(S^{(t)}, m_t, ca_t, cb_t)$

At t-th clock, the cipher executes the state update function: $S^{(t+1)} = \text{State} - \text{Update}\,128\,(S^{(t)}, m_t, ca_t, cb_t)$, which is given as follows:

Step 1. Linear feedback update

$$s_{t,289} \longleftarrow s_{t,289} \oplus s_{t,235} \oplus s_{t,230}$$
$$s_{t,230} \longleftarrow s_{t,230} \oplus s_{t,196} \oplus s_{t,193}$$
$$s_{t,193} \longleftarrow s_{t,193} \oplus s_{t,160} \oplus s_{t,154}$$
$$s_{t,154} \longleftarrow s_{t,154} \oplus s_{t,111} \oplus s_{t,107}$$
$$s_{t,107} \longleftarrow s_{t,107} \oplus s_{t,66} \oplus s_{t,61}$$
$$s_{t,61} \longleftarrow s_{t,61} \oplus s_{t,23} \oplus s_{t,0}$$

Step 2. Generate keystream bit

$$z_t \longleftarrow s_{t,12} \oplus s_t\ , 154 \oplus s_{t,235} \cdot s_{t,61} \oplus s_{t,235} \cdot s_{t,193} \oplus s_{t,61} \cdot s_{t,193} \oplus\ s_{t,230} \cdot s_{t,111} \oplus (s_{t,230} \oplus 1) \cdot s_{t,66}$$

Step 3. Generate the nonlinear feedback bit

$$f_t \longleftarrow s_{t,0} \oplus s_{t,107} \oplus 1 \oplus s_{t,244} \cdot s_{t,23} \oplus s_{t,244} \cdot s_{t,160} \oplus\ s_{t,23} \cdot s_{t,160} \oplus s_{t,230} \oplus z_t$$

Step 4. Shift the 293-bit register with the feedback bit f_t

$$s_{t+1,i} \longleftarrow s_{t,i+1} \text{ for } i = 0, 1, \ldots, 291$$
$$s_{t+1,292} \longleftarrow f_t \oplus m_t$$

3.2. Algorithm for Algebraic Degree Estimation of ACORN v3. In this section, we will propose an efficient algorithm for algebraic degree estimation of ACORN v3 using numeric mapping, as depicted in Algorithm 1.

Algorithm 1 gives a numeric degree $\mathbf{DEG}(f, X)$ of the output function f after N rounds over initial input variables $X = (x_1, x_2, \ldots, x_{128})$ as output, which gives an upper bound on the algebraic degree of the first output bit after N rounds.

The time complexity of Algorithm 1 mainly depends on the values of N and the ANFs of the update function

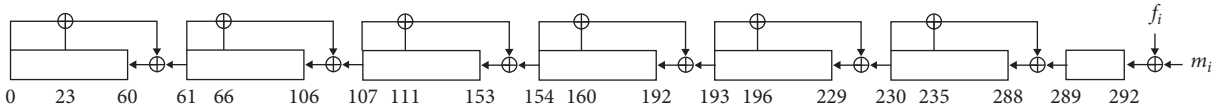


Figure 1: The structure of authenticated encryption cipher ACORN v3.

Require: Given the ANFs of the initial state $S^{(0)} = \{s_{0,0}, s_{0,1}, \ldots, s_{0,292}\}$, the ANFs of the update function State Update 128 and the keystream output function f, and the set of initial input variables $X = (x_1, x_2, \ldots, x_{128})$.
(1) Set $D^{(0)}$ to $\deg(S^{(0)}, X)$;
(2) For t from 1 to N do:
(3) $\quad D^{(t)} \longleftarrow \mathbf{DEG}(\text{State Update 128}(S^{(t-1)}), D^{(t-1)})$;
(4) Compute $\mathbf{DEG}(f, D^{(N)})$;
(5) Return $\mathbf{DEG}(f, D^{(N)})$.

Algorithm 1: Algebraic degree estimation of ACORN v3 using numeric mapping.

State Update 128. Since all of the update function State Update 128 are shifting operations except one quadratic function and six linear functions, Algorithm 1 has a time complexity of $\mathcal{O}(N)$. Algorithm 1 requires to store $D^{(t)}$ for $t = 1, 2, \ldots, N$. Since the number of initial input variables is constant for ACORN v3, it leads to a negligible memory complexity of $\mathcal{O}(N)$.

3.3. Experimental Results. By using Algorithm 1, we will investigate the mixing efficiency of ACORN v3 and exploit new distinguishing attacks on the cipher.

3.3.1. When Will the Initial Input Variables Be Sufficiently Mixed? By applying Algorithm 1, we investigate the mixing efficiency of ACORN v3. When taking all the key and IV bits as initial input variables, the result shows that the maximum number of initialization rounds of ACORN v3 such that the generated keystream bit does not achieve maximum algebraic degree is at least 669 (out of 1792). When taking all the IV bits as input variables, the result shows that the maximum number of initialization rounds of ACORN v3 such that the generated keystream bit does not achieve maximum algebraic degree is at least 708 (out of 1792). The results are listed in Table 2. Note that both of these two results are lower bounds on the maximum number of initialization rounds of ACORN v3 such that the generated keystream bit does not achieve maximum algebraic degree. In other words, the true maximum numbers of initialization rounds which do not achieve maximum algebraic degree could be higher.

Furthermore, we also take a subset of IV bits as initial input variables X and apply Algorithm 1 to ACORN v3. Since the IV bits are sequentially loaded into the internal state in the second 128 initialization rounds, it is a natural and reasonable idea that we select the latter IV bits into the cube. We consider an exhaustive search on the subset $\{iv_p, iv_{p+1}, \ldots, iv_{127}\}$ of all 128 IV bits for all $1 \le p \le 127$. Some results we have found are listed in Table 3. All these results are obtained on a common PC with 2.5 GHz Intel Pentium 4 processor within one second. In Table 3, the cube size d means that the cube $\{iv_{128-d}, iv_{128-(d-1)}, \ldots, iv_{127}\}$ is used in our attack. As for 676 rounds of ACORN v3, when $d = 36$, the best result $\mathbf{DEG}(f, X) = 35$ is found, which leads to a practical distinguishing attack on it with time complexity of 2^{36} and improves the previous distinguishing attack [17] by a factor of $2^{4.64}$. Furthermore, the distinguishing advantage of our attack is 1, while the attack of [17] is based on limited chi-square statistical test and its distinguishing advantage is certainly smaller than 1. As for 721 rounds of ACORN v3, when $d = 95$, the best result $\mathbf{DEG}(f, X) = 94$ is found, which leads to a distinguishing attack on it with time complexity of 2^{95}. This is the best result we have found. Clearly, our results are the best distinguishing attacks on round reduced variants of ACORN v3 so far. Note that all our attacks are deterministic rather than statistical, that is, our attacks hold with probability 1.

Table 2: Lower bound on the maximum number of rounds of not achieving maximum degree for ACORN v3 with initial input variables X.

Cipher	$X = (K, \text{IV})$		$X = \text{IV}$	
	(# key + # IV)	# rounds	# IV	# rounds
ACORN v3	256	669	128	708

Table 3: Our distinguishing attacks on round reduced variants of ACORN v3.

# rounds	Size of cube d	Cube	Time complexity
647	21	$\{iv_{107}, iv_{108}, \ldots, iv_{127}\}$	2^{21}
649	24	$\{iv_{104}, iv_{105}, \ldots, iv_{127}\}$	2^{24}
676	36	$\{iv_{92}, iv_{93}, \ldots, iv_{127}\}$	2^{36}
704	61	$\{iv_{67}, iv_{68}, \ldots, iv_{127}\}$	2^{61}
721	95	$\{iv_{33}, iv_{34}, \ldots, iv_{127}\}$	2^{95}

3.3.2. Experiments. Since 2^{21}, 2^{24}, and 2^{36} in Table 3 are practical, we verify these results by carrying out a test for random 100 keys within half a day on a common PC with 2.5 GHz Intel Pentium 4 processor. All outputs of 647, 649, and 670 rounds of

ACORN v3 over the cubes $\{iv_{107}, iv_{108}, \ldots, iv_{127}\}$, $\{iv_{104}, iv_{105}, \ldots, iv_{127}\}$ and $\{iv_{93}, iv_{94}, \ldots, iv_{127}\}$, respectively, always sum to 0. This clearly confirms the effectiveness and accuracy of our algorithm.

4. Conclusions

In this paper, we focus on proposing an efficient algorithm for algebraic degree estimation of ACORN v3. By applying our algorithm, we investigate the mixing efficiency of ACORN v3 and exploit distinguishing attacks on it. As a result, new distinguishing attacks on 647, 649, 670, 704, and 721 initialization rounds of ACORN v3 are obtained, respectively. So far as we know, all of our distinguishing attacks on ACORN v3 are the best. The effectiveness and accuracy of our algorithm is confirmed by the experimental results.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants 61602514, 61802437, 61272488, 61202491, 61572516, 61272041, and 61772547, National Cryptography Development Fund under Grant MMJJ20170125, and National Postdoctoral Program for Innovative Talents under Grant BX201700153.

References

[1] H. Wu, "ACORN: a lightweight authenticated cipher (v1). caesar first round submission," 2014, http://competitions.cr.yp.to/round1/acornv1.pdf.

[2] Caesar, "Competition for authenticated encryption: security, applicability, and robustness," http://competitions.cr.yp.to/index.html.

[3] H. Wu, "ACORN: a lightweight authenticated cipher (v2). caesar second round submission," 2015, http://competitions.cr.yp.to/round2/acornv2.pdf.

[4] H. Wu, "ACORN: a lightweight authenticated cipher (V3). CAESAR submission," 2016, http://competitions.cr.yp.to/round3/acornv3.pdf.

[5] L. Jiao, B. Zhang, and M. Wang, "Two generic methods of analyzing stream ciphers," in *ISC 2015. LNCS*, J. Lopez and C. J. Mitchell, Eds., vol. 9290, pp. 379–396, Springer, Cham, Switzerland, 2015.

[6] M. I. Salam, K. K. H. Wong, H. Bartlett, L. Simpson, E. Dawson, and J. Pieprzyk, "Finding state collisions in the authenticated encryption stream cipher acorn," Cryptology ePrint Archive, Report 2015/908, 2015, https://eprint.iacr.org/2015/908.

[7] F. Lafitte, L. Lerman, O. Markowitch, and D. Van Heule, "SAT-based cryptanalysis of ACORN," Cryptology ePrint Archive, Report 2016/521, 2016, https://eprint.iacr.org/2016/521.

[8] D. Roy and S. Mukhopadhyay, "Some results on ACORN," Cryptology ePrintArchive, Report 2016/1132, 2016, https://eprint.iacr.org/2016/1132.

[9] P. Dey, R. S. Rohit, and A. Adhikari, "Full key recovery of ACORN with a single fault," *Journal of Information Security and Applications*, vol. 29, pp. 57–64, 2016.

[10] D. K. Dalai and D. Roy, "A state recovery attack on ACORN-v1 and ACORN-v2," in *NSS 2017, LNCS*, Z. Yan, Ed., vol. 10394, pp. 332–345, Springer, Finland, 2017.

[11] X. Zhang, X. Feng, and D. Lin, "Fault attack on the authenticated cipher ACORN v2," *Security and Communication Networks*, vol. 2017, Article ID 3834685, 16 pages, 2017.

[12] M. I. Salam, H. Bartlett, E. Dawson, J. Pieprzyk, L. Simpson, and K. K.-H. Wong, "Investigating cube attacks on the authenticated encryption stream cipher ACORN," in *ATIS 2016. CCIS*, L. Batten and G. Li, Eds., vol. 651, pp. 15–26, Springer, Singapore, 2016.

[13] A. A. Siddhanti, S. Maitra, and N. Sinha, "Certain observations on ACORN v3 and the implications to TMDTO attacks," in *Space 2017. LNCS*, S. Ali, JL. Danger, and T. Eisenbarth, Eds., vol. 10662, pp. 264–280, Springer, Cham, Switzerland, 2017.

[14] X. Zhang and D. Lin, "Cryptanalysis of acorn in nonce-reuse setting," in *Inscrypt 2017. LNCS*, X. Chen, D. Lin, and M. Yung, Eds., vol. 10726, pp. 342–361, Springer, Cham, Switzerland, 2017.

[15] X. Zhang, X. Feng, and D. Lin, "Fault attack on ACORN v3," *The Computer Journal*, vol. 61, no. 8, pp. 1166–1179, 2018.

[16] A. Adomnicai, L. Masson, and J. J. A. Fournier, "Practical algebraic side-channel attacks against ACORN," in *ICISC 2018. LNCS*, K. Lee, Ed., vol. 11396, pp. 325–340, Springer, Cham, Switzerland, 2018.

[17] V. A. Ghafari and H. Hu, "A new chosen IV statistical distinguishing framework to attack symmetric ciphers, and its application to ACORN-v3 and Grain-128a," Cryptology ePrint Archive, Report 2017/1103, 2017, https://eprint.iacr.org/2017/1103.pdf.

[18] V. A. Ghafari and H. Hu, "A new chosen IV statistical distinguishing framework to attack symmetric ciphers, and its application to ACORN-v3 and Grain-128a," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 6, pp. 2393–2400, 2018, https://doi.org/10.1007/s12652-018-0897-x.

[19] Y. Todo, T. Isobe, Y. Hao, and W. Meier, "Cube attacks on non-blackbox polynomials based on division property," in *Crypto 2017. LNCS*, J. Katz and H. Shacham, Eds., vol. 10403, pp. 250–279, Springer, Cham, Switzerland, 2017.

[20] Q. Wang, Y. Hao, Y. Todo, C. Li, T. Isobe, and W. Meier, "Improved division property based cube attacks exploiting algebraic properties of superpoly," in *CRYPTO 2018, LNCS*, H. Shacham and A. Boldyreva, Eds., vol. 10991, pp. 275–305, Springer, Cham, Switzerland, 2018.

[21] Q. Wang, Y. Hao, Y. Todo, C. Li, T. Isobe, and W. Meier, "Improved division property based cube attacks exploiting algebraic properties of superpoly (full version)," Cryptology ePrint Archive, Report 2017/1063, 2017, https://eprint.iacr.org/2017/1063.

[22] M. Liu, "Degree evaluation of NFSR-based cryptosystems," in *CRYPTO 2017. LNCS*, J. Katz and H. Shacham, Eds., vol. 10403, pp. 227–249, Springer, Cham, 2017.

[23] I. Dinur and A. Shamir, "Cube attacks on tweakable black box polynomials," in *Eurocrypt 2009. LNCS*, A. Joux, Ed., vol. 5479, pp. 278–299, Springer, Heidelberg, Germany, 2009.

[24] J.-P. Aumasson, I. Dinur, W. Meier, and A. Shamir, "Cube testers and key recovery attacks on reduced-round MD6 and trivium," in *FSE 2009. LNCS*, O. Dunkelman, Ed., vol. 5665, pp. 1–22, Springer, Heidelberg, Germany, 2009.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 9580862, 5 pages
https://doi.org/10.1155/2019/9580862

Research Article

Analysis of DES Plaintext Recovery Based on BP Neural Network

Sijie Fan[1,2] **and Yaqun Zhao**[1,2]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China*
[2]*Information Engineering University, Zhengzhou, China*

Correspondence should be addressed to Sijie Fan; 826733148@qq.com

Received 16 May 2019; Accepted 22 October 2019; Published 11 November 2019

Guest Editor: Leonel Sousa

Backpropagation neural network algorithms are one of the most widely used algorithms in the current neural network algorithm. It uses the output error rate to estimate the error rate of the direct front layer of the output layer, so that we can get the error rate of each layer through the layer-by-layer backpropagation. The purpose of this paper is to simulate the decryption process of DES with backpropagation algorithm. By inputting a large number of plaintext and ciphertext pairs, a neural network simulator for the decryption of the target cipher is constructed, and the ciphertext given is decrypted. In this paper, how to modify the backpropagation neural network classifier and apply it to the process of building the regression analysis model is introduced in detail. The experimental results show that the final result of restoring plaintext of the neural network model built in this paper is ideal, and the fitting rate is higher than 90% compared with the true plaintext.

1. Introduction

The study of cryptography mainly includes two aspects, cryptographic design and cryptanalysis. There are independent and mutually unified relationships between them [1]. Block cipher is an important branch of symmetric cryptography. It uses the same key in encryption and decryption and plays a very important role in information and communication security. We hope to use existing cryptanalysis methods to design cryptanalysis methods that can resist all cryptanalysis methods. At the same time, we also hope to use the updated cryptanalysis methods to find some security flaws in cryptanalysis algorithms.

Modern cryptosystems often use methods to expand the key space or increase the complexity of encryption and decryption, and use some mathematical problems as the theoretical basis, which greatly improves the requirement of computational power in cryptographic deciphering. Sometimes the cost of traditional cryptographic deciphering methods may exceed the value of cryptographic deciphering. Artificial neural network (ANN) is the same discipline as cryptography for studying information processing. Neural network algorithm has the characteristics of nonlinear massively parallel-distributed processing and has strong high-speed information processing and uncertainty information processing capability. Using neural networks to solve cryptographic problems will provide a new research idea for cryptography.

In 2008, Bafghi et al. used a recurrent neural network to solve the problem of finding the least-weight multibranch path between two known nodes in the differential operation graph of block cipher. The main idea was to minimize the loss function of the neural network [2]. In 2010, Alallayah et al. considered the black box characteristics of neural networks, combined with system identification technology and adaptive system technology, simulated the neural model of the cryptanalysis target system and could guess the key from a given ciphertext [3]. In 2012, Alani et al. used a new cryptanalysis attack on DES (Data Encryption Standard) and 3DES (Triple Data Encryption Algorithm) cryptographic algorithms. The attack implemented was a known plaintext attack based on a neural network. In this attack, they trained a neural network to retrieve plaintext from ciphertext without retrieving the keys used in encryption. Compared with other attacks, this method reduced the number of known plaintexts required and reduced the time required to

perform a full attack [4]. The above methods were less able to directly restore the plaintext sequence, and the experimental procedures of the related literature were mostly based on the simplified cryptographic encryption algorithm and had very high requirements on the computing power.

In this paper, we choose DES algorithm as a case study of block cipher. We propose to use BP (backpropagation) neural network algorithm to simulate the mapping relationship between ciphertext and plaintext. The ciphertext obtained by DES encryption is converted into binary string, which is fed to our improved BP neural network as input after processing according to the preprocessing method defined in this paper. The difference between predicted output and true plaintext is compared for the purpose of cryptanalysis. Compared with previous work, the plaintext recovered by this experiment has a better fitting effect with true plaintext. According to the error rate defined in this paper, the experimental error rate can be controlled below 10%.

The second section of this paper briefly introduces the development history and basic working principle of block cipher. The third section briefly introduces the principle of BP algorithm and the modification we have made to it, thus successfully building a regression model. The fourth section shows our experimental process and results.

2. Brief Introduction to Block Ciphers

Block cipher is one of the important systems in modern cryptography, which is an important part of many cryptosystems. Block cipher usually refers to a kind of cipher algorithm that can only deal with a piece of data of a certain length at a time. Here, the "piece" is called a block. The number of bits in a block is called the block length. Specifically, the principle of block cipher is to divide the plaintext message sequence into a group $(m_1, m_2, \ldots, m_n)$, $(m_{n+1}, m_{n+2}, \ldots, m_{2n}), \ldots$ encrypts it according to a set of fixed encryption algorithms under the control of the key $K = k_1, k_2, \ldots, k_m$, and outputs a group of ciphertext $(c_1, c_2, \ldots, c_n)$, $(c_{n+1}, c_{n+2}, \ldots, c_{2n}), \ldots$. The model is shown in Figure 1.

Under the same key, the block cipher transforms the input plaintext group with length i equally, so it only needs to study the transformation rules for any group [5].

A cryptosystem consists of five parts (plaintext P, ciphertext C, key K, encryption transformation E, and decryption transformation D). It satisfies the following conditions [6]:

(1) $P = \{p_1, p_2, \ldots, p_n\}$ is a limited set of plaintext

(2) $C = \{c_1, c_2, \ldots, c_n\}$ is a limited set of ciphertext

(3) $K = \{k_1, k_2, \ldots, k_m\}$ is a limited set of keys

(4) $E = \{e_1, e_2, \ldots, e_n\}$ is a limited set of encryption change rules

(5) $D = \{d_1, d_2, \ldots, d_n\}$ is a limited set of decryption change rules

(6) $\forall k \in K, \exists e_k \in E, d_k \in D, \text{s.t.}, d_k(e_k(p)) = p$, $(\forall p \in P), e_k : P \longrightarrow C, d_k : C \longrightarrow P$

DES is a method of encrypting 64-bit plaintext m by 16 rounds of encryption processing with 56-bit key and obtaining 64-bit ciphertext. We choose DES as the block cipher for research because it can change the encryption key and network level faster, encrypting at a faster rate and reducing the impact of other factors. The specific description is as follows:

(1) Enter 64-bit plaintext and perform initial replacement IP

(2) Divide the plaintext into two parts, each part of 32 bits, which are represented by L_0 and R_0, respectively

(3) After adding the key, perform 16 rounds of operation f, $f : \{0,1\}^{32} \times \{0,1\}^{48} \longrightarrow \{0,1\}^{32}$

(4) After 16 rounds, the left and right bit strings are exchanged and then connected for inverse replacement

(5) Output 64-bit ciphertext

3. Backpropagation Neural Network

Artificial neural network is a cross-disciplinary field of multidisciplinary research in brain science, neuropsychology and information science. It is a research hotspot in high-tech fields in recent years. Its research goal is to explore the mystery of human intelligence by studying the composition mechanism and thinking mode of the human brain and then to make the machine have human-like intelligence by simulating the structure and working methods of the human brain [7].

BP (backpropagation) neural network usually refers to the multilayer forward neural network based on error rate backpropagation algorithm, and the error rate backpropagation algorithm is the most successful neural network learning algorithm to date. It uses the error rate after output to estimate the error rate of the direct predecessor layer of the output layer and then uses this error rate to estimate the error rate of the previous layer. After such a layer of backpropagation, the error rate estimates of all other layers are obtained [8]. The BP neural network topology includes an input layer, a hidden layer, and an output layer. The model is shown in the following Figure 2.

The BP neural network model is often used for classification. It has high self-learning, self-adaptive, and fault-tolerant ability. That is to say, BP neural network can simulate the mapping relationship between input and output through continuous learning, and this process is reflected in the dynamic adjustment of network weights and thresholds. After repeated training, the error rate is stable in an acceptable range. At this time, the corresponding network parameters can be finally determined to achieve local optimum. If the local nerve unit of BP neural network is damaged, it has little effect on the global training results [9].

Based on the above work, we modify a classifier based on BP algorithm and realize the regression model of BP neural algorithm. A large number of plaintext pairs are fed into the model to get the difference between the output plaintext and the true plaintext. The modified model is as follows.

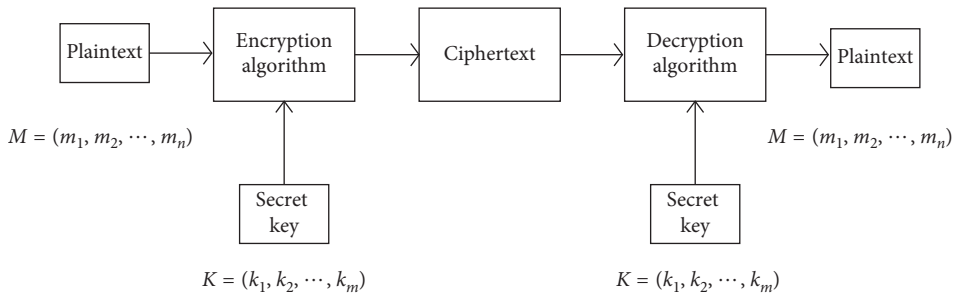


Figure 1: Block cipher workflow.

3.1. Forward Propagation

$$\begin{aligned} &\text{Input Layer}: a_0 = x, \\ &\text{Layer 1}: z = \sigma(w_1 x + b_1), \\ &\text{Layer 2}: y = \sigma(w_2 z + b_2). \end{aligned} \tag{1}$$

3.2. Backpropagation

$$\text{Loss function}: L = \frac{1}{2}[f(x) - y]^2,$$

$$\text{Layer 2}: y = k_2, \quad k_2$$

$$\text{Error} = \delta_2$$

$$\begin{aligned} \frac{\partial L}{\partial b_2} &= \frac{\partial L}{\partial k_2}\frac{\partial k_2}{\partial b_2} \\ \frac{\partial L}{\partial w_2} &= \frac{\partial L}{\partial k_2}\frac{\partial k_2}{\partial w_2} \end{aligned} \tag{2}$$

$$\text{Layer 1}:$$

$$\begin{aligned} \text{Error} = \delta_1 &= \frac{\partial L}{\partial k_1} \\ &= \frac{\partial L}{\partial k_2}\frac{\partial k_2}{\partial z}\frac{\partial z}{\partial k_1} \\ &= (y - f(x)) w_2 \odot \sigma'(z) \odot \sigma'(k_1), \end{aligned}$$

where $k_1 = w_1 x + b_1$ and $\odot$ represents the multiplication of the corresponding position of the matrix.

$$\begin{aligned} \frac{\partial L}{\partial b_1} &= \frac{\partial L}{\partial k_1}\frac{\partial k_2}{\partial b_1} = \delta_1, \\ \frac{\partial L}{\partial w_1} &= \frac{\partial L}{\partial k_1}\frac{\partial k_1}{\partial w_1} = \delta_1 x. \end{aligned} \tag{3}$$

4. Experimental Process

We use DES blocks with variable plaintext and constant keys to convert variable plain text to binary text, which are stored in different document texts. Each document has a 3.2 million-bit binary number. We use it as plaintext, after DES (electronic codebook mode) encryption, the corresponding cipher is obtained. In the data collection phase, data from the California Institute of Technology Caltech-256 dataset are selected, and the data into 1001 files are stitched as plaintext, and the size of files are 512 KB, ten of them were selected as our experimental data [10]. The intercepted image of the plaintext file and the ciphertext file obtained by encrypting them are shown respectively in Figures 3 and 4.

Since the single plaintext document is too large and the computer computing power is limited, we do some of the same processing for each ciphertext text and compress the ciphertext structure. We take an 8-bit binary number from start to finish in turn to convert it to a decimal number, so that each ciphertext becomes a 100000×1 matrix. Similarly, for each plaintext, we take a 8-bit binary number from start to finish in turn to convert it to a decimal number, so that each plaintext becomes a 100000×1 matrix. Then we input all the samples of the preprocessed ciphertext and the plaintext into the modified BP neural network and obtain the output. We compare this output with the expected plaintext to make the BP neural network a simulated decryption system. Convert each decimal digit of the output of the neural network into binary. If the number of digits is not enough, the high digits are filled with 0 and then all of them are connected to restore the plaintext effect.

We define the mean squared error of the output matrix and the processed matrix representing the true plaintext as the evaluation criteria for the experimental effect. That is, the output matrix of the modified BP neural network is $A' = (a'_{ij})$, the processed plaintext text matrix is $A = (a_{ij})$, and the evaluation criterion is

$$\text{error} = \sum \frac{\left(a_{ij} - a'_{ij}\right)^2}{n}. \tag{4}$$

The experimental process is as follows.

4.1. Input Plaintext. The known plaintext (binary text) is encrypted according to des (ecb mode) to obtain the corresponding ciphertext, and the known ciphertext is processed and converted into a format that can be fed into the neural network;

4.2. Neural Network Training. Modify the BP neural network model, change it from the classification model to the

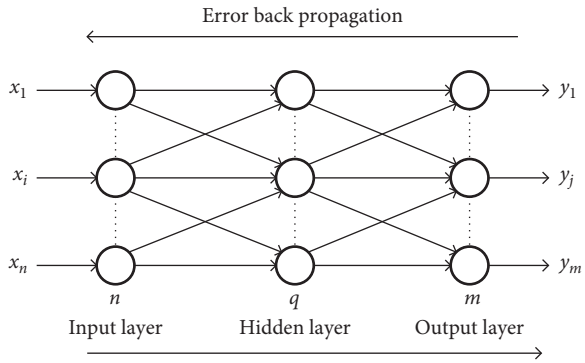


Figure 2: Backpropagation neural network model.

```
00010001101111011000000111011111011111000000001100000000010000
00001000000000010000000100000010100010111101110000011000000100
00000010101101100100111110100000011011010111001000111000101000
10011100010111101110000000110000011010000101000000010000100011
01111011000000111011111011111000000001100000000001000000010000
00000010000000100000010000010111011101100000110000000100000001
01011010101010001011010000001100010100000100011111010100111111
00000000100001011011011000001100000100100000010000000101101110
```

Figure 3: Plaintext sample.

```
00011001101000000010011011000101001001011001010001001000111010 1
01001111111011001110011111100011010110111111001001000110111000 0
11011111001001100011101110001001110100011110110110011111101000 1
10101100001001101010001110111011111111101010001110000000100001 0
11010011101011111001110111000011110001001011101111101111101010 11
00000001000000001010110101010100010111010000001100010100000100 0
11111010100111111000000001000010111011110010011000111011100010 0
```

Figure 4: Ciphertext sample.

regression model, and constantly adjust the internal parameters of the neural network until the best training effect is achieved.

4.3. Output Result. The experimental error rate is stable at an acceptable level, and the improved model is used to achieve plaintext recovery.

5. Results and Discussion

After repeated training, the parameters of BP neural network are constantly adjusted. We used the sigmoid activation function, and the number of trainings exceeds 1000. The error rate can be stabilized at about 10%. The experimental results are shown in the following Table 1:

The correlation between data is too large, which will inevitably lead to unreliable and unpredictable networks. Therefore, our input is encrypted by DES blocks of constant keys, but special attention should be paid to the number of trainings for controlling neural networks. Overtraining the network, the network may become over-fitting [11, 12], so it may not be possible to accurately predict the plaintext outside the training set, resulting in an excessive error rate.

Table 1: Experimental results.

Number of experiments	Error rate
1	0.3632983
2	0.3375012
3	0.2215154
.	.
.	.
.	.
998	0.1022956
999	0.1022163
1000	0.1021363

The modified BP neural network optimizes the weight in backpropagation by the steepest descent method, which converges straight down to the local minimum point in the weight space. However, in addition to trying several different weight initial values, there is no better suggestion than the difference in the output of the neural network [13]. Correct selection of the learning rate effectively controls the size of the step size used to modify each weight in the multidimensional weight space [14]. If the selected learning rate is too large, the local minimum may be continually overrun, causing oscillations and slowly converge to a lower error rate. If the learning rate is too low, the number of iterations required may be too large, resulting in a slow neural network performance [15].

6. Conclusion

In this experiment, we propose to apply the modified BP neural network algorithm to cryptanalysis and implement it. Here, we define mean-square error for analyzing output. Efficiency can be further improved by increasing the number of samples used to train the neural network and adjusting the weights and biases of the neurons in each layer.

Although the DES algorithm is no longer used in new commercial and public applications, the reason why we choose the DES algorithm for cryptanalysis is that the design structure of the DES algorithm is also reflected in other cryptographic algorithms, such as the gost algorithm and the camella algorithm. In addition, many software still compatible with DES algorithm, because there is no real way to completely crack DES algorithm.

In the future, a lot of work on weight selection and adaptation (training) of neural networks still needs to be completed, especially the possibility of hardware implementation is still an area worthy of further study. There may be different types of neural networks for cryptanalysis [16], resulting in unexpected results.

Data Availability

The data used to support the findings of this study are included within the article. Data can be used for free by everyone to verify the experimental results.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by National Key Research and Development Project 2016–2018 (2016YFE0100600), State Key Laboratory of Information Assurance Technology Open Fund Project (KJ-15-008), and State Key Laboratory of Cryptography and Science.

Supplementary Materials

The supplementary material contains experimental data, including 10 plain text data, with a total size of 24 MB, and 10 cipher text data, obtained by encrypting ten plain text data with DES encryption algorithm, with a total size of 40 MB. The plain text data source is from the California Institute of Technology Cal tech-256 data set. (*Supplementary Materials*)

References

[1] Cheng and Hai and Qun Ding, "Overview of the block cipher," in *Proceedings of the 2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control*, IEEE, Harbin, China, December 2012.

[2] A. G. Bafghi, R. Safabakhsh, and B. Sadeghiyan, "Finding the differential characteristics of block ciphers with neural networks," *Information Sciences*, vol. 178, no. 15, pp. 3118–3132, 2008.

[3] K. M. Alallayah, A. H. Alhamami, W. AbdElwahed, and M. Amin, "Applying neural networks for simplified data encryption standard (SDES) cipher system cryptanalysis," *International Arab Journal of Information Technology*, vol. 9, no. 2, pp. 163–169, 2012.

[4] M. M. Alani, *Neuro-Cryptanalysis of DES and Triple-DES. Neural Information Processing*, Springer, Berlin, Germany, 2012.

[5] C. de Canniere, A. Biryukov, and B. Preneel, "An introduction to block cipher cryptanalysis," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 346–356, 2006.

[6] D. Mills, "Review of cryptography: theory and practice by D. R. Stinson," *Cryptologia*, vol. 31, no. 1, pp. 87-88, 2007.

[7] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[8] T. Kohonen, *Self-Organizing Feature Maps. Self-Organization and Associative Memory*, Springer, Berlin, Germany, 1988.

[9] H. D. Landahl, W. S. Mcculloch, and W. Pitts, "A statistical consequence of the logical calculus of nervous nets," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 135–137, 1943.

[10] G. Griffin, A. Holub, and P. Perona, *Caltech-256 Object Category Dataset*, California Institute of Technology, Pasadena, CA, USA, 2007.

[11] I. V. Tetko, A. E. P. Villa, T. I. Aksenova et al., "Application of a pruning algorithm to optimize artificial neural networks for pharmaceutical fingerprinting," *Journal of Chemical Information and Computer Sciences*, vol. 38, no. 4, pp. 660–668, 1998.

[12] I. V. Tetko, T. I. Aksenova, V. V. Volkovich et al., "Polynomial neural network for linear and non-linear model selection in quantitative-structure activity relationship studies on the internet," *SAR and QSAR in Environmental Research*, vol. 11, no. 3-4, pp. 263–280, 2000.

[13] H. R. Guo and Z. M. Li, "A method of improving generalization ability for neural network based on genetic algorithm," in *Proceedings of the 2010 IEEE International Conference on Intelligent Computing & Intelligent Systems*, October 2010.

[14] J. P. Yang, Q. Li, Z. Liu, and X. L. Yuan, "Research of improved bp algorithm based on self-adaptive learning rate," *Computer Engineering & Applications*, vol. 45, no. 11, pp. 56–58, 2009.

[15] S. Wermter, C. Weber, W. Duch et al., *Artificial Neural Networks and Machine Learning—ICANN 2014*, Springer, Berlin, Germany, 2014.

[16] S. Aditya and N. Nadir, "Cryptography based on artificial neural networks and chaos theory," *International Journal of Computer Applications*, vol. 133, no. 4, pp. 25–30, 2016.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 8567534, 10 pages
https://doi.org/10.1155/2019/8567534

Research Article

An Indistinguishably Secure Function Encryption Scheme

Ping Zhang, Yamin Li, and Muhua Liu

School of Mathematics and Statistics, Henan University of Science and Technology, Luoyang, China

Correspondence should be addressed to Yamin Li; 2390043823@qq.com

Received 1 April 2019; Revised 17 August 2019; Accepted 30 August 2019; Published 5 November 2019

Guest Editor: Leonel Sousa

In this work, we first design a function encryption scheme by using key encapsulation. We combine public key encryption with symmetric encryption to implement the idea of key encapsulation. In the key encapsulation, we use a key to turn a message (plaintext) into a ciphertext by symmetric encryption, and then we use public key encryption to turn this key into another ciphertext. In the design of function encryption scheme, we use the public key encryption system, symmetric encryption system, noninteractive proof system, indistinguishable obfuscator, and commitment scheme. Finally, we prove the indistinguishable security of our function encryption scheme.

1. Introduction

Nowadays, network technology has been rapidly developed and widely used. Due to the emergence of hacking, virus, and electronic fraud and theft, various information leakage incidents occur frequently. Both personal information security and enterprise information security are getting more and more attention, which makes information security more important than ever.

With the rise of cloud computing, in order to save local storage resources, more and more companies or individuals choose to store local data on third-party servers (such as cloud computing platforms). To prevent user data from leaking, the data can be encrypted before storing it on third-party servers. It is noteworthy that in this application, third-party servers need to operate something on the user's ciphertext, such as data mining, query, and statistics of the ciphertext. However, the traditional encryption system does not support ciphertext computing.

This problem can be solved by using fully homomorphic encryption [1]. It is possible to compute on the ciphertext using fully homomorphic encryption. In other words, a nontrusted information processing system can process information in ciphertext environment without disclosing any information of users. However, the main problem of fully homomorphic encryption is that the results are encrypted. That is to say, although the third-party servers can perform ciphertext calculation, it cannot obtain the calculation results. In many application scenarios, third-party servers need to make certain decisions based on the results of the calculations. Therefore, fully homomorphic encryption does not meet our requirements.

Function encryption firstly appeared in the article [2]. It is an extension of the articles [2–7]. It enables the third party to operate and output a function of plaintext without decryption. Therefore, it is very suitable for the computing scenario of data outsourcing encryption. Specifically, in function encryption for a function family, the key generation algorithm can generate the corresponding decryption key for each function in this function family by using the master private key. Given the ciphertext of a plaintext, the owner of the decryption key of each function can calculate the corresponding function value for this plaintext. A secure function encryption system should satisfy that users with decryption key can only get a function value of the plaintext but not any information about the plaintext. The traditional public key encryption scheme is too extreme. It enables users with decrypted key to get all or no messages from the ciphertext. Therefore, it is of great significance to study function encryption.

Function encryption is a perfect noninteractive solution to many problems that arise in delegating services to external servers. Consider the following scenario: Suppose a bank subscribes to an external cloud server for storing financial

records of its customers. In order to ensure the security of these records and perform various computations on the outsourced data remotely from time to time, a cost-effective choice for the bank is to use a function encryption scheme to encrypt the records locally prior to uploading to the cloud server. Now, suppose the researchers wish to retrieve investment amount summary of all customers who have joined a certain wealth management product from the cloud server. For this, the bank needs to provide the researchers a decryption key for the corresponding functionality. However, if the encryption scheme used by the bank possesses no function privacy, then the researchers would get to know the specific customer's information from the decryption key provided by the bank. Thus, after performing the assigned computation, for financial gain, the researchers may leak the information to someone. This is clearly undesirable from the privacy point of view. Our function encryption scheme can solve the aforementioned information leak problem, and it can be applied to the following application scenarios such as spam filtering in the mail service, patient record privacy protection in the hospital, and partial decryption of information in the encrypted information store.

1.1. Our Result. The design of our function encryption scheme depends on the idea of key encapsulation. Key encapsulation (also known as hybrid encryption) consists of two parts: an external scheme and an internal scheme. To encrypt messages in a hybrid encryption scheme, a new key is generated for the internal scheme and used to encrypt messages. The key itself is then encrypted using an external scheme. The final hybrid ciphertext consists of two parts: the external ciphertext and the internal ciphertext. If someone wants to decrypt, he should firstly decrypt the external ciphertext to get the key and use this key to decrypt the internal ciphertext. While the internal scheme is symmetric encryption and the external scheme is public key encryption, the hybrid scheme still belongs to public key encryption. Why use the idea of key encapsulation? The external scheme is inefficient, so it is only used to encrypt relatively short keys, while relatively long messages are encrypted by the internal scheme.

We choose symmetric encryption and public key encryption as the internal and external schemes of key encapsulation, respectively. In the design of the function encryption scheme, our idea is as follows: at first, we choose a key k of the internal scheme to encrypt the message x, and then use the external scheme to encrypt the key k to get the two parts of a complete ciphertext. This will allow for decryption during the two-step process described above. First, the key k of the internal scheme is obtained by decrypting the external ciphertext in the external scheme, and then $f(x)$ is obtained by using this key in the internal scheme. Specifically, we describe how to use the public key encryption system, symmetric encryption system, noninteractive proof system, indistinguishable obfuscator, and commitment scheme to design our function encryption scheme. Our scheme achieves indistinguishable security [8].

1.2. Related Works. The concept of function encryption originates from inner product encryption [6] and attribute-based encryption [5]. Early function encryption generally refers to attribute-based encryption, predicate encryption, or inner product encryption. These three kinds of encryption are the preliminary stages of function encryption. Sahai and Waters firstly created the concept of function encryption in 2008 [9]. Later, O'Neill proposed the security definition of the general function encryption [7], which laid the foundation for the research of function encryption.

In 2011, Boneh et al. [3] not only gave the general form of function encryption, but also gave the formal definition and various security definitions of function encryption. In 2012, Gorbunov et al. [10] proposed a nonconcise general-purpose function encryption scheme based on multiparty security calculation. In 2013, Waters [11], Garg et al., [12] and Ananth et al. [13] further improved the efficiency of universal function encryption system. Subsequently, the function encryption scheme for regular language [14] and the function encryption scheme-based deterministic finite automata [15] were proposed. In the same year, Garg et al. [16] constructed a general function encryption scheme for the first time using indistinguishable obfuscation, which set up a new direction for the research of universal function encryption. Goldwasser et al. [17] proposed a simple universal function encryption scheme which uses fully homomorphic encryption, attribute-based encryption, and obfuscation circuit technology as the underlying modules. This scheme is a classical function encryption scheme. Later, Goldwasser et al. [18] firstly proposed multi-input function encryption. They not only proved the adaptive indistinguishable security of the scheme but also proved the simulation-based security of the scheme. It has to be noted that multi-input function encryption scheme is a refinement of function encryption, which can be applied in many occasions and can realize the security processing of multiuser information at different times.

With the deepening of research on function encryption and fully homomorphic encryption, a new full key homomorphic encryption scheme (FKHE) [19] was proposed at the Eurocrypt 2014. It combines fully homomorphic encryption with function encryption. It overcomes the shortcomings of single key shortage of fully homomorphic encryption and provides possibilities for the use of function encryption in the outsourcing computing environment. In 2015, Goyal et al. [20] proposed for the first time the definition and construction of function encryption for random functions, which proved that the construction scheme was simulation-based security. Ananth et al. [13] proved that any selective secure function encryption can be transformed into adaptive secure function encryption and designed an adaptive function encryption scheme that combines public key function encryption with private key function encryption. Brakerski and Segev [21] studied the symmetric function encryption system with function hiding. Abdalla et al. [22] firstly proposed an internal product function encryption scheme based on public key, which set up a new direction of practical internal product function encryption scheme. Bishop et al. [23] firstly proposed the inner product

function encryption scheme of the function hidden in symmetric case. Subsequently, Datta et al. [24] improved Bishop's scheme. In 2018, Kim et al. [25] improved the inner product function encryption scheme of the function hidden to improve its efficiency. They also described three application scenarios of inner product function encryption of the function hidden. Kim et al. made it possible to use the inner product function encryption of the function hidden in practical applications.

1.3. Organization. This paper is organized as follows. The first section is the introduction of this paper. The second section is the existing definition of function encryption. In the third section, we recall some definitions of cryptographic primitives used in scheme construction. In the fourth section, we propose the design of our function encryption. In the fifth section, we prove the security of our constructed scheme. Finally, in the sixth section, we summarize the full text.

1.4. Basic Notation. In the following sections, the security parameter is $\lambda \in \mathbb{N}$. If $|\mathrm{negl}(\lambda)| < 1/\mathrm{poly}(\lambda)$ holds for all polynomials $\mathrm{poly}(\lambda)$ and all sufficiently large λ, we call $\mathrm{negl}(\lambda)$ as negligible function. In this paper, "PPT" represents probabilistic polynomial time and "$x \| y$" represents a concatenation of x and y. Let $\mathscr{F} = \mathscr{F}(\lambda)$ represent an ensemble, each of which is a finite function.

2. Function Encryption

In the second section, the concept of function encryption [3] and its indistinguishable security are rementioned. Indistinguishable security is not only the first consideration of function encryption but also of predicate encryption [4, 6].

2.1. Syntax. For a class of functions $\mathscr{F} = \mathscr{F}(\lambda)$ over message space $\mathscr{M} = \mathscr{M}_\lambda$, a function encryption scheme is composed of the following four algorithms:

(i) $\mathrm{Setup}(1^\lambda) \longrightarrow (\mathrm{mpk}, \mathrm{msk})$: This is a random algorithm. This algorithm requires a security parameter λ as the input of the algorithm, requires the master public key mpk, and the master secret key msk as the output of the algorithm.

(ii) $\mathrm{Enc}(\mathrm{mpk}, m) \longrightarrow \mathrm{CT}$: This algorithm requires the master public key mpk and the plaintext $m \in \mathscr{M}$ as the input of the algorithm and requires a ciphertext CT as the output of the algorithm.

(iii) $\mathrm{KeyGen}(\mathrm{msk}, f) \longrightarrow \mathrm{sk}_f$: This algorithm requires the master secret key msk and the randomized function $f \in \mathscr{F}$ as the input of the algorithm and requires a secret key sk_f as the output of the algorithm.

(iv) $\mathrm{Dec}(\mathrm{sk}_f, \mathrm{CT}) \longrightarrow f(m)$: This algorithm requires the ciphertext CT and the secret key sk_f as the input of the algorithm and requires a string $f(m)$ as the output of the algorithm.

Definition 1 (correctness). If the following conclusion holds for every function $f \in \mathscr{F}$ and every message $m \in \mathscr{M}$, the function encryption scheme for $\mathscr{F}$ is correct.

$$\Pr\Big[(\mathrm{mpk}, \mathrm{msk}) \longleftarrow \mathrm{Setup}\left(1^\lambda\right); \mathrm{Dec}(\mathrm{KeyGen}(\mathrm{msk}, f), \mathrm{Enc}(\mathrm{mpk}, m)) \neq f(m)\Big] = \mathrm{negl}(\lambda). \tag{1}$$

Definition 2 (indistinguishable security of function encryption). The indistinguishable security can be seen as a game between the attacker $\mathscr{A}$ and challenger. This game is divided into five phases.

(i) Setup phase: the challenger generates master key by the Setup algorithm $(\mathrm{mpk}, \mathrm{msk}) \longleftarrow \mathrm{Setup}(1^\lambda)$ and then makes the attacker $\mathscr{A}$ get the master public key mpk.

(ii) Query phase 1: the attacker $\mathscr{A}$ chooses f_i in $\mathscr{F}$ adaptively and makes the challenger get this f_i. The challenger generates the key sk_{f_i} of function f_i by the KeyGen algorithm $\mathrm{sk}_{f_i} \longleftarrow \mathrm{KeyGen}(\mathrm{msk}, f_i)$ and sends it to the attacker. The attacker can repeat this step in any polynomial number of times.

(iii) Challenge phase: the attacker $\mathscr{A}$ chooses two messages $m_0, m_1 \in \mathscr{M}$ such that $f_i(m_0) = f_i(m_1)$ and makes the challenger get it. The Challenger chooses m_b $(b \longleftarrow \{0, 1\})$ from m_0 and m_1, runs the Enc algorithm $\mathrm{CT} \longleftarrow \mathrm{Enc}(\mathrm{mpk}, m_b)$ $(b \longleftarrow \{0, 1\})$, and makes the attacker get the ciphertext CT.

(iv) Query phase 2: key queries continue to be initiated by the attacker $\mathscr{A}$ as before, but it also needs to satisfy that for any query f_i, there is $f_i(m_0) = f_i(m_1)$ holds.

(v) Guess phase: the attacker $\mathscr{A}$ guesses whether the ciphertext CT is encryption for message m_0 or message m_1. Finally, the attacker $\mathscr{A}$ give his guess b' $(b' \longleftarrow \{0, 1\})$.

In this game, the advantage of the attacker $\mathscr{A}$ is $\mathrm{Adv}_{\mathscr{A}} = \Pr[b' = b] - (1/2)$.

3. Preliminaries

In the third section, we recall some concepts of primitives in cryptography, which are used in the construction of function encryption. Here, we omit not only the formal definitions of standard semantically secure public key encryption scheme PKE = (PKE.KeyGen, PKE.Enc, PKE.Dec) but also the formal definitions of standard semantically secure symmetric encryption scheme SE = (SE.Enc, SE.Dec). Next, we describe the formal definitions

of indistinguishable obfuscation, commitment scheme, and witness indistinguishable proof system in detail.

3.1. Indistinguishable Obfuscation. According to the syntax of [16], the concept of indistinguishable obfuscation was recalled.

Definition 3 (Indistinguishable Obfuscation($i\mathcal{O}$)). If the following conclusions hold, then the uniform PPT machine $i\mathcal{O}$ is known as an indistinguishable obfuscator of a circuit class $\{\mathcal{C}_\lambda\}$.

(i) Correctness: for every security parameter $\lambda \in \mathbb{N}$, $C \in \mathcal{C}_\lambda$, and every input x, the following formula holds:

$$\Pr\left[C'(x) = C(x) : C' \longleftarrow i\mathcal{O}(\lambda, C)\right] = 1. \tag{2}$$

(ii) Indistinguishability: there is a negligible function negl that makes the following conclusion hold for every PPT distinguisher Samp, $\mathcal{D}$ (which is not necessarily uniform). For every security parameter $\lambda \in \mathbb{N}$ and every pair of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, if $C_0(x) = C_1(x)$ for every input x, then

$$\left|\Pr\left[D\left(i\mathcal{O}(\lambda, C_0)\right) = 1\right] - \Pr\left[D\left(i\mathcal{O}(\lambda, C_1)\right) = 1\right]\right| \leq \mathrm{negl}(\lambda). \tag{3}$$

3.2. Commitment Scheme. According to the syntax of [26], a commitment scheme Com takes a random number r and a string x as the input and takes $c \longleftarrow \mathrm{Com}(x; r)$ as the output. The following two properties are necessary for a perfectly binding commitment scheme:

(i) Perfectly binding property: this property means that the commitments for two different strings must also be different. More formally, $\forall\ x_1 \neq x_2$ and r_1, r_2, $\mathrm{Com}(x_1; r_1) \neq \mathrm{Com}(x_2; r_2)$.

(ii) Computational hiding property: for every string x_0 and x_1 (the length of x_0 and x_1 is the same) and for every PPT adversary $\mathcal{A}$, the following formula holds:

$$\left|\Pr\left[\mathcal{A}\left(\mathrm{Com}\left(x_0\right)\right) = 1\right] - \Pr\left[\mathcal{A}\left(\mathrm{Com}\left(x_1\right)\right) = 1\right]\right| \leq \mathrm{negl}(\lambda). \tag{4}$$

3.3. Noninteractive Witness Indistinguishable Proof. The syntax of the noninteractive proof system was firstly recalled, and then the formal concept of noninteractive witness indistinguishable (NIWI) proof [27] was also recalled.

3.3.1. Syntax. An efficiently computable relation R is composed of pairs (x, w), in which x and w are named as the statement and the witness, respectively. The language consisting of statements in R is denoted by L. The following three algorithms constitute a noninteractive proof system for a language L:

(i) $\mathrm{NIWI.Setup}(1^\lambda) \longrightarrow \mathrm{crs}$: this algorithm requires a security parameter 1^λ as the input of the algorithm and requires a common reference string crs as the output of the algorithm.

(ii) $\mathrm{NIWI.Prove}(\mathrm{crs}, x, w) \longrightarrow \pi$: this prove algorithm requires the common reference string crs and a statement x along with a witness w as the input of the algorithm. This prove algorithm outputs a proof string π when $(x, w) \in R$ or outputs fail when $(x, w) \notin R$.

(iii) $\mathrm{NIWI.Verify}(\mathrm{crs}, x, \pi) \longrightarrow \{0, 1\}$: this verify algorithm requires the common reference string crs and a statement x with a corresponding proof π as the input. If the proof is valid, this algorithm outputs 1 and otherwise 0.

Definition 4 (NIWI). A noninteractive witness indistinguishable proof system for a language L with a PPT relation R needs to satisfy the following properties:

(i) Perfect completeness property: for all $(x, w) \in R$, the following formula holds:

$$\Pr\left[\mathrm{NIWI.Verify}(\mathrm{crs}, x, \mathrm{NIWI.Prove}(\mathrm{crs}, x, w)) = 1\right] = 1. \tag{5}$$

Here, the reference string crs is generated by the algorithm $\mathrm{crs} \longleftarrow \mathrm{NIWI.Setup}(1^\lambda)$, and the probability is taken over the coins of NIWI.

(ii) Statistical soundness property: for all adversary $\mathcal{A}$, the following formula holds:

$$\Pr\left[\mathrm{NIWI.Verify}(\mathrm{crs}, x, \pi) = 1 \wedge x \notin L \;\middle|\; \begin{array}{l} \mathrm{crs} \longleftarrow \mathrm{NIWI.Setup}(1^\lambda); \\ (x, \pi) \longleftarrow \mathcal{A}(\mathrm{crs}) \end{array}\right] = \mathrm{negl}\left(1^\lambda\right). \tag{6}$$

(iii) Witness indistinguishability property: for any triplet (x, w_0, w_1) ($(x, w_0) \in R$ and $(x, w_1) \in R$), the distributions $\{\mathrm{crs}, \mathrm{NIWI.Prove}(\mathrm{crs}, x, w_0)\}$ and $\{\mathrm{crs}, \mathrm{NIWI.Prove}(\mathrm{crs}, x, w_1)\}$ are computationally indistinguishable (here the reference string crs is generated by the algorithm $\mathrm{crs} \longleftarrow \mathrm{NIWI.Setup}(1^\lambda)$).

4. Construction

In the fourth section, we propose the design of function encryption scheme. In our construction, we set the key space of the symmetric encryption scheme SE to $\{0, 1\}^{\ell_{\mathrm{SE}}}$.

In our scheme, we set the ciphertext length of the public key encryption scheme PKE = (PKE.KeyGen, PKE.Enc, PKE.Dec) to $\mathrm{len}_c = \mathrm{len}_c(1^\lambda)$. In the design of our scheme, the parameter $\mathrm{len} = 2 \cdot \mathrm{len}_c$ will be used, and the symmetric encryption scheme SE needs to satisfy this property:

$$\Pr\left[\mathrm{SE.Dec}(k_1, C') \neq \perp : (k_0, k_1) \longleftarrow \{0,1\}^{\ell_{\mathrm{SE}}},\ k_0 \neq k_1,\ C' \longleftarrow \mathrm{SE.Enc}(k_0, m)\right] \leq \mathrm{negl}(\lambda), \tag{7}$$

where symbol "$\perp$" denotes an error.

In the design of our scheme, the NIWI proof system used is denoted by (NIWI.Setup, NIWI.Prove, NIWI.Verify), the perfectly binding commitment scheme used is denoted by Com, and the indistinguishable obfuscator used is denoted by $i\mathcal{O}$. Now, we begin to give our scheme FE = (Setup, Enc, KeyGen, Dec).

(i) $\mathrm{Setup}(1^\lambda) \longrightarrow (\mathrm{mpk}, \mathrm{msk})$

(1) At first, two pairs of key pairs are generated using the key generation algorithm $(\mathrm{pk}_1, \mathrm{sk}_1) \longleftarrow \mathrm{PKE.KeyGen}(1^\lambda)$ and $(\mathrm{pk}_2, \mathrm{sk}_2) \longleftarrow \mathrm{PKE.KeyGen}(1^\lambda)$ of the public key encryption scheme (note: $(\mathrm{pk}, \mathrm{sk}) \longleftarrow \mathrm{PKE.KeyGen}(1^\lambda)$ is a random algorithm)

(2) Compute a common reference string $\mathrm{crs} \longleftarrow \mathrm{NIWI.Setup}(1^\lambda)$ using the NIWI proof system

(3) Choose a symmetric encryption scheme SE

(4) Compute $C \longleftarrow \mathrm{Com}(0^{\mathrm{len}})$

(5) Let the master public key and the master secret key be $\mathrm{mpk} = (\mathrm{pk}_1, \mathrm{pk}_2, \mathrm{crs}, C, \mathrm{SE})$ and $\mathrm{msk} = \mathrm{sk}_1$, respectively

(ii) $\mathrm{Enc}(\mathrm{mpk}, m) \longrightarrow \mathrm{CT}$:

(1) Choose a secret key K_{SE} of the symmetric encryption and compute the ciphertexts
$\mathrm{CT}_1 \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_1, K_{\mathrm{SE}})$
$\mathrm{CT}_2 \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_2, K_{\mathrm{SE}})$
$\mathrm{CT}_3 \longleftarrow \mathrm{SE.Enc}(K_{\mathrm{SE}}, m)$

(2) Compute $\pi \longleftarrow \mathrm{NIWI.Prove}(\mathrm{crs}, y, w)$ $(y = (\mathrm{CT}_1, \mathrm{CT}_2, C, \mathrm{pk}_1, \mathrm{pk}_2))$

(i) Either CT_1 and CT_2 are encryptions for the same plaintext, or

(ii) C is a commitment for $\mathrm{CT}_1 \| \mathrm{CT}_2$

The real witness $w_{\mathrm{real}} = (K_{\mathrm{SE}}, r_1, r_2)$ is used to prove the first part of the statement, where the random numbers r_1 and r_2 are used to compute the ciphertexts CT_1 and CT_2, respectively; the trapdoor witness $w_{\mathrm{trap}} = s$ is used to prove the second part of the statement, where the random number s is used to compute C

(3) The ciphertext is $\mathrm{CT} = (\mathrm{CT}_1, \mathrm{CT}_2, \mathrm{CT}_3, \pi)$

(iii) $\mathrm{KeyGen}(\mathrm{msk}, f) \longrightarrow \mathrm{sk}_f$

(1) Compute the decryption key $\mathrm{sk}_f \longleftarrow i\mathcal{O}(\mathcal{G})$ for function f, where the circuit $\mathcal{G}$ is showed in Figure 1; it should be noted that the circuit $\mathcal{G}$ contains the random function f, the secret key sk_1, and the master public key mpk

(iv) $\mathrm{Dec}(\mathrm{sk}_f, \mathrm{CT}) \longrightarrow f(m)$

(1) It inputs CT and decryption key sk_f and outputs $f(m)$

The correctness of the scheme we design is easily derived from the correctness of its components. Next, we will demonstrate the security of this scheme.

5. Proof of Security

Theorem 1. *Assume that the aforementioned function encryption instantiated with a standard semantically secure public key encryption, a standard semantically secure symmetric encryption, a computational hiding commitment, a noninteractive witness indistinguishable proof, and indistinguishably secure obfuscator, it is indistinguishably secure.*

Now, we prove that if the aforementioned assumption is true, no polytime attacker can break our scheme. We will prove indistinguishable security of the function encryption using indistinguishable game. We assume that a polytime attacker $\mathcal{A}$ makes q private key queries. Let f_i $(i \in [q])$ denote the ith function query. There is a constraint $f_i(m_0) = f_i(m_1)$ for each function query.

We need to define a sequence of games to prove theorem 1. The first game is the experiment with the challenge message m_0. Next, we prove that any PPT adversary has almost the same advantage in each game as that of the previous game.

Game 1. The challenger encrypts message m_0 in the challenge ciphertext.

(i) *Setup Phase.* The challenger firstly computes the key pair $(\mathrm{mpk}, \mathrm{msk}) \longleftarrow \mathrm{Setup}(1^\lambda)$ and gives master public mpk to $\mathcal{A}$. Choose a symmetric encryption scheme SE and three secret keys $k_1 \longleftarrow \{0,1\}^{\ell_{\mathrm{SE}}}$, $k_0 \longleftarrow \{0,1\}^{\ell_{\mathrm{SE}}}$, and $k_2 \longleftarrow \{0,1\}^{\ell_{\mathrm{SE}}}$ for the symmetric encryption scheme. Compute the ciphertexts:

$$\begin{aligned}
&\mathrm{CT}^*_{1,1} \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_1, k_1),\\
&\mathrm{CT}^*_{2,1} \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_2, k_1).\\
&\mathrm{CT}^*_{1,0} \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_1, k_0),\\
&\mathrm{CT}^*_{2,0} \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_2, k_0).\\
&\mathrm{CT}^*_{1,2} \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_1, k_2),\\
&\mathrm{CT}^*_{2,2} \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_2, k_2).
\end{aligned}$$

Set $c_1 = \mathrm{CT}^*_{1,1} \| \mathrm{CT}^*_{2,1}$, $c_2 = \mathrm{CT}^*_{1,0} \| \mathrm{CT}^*_{2,0}$, $c_3 = \mathrm{CT}^*_{1,2} \| \mathrm{CT}^*_{2,2}$, $c_4 = \mathrm{CT}^*_{1,1} \| \mathrm{CT}^*_{2,0}$, $c_5 = \mathrm{CT}^*_{1,2} \| \mathrm{CT}^*_{2,0}$.

(ii) *Query Phase.* The adversary $\mathcal{A}$ submits query of f adaptively. The challenger sends $\mathrm{sk}_f \longleftarrow \mathrm{KeyGen}(\mathrm{msk}, f)$ to $\mathcal{A}$.

(iii) *Challenge Phase.* The challenger chooses a challenge plaintext m_0 from two plaintexts and a secret key k_1 of the symmetric encryption, computes the ciphertexts $\mathrm{CT}^*_1 \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_1, k_1)$, $\mathrm{CT}^*_2 \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_2, k_1)$, and $\mathrm{CT}^*_3 \longleftarrow \mathrm{SE.Enc}(k_1, m_0)$. Compute a NIWI proof $\pi^* \longleftarrow \mathrm{NIWI.Prove}(\mathrm{crs}, y^*, w)$ $(y^* = (\mathrm{CT}^*_1, \mathrm{CT}^*_2, C, \mathrm{pk}_1, \mathrm{pk}_2))$. Set $\mathrm{st} = (\mathrm{CT}^*_1, \mathrm{CT}^*_2, k_1, \pi^*)$. At last, it returns $\mathrm{CT}^* = (\mathrm{CT}^*_1, \mathrm{CT}^*_2, \mathrm{CT}^*_3, \pi^*)$.

Input: Ciphertext CT
Constants: Master public key mpk, secret key sk_1, random function f

(1) Parse $CT = (CT_1, CT_2, CT_3, \pi)$.
(2) If NIWI.Verify $(crs, y, \pi) = 0$, output $\perp$ and stop. If NIWI.Verify $(crs, y, \pi) = 1$, continue to the next step. Here the statement corresponding to π is $y = (CT_1, CT_2, C, pk_1, pk_2)$.
(3) Compute $K_{SE} \longleftarrow$ PKE.Dec (sk_1, CT_1), $m \longleftarrow$ SE.Dec (K_{SE}, CT_3).
(4) Output $f(m)$.

Figure 1: Functionality $\mathcal{G}$.

Game 2. This game is exactly the same as the previous game (Game 1) except for a little difference. The difference is that for all key queries of f, the corresponding secret key sk_f is computed by $sk_f \longleftarrow i\mathcal{O}(\mathcal{G}_f)$, where the circuit $\mathcal{G}_f$ is described in Figure 2.

Game 3. This game is exactly the same as the previous game (Game 2) except for a little difference. The difference is that the commitment C is computed as follows: the challenge ciphertext is denoted by $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$ and $C \longleftarrow \mathrm{Com}(CT_1^* \| CT_2^*)$.

Game 4. This game is exactly the same as the previous game (Game 3) except for a little difference. The difference is that in every challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$ π^* is computed using the trapdoor witness.

Game 5. This game is exactly the same as the previous game (Game 4) except for a little difference. The difference is that we modify the challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$: the second ciphertext CT_2^* is an encryption of k_0, that is, $CT_2^* \longleftarrow$ PKE.Enc (pk_2, k_0).

Game 6. This game is exactly the same as the previous game (Game 5) except for a little difference. The difference is that for all key queries of f, the corresponding secret key sk_f is computed by $sk_f \longleftarrow i\mathcal{O}(\mathcal{G}_f^*)$, where the circuit $\mathcal{G}_f^*$ is described in Figure 3.

Game 7. This game is exactly the same as the previous game (Game 6) except for a little difference. The difference is that we modify the challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$: the first ciphertext CT_1^* is an encryption of k_0, that is, $CT_1^* \longleftarrow \mathrm{PKE.Enc}(pk_1, k_0)$.

Game 8. This game is exactly the same as the previous game (Game 7) except for a little difference. The difference is that the symmetric encryption key k_1 is replaced by another key k_2.

Game 9. This game is exactly the same as the previous game (Game 8) except for a little difference. The difference is that we modify the challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$: the third ciphertext CT_3^* is an encryption of m_1, that is, $CT_3^* \longleftarrow \mathrm{SE.Enc}(k_2, m_1)$.

Game 10. This game is exactly the same as the previous game (Game 9) except for a little difference. The difference is that we modify the challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$: the first ciphertext CT_1^* is an encryption of k_2, that is $CT_1^* \longleftarrow \mathrm{PKE.Enc}(pk_1, k_2)$.

Game 11. This game is exactly the same as the previous game (Game 10) except for a little difference. The difference is that for all key queries f, the corresponding secret key sk_f is computed by $sk_f \longleftarrow i\mathcal{O}(\mathcal{G}_f)$.

Game 12. This game is exactly the same as the previous game (Game 11) except for a little difference. The difference is that we modify the challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$: the second ciphertext CT_2^* is an encryption of k_2, that is, $CT_2^* \longleftarrow \mathrm{PKE.Enc}(pk_2, k_2)$.

Game 13. This game is exactly the same as the previous game (Game 12) except for a little difference. The difference is that in every challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$, the proof π^* is computed using the real witness.

Game 14. This game is exactly the same as the previous game (Game 13) except for a little difference. The difference is that the commitment C is computed as follows: $C \longleftarrow \mathrm{Com}(0^{\mathrm{len}})$. Note that this game corresponds to the selective indistinguishable game that has been honestly executed, where the challenger encrypts the message m_1 in the challenge ciphertext.

The description for the sequence of games is completed. Next, we prove that the neighbouring games are indistinguishable.

Lemma 1. *Game 1 and Game 2 are computationally indistinguishable when $i\mathcal{O}$ is an indistinguishable obfuscator.*

Proof. We can see that the difference between Game 1 and Game 2 lies only in the calculation method of the secret key sk_f. In the former experiment, Game 1, for any key query f, the secret key sk_f is outputted by $i\mathcal{O}(\mathcal{G})$; however, in the latter experiment, Game 2, the secret key sk_f is outputted by $i\mathcal{O}(\mathcal{G}_f)$. If we want to prove that Game 1 and Game 2 are computationally indistinguishable, we need to prove that for any input CT, $\mathcal{G}$ and $\mathcal{G}_f$ have identical output for identical input. Then, according to the security of indistinguishable obfuscator, we can get that $i\mathcal{O}(\mathcal{G})$ and $i\mathcal{O}(\mathcal{G}_f)$ are computationally indistinguishable, which means that Game 1 and Game 2 are computationally indistinguishable.

First, we will prove that for any input $CT = (CT_1, CT_2, CT_3, \pi)$, $\mathcal{G}$ outputs $\perp$ if and only if $\mathcal{G}_f$ outputs $\perp$.

We can find that both $\mathcal{G}$ and $\mathcal{G}_f$ output $\perp$ if and only if the proof π is invalid, that is, NIWI.Verify $(crs, y, \pi) = 0$. Here, $y = (CT_1, CT_2, C, pk_1, pk_2)$. If the proof π is valid, we define that an input $CT = (CT_1, CT_2, CT_3, \pi)$ is valid.

Next, we prove that for any valid input $CT = (CT_1, CT_2, CT_3, \pi)$, $\mathcal{G}(CT) = \mathcal{G}_f(CT)$.

Input: Ciphertext CT
Constants: sk_1, mpk, f, k_0, k_1, k_2, c_1, c_2, c_3, c_4, c_5

(1) Parse $\mathrm{CT} = (\mathrm{CT}_1, \mathrm{CT}_2, \mathrm{CT}_3, \pi)$.
(2) If NIWI.Verify (crs, y, π) = 0, output $\perp$ and stop. If NIWI.Verify (crs, y, π) = 1, continue to the next step. Here the statement corresponding to π is $y = (\mathrm{CT}_1, \mathrm{CT}_2, C, \mathrm{pk}_1, \mathrm{pk}_2)$.
(3) If there exists a $c_i = \mathrm{CT}_1^* \,||\, \mathrm{CT}_2^*$, then if SE.Dec $(k_0, \mathrm{CT}_3) = m$, and $m \neq \perp$, output $f(m)$ and stop, else if SE.Dec $(k_1, \mathrm{CT}_3) = m$, and $m \neq \perp$, output $f(m)$ and stop, else SE.Dec $(k_2, \mathrm{CT}_3) = m$, and $m \neq \perp$, output $f(m)$ and stop, otherwise continue to the next step.
(4) Compute $K_{\mathrm{SE}} \longleftarrow$ PKE.Dec $(\mathrm{sk}_1, \mathrm{CT}_1)$, $m \longleftarrow$ SE.Dec $(K_{\mathrm{SE}}, \mathrm{CT}_3)$.
(5) Output $f(m)$.

Figure 2: Functionality $\mathcal{G}_f$.

Input: Ciphertext CT
Constants: sk_2, mpk, f, k_0, k_1, k_2, c_1, c_2, c_3, c_4, c_5

(1) Parse $\mathrm{CT} = (\mathrm{CT}_1, \mathrm{CT}_2, \mathrm{CT}_3, \pi)$.
(2) If NIWI.Verify (crs, y, π) = 0, output $\perp$ and stop. If NIWI.Verify (crs, y, π) = 1, continue to the next step. Here the statement corresponding to π is $y = (\mathrm{CT}_1, \mathrm{CT}_2, C, \mathrm{pk}_1, \mathrm{pk}_2)$.
(3) If there exists a $c_i = \mathrm{CT}_1^* \,||\, \mathrm{CT}_2^*$, then if SE.Dec $(k_0, \mathrm{CT}_3) = m$, and $m \neq \perp$, output $f(m)$ and stop, else if SE.Dec $(k_1, \mathrm{CT}_3) = m$, and $m \neq \perp$, output $f(m)$ and stop, else SE.Dec $(k_2, \mathrm{CT}_3) = m$, and $m \neq \perp$, output $f(m)$ and stop, otherwise continue to the next step.
(4) Compute $K_{\mathrm{SE}} \longleftarrow$ PKE.Dec $(\mathrm{sk}_2, \mathrm{CT}_2)$, $m \longleftarrow$ SE.Dec $(K_{\mathrm{SE}}, \mathrm{CT}_3)$.
(5) Output $f(m)$.

Figure 3: Functionality $\mathcal{G}_f^*$.

Here we have to consider two cases:

(i) *Case 1*. There does not exist a c_i, such that $c_i = \mathrm{CT}_1^* \| \mathrm{CT}_2^*$.
(ii) *Case 2*. There exists a c_i, such that $c_i = \mathrm{CT}_1^* \| \mathrm{CT}_2^*$.

For the case 1, both $\mathcal{G}$ and $\mathcal{G}_f$ compute k_1 with sk_1 by decrypting CT_1, compute m with k_1 by decrypting CT_3, and outputs $f(m)$. In the case 2, $\mathcal{G}$ computes k_1 with sk_1 by decrypting CT_1, computes m with k_1 by decrypting CT_3, and also outputs $f(m)$. What's more, because $c_i = \mathrm{CT}_1^* \| \mathrm{CT}_2^*$, $\mathcal{G}_f$ also outputs $f(m)$. So we can get that $\mathcal{G}(\mathrm{CT}) = \mathcal{G}_f(\mathrm{CT})$.

The experiment $G_{2,i}$ $(0 \leq i \leq q)$ is defined as follows: in $G_{2,i}$, the first ith queries are answered by $\mathcal{G}$ and the remaining $(i+1)$th to qth queries are answered by $\mathcal{G}_f$. We can see that $G_{2,0}$ happens to be Game 1 and $G_{2,q}$ happens to be Game 2.

Here, we prove that if there is a PPT adversary $\mathcal{A}$ which can distinguish the experiments $G_{2,i}$ and $G_{2,i+1}$ with nonnegligible advantage, there is another PPT adversary $\mathcal{B}$ which can break the security of indistinguishable obfuscator with nonnegligible advantage.

We construct $\mathcal{B}$ by $\mathcal{A}$ as follows:

(1) First of all, the adversary $\mathcal{B}$ honestly runs the Game 1.
(2) For the first ith key queries f, the adversary $\mathcal{B}$ computes the key sk_f by $\mathcal{G}_f$. For the remaining $(i+2)$th to qth key queries f, the adversary $\mathcal{B}$ computes the key sk_f by $\mathcal{G}$.
(3) For the $(i+1)$th key queries f, the adversary $\mathcal{B}$ respectively constructs the circuit $\mathcal{G}$ and $\mathcal{G}_f$ and sends them to the challenger of the $i\mathcal{O}$ and receives an obfuscation sk_f. Then, the adversary $\mathcal{B}$ gives sk_f to $\mathcal{A}$.
(4) The adversary $\mathcal{B}$ runs the rest of the experiment according to the Game 1.
(5) At last, the adversary $\mathcal{B}$ gives the output of the game to the adversary $\mathcal{A}$ and the obfuscation challenger.

We happen to be in experiment $G_{2,i}$ when the challenger of $i\mathcal{O}$ chose the circuit $\mathcal{G}$; We happen to be in experiment $G_{2,i+1}$ when the challenger of $i\mathcal{O}$ chose the circuit $\mathcal{G}_f$. Therefore, if the adversary $\mathcal{A}$ can distinguish between the two experiments with nonnegligible advantage, then the adversary $\mathcal{B}$ can break the security of indistinguishable obfuscator with the same advantage.

In summary, Game 1 is computationally indistinguishable from Game 2. □

Lemma 2. *If Com is a computationally hiding commitment scheme, Game 2 is computationally indistinguishable from Game 3.*

Proof. We can see that the difference between Game 2 and Game 3 lies only in the calculation method of the commitment C. In the former game, Game 2, C is a commitment for 0^{len}; however, in the latter game, Game 3, C is a commitment for $\mathrm{CT}_1^* \| \mathrm{CT}_2^*$. It is important to note that the random number used to compute C is never used anywhere.

Therefore, the property of computational hiding in the commitment guarantees that Game 2 is computationally indistinguishable from Game 3. □

Lemma 3. *Because of witness indistinguishability of NIWI, Game 3 is computationally indistinguishable from Game 4.*

Proof. We can see that the difference between Game 3 and Game 4 lies only in the witness w used. In Game 3, for proving that CT_1^* and CT_2^* are encryptions of the same message, we use the real witness to compute π. However, in Game 4, for proving that C is a commitment for $\mathrm{CT}_1^* \| \mathrm{CT}_2^*$, we use the trapdoor witness to compute π. Since NIWI is witness indistinguishable, Game 3 is computationally indistinguishable from Game 4. □

Lemma 4. *If* (*PKE.KeyGen*, *PKE.Enc*, *PKE.Dec*) *is a semantically secure public key encryption scheme, Game 4 is computationally indistinguishable from Game 5.*

Proof. We can see that the difference between Game 4 and Game 5 lies only in the calculation method of the second ciphertexts CT_2^* in the challenge ciphertexts CT^*. In Game 4, CT_2^* is an encryption of the random message k_1, while in Game 5, CT_2^* is an encryption of the random message k_0. Next, we show that if there is an adversary $\mathscr{A}$ who can distinguish Game 4 from Game 5, there is an adversary $\mathscr{B}$ who can break the semantic security of the public key encryption system. The adversary $\mathscr{B}$ is designed as follows:

(1) At first, the adversary $\mathscr{B}$ received a public key pk from the challenger.

(2) $\mathscr{B}$ generates $(\mathrm{pk}_1, \mathrm{sk}_1) \longleftarrow \mathrm{PKE.KeyGen}(1^\lambda)$, $\mathrm{crs} \longleftarrow \mathrm{NIWI.Setup}(1^\lambda)$, and chooses a symmetric encryption scheme SE. Next, the adversary $\mathscr{B}$ encrypts the string k_1 using pk_1 to compute the ciphertext CT_1^*.

(3) The adversary $\mathscr{B}$ sends (k_1, k_0) to the challenger and receives a ciphertext CT_2^*. Then, $\mathscr{B}$ computes the commitment $C = \mathrm{Com}(\mathrm{CT}_1^* \| \mathrm{CT}_2^*)$.

(4) The adversary $\mathscr{B}$ runs the rest of the experiment according to Game 4 and Game 5.

(5) At last, the adversary $\mathscr{A}$ received the output of the experiment from the adversary $\mathscr{B}$.

(6) If adversary $\mathscr{A}$ outputs Game 4, the adversary $\mathscr{B}$ outputs that CT_2^* is an encryption of k_1, otherwise outputs that CT_2^* is an encryption of k_0.

We can see that the adversary $\mathscr{B}$ just runs Game 4 when CT_2^* is an encryption of k_1, and $\mathscr{B}$ just runs Game 5 when CT_2^* is an encryption of k_0. Therefore, if there exists an adversary who can distinguish the outputs of the two games with nonnegligible advantage, we can construct another adversary who can break the semantic security of public key encryption with nonnegligible advantage. □

Lemma 5. *If Com is perfectly binding, and NIWI is statistically sound, $i\mathcal{O}$ is an indistinguishable obfuscator and Game 5 and Game 6 are computationally indistinguishable.*

Proof. Similar to the proof of Lemma 1, we firstly prove that for any input $\mathrm{CT} = (\mathrm{CT}_1, \mathrm{CT}_2, \mathrm{CT}_3, \pi)$, $\mathscr{G}_f$ outputs $\perp$ if and only if $\mathscr{G}_f^*$ outputs $\perp$.

We can see that both $\mathscr{G}_f$ and $\mathscr{G}_f^*$ output $\perp$ if and only if the proof π is invalid, that is, $\mathrm{NIWI.Verify}(\mathrm{crs}, y, \pi) = 0$ ($y = (\mathrm{CT}_1, \mathrm{CT}_2, C, \mathrm{pk}_1, \mathrm{pk}_2)$).

So we need to consider valid inputs only. Then, we will prove that all valid inputs $\mathrm{CT} = (\mathrm{CT}_1, \mathrm{CT}_2, \mathrm{CT}_3, \pi)$ meet one of the following properties:

(i) CT_1 and CT_2 are encryptions for the same message

(ii) There exists an i such that $\mathrm{CT}_1 \| \mathrm{CT}_2 = c_i$

We are going to use a proof by contradiction here. We assume that there is a valid input that satisfies neither of the above properties. Because NIWI is statistically sound, the statement $y = (\mathrm{CT}_1, \mathrm{CT}_2, C, \mathrm{pk}_1, \mathrm{pk}_2)$ must have either a real witness or a trapdoor witness when the input is valid. However, because CT_1 and CT_2 are encryptions of different messages, the real witness does not exist. Therefore, there must be a trapdoor witness to make the input valid. That means there is s such that $C = \mathrm{Com}(\mathrm{CT}_1 \| \mathrm{CT}_2; s)$. On the other hand, since $C = \mathrm{Com}(\mathrm{CT}_1^* \| \mathrm{CT}_2^*; s)$ and Com is perfectly binding, $\mathrm{CT}_1 \| \mathrm{CT}_2 = \mathrm{CT}_1^* \| \mathrm{CT}_2^* = c_5$. Thus, we get a contradiction, and the assumption is not true.

Next, we will prove that for any input $\mathrm{CT} = (\mathrm{CT}_1, \mathrm{CT}_2, \mathrm{CT}_3, \pi)$, $\mathscr{G}_f(\mathrm{CT}) = \mathscr{G}_f^*(\mathrm{CT})$.

If both CT_1 and CT_2 are encryptions for the same message, then $\mathrm{PKE.Dec}(\mathrm{sk}_1, \mathrm{CT}_1) = \mathrm{PKE.Dec}(\mathrm{sk}_2, \mathrm{CT}_2) = k_1$, and $\mathrm{SE.Dec}(k_1, \mathrm{CT}_3) = m$. Therefore, both G_f and G_f^* output $f(m)$. On the other hand, if there is an i satisfying $\mathrm{CT}_1 \| \mathrm{CT}_2 = c_i$, then both G_f and G_f^* output $f(m)$. Therefore, for all valid inputs, G_f and G_f^* have same output for the same input, that is, $\mathscr{G}_f(\mathrm{CT}) = \mathscr{G}_f^*(\mathrm{CT})$.

In summary, if there exists an adversary which can distinguish the outputs of these games with nonnegligible advantage, we can construct another adversary which can break the security of indistinguishable obfuscation with the same advantage. □

Lemma 6. *Game 6 is computationally indistinguishable from Game 7, when PKE is a semantically secure public key encryption scheme.*

Proof. The proof method of this lemma is the same as the proof method of lemma 4, so it is omitted here. □

Lemma 7. *Game 7 is computationally indistinguishable from Game 8.*

Proof. We can see that Game 7 and Game 8 differ in the secret key used in the symmetric encryption scheme. In Game 7, k_1 is used to encrypt m_0, while in Game 8, k_2 is used to encrypt m_0. We can find that both k_1 and k_2 are chosen

randomly from the key space of symmetric encryption scheme. So Game 7 is computationally indistinguishable from Game 8. □

Lemma 8. *The outputs of Game 8 and Game 9 are computationally indistinguishable when* (*SE.KeyGen*, *SE.Enc*, *SE.Dec*) *is a semantically secure symmetric encryption scheme.*

Proof. Since the proof method of this lemma is the same as that of lemma 4, the proof process is omitted here. □

Lemma 9. *The outputs of Game 9 and Game 10 are computationally indistinguishable when* (*PKE.KeyGen*, *PKE.Enc*, *PKE.Dec*) *is a semantically secure public key encryption scheme.*

Proof. Since the proof method of this lemma is the same as that of lemma 4, the proof process is omitted here. □

Lemma 10. *Game 10 and Game 11 are computationally indistinguishable when NIWI is statistically sound, Com is perfectly binding, and $i\mathcal{O}$ is an indistinguishable obfuscator.*

Proof. Since the proof method of this lemma is the same as that of lemma 5, the proof process is omitted here. □

Lemma 11. *The outputs of Game 11 and Game 12 are computationally indistinguishable when* (*PKE.KeyGen*, *PKE.Enc*, *PKE.Dec*) *is a semantically secure public key encryption scheme.*

Proof.
Since the proof method of this lemma is the same as that of lemma 4, the proof process is omitted here. □

Lemma 12. *Because of witness indistinguishability of NIWI, Game 12 and Game 13 are computationally indistinguishable.*

Proof. Since the proof method of this lemma is the same as that of lemma 3, the proof process is omitted here. □

Lemma 13. *Game 13 is computationally indistinguishable from Game 14 when Com is a computationally hiding commitment scheme.*

Proof. Since the proof method of this lemma is the same as that of lemma 2, the proof process is omitted here. □

6. Conclusion

We firstly design a function encryption scheme using the key encapsulation mechanism in this paper. This mechanism combines public key encryption with symmetric encryption. We encrypt the message using symmetric encryption, and then we encrypt the key of symmetric encryption using public key encryption. In the construction of function encryption scheme, we use noninteractive witness indistinguishable proof, commitment scheme, and indistinguishable obfuscator. Finally, the indistinguishable security of the designed function encryption is proven.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 11401172) and Science and Technology Project of Henan Educational Committee of China (No. 20A520012).

References

[1] G. Craig, *A Fully Homomorphic Encryption Scheme*, Stanford University, Stanford, CA, USA, 2009.

[2] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473, Aarhus, Denmark, May 2005.

[3] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: definitions and challenges," in *Proceedings of the 8th Theory of Cryptography Conference, TCC 2011*, pp. 253–273, Providence, RI, USA, March 2011.

[4] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proceedings of the 4th Theory of Cryptography Conference, TCC 2007*, pp. 535–554, Amsterdam, The Netherlands, February 2007.

[5] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006*, pp. 89–98, Alexandria, VA, USA, October-November 2006.

[6] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proceedings of the 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 146–162, Istanbul, Turkey, April 2008.

[7] A. O'Neill, "Definitional issues in functional encryption," Cryptology ePrint Archive: Report 2010/556, 2010.

[8] S. Goldwasser and M. Bellare, *Lecture Notes on Cryptography*, MIT Laboratory of Computer Science, Cambridge, MA, USA, 2001.

[9] A. Sahai and B. Waters, Slides on Functional Encryption, PowerPoint Presentation, 2008.

[10] S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Functional encryption with bounded collusions via multi-party computation," in *Proceedings of the 32nd Annual Cryptology Conference*, pp. 162–179, Santa Barbara, CA, USA, August 2012.

[11] B. Waters, "A punctured programming approach to adaptively secure functional encryption," in *Proceedings of the 35th Annual Cryptology Conference*, pp. 678–697, Santa Barbara, CA, USA, August 2015.

[12] S. Garg, C. Gentry, S. Halevi, and M. Zhandry, "Functional encryption without obfuscation," in *Proceedings of the 13th International Conference on Theory of Cryptography, TCC 2016*, pp. 480–511, Tel Aviv, Israel, January 2016.

[13] P. Ananth, Z. Brakerski, G. Segev, and V. Vaikuntanathan, "From selective to adaptive security in functional encryption," in *Proceedings of the 35th Annual Cryptology Conference*, pp. 657–677, Santa Barbara, CA, USA, August 2015.

[14] B. Waters, "Functional encryption for regular languages," in *Proceedings of the 32nd Annual Cryptology Conference*, pp. 218–235, Santa Barbara, CA, USA, August 2012.

[15] S. C. Ramanna, "DFA-based functional encryption: adaptive security from dual system encryption," IACR Cryptology ePrint Archive: Report 2013/638, 2013.

[16] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pp. 40–49, Berkeley, CA, USA, October 2013.

[17] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich, "Reusable garbled circuits and succinct functional encryption," in *Proceedings of the Symposium on Theory of Computing Conference, STOC'13*, pp. 555–564, Palo Alto, CA, USA, June 2013.

[18] S. Goldwasser, V. Goyal, A. Jain, and A. Sahai, "Multi-input functional encryption," IACR Cryptology ePrint Archive Report: 2013/727, 2013.

[19] D. Boneh, C. Gentry, S. Gorbunov et al., "Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits," in *Proceedings of the 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 533–556, Copenhagen, Denmark, May 2014.

[20] V. Goyal, A. Jain, V. Koppula, and A. Sahai, "Functional encryption for randomized functionalities," in *Proceedings of the 12th Theory of Cryptography Conference, TCC 2015*, pp. 325–351, Warsaw, Poland, March 2015.

[21] Z. Brakerski and G. Segev, "Function-private functional encryption in the private-key setting," in *Proceedings of the 12th Theory of Cryptography Conference, TCC 2015*, pp. 306–324, Warsaw, Poland, March 2015.

[22] M. Abdalla, F. Bourse, A. D. Caro, and D. Pointcheval, "Simple functional encryption schemes for inner products," in *Proceedings of the 18th IACR International Conference on Practice and Theory in Public-Key Cryptography*, pp. 733–751, Gaithersburg, MD, USA, March-April 2015.

[23] A. Bishop, A. Jain, and L. Kowalczyk, "Function-hiding inner product encryption," in *Proceedings of the 21st International Conference on the Theory and Application of Cryptology and Information Security*, pp. 470–491, Auckland, New Zealand, November-December 2015.

[24] P. Datta, R. Dutta, and S. Mukhopadhyay, "Functional encryption for inner product with full function privacy," in *Proceedings of the 19th IACR International Conference on Practice and Theory in Public-Key Cryptography*, pp. 164–195, Taipei, Taiwan, March 2016.

[25] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu, "Function-hiding inner product encryption is practical," in *Proceedings of the 11th International Conference on Security and Cryptography for Networks, SCN 2018*, pp. 544–562, Amalfi, Italy, September 2018.

[26] O. Goldreich, *Foundations of Cryptographyčž Basic Tools*, The Press Syndicate of the University of Cambridge, Cambridge, UK, 2001.

[27] U. Feige and A. Shamir, "Witness indistinguishable and witness hiding protocols," in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pp. 416–426, Baltimore, MA, USA, May 1990.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 4187892, 15 pages
https://doi.org/10.1155/2019/4187892

Research Article

A Novel Construction of Constrained Verifiable Random Functions

Muhua Liu, Ping Zhang, and Qingtao Wu

Control Science and Engineering Postdoctoral Mobile Station, Henan University of Science and Technology, Luoyang 471023, China

Correspondence should be addressed to Muhua Liu; lxk0379@126.com

Received 4 April 2019; Revised 30 August 2019; Accepted 11 October 2019; Published 3 November 2019

Guest Editor: Giovanni Agosta

Constrained verifiable random functions (VRFs) were introduced by Fuchsbauer. In a constrained VRF, one can drive a constrained key sk_S from the master secret key sk, where S is a subset of the domain. Using the constrained key sk_S, one can compute function values at points which are not in the set S. The security of constrained VRFs requires that the VRFs' output should be indistinguishable from a random value in the range. They showed how to construct constrained VRFs for the bit-fixing class and the circuit constrained class based on multilinear maps. Their construction can only achieve selective security where an attacker must declare which point he will attack at the beginning of experiment. In this work, we propose a novel construction for constrained verifiable random function from bilinear maps and prove that it satisfies a new security definition which is stronger than the selective security. We call it semiadaptive security where the attacker is allowed to make the evaluation queries before it outputs the challenge point. It can immediately get that if a scheme satisfied semiadaptive security, and it must satisfy selective security.

1. Introduction

Pseudorandom functions (PRFs) are one of the basic concepts in modern cryptography, which were introduced by Goldreich et al. [1]. A PRF is an efficiently computable function $F : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$. For a randomly chosen key $sk \in \mathcal{K}$, a polynomial probabilistic time (PPT) adversary cannot distinguish the outputs $F(sk, x)$ of the function for any $x \in \mathcal{X}$ from a randomly chosen values from $\mathcal{Y}$.

Boneh and Waters [2] put forward the concept of PRFs and presented a new notion which was called constrained pseudorandom functions. A constrained PRF is the same as the standard PRF except that it is associated with a set $S \subset \mathcal{X}$. It contains a master key $sk \in \mathcal{K}$ which can be used to evaluate all points that belonged to the domain $\mathcal{X}$. Given the master key $sk \in \mathcal{K}$ and a set $S \subset \mathcal{X}$, it can generate a constrained key sk_S which can be used to evaluate $F(sk, x)$ for any $x \notin S$. Pseudorandomness requires that given several constrained keys for sets $S_1, \ldots, S_{q_1} \subset \mathcal{X}$ and several function values at points $x_1, \ldots, x_{q_2} \in \mathcal{X}$ which were chosen adaptively by the adversary, the adversary cannot distinguish a function value $F(sk, x)$ from a random value for all $x \neq x_i$, $\forall i \in \{1, \ldots, q_2\}$, and $x \in \cap_{j=1}^{q_1} S_j$. Constrained PRFs have been used to optimize the ciphertext length of broadcast encryption [2] and construct multiparty key exchange [3].

Verifiable random functions were introduced by Micali et al. [4]. A VRF is similar to a pseudorandom function. It also preserves the pseudorandomness that a PPT adversary cannot distinguish an evaluated value $F(sk, x)$ from a random value even if it is given several values at other points. A VRF has an additional property that the party holding the secret key is allowed to evaluate F on $x \in \mathcal{X}$ associated with a noninteractive proof. With the proof, anyone can verify the correctness of a given evaluation by the public key. In addition, the evaluation of $F(sk, x)$ should remain pseudorandomness and even an adversary can query values and proofs at other points. Lastly, the verification should remain

sound even if the public key was computed maliciously. VRFs have been used to construct zero knowledge proofs [5], and electronic payment schemes [6], and so on.

In SCN 2014, Fuchsbauer [7] extended the notion of VRFs to a new notion, which was called constrained VRFs. In addition to three polynomial time algorithms: Setup, Prove, and Verify, they defined another algorithm Constrain, which was used to drive a constrained key. For constrained VRFs, it generates a pair key (pk, sk) in the Setup algorithm. Given a constrained key sk_S for a set $S \subset \mathcal{X}$, the algorithm Prove computes a value $y = F(sk, x)$ associated with a prove π which can be used to verify the correctness of $y = F(sk, x)$ by the public key pk. A constrained VRF should satisfy the security notions of provability, uniqueness, and pseudorandomness. The pseudorandomness requires that the evaluation of $F(sk, x)$ should be indistinguishable from a random value, even if the adversary is given several constrained keys for subset $S_1, \ldots, S_{q_1} \subset \mathcal{X}$ and several function values associated with proofs at points $x_1, \ldots, x_{q_2}$, where $x \neq x_i$, $\forall i \in [q_2]$, and $x \in \cap_{j=1}^{q_1} S_j$.

A possible application of constrained VRFs is micropayments [8]. Micropayment schemes emphasized the ability to make payments of small amounts. In the micropayment based on probability, a large number of users and merchants jointly select a user to pay the cheque. It can realize the micropayment of a large number of users to be converted into a macropayment of a certain user with a small probability. In this scheme, how do we decide which cheque C should be payable in fair way? Using the VRFs, merchant M publishes pk_M for VRF with range $\mathcal{Y} \in [0, 1]$. Cheque C is payable if $F(sk_M, C) < s$, where s is a known selection rate. However, it has a drawback which needs a public key infrastructure (PKI) for merchants' keys pk_M. By the constrained VRFs, every merchant uses the same key sk. Merchant M gets constrained key sk_M for set (id_M, C), where id_M is the identity of merchant M. Cheque C is payable if $F(sk_M, id_M \| C) < s$. Anybody can check the result by the same public key pk. Therefore, it does not need the PKI for merchants.

Fuchsbauer [7] gave two constructions from the multilinear maps based on constrained PRFs proposed by Boneh and Waters [2]. The first one is bit-fixing VRFs, in which the constrained keys can be derived for any set $S_\upsilon \subset \{0, 1\}^n$, where S_υ is described by a vector $\upsilon \in \{0, 1, \perp\}$ as the set of all strings such that it matches υ at all coordinates that are not $\perp$. The second one is circuit constrained VRFs, in which the constrained keys can be derived for any set that is decidable by a polynomial size circuit.

However, Fuchsbauer's constructions [7] can only achieve selective security—a weaker notion where the adversary must commit to a challenge point x^* at the beginning of the experiment. By the technology of complexity leveraging, any selective security can be converted into adaptive security where the adversary can make its challenge query at any point. The reduction simply guesses beforehand which challenge value the adversary will query. Therefore, it leads to a security loss that is exponential in the input length. In this work, we attempt to ask an ambitious question: is it possible to construct a constrained VRF which satisfies a more stronger security compared with the selective security?

In this work, we propose a novel construction based on the bilinear maps. Inspired by the constrained PRFs of Hohenberber et al. [9], we construct a VRF with constrained keys for any sets of polynomial size and define a new security named semiadaptive security. It allows the adversary to query the evaluation oracle before it outputs a challenge point, while the public key is returned to the adversary associated with the challenge evaluation. This definition is stronger than the selective security, which can be verified easily.

Our scheme is derived from the constructions of constrained PRFs given by Hohenberger et al. [9]. It is defined over a bilinear group, which contains three groups G_1, G_2, and G_T with composite order $N = pq$, equipped with bilinear maps $e : G_1 \times G_2 \longrightarrow G_T$. The constrained VRFs map an input from $\{0, 1\}^{\ell(\lambda)}$ to an element of G_T. The secret key is a tuple $sk = (v, w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}), h)$, where $v \in G_1, w \in G_2, \gamma \longleftarrow \mathbb{Z}_N^2, \{d_{i,0}, d_{i,1}\}_{i=1}^n \longleftarrow \mathbb{Z}_N^2, h : \{0, 1\}^{\ell(\lambda)} \longrightarrow \{0, 1\}^{n(\lambda)}$ is an admissible hash function. VRFs are defined as

$$F(sk, x) := e\left(v^{\prod_{i=1}^n d_{i,h(x)_i}}, w^\gamma\right), \tag{1}$$

associated with a proof

$$P(sk, x) := v^{\prod_{i=1}^n d_{i,h(x)_i}}, \tag{2}$$

where $h(x)_i$ is the ith bit of $h(x)$.

In order to verify the correctness of evaluation, we define a public key as $pk = (w, w^\gamma, i\mathcal{O}(\mathcal{C}))$, where $i\mathcal{O}(\mathcal{C})$ is an obfuscation of a circuit which takes a point x as input and outputs an element $D(x) := e(v^{\prod_{i=1}^n d_{i,h(x)_i}}, w)$ from G_T. The verifier only needs to check $e(P(sk, x), w) = D(x)$ and $e(P(sk, x), w^\gamma) = F(sk, x)$. The constrained key is an obfuscation of a circuit that has the secret key sk and the constrained set S hardwired in it. On input a value $x \notin S$, it outputs $(F(sk, x), P(sk, x))$. While this solution would work only if the obfuscator achieves a black box obfuscation definition [10], there is no reason to believe that an indistinguishability obfuscator would necessarily hide the secret key sk.

We solve this problem by a new technique which was introduced by Hohenberger [9]. We divide the domain into two disjoint sets by the admissible hash function: computable set and challenge set. The proportion of computable set in the domain is about $1 - 1/Q(\lambda)$, and the proportion of challenge set in the domain is about $1/Q(\lambda)$, where $Q(\lambda)$ is the number of queries made by the adversary. In the evaluation queries before the adversary outputs the challenge point, we use the secret key sk to answer the evaluation query x and abort the experiment if x belonged to the challenge set. After the adversary outputs a challenge point x^*, we use a freshly chosen secret key sk' to answer the evaluation queries. Via a hybrid argument, we reduce weak Bilinear Diffie–Hellman Inversion (BDHI) assumption to the pseudorandomness of constrained VRFs.

1.1. Related Works. Lysyanskaya [11] gave a construction of VRFs in bilinear groups, but the size of proofs and keys is linear in input size, which may be undesirable in resource constrained user. Dodis and Yampolskiy [12] gave a simple and efficient construction of VRFs based on bilinear mapping. Their VRFs' proofs and keys have constant size, but it is only suitable for small input spaces. Hohenberger and Waters [13] presented the first VRFs for exponentially large input spaces under a noninteractive assumption. Abdalla et al. [14] showed a relation between VRFs and identity-based key encapsulation mechanisms and proposed a new VRF-suitable identity-based key encapsulation mechanism from the decisional ℓ–weak Bilinear Diffie–Hellman Inversion assumption.

Fuchsbauer et al. [15] studied the adaptive security of the GGM construction for constrained PRFs and gave a new reduction that only loses a quasipolynomial factor $q^{O(\log \lambda)}$, where q is the number of adversary's queries. Hofheinz et al. [16] gave a new constrained PRF construction for circuit that has polynomial reduction to indistinguishability obfuscation in the random oracle model.

Kiayias et al. [17] introduced a novel cryptographic primitive called delegatable pseudorandom function, which enables a proxy to evaluate a pseudorandom function on a strict subset of its domain using a trapdoor derived from the delegatable PRF's secret key. Boyle et al. [18] introduced functional PRFs which can be seen as constrained PRFs. In functional PRFs, in addition to a master secret key, there are other secret keys for a function f, which allows one to evaluate the pseudorandom function on any y for which there exists an x such that $f(x) = y$. Chandran et al. [19] showed constructions of selectively secure constrained VRFs for the class of all polynomial-sized circuits.

Notations. In what follows, we will denote with $\lambda \in \mathbb{N}$ a security parameter. We say $\mathrm{negl}(\lambda)$ is negligible if $|\mathrm{negl}(\lambda)| < 1/\mathrm{poly}(\lambda)$ holds for all polynomials $\mathrm{poly}(\lambda)$ and all sufficiently large λ. Denote PPT as probabilistic polynomial time. Denote $[n]$ as the set $\{1, \ldots, n\}$.

2. Preliminaries

We first give a definition of admissible hash functions which is introduced by Boneh and Boyen [20].

Definition 1 (see [20]). Let ℓ, n, and θ be efficiently computable univariate polynomials. An efficiently computable function $h : \{0,1\}^{\ell(\lambda)} \longrightarrow \{0,1\}^{n(\lambda)}$ and an efficient randomized algorithm AdmSample are θ–admissible if for any $u \in \{0,1,\perp\}^{n(\lambda)}$, define $H_u : \{0,1\}^{\ell(\lambda)} \longrightarrow \{0,1\}$ as follows: $H_u(x) = 0$ if for all $1 \le j \le n(\lambda)$ and $h(x)_j \neq u_j$, else $H_u(x) = 1$. For any efficiently computable polynomial $Q(\lambda)$, $\forall x_1, \ldots, x_{Q(\lambda)}$, $z \in \{0,1\}^{\ell(\lambda)}$, where $z \neq x_i$, $\forall i \in [Q(\lambda)]$, we have that

$$\Pr\left[\forall i \le Q(\lambda), H_u(x_i) = 1 \wedge H_u(z) = 0\right] \ge 1/\theta(Q(\lambda)), \quad (3)$$

where the probability is taken only over $u \longleftarrow \mathrm{AdmSample}(1^\lambda, Q(\lambda))$.

Next, we present the formal definition of indistinguishability obfuscation following the syntax of Garg et al. [21].

Definition 2 (indistinguishability obfuscation (iO)). A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}$ if the following holds:

(i) Correctness: for all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, and for all inputs x, we have

$$\Pr\left[C'(x) = C(x) : C' \longleftarrow i\mathcal{O}(\lambda, C)\right] = 1. \quad (4)$$

(ii) Indistinguishability: for any (not necessarily uniform) PPT distinguisher $\mathrm{Samp}, \mathcal{D}$, there exists a negligible function negl such that the following holds: if $\Pr[\forall x, C_0(x) = C_1(x); (C_0, C_1, \sigma) \longleftarrow \mathrm{Samp}(1^\lambda)] > 1 - \mathrm{negl}(\lambda)$, then

$$\begin{aligned}&\left|\Pr\left[\mathcal{D}(\sigma, i\mathcal{O}(\lambda, C_0)) = 1 : (C_0, C_1, \sigma) \longleftarrow Samp\left(1^\lambda\right)\right]\right.\\ &\left. - \Pr\left[\mathcal{D}(\sigma, i\mathcal{O}(\lambda, C_1)) = 1 : (C_0, C_1, \sigma) \longleftarrow Samp\left(1^\lambda\right)\right]\right|\\ &\le \mathrm{negl}(\lambda).\end{aligned} \quad (5)$$

2.1. Assumptions. Let $\mathcal{G}$ be a PPT group algorithm that takes a security parameter 1^λ as input and outputs as tuple $(N, G_p, G_q, G_1, G_2, G_T, e)$, in which p and q are independent uniform random λ–bit primes. G_1, G_2, and G_T are groups of order $N = pq$, $e : G_1 \times G_2 \longrightarrow G_T$ is a bilinear map, and G_p and G_q are the subgroups of G_1 with the order p and q, respectively.

The subgroup decision assumption [22] in the bilinear group is said that the uniform distribution on G_1 is computationally indistinguishable from the uniform distribution on a subgroup of G_p or G_q.

Assumption 1 (subgroup hiding for composite order bilinear groups). Let $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^\lambda)$ and $b \longleftarrow \{0,1\}$. Let $T \longleftarrow G_1$ if $b = 0$, else $T \longleftarrow G_p$. The advantage of algorithm $\mathcal{A}$ in solving the subgroup decision problem is defined as

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{SGH}} = \left|\Pr\left[b \longleftarrow \mathcal{A}\left(N, G_p, G_q, G_1, G_2, G_T, e, T\right)\right] - 1/2\right|. \quad (6)$$

We say that the subgroup decision problem is hard if for all PPT $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{SGH}}$ is negligible in λ.

Assumption 2 (weak Bilinear Diffie–Hellman Inversion). Let $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^\lambda)$, $g_1 \longleftarrow G_1, a \longleftarrow \mathbb{Z}_N^*$, and $g_2 \longleftarrow G_2, \gamma \longleftarrow \mathbb{Z}_N^*$. Let $D = (N, G_p, G_q, G_1, G_2, G_T, e, g_1, g_1^a, \ldots, g_1^{a^{n-1}}, g_2, g_2^\gamma)$. Let $T = e(g_1^{a^n}, g_2^\gamma)$ if $b = 0$, else

$T \longleftarrow G_T$. The advantage of algorithm $\mathcal{A}$ in solving the problem is defined as

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{BDHI}} = |\Pr[b \longleftarrow \mathcal{A}(D, T)] - 1/2|. \quad (7)$$

We say that the weak bilinear Diffie–Hellman inversion problem is hard if for all PPT $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{BDHI}}$ is negligible in λ.

Chase et al. [22] showed that many q–type assumptions can be implied by subgroup hiding in bilinear groups of composite order.

3. Definition

We recall the definition of constrained VRFs which was given by Fuchsbauer [7].

Let $F : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$ be an efficiently computable function, where $\mathcal{K}$ is the key space, $\mathcal{X}$ is the input domain, and $\mathcal{Y}$ is the range. F is said to be constrained VRFs with regard to a set $S \subset \mathcal{X}$ if there exists a constrained key space $\mathcal{K}'$, a proof space $\mathcal{P}$, and four algorithms (Setup, Constrain, Prove, and Verify) :

(i) $\mathrm{Setup}(1^\lambda) \longrightarrow (pk, sk)$: it is a PPT algorithm that takes the security parameter λ as input and outputs a pair of keys (pk, sk), a description of the key space $\mathcal{K}$, and a constrained key space $\mathcal{K}'$

(ii) $\mathrm{Constrain}(sk, S) \longrightarrow sk_S$: this algorithm takes the secret key sk and a set $S \subset \mathcal{X}$ as input and outputs a constrained key $sk_S \in \mathcal{K}'$

(iii) $\mathrm{Prove}(sk_S, x) \longrightarrow (y, \pi)$ or $(\perp, \perp)$: this algorithm takes the constrained key sk_S and a value x as input and outputs a pair $(y, \pi) \in \mathcal{Y} \times \mathcal{P}$ of a function value and a proof if $x \notin S$, else outputs $(\perp, \perp)$

(iv) $\mathrm{Verify}(pk, x, y, \pi) \longrightarrow \{0, 1\}$: this algorithm takes the public key pk, an input x, a function value y, and a proof π as input and outputs a value in $\{0, 1\}$, where "1" indicates that $y = F(sk, x)$

3.1. Provability. For all $\lambda \in \mathbb{N}$, $(pk, sk) \longleftarrow \mathrm{Setup}(1^\lambda)$, $S \subset \mathcal{X}$, $sk_S \longleftarrow \mathrm{Constrain}(sk, S)$, $x \in \mathcal{X}$, and $(y, \pi) \longleftarrow \mathrm{Prove}(sk_S, x)$, it holds that

(i) If $x \notin S$, then $y = F(sk, x)$ and $\mathrm{Verify}(pk, x, y, \pi) = 1$

(ii) If $x \in S$, then $(y, \pi) = (\perp, \perp)$

3.2. Uniqueness. For all $\lambda \in \mathbb{N}$, $(pk, sk) \longleftarrow \mathrm{Setup}(1^\lambda)$, $x \in \mathcal{X}$, $y_0, y_1 \in \mathcal{Y}$, and $\pi_0, \pi_1 \in \mathcal{P}$, one of the following conditions holds:

(i) $y_0 = y_1$,

(ii) $\mathrm{Verify}(pk, x, y_0, \pi_0) = 1$, or

(iii) $\mathrm{Verify}(pk, x, y_1, \pi_1) = 1$,

which implies that for every x there is only one value y such that $F(sk, x) = y$.

3.3. Pseudorandomness. We consider the following experiment $\mathrm{Exp}_{\mathcal{A}}^{\mathrm{VRF}}(1^\lambda, b)$ for $\lambda \in \mathbb{N}$:

(i) The challenger first chooses $b \longleftarrow \{0, 1\}$ and then generates (pk, sk) by running the algorithm $\mathrm{Setup}(1^\lambda)$ and returns pk to the adversary $\mathcal{A}$

(ii) The challenger initializes two sets V and E and sets $V := \emptyset, E := \emptyset$, where V will contain the points that the adversary $\mathcal{A}$ cannot evaluate and E contains the points at which the adversary queries the evaluation oracle

(iii) The adversary $\mathcal{A}$ is given the following oracle:

(1) Constrain: on input a set $S \subset \mathcal{X}$, if $V \cap S \neq \emptyset$, return $sk_S \longleftarrow \mathrm{Constrain}(sk, S)$ and set $V := V \cap S$; else return $\perp$

(2) Evaluation: given $x \in \mathcal{X}$, return $(F(sk, x)$ and $P(sk, x))$ and set $E := E \cup \{x\}$

(3) Challenge: on input $x^* \in \mathcal{X}$, if $x^* \in E$ or $x^* \notin V$, then it returns $\perp$. Else, it returns $F(sk, x^*)$ if $b = 0$, or it returns a random value from $\mathcal{Y}$ if $b = 1$

(iv) $\mathcal{A}$ outputs a bit b'; if $b = b'$, the experiment outputs 1

A constrained VRF is pseudorandomness if for all PPT adversary $\mathcal{A}$, it holds that

$$\left|\Pr\left[\mathrm{Exp}_{\mathcal{A}}^{\mathrm{VRF}}\left(1^\lambda, b\right) = 1\right] - 1/2\right| \leq \mathrm{negl}(\lambda). \quad (8)$$

3.4. Semiadaptive Security. We give a weak definition for pseudorandomness which is called semiadaptive security. It allows the adversary to query the evaluation before it outputs a challenge point, while the public key is returned to the adversary after the adversary commits a challenge point. In the selective security, the adversary must commit a challenge input at the beginning of the experiment. Therefore, we can find that if a scheme satisfies the semiadaptive security, it must satisfy selective security. Conversely, it may be not true.

3.5. Puncturable Verifiable Random Functions. Puncturable VRFs are a special class of constrained VRFs, in which the constrained set contains only one value, i.e., $S = \{x^*\}$. The properties of provability, uniqueness, and pseudorandomness are similar to the constrained VRFs. To avoid repetition, we omit the formal definitions.

4. Construction

In this section, we give our construction for puncturable VRFs. A puncturable VRF $F : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$ consists of four algorithms (Setup, Puncture, Prove, and Verify). The input domain is $\mathcal{X} \longleftarrow \{0, 1\}^\ell$, where $\ell = \ell(\lambda)$. The key space $\mathcal{K}$ and range space $\mathcal{Y}$ are defined as a part of the setup algorithm.

(i) $\mathrm{Setup}(1^\lambda) \longrightarrow (pk, sk)$: On input the security parameter 1^λ, run $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^\lambda)$ such that $e : G_1 \times G_2 \longrightarrow G_T$ and G_p and G_q

are subgroups of G_1. Let n, θ be polynomials such that there exists a θ–admissible hash function $h : \{0,1\}^{\ell(\lambda)} \longrightarrow \{0,1\}^{n(\lambda)}$.

The key space is $\mathcal{K} = G_1 \times G_2 \times \mathbb{Z}_N \times (\mathbb{Z}_N^2)^n$, the range is $\mathcal{Y} = G_T$, and the proof space is $\mathcal{P} = G_1$. The setup algorithm chooses $v \in G_1, w \in G_2, \gamma \in \mathbb{Z}_N$, and $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$ uniformly at random and sets $sk = (v, w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$. The public key contains an obfuscation of a circuit $\mathscr{C}_1$, where $\mathscr{C}_1$ is described in Figure 1. Note that $\mathscr{C}_1$ has $v, w, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}), h$ hardwired in it. Set $pk = (w, w^\gamma, i\mathcal{O}(\mathscr{C}_1))$, where $\mathscr{C}_1$ is padded to be of appropriate size. The puncturable VRF F is defined as follows. Let $h(x) = b_1, \ldots, b_n$, where $b_i \in \{0,1\}$. Then,

$$F(sk, x) = e\left(v^{\prod_{j=1}^{n} d_{j,b_j}}, w^\gamma\right), P(sk, x) = v^{\prod_{j=1}^{n} d_{j,b_j}}. \tag{9}$$

(ii) Puncture$(sk, x') \longrightarrow sk_{x'}$: This algorithm computes an obfuscation of a circuit $\mathscr{C}_2$ which is defined in Figure 2. Note that $\mathscr{C}_2$ has the secret key sk and the puncturable value x' hardwired in it. Set $sk_{x'} \longleftarrow i\mathcal{O}(\mathscr{C}_2)$ where $\mathscr{C}_2$ is padded to be of appropriate size.

(iii) Prove$(sk_{x'}, x) \longrightarrow (y, \pi)$ or $(\perp, \perp)$: The punctured key $sk_{x'}$ is a program that takes an ℓ–bit input x. We define

$$\text{Prove}(sk_{x'}, x) = sk_{x'}(x). \tag{10}$$

(iv) Verify$(pk, x, y, \pi) \longrightarrow \{0,1\}$: To verify $(x, y, \pi) \in \{0,1\}^{\ell(\lambda)} \times G_T \times G_1$ with regard to $pk = (w, w^\gamma, i\mathcal{O}(\mathscr{C}_1))$, compute $D(x) \coloneqq \mathscr{C}_1(x) = e(v^{\prod_{j=1}^{n} d_{j,b_j}}, w)$ and output 1 if the following equations are satisfied:

$$e(\pi, w) = D(x), \quad e(\pi, w^\gamma) = y. \tag{11}$$

4.1. Properties

4.1.1. Provability. From the definition of F and P, we observe that for $(pk, sk) \longleftarrow \text{Setup}(1^\lambda)$, $x \in \mathcal{X}, sk_{x'} \longleftarrow \text{Puncture}(sk, x')$, $(y, \pi) = \text{Prove}(sk_{x'}, x)$, if $x \neq x'$:

$$\begin{aligned} e(\pi, w) &= e\left(v^{\prod_{j=1}^{n} d_{j,b_j}}, w\right) = D(x), \\ e(\pi, w^\gamma) &= e\left(v^{\prod_{j=1}^{n} d_{j,b_j}}, w^\gamma\right) = y = F(sk, x). \end{aligned} \tag{12}$$

Therefore, we have Verify$(pk, x, y, \pi) = 1$. When $x = x'$, we can get that Prove$(sk_{x'}, x') = (\perp, \perp)$. This completes the proof of provability.

4.1.2. Uniqueness. Consider a public key $pk = (w, w^\gamma, i\mathcal{O}(\mathscr{C}_1))$, where $w \in G_2, \gamma \in \mathbb{Z}_N$, and $\mathscr{C}_1$ is described in Figure 1. Given a value $x \in \{0,1\}^{\ell(\lambda)}$ and two pair values (y_0, π_0) and $(y_1, \pi_1) \in G_T \times G_1$ that satisfy Verify$(pk, x, y_b, \pi_b) = 1$ for $b \in \{0,1\}$. We show that $y_0 = y_1$.

Input: a value x
Constants: $v, w, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}), h$

(1) compute $h(x) = b_1, \ldots, b_n$;
(2) output $D(x) = e(v^{\prod_{j=1}^{n} d_{j,b_j}}, w)$.

Figure 1: Circuit $\mathscr{C}_1$.

Input: a value x
Constants: the secret key sk, the puncturable tvalue $x' \in \{0,1\}^{\ell(\lambda)}$

(1) compute $h(x) = b_1, \ldots, b_n$;
(2) if $x = x'$, output $(\perp, \perp)$;
(3) else,output $y = e(v^{\prod_{j=1}^{n} d_{j,b_j}}, w^\gamma)$, $\pi = v^{\prod_{j=1}^{n} d_{j,b_j}}$.

Figure 2: Circuit $\mathscr{C}_2$.

From the verification equations, we observe that $e(\pi_b, w) = D(x)$ and $e(\pi_b, w^\gamma) = y_b$. Because $D(x) = e(v^{\prod_{j=1}^{n} d_{j,b_j}}, w)$, then $\pi_0 = \pi_1$. Therefore, we can get that $y_0 = e(\pi_0, w^\gamma) = e(\pi_1, w^\gamma) = y_1$.

4.2. Proof of Pseudorandomness. In this section, we prove that our construction is secure puncturable VRFs as defined in Section 3.

Theorem 1. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and the subgroup hiding assumption for composite order bilinear groups holds, then our construction described as above satisfies the semiadaptive security as defined in Section 3.*

Proof. To prove the above theorem, we first define a sequence of games where the first one is the original pseudorandomness security game and show that each adjacent games is computationally indistinguishable for any PPT adversary $\mathscr{A}$. Without loss of generality, we assume that the adversary $\mathscr{A}$ makes $Q = Q(\lambda)$ evaluation queries before outputting the challenge point, where $Q(\lambda)$ is a polynomial. We present a full description of each game and underline the changes from the present one to the previous one. Each such game is completely characterized by its key generation algorithm and its challenge answer. The differences between these games are summarized in Table 1. □

4.2.1. Game 1. The first game is the original security for our construction. Here, the challenger first chooses a puncturable VRF key. Then, $\mathscr{A}$ makes evaluation queries and finally outputs a challenge point. The challenger responds with either a PRF evaluation or a random value.

(1) The challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^\lambda)$, chooses $v \in G_1, w \in G_2, \gamma \in \mathbb{Z}_N, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$, and sets $sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^\gamma, i\mathcal{O}(\mathscr{C}_1))$. Then, the challenger flips a coin $\alpha \longleftarrow \{0,1\}$.

Table 1: The differences between each adjacent games.

Game	Key generation	Challenge answer
Game 1	$sk = (v, w, \gamma, \{(d_{i,0}, d_{i,1})\}_{i=1}^{n})$	$y_0 = e(v^{\prod_{j=1}^{n} d_{j,b_j^*}}, w^{\gamma})$
Game 2	$sk = (v, w, \gamma, \{(d_{i,0}, d_{i,1})\}_{i=1}^{n})$	If $H_u(x^*) = 1$, abort, else $y_0 = e(v^{\prod_{j=1}^{n} d_{j,b_j^*}}, w^{\gamma})$
Game 3	If $H_u(x) = 1$, $sk = (v, w, \gamma, \{(d_{i,0}, d_{i,1})\}_{i=1}^{n})$. Else, $sk' = (v_1, w, \gamma, \{(e_{i,0}, e_{i,1})\}_{i=1}^{n})$	If $H_u(x^*) = 1$, abort, else $y_0 = e(v_1^{\prod_{j=1}^{n} e_{j,b_j^*}}, w^{\gamma})$
Game 4	If $H_u(x) = 1$, $sk = (v, w, \gamma, \{(d_{i,0}, d_{i,1})\}_{i=1}^{n})$. Else, $sk' = (v_1, w, \gamma, \{(e_{i,0}, e_{i,1})\}_{i=1}^{n})$ where $e_{i,b} = e'_{i,0} \cdot a$, if $H_u(x^*)_i = b$	If $H_u(x^*) = 1$, abort, else $y_0 = e(v_1^{\prod_{j=1}^{n} e_{j,b_j^*}}, w^{\gamma})$
Game 5	If $H_u(x) = 1$, $sk = (v, w, \gamma, \{(d_{i,0}, d_{i,1})\}_{i=1}^{n})$. Else, $sk' = (V, w, \gamma, \{(e'_{i,0}, e'_{i,1})\}_{i=1}^{n})$ where $V = (v_1, v_1^{a}, \ldots, v_1^{a^{n-1}})$	If $H_u(x^*) = 1$, abort, else $y_0 = e(v_1^{a^n \prod_{j=1}^{n} e'_{j,b_j^*}}, w^{\gamma})$
Game 6	If $H_u(x) = 1$, $sk = (v, w, \gamma, \{(d_{i,0}, d_{i,1})\}_{i=1}^{n})$. Else, $sk' = (V, w, \gamma, \{(e'_{i,0}, e'_{i,1})\}_{i=1}^{n})$ where $V = (v_1, v_1^{a}, \ldots, v_1^{a^{n-1}})$	If $H_u(x^*) = 1$, abort, else $y_0 = T^{\prod_{j=1}^{n} e'_{j,b_j^*}}$

(2) The adversary $\mathcal{A}$ makes a evaluation query $x_i \in \{0,1\}^{\ell(\lambda)}$. Then, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^{n} d_{j,b_j^i}}, w^{\gamma}), v^{\prod_{j=1}^{n} d_{j,b_j^i}})$.

(3) The adversary $\mathcal{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger computes $sk_{x^*} \longleftarrow i\mathcal{O}(\mathcal{C}_2)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, sets $y_0 = e(v^{\prod_{j=1}^{n} d_{j,b_j^*}}, w^{\gamma})$ and $y_1 \longleftarrow G_T$, and returns (pk, sk_{x^*}, y_α) to the adversary $\mathcal{A}$.

(5) The adversary $\mathcal{A}$ outputs a bit $\alpha\prime$ and wins if $\alpha' = \alpha$.

4.2.2. Game 2. This game is the same as the Game 1 except that a partitioning game is simulated. If the undesirable partition is queried, we abort the game. The partition game is defined as follows: the challenger samples a string $u \in \{0, 1, \perp\}^n$ by the algorithm AdmSample of admissible hash function and aborts if either there exists a evaluation query x such that $H_u(x) = 0$ or the challenge query x^* such that $H_u(x^*) = 1$.

(1) The challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^{\lambda})$, chooses $v \in G_1, w \in G_2, \gamma \in \mathbb{Z}_N$, $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$, and sets $sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^{\gamma}, i\mathcal{O}(\mathcal{C}_1))$. Then, the challenger flips a coin $\alpha \longleftarrow \{0,1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^{\lambda}, Q)$.

(2) The adversary $\mathcal{A}$ makes a evaluation query $x_i \in \{0,1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$ if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^{n} d_{j,b_j^i}}, w^{\gamma}), v^{\prod_{j=1}^{n} d_{j,b_j^i}})$.

(3) The adversary $\mathcal{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger computes $sk_{x^*} \longleftarrow i\mathcal{O}(\mathcal{C}_2)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, sets $y_0 = e(v^{\prod_{j=1}^{n} d_{j,b_j^*}}, w^{\gamma})$ and $y_1 \longleftarrow G_T$, and returns (pk, sk_{x^*}, y_α) to the adversary $\mathcal{A}$.

(5) The adversary $\mathcal{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 1. *For any PPT adversary $\mathcal{A}$, if $\mathcal{A}$ wins with advantage ϵ in Game 1, then it wins with advantage $\epsilon/\theta(Q(\lambda))$ in Game 2.*

Proof. The difference between Game 1 and Game 2 is that we add an abort condition in Game 2. From the θ−admissible of hash function h, we can get that for all $x_1, \ldots, x_Q, x^*$, $\Pr_{u \longleftarrow \mathrm{AdmSample}(1^{\lambda}, Q(\lambda))}[\forall i, H_u(x_i) = 1 \wedge H_u(x^*) = 0] \geq 1/\theta(Q(\lambda))$. The two experiments are equal if Game 2 does not abort. Therefore, if $\mathcal{A}$ wins with advantage ϵ in Game 1, then it wins with advantage at least $\epsilon/\theta(Q(\lambda))$ in Game 2. □

4.2.3. Game 3. This game is the same as the previous one except that the public key and the punctured key are obfuscation of two other circuits defined in Figures 3 and 4, respectively. On inputs x such that $H_u(x) = 1$, the public key and the punctured key use the same secret key sk as before. However, if $H_u(x) = 0$, the public key and the punctured key use a different secret key $sk\prime$ which is randomly chosen from the key space. The detailed description is given as follows:

(1) The challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^{\lambda})$, chooses $v \in G_1, w \in G_2, \gamma \in \mathbb{Z}_N$, and $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$, and sets $sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^{\gamma}, i\mathcal{O}(\mathcal{C}_1))$. Then, the challenger flips a coin $\alpha \longleftarrow \{0,1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^{\lambda}, Q)$.

(2) The adversary $\mathcal{A}$ makes a evaluation query $x_i \in \{0,1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$ if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes

Input: a value x
Constants: v, w^{γ}, $(d_{1,0}, d_{1,1})$, ...,$(d_{n,0}, d_{n,1})$, h, the secret key $sk' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, and the random string $u \in \{0, 1, \perp\}^n$

(a) compute $h(x) = b_1, \ldots, b_n$;
(b) if $H_u(x) = 0$, then output $D(x) = e(v_1^{\prod_{j=1}^{n} d_{j,b_j}}, w)$;
(c) else, output $D(x) = e(v^{\prod_{j=1}^{n} d_{j,b_j}}, w)$

Figure 3: Circuit $\mathscr{C}_3$.

Input: a value x
Constants: the secret key sk, the puncturable value $x^* \in \{0, 1\}^{\ell(\lambda)}$, the secret key $sk' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, and the random string $u \in \{0, 1, \perp\}^n$

(a) compute $h(x) = b_1, \ldots, b_n$;
(b) if $x = x^*$, output $(\perp, \perp)$;
(c) else if $H_u(x) = 0$, output $y = e(v^{\prod_{j=1}^{n} e_{j,b_j}}, w^{\gamma})$, $\pi = v_1^{\prod_{j=1}^{n} e_{j,b_j}}$;
(d) if $H_u(x) = 1$, output $y = e(v^{\prod_{j=1}^{n} d_{j,b_j}}, w^{\gamma})$, $\pi = v^{\prod_{j=1}^{n} d_{j,b_j}}$.

Figure 4: Circuit $\mathscr{C}_4$.

$h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^{n} d_{j,b_j^i}}, w^{\gamma}), v^{\prod_{j=1}^{n} d_{j,b_j^i}})$.

(3) The adversary $\mathscr{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger chooses $v_1 \in G_1$, $(e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}) \in \mathbb{Z}_N^2$, sets $sk' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, computes $pk' = (w, w^{\gamma}, i\mathcal{O}(\mathscr{C}_3))$, $sk_{x^*} \longleftarrow i\mathcal{O}(\mathscr{C}_4)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, and sets $y_0 = e(v_1^{\prod_{j=1}^{n} e_{j,b_j^*}}, w^{\gamma})$ and $y_1 \longleftarrow G_T$. Then, it returns $(pk', sk_{x^*}, y_{\alpha})$ to the adversary $\mathscr{A}$.

(5) The adversary $\mathscr{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 2. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and the assumption 1 holds, Game 2 and Game 3 are computationally indistinguishable.*

This proof is given in Section 4.3.

4.2.4. Game 4. This game is the same as the previous one except that the generation way of secret key sk' is different. We make some elements of secret key sk' to contain a factor a, for use on inputs x where $H_u(x) = 0$. The detailed description is given as follows:

(1) The challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^{\lambda})$, chooses $v \in G_1, w \in G_2, \gamma \in \mathbb{Z}_N$, $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$, and sets $sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^{\gamma}, i\mathcal{O}(\mathscr{C}_1))$. Then, the challenger flips a coin $\alpha \longleftarrow \{0, 1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^{\lambda}, Q)$.

(2) The adversary $\mathscr{A}$ makes a evaluation query $x_i \in \{0, 1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$ if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^{n} d_{j,b_j^i}}, w^{\gamma}), v^{\prod_{j=1}^{n} d_{j,b_j^i}})$.

(3) The adversary $\mathscr{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger chooses $v_1 \in G_1$, $a \in \mathbb{Z}_N$, $(e'_{1,0}, e'_{1,1}), \ldots, (e'_{n,0}, e'_{n,1}) \in \mathbb{Z}_N^2$. Set $e_{i,b} = e'_{i,b} \cdot a$, if $h(x^*)_i = b$, else $e_{i,b} = e'_{i,b}$. Let $sk' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, compute $pk' = (w, w^{\gamma}, i\mathcal{O}(\mathscr{C}_3))$, $sk_{x^*} \longleftarrow i\mathcal{O}(\mathscr{C}_4)$, and $h(x^*) = b_1^*, \ldots, b_n^*$ and set $y_0 = e(v_1^{\prod_{j=1}^{n} e_{j,b_j^*}}, w^{\gamma})$ and $y_1 \longleftarrow G_T$. Then, it returns $(pk', sk_{x^*}, y_{\alpha})$ to the adversary $\mathscr{A}$.

(5) The adversary $\mathscr{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 3. *The outputs of Game 3 and Game 4 are statistically indistinguishable.*

Proof. Recall that the difference between Game 3 and Game 4 is the manner in which $\{e_{i,b}\}_{i \in [n], b \in \{0,1\}}$ are chosen. In Game 3, $\{e_{i,b}\}_{i \in [n], b \in \{0,1\}}$ are chosen randomly from $\mathbb{Z}_N$, while in Game 4, the challenger first chooses $e'_{i,b} \longleftarrow \mathbb{Z}_N$ and $a \longleftarrow \mathbb{Z}_N$ and sets $e_{i,b} = e'_{i,b} \cdot a$, if $h(x^*)_i = b$, else $e_{i,b} = e'_{i,b}$. Since $a \in \mathbb{Z}_N$ which is invertible with overwhelming probability, $e_{i,b} = e'_{i,b} \cdot a$ is a uniform element in $\mathbb{Z}_N$. Hence, the two experiments are statistically indistinguishable. □

4.2.5. Game 5. This game is the same as the previous one except that the hardwire of circuits $\mathscr{C}_3$ and $\mathscr{C}_4$ is changed. The two circuits contain some constants $\{v_1^{a^i}\}_{i=1}^{n-1}$. When $H_u(x) = 0$, the related function values are computed using

the constants $\{v_1^{a^i}\}_{i=1}^n$. The detailed description is given as follows:

(1) The challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^\lambda)$, chooses $v \in G_1, w \in G_2, \gamma \in \mathbb{Z}_N$, and $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$, and sets $sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^\gamma, i\mathcal{O}(\mathcal{C}_1))$. Then, the challenger flips a coin $\alpha \longleftarrow \{0,1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^\lambda, Q)$.

(2) The adversary $\mathcal{A}$ makes an evaluation query $x_i \in \{0,1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$ if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^n d_{j,b_j^i}}, w^\gamma), v^{\prod_{j=1}^n d_{j,b_j^i}})$.

(3) The adversary $\mathcal{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger chooses $v_1 \in G_1, a \in \mathbb{Z}_N, (e'_{1,0}, e'_{1,1}), \ldots, (e'_{1,0}, e'_{n,1}) \in \mathbb{Z}_N^2$. Set $\overrightarrow{V} = (v_1, v_1^a, \ldots, v_1^{a^{n-1}})$. Let $sk' = ((v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, compute $pk' = (w, w^\gamma, i\mathcal{O}(\mathcal{C}_5))$, $sk_{x^*} \longleftarrow i\mathcal{O}(\mathcal{C}_6)$, $h(x^*) = b_1^*, \ldots, b_n^*$, and $D_{x^*} = e(v_1^{a^n \prod_{j=1}^n e'_{j,b_j^*}}, w)$. Set $y_0 = e(v_1^{a^n \prod_{j=1}^n e'_{j,b_j^*}}, w)$ and $y_1 \longleftarrow G_T$, where the descriptions of $\mathcal{C}_5$ and $\mathcal{C}_6$ are given in Figures 5 and 6, respectively. Then, it returns $(pk', sk_{x^*}, y_\alpha)$ to the adversary $\mathcal{A}$.

(5) The adversary $\mathcal{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 4. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, Game 4 and Game 5 are computationally indistinguishable.*

Proof. We will introduce an intermediate experiment 4_A and prove that Game 4 and 4_A are computationally indistinguishable and Game 4_A and Game 5 are computationally indistinguishable.

The experiment 4_A is the same as Game 5 except that sk_{x^*} is generated by obfuscating the circuit $\mathcal{C}_4$ in Step 4. Assume that there exists a PPT adversary $\mathcal{A}$ that distinguishes the outputs of Game 4 and Game 4_A, we construct a PPT adversary $\mathcal{B}$ that breaks the $i\mathcal{O}$ security with the same probability. $\mathcal{B}$ runs Step 1 and Step 3 as in experiment 4. If the experiment does not abort, $\mathcal{B}$ chooses values to construct the circuits: $v_1 \in G_1, a \longleftarrow \mathbb{Z}_N, (e'_{1,0}, e'_{1,1}), \ldots, (e'_{n,0}, e'_{n,1}) \in \mathbb{Z}_N^2$. Set $e_{i,b} = e'_{i,b} \cdot a$, if $h(x^*)_i = b$, else $e_{i,b} = e'_{i,b}$. Let $sk' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, $\overrightarrow{V} = (v_1, v_1^a, \ldots, v_1^{a^{n-1}})$, and $sk'' = (v_1, w, \gamma, (e'_{1,0}, e'_{1,1}), \ldots, (e'_{n,0}, e'_{n,1}))$. $\mathcal{B}$ constructs circuits $C_0 = \mathcal{C}_3$ and $C_1 = \mathcal{C}_5$, where sk' is replaced by sk''. Then, he sends C_0 and C_1 to the $i\mathcal{O}$ challenger and gets $pk' = i\mathcal{O}(C_\beta)$. $\mathcal{B}$ computes $sk_{x^*} = i\mathcal{O}(\mathcal{C}_4), h(x^*) = b_1^*, \ldots, b_n^*$, $y_0 = e(v_1^{\prod_{j=1}^n e_{j,b_j^*}}, w^\gamma)$, and $y_1 \longleftarrow G_T$ and returns $(pk', sk_{x^*}, y_\alpha)$ to the adversary $\mathcal{A}$. $\mathcal{A}$ outputs α', if $\alpha = \alpha'$ and $\mathcal{B}$ outputs 0, else outputs 1. The circuits C_0 and C_1 have identical functionality. We observe that for any string x, $h(x) = b_1, \ldots, b_n$:

(i) For any $x \in \{0,1\}^\ell$ such that $H_u(x) = 1$, both circuits output $D(x) = e(v^{\prod_{j=1}^n d_{j,b_j}}, w)$

(ii) For any $x \in \{0,1\}^\ell$ such that $H_u(x) = 0, x \neq x^*$, we have $C_0(x) = (v_1^{\prod_{j=1}^n e_{j,b_j}}, w) = ((v_1^{a^{n(x)}})^{\prod_{j=1}^n e'_{j,b_j}}, w) = C_1(x)$, where $n(x) = |\{i : b_i = b_i^*\}|$

(iii) For any $x \in \{0,1\}^\ell$ such that $H_u(x) = 0, x = x^*$, we have $C_0(x) = (v_1^{\prod_{j=1}^n e_{j,b_j}}, w) = D_{x^*} = C_1(x)$

Therefore, if there exists an adversary $\mathcal{A}$ that distinguishes the outputs of Game 4 and Game 4_A with advantage ϵ, then there exists an adversary $\mathcal{B}$ that breaks the security of $i\mathcal{O}$ with the same advantage.

The indistinguishability of Game 4_A and Game 5 is similar to the previous one (Game 4 and Game 4_A). □

4.2.6. Game 6. This game is the same as the previous one except that $e(v_1^{a^n}, w^\gamma)$ is replaced by a random element from G_T. Formally, the challenger chooses a random element $T \longleftarrow G_T$, and uses $y_0 = T^{\prod_{j=1}^n e'_{j,b_j^*}}$ to replace $y_0 = e(v_1^{a^n \prod_{j=1}^n e'_{j,b_j^*}}, w^\gamma)$.

Lemma 5. *If there exists an adversary $\mathcal{A}$ that distinguishes Game 5 and Game 6 with advantage ϵ, then there exists an adversary $\mathcal{B}$ that breaks assumption 2 with advantage ϵ.*

Proof. We observe that the difference between Game 5 and Game 6 is that the element $e(v_1^{a^n}, w^\gamma)$ in Game 5 is replaced by a random element in Game 6. $\mathcal{B}$ receives an instance $(N, G_p, G_1, G_1, G_2, G_T, e, g_1, g_1^a, \ldots, g_1^{a^{n-1}}, g_2, g_2^\gamma, T)$, where T is either equal to $e(g_1^{a^n}, g_2^\gamma)$ or a random element of G_T. Then, $\mathcal{B}$ simulates Game 5 except that $y_\alpha = T^{\prod_{j=1}^n e'_{j,b_j^*}}$. $\mathcal{A}$ outputs α',. If $\alpha = \alpha'$, $\mathcal{B}$ outputs 0, which indicates that $T = e(g_1^{a^n}, g_2^\gamma)$; else, $\mathcal{B}$ outputs 1, which implies that T is a random element from G_T.

We observe that both y_0 and y_1 are chosen randomly from G_T in Game 6. This completes the proof of Theorem 1. □

4.3. Proof of Lemma 2. The major difference between Game 2 and Game 3 is the 'challenge partition' inputs x where $H_u(x) = 0$. Therefore, in order to show that for any PPT adversary $\mathcal{A}$, the outputs of Game 2 and Game 3 are indistinguishable; we give a sequence of subexperiments Game 2_A to Game 2_F and prove that any PPT attacker's advantage in each game must be negligible close to the previous one. We omit the previous experiment Game 2 and describe the intermediate experiments. In the first game, we change the secret key such that the circuit computes the output in a different manner and the output is the same as in the original circuit. Next, using the weak bilinear Diffie–Hellman inversion assumption, we modify the constants hardwired in

Input: a value x
Constants: v,w$^{\gamma}$, $(d_{1,0}, d_{1,1}), \ldots,(d_{n,0}, d_{n,1})$, h, the puncturable point $x^* \in \{0,1\}^{\ell(\lambda)}$, the vector $\boldsymbol{V} = (v_1, v_1^{a}, \ldots, v_1^{a^{n-1}})$, the secret key $sk' = (v_1, w, \gamma, (e'_{1,0}, e'_{1,1}), \ldots,(e'_{n,0}, e'_{n,1}))$, the random string $u \in \{0,1,\perp\}^n$, and the value Dx^*
(a) compute $h(x) = b_1, \ldots, b_n$, and $h(x^*) = b_1^*, \ldots, b_n^*$, let $n(x) = |\{i : b_i = b_i^*\}|$;
(b) if $H_u(x) = 0, x \neq x^*$, then output $D(x) = e((v_1^{a^{n(x)}})^{\Pi_{j=1}^n e_{j,b_j}}, w)$;
(c) if $H_u(x) = 0, x = x^*$, then output Dx^*;
(d) if $H_u(x) = 1$, output $D(x) = e(v^{\Pi_{j=1}^n d_{j,b_j}}, w)$.

Figure 5: Circuit $\mathscr{C}_5$.

Input: a value x
Constants: the secret key sk, the puncturable point $x^* \in \{0,1\}^{\ell(\lambda)}$, the secret key $sk' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots,(e_{n,0}, e_{n,1}))$, the vector $\boldsymbol{V} = (v_1, v_1^{a}, \ldots, v_1^{a^{n-1}})$, and the random string $u \in \{0,1,\perp\}^n$
(a) compute $h(x) = b_1, \ldots, b_n$, and $h(x^*) = b_1^*, \ldots, b_n^*$, let $n(x) = |\{i : b_i = b_i^*\}|$;
(b) if $x = x^*$, output$(\perp, \perp)$;
(c) else if $H_u(x) = 0$, then output $y = e((v_1^{a^{n(x)}})^{\Pi_{j=1}^n d_{j,b_j}}, w^{\gamma})$, $\pi = (v_1^{a^{n(x)}})^{\Pi_{j=1}^n d_{j,b_j}}$;
(d) if $H_u(x) = 1$,output $y = e(v^{\Pi_{j=1}^n d_{j,b_j}}, w^{\gamma})$, $\pi = v^{\Pi_{j=1}^n d_{j,b_j}}$.

Figure 6: Circuit $\mathscr{C}_6$.

the program such that the output of all challenge partition inputs is changed. Essentially, a different base for the challenge partition is used in the two programs, respectively. Finally, using Subgroup Hiding Assumption and Chinese Remainder Theorem, we can change the exponents for the challenge partition and ensure that the original circuit (in Game 2) and final circuit (in Game 3) use different secret keys for the challenge partition.

4.3.1. Game 2_A. In this game, we modify the secret key $d_{i,b}$ for $j \in \{1, \ldots, n\}$ and $b \in \{0,1\}$. It is easy to verify that the two experiments are statistically indistinguishable. The detailed description is given as follows:

(1) The challenger flips a coin $\alpha \longleftarrow \{0,1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^{\lambda}, Q)$. Then, the challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathscr{G}(1^{\lambda})$, chooses $v \in G_1$, $w \in G_2, a, \gamma \in \mathbb{Z}_N$, and $(d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,1}) \in \mathbb{Z}_N^2$, sets $d_{i,b} = d'_{i,b}$ if $u_i = b$, else $d_{i,b} = a \cdot d'_{i,b}$, and $sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^{\gamma}, i\mathcal{O}(\mathscr{C}_1))$.

(2) The adversary $\mathscr{A}$ makes an evaluation query $x_i \in \{0,1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$, if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^n d_{j,b_j^i}}, w^{\gamma}), v^{\prod_{j=1}^n d_{j,b_j^i}})$.

(3) The adversary $\mathscr{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger computes $sk_{x^*} \longleftarrow i\mathcal{O}(\mathscr{C}_2)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, sets $y_0 = e(v^{\prod_{j=1}^n d_{j,b_j^*}}, w^{\gamma})$ and $y_1 \longleftarrow G_T$, and returns $(pk, sk_{x^*}, y_{\alpha})$ to the adversary $\mathscr{A}$.

(5) The adversary $\mathscr{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 6. *The outputs of Game 2 and Game* 2_A *are statistically indistinguishable.*

Proof. We observe the difference between Game 2 and Game 2_A is the manner in which $d_{i,b}$ are chosen. In Game 2, $d_{i,b}$ are chosen randomly from $\mathbb{Z}_N$, while in Game 2_A, the challenger first chooses $d'_{i,b} \longleftarrow \mathbb{Z}_N$ and $a \longleftarrow \mathbb{Z}_N$ and sets $d_{i,b} = d'_{i,b}$, if $u_i = b$, else $d_{i,b} = d'_{i,b} \cdot a$. Since $a \in \mathbb{Z}_N$, which is invertible with overwhelming probability, $d_{i,b} = d'_{i,b} \cdot a$ is a uniformly random element in $\mathbb{Z}_N$. Therefore, the two experiments are statistically indistinguishable. □

4.3.2. Game 2_B. This game is the same as the previous one except the hardwire of the circuit is changed. The domain is divided into two disjoint sets by the admissible hash function. When $H_u(x) = 0$, all elements $d_{i,b}$ used to compute function values y contain a factor a. Therefore, the related function values can be computed by $v' = v^{a^n}$. On the other hand, only some elements $d_{i,b}$ used to compute function values y contain the factor a when $H_u(x) = 1$. Therefore, the related function values can only be computed by $(v, v^a, \ldots, v^{a^{n-1}})$.

(1) The challenger flips a coin $\alpha \longleftarrow \{0,1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^{\lambda}, Q)$. Let $m(x) = |\{i : u_i \neq h(x)_i\}|$. Then, the challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathscr{G}(1^{\lambda})$, chooses $v \in G_1, w \in G_2, a, \gamma \in \mathbb{Z}_N$, and $(d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,1}) \in \mathbb{Z}_N^2$, sets $d_{i,b} = d'_{i,b}$ if $u_i = b$, else $d_{i,b} = a \cdot d'_{i,b}$, and $D = ((d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,1}))$, $\overrightarrow{V} = (v, v^a, \ldots, v^{a^{n-1}})$, $v' = v^{a^n}$, and $pk = (w, w^{\gamma}, i\mathcal{O}(\mathscr{C}_7))$, where the description of $\mathscr{C}_7$ is given in Figure 7.

(2) The adversary $\mathscr{A}$ makes an evaluation query $x_i \in \{0,1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$, if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes

Input: a value x
Constants: the element $w \in G_2$, $v' \in G_1$, $D = ((d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,1}))$, the random string $u \in \{0, 1, \perp\}^n$, and $\boldsymbol{V} = (v, v^a, \ldots, v^{a^{n-1}})$
(a) compute $h(x) = b_1, \ldots, b_n$, and $m(x) = |\{i : u_i \neq h(x)_i\}|$;
(b) if $H_u(x) = 0$, output $D(x) = e((v')^{\Pi_{j=1}^n d_{j,b_j}}, w)$;
(c) if $H_u(x) = 1$, output $D(x) = e((v^{a^{m(x)}})^{\Pi_{j=1}^n d_{j,b_j}}, w)$.

Figure 7: Circuit $\mathscr{C}_7$.

$h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^n d_{j,b_j^i}}, w^\gamma), v^{\prod_{j=1}^n d_{j,b_j^i}})$.

(3) The adversary $\mathscr{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger computes $sk_{x^*} \longleftarrow i\mathcal{O}(\mathscr{C}_8)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, sets $y_0 = e(v^{\prod_{j=1}^n d_{j,b_j^*}}, w^\gamma)$ and $y_1 \longleftarrow G_T$, and returns (pk, sk_{x^*}, y_α) to the adversary $\mathscr{A}$, where the description of $\mathscr{C}_8$ is given in Figure 8.

(5) The adversary $\mathscr{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 7. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, Game 2_A and Game 2_B are computationally indistinguishable.*

Proof. The proof method is similar to Lemma 4. □

4.3.3. Game 2_C. This game is the same as the previous one except that v' is chosen randomly from G_1 in Step 1, and $y_0 = e((v')^{\prod_{j=1}^n d'_{j,b_j^*}}, w^\gamma)$ in Step 4.

Lemma 8. *If there exists an adversary $\mathscr{A}$ that distinguishes the Game 2_B and Game 2_C with advantage ϵ, then there exists an adversary $\mathscr{B}$ that breaks assumption 2 with advantage ϵ.*

Proof. We observe that the difference between Game 2_B and Game 2_C is that the term v^{a^n} is replaced by a random element of G_1. This proof is similar to the proof of Lemma 5. □

4.3.4. Game 2_D. This game is the same as the previous one except that v is chosen randomly from the subgroup G_p, and $v\prime$ is chosen randomly from the subgroup G_q in Step 1.

Lemma 9. *Assuming assumption 1 holds, Game 2_C and Game 2_D are computationally indistinguishable.*

Proof. We introduce an intermediate experiment 2_{C_1} and show that Game 2_{C_1} and 2_C are computationally indistinguishable. Similarly, Game 2_{C_1} and Game 2_D are computationally indistinguishable.

Game 2_{C_1} is the same as Game 2_C except that v is chosen from G_p. Suppose that there exists an adversary $\mathscr{A}$ which can distinguish Game 2_{C_1} and 2_C, we construct an adversary $\mathscr{B}$ that breaks assumption 1. $\mathscr{B}$ receives $(N, G_p, G_q, G_1, G_2, G_T, e, T)$, where $T \longleftarrow G_1$ or $T \longleftarrow G_p$. $\mathscr{B}$ sets $v = T$, chooses $v' \longleftarrow G_1, w \longleftarrow G_2, a, \gamma \xleftarrow{} \mathbb{Z}_N$, and $(d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,1}) \in \mathbb{Z}_N^2$, and computes V, D, and pk as in Game 2_C. Then, $\mathscr{B}$ runs the rest steps as in Game 2_C. At last, $\mathscr{A}$ outputs α', if $\alpha = \alpha'$ and $\mathscr{B}$ guesses $T \in G_1$, else $\mathscr{B}$ guesses $T \in G_p$. Note that $\mathscr{B}$ simulates exactly Game 2_C when $T \in G_1$, and $\mathscr{B}$ simulates exactly Game 2_{C_1} when $T \in G_p$. Therefore, if there exists an adversary $\mathscr{A}$ that distinguishes the outputs 2_C and 2_{C_1} with advantage ϵ, then there exists an adversary $\mathscr{B}$ that breaks the assumption 1. □

4.3.5. Game 2_E. This game is the same as the previous one except that the secret key is divided into two parts sk and sk'. If $H_u(x) = 0$, the related function values are computed by sk'. Else, the related function values are computed by sk.

(1) The challenger flips a coin $\alpha \longleftarrow \{0, 1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^\lambda, Q)$. Let $m(x) = |\{i : u_i \neq h(x)_i\}|$. Then, the challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathscr{G}(1^\lambda)$, chooses $v \in G_p, v' \in G_q, w \in G_2, a, \gamma \in \mathbb{Z}_N$, and $(d_{1,0}, d_{1,1}, \ldots, (d_{n,0}, d_{n,1})) \in \mathbb{Z}_N^2$, and sets $sk = (v, w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$, $sk' = (v', w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$, and $pk = (w, w^\gamma, i\mathcal{O}(\mathscr{C}_3))$.

(2) The adversary $\mathscr{A}$ makes an evaluation query $x_i \in \{0, 1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$ if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^n d_{j,b_j^i}}, w^\gamma), v^{\prod_{j=1}^n d_{j,b_j^i}})$.

(3) The adversary $\mathscr{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger computes $sk_{x^*} \longleftarrow i\mathcal{O}(\mathscr{C}_4)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, sets $y_0 = e((v')^{\prod_{j=1}^n d'_{j,b_j^*}}, w^\gamma)$ and $y_1 \longleftarrow G_T$, and returns (pk, sk_{x^*}, y_α) to the adversary $\mathscr{A}$.

(5) The adversary $\mathscr{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 10. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, Game 2_D and Game 2_E are computationally indistinguishable.*

Input: a value x
Constants: the puncturable value x^*, the element $w^\gamma \in G_2, v' G_t$, $D = ((d'_{1,0}, d'_{1,1}), \cdots, (d'_{n,0}, d'_{n,1}))$, the random string $u \in \{0, 1, \perp\}^n$, and $\boldsymbol{V} = (v, v^a, \ldots, v^{a^{n-1}})$
(a) compute $h(x) = b_1, \ldots, b_n$, and $m(x) = |\{i : u_i \neq h(x)_i\}|$;
(b) if $x = x^*$, output $(\perp, \perp)$;
(c) else if $H_u(x) = 0$, output $y = e((v')^{\Pi_{j=1}^n d'_{j,b_j}}, w^\gamma)$, $\pi = (v')^{\Pi_{j=1}^n d'_{j,b_j}}$;
(d) if $H_u(x) = 1$, output $D(x) = e((v^{a^{m(x)}})^{\Pi_{j=1}^n d'_{j,b_j}}, w^\gamma)$, $\pi = (v^{a^{m(x)}})^{\Pi_{j=1}^n d'_{j,b_j}}, w^\gamma$

Figure 8: Circuit $\mathscr{C}_8$.

Proof. We introduce two intermediate experiments 2_{D_1} and 2_{D_2} and show that Game 2_D and 2_{D_1} are computationally indistinguishable, Game 2_{D_1} and Game 2_{D_2} are computationally indistinguishable, and Game 2_{D_2} and Game 2_E are computationally indistinguishable.

First, we define the experiments 2_{D_1} and 2_{D_2}. In experiment 2_{D_1}, the challenger samples v, v', w, a, γ, and $d'_{i,b}$ as in Game 2_D, sets $d_{i,b} = d'_{i,b}$ if $u_i = b$, else $d_{i,b} = a \cdot d'_{i,b}$. Then, it answers the evaluation queries of $\mathscr{A}$ exactly as in Game 2_D. For the challenge queries, it sets $v'' = (v')^{1/a^n}$ and computes $y_0 = e((v'')^{\prod_{j=1}^n d_{j,b_j^*}}, w^\gamma)$ and $y_1 \longleftarrow G_T$. The public key pk is computed by the circuit $\mathscr{C}_3$, where $sk = (v, w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $sk' = (v'', w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$. The constrained key sk_{x^*} is computed as in Game 2_D.

Game 2_{D_2} is the same as the game 2_{D_1}, except that the constrained key sk_{x^*} is computed by the circuit $\mathscr{C}_4$. □

Claim 1. Assuming $\mathscr{A}$ is a secure indistinguishability obfuscator, Game α' and Game $\alpha' = \alpha$ are computationally indistinguishable.

Proof. We construct a PPT adversary $i\mathcal{O}$ that uses 2_A to break the security of 2_B. 2_C runs Step 1 and Step 3 as in Game v'. On receiving the challenge point G_1, $y_0 = e((v')^{\prod_{j=1}^n d'_{j,b_j^*}}, w^\gamma)$ sets $\mathscr{A}$ and 2_B as in 2_C and constructs circuits $\mathscr{B}$ and 2_B. Then, he sends 2_C to the v^{a^n} challenger and receives G_1. 2_D computes G_p as in Game v' and sends G_q to 2_C. 2_D returns 2_{C_1}, if 2_{C_1}, 2_C outputs 0, else outputs 1.

Next, we only show that the circuit 2_{C_1} and 2_D have the identical functionality. For any 2_{C_1} such that 2_C. For any G_p such that $\mathscr{A}$. Therefore, the two circuits are functionally equivalent. Hence, if there exists an adversary that can distinguish the two games, then we can construct an adversary 2_{C_1} that breaks the 2_C security. □

Claim 2. Assuming $\mathscr{B}$ is a secure indistinguishability obfuscator, Game $\mathscr{B}$ and Game $(N, G_p, G_q, G_1, G_2, G_T, e, T)$ are computationally indistinguishable.

Proof. The proof method is similar to the previous one. □

Claim 3. Game $T \longleftarrow G_1$ and Game $T \longleftarrow G_p$ are statistically indistinguishable.

Proof. The difference between Game $\mathscr{B}$ and Game $v = T$ is the chosen way of $v' \longleftarrow G_1, w \longleftarrow G_2, a, \gamma \in \mathbb{Z}_N$. In addition, $(d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,}) \in \mathbb{Z}_N^2$ is replaced by the value $\overrightarrow{V}$, D, and pk in Game 2_C. Since $\mathscr{B}$, a is invertible with overwhelming probability. Therefore, 2_C is a uniform element from $\mathscr{A}$ and α' is also a uniformly random element from $\alpha = \alpha'$ in Game $\mathscr{B}$. It follows that the two experiments are statistically indistinguishable. □

4.3.6. Game 2_F. This game is the same as the previous one except that the generation way of secret key sk' is different to the one of sk.

(1) The challenger flips a coin $\alpha \longleftarrow \{0, 1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^\lambda, Q)$. Let $m(x) = |\{i : u_i \neq h(x)_i\}|$. Then, the challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathscr{G}(1^\lambda)$, chooses $v \in G_p, v' \in G_q, w \in G_2, a, \gamma \in \mathbb{Z}_N$, $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$, and $(e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}) \in \mathbb{Z}_N^2$, and sets $sk = (v, w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$, $sk' = (v', w^\gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, and $pk = (w, w^\gamma, i\mathcal{O}(\mathscr{C}_3))$.

(2) The adversary $\mathscr{A}$ makes an evaluation query $x_i \in \{0, 1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$ if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^n d_{j,b_j^i}}, w^\gamma), v^{\prod_{j=1}^n d_{j,b_j^i}})$.

(3) The adversary $\mathscr{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger computes $sk_{x^*} \longleftarrow i\mathcal{O}(\mathscr{C}_4)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, sets $y_0 = e((v')^{\prod_{j=1}^n d'_{j,b_j^*}}, w^\gamma)$ and $y_1 \longleftarrow G_T$, and returns (pk, sk_{x^*}, y_α) to the adversary $\mathscr{A}$.

(5) The adversary $\mathscr{A}$ outputs a bit $\alpha\prime$ and wins if $\alpha' = \alpha$.

Lemma 11. *Game 2_E and Game 2_F are statistically indistinguishable.*

Proof. The only difference between Game 2_E and Game 2_F is the chosen way of the secret key sk and sk'. In Game 2_E, the challenger chooses $v \in G_p, v' \in G_q, w \in G_2, \gamma, d_{i,b} \in \mathbb{Z}_N$ and sets $sk = (v, w^\gamma, (d_{1,0}, d_{1,1})), \ldots, (d_{n,0}, d_{n,1}))$ and $sk' = (v', w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$. While the challenger chooses $d_{i,b}, e_{i,b} \in \mathbb{Z}_N$ and sets $sk = (v, w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $sk' = (v', w^\gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$ in Game 2_F. Using Chinese remainder theorem, the

distributions $\{x \bmod p, x \bmod q : x \longleftarrow \mathbb{Z}_N\}$ and $\{x \bmod p, y \bmod q : x, y \longleftarrow \mathbb{Z}_N\}$ are statistically indistinguishable. Therefore, Game 2_E and Game 2_F are statistically indistinguishable. □

Lemma 12. *Assuming assumption 1 holds, Game 2_F and Game 3 are computationally indistinguishable.*

Proof. The proof method is similar to Lemma 9. □

5. Constrained Verifiable Random Function

In this section, we give our construction of constrained verifiable random function with polynomial size of the constrained set. We embed the puncturable VRFs in the constrained VRFs. Informally, our algorithm works as follows: The setup algorithm is the same as the puncturable VRFs. The constrained key sk_S for the subset S is a circuit which has the secret key sk hardwired in it. On input a value x, the circuit computes the function value and proof by the puncturable VRFs if $x \notin S$. The verifiable algorithm is the same as the puncturable VRFs. When proving the pseudorandomness, we translate puncturable VRFs into constrained VRFs with polynomial size of the constrained set by means of hybrid argument. Once the adversary queries the constrained key for the polynomial set S_1, the challenger can guess the challenge point x^* with a probability of $1/|S_1|$. Subsequently, the secret key sk can be replaced by a constrained key sk_{x^*} of puncturable VRFs. Via a hybrid argument, we reduce pseudorandomness of puncturable VRFs to the pseudorandomness of constrained VRFs.

Let $F : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$ be a puncturable VRF (Setup, Puncture, Prove, and Verify), and $P : \mathcal{K} \times \mathcal{X} \longrightarrow G_1$ be a function of generation proof. We construct constrained VRFs (F.Setup, F.Constrain, F.Prove, and F.Verify) by invoking the puncturable VRFs:

(i) $F.\text{Setup}(1^\lambda) \longrightarrow (pk, sk)$: Run the algorithm $(pk_1, sk_1) \longleftarrow \text{Setup}(1^\lambda)$. Set $pk = pk_1$ and $sk = sk_1$.

(ii) $F.\text{Constrain}(sk, S) \longrightarrow sk_S$: This algorithm takes the secret key sk and the constrained set S as inputs, where $|S|$ = poly and computes an obfuscation of a circuit $\mathcal{C}_{sk,S}$ defined as in Figure 9. $\mathcal{C}_{sk,S}$ has the secret key, the function descriptions F and P, and the constrained set S hardwired in it. Sets $sk_S \longleftarrow i\mathcal{O}(C_{sk}, S)$ where $\mathcal{C}_{sk,S}$ is padded to be of appropriate size.

(iii) $F.\text{Prove}(sk_S, x) \longrightarrow (y, \pi)$ or $(\perp, \perp)$: The constrained key sk_S is a program that takes x as the input. Define $F.\text{Prove}(sk_S, x) = sk_S(x)$.

(iv) $F.\text{Verify}(pk, x, y, \pi) \longrightarrow \{0, 1\}$: This algorithm is the same as Verify.

The provability and uniqueness follow from the puncturable VRFs. We omit the detailed description. Next, we show that this construction satisfies the pseudorandomness defined in Section 3.

Theorem 2. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and (Setup, Puncture, Prove, and Verify) is a secure puncturable VRF. Then, the construction defined above satisfies the pseudorandomness.*

Proof. Without loss of generality, we assume the adversary makes q_1 evaluation queries and q_2 constrained queries. We present a full description of each game and underline the changes from the presented one to the previous one. □

5.1. Game 1. The first game is the original security game for our construction. Here, the challenger first chooses a pair constrained VRF key (pk, sk). Then, $\mathcal{A}$ makes evaluation queries and constrained key queries and outputs a challenge point. The challenger responds with either a VRF evaluation or a random element.

(i) The challenger chooses $b \longleftarrow \{0, 1\}$ and then generates (pk, sk) by running the algorithm F.Setup (1^λ)

(ii) The adversary makes evaluation queries or constrained queries:

(1) If $\mathcal{A}$ sends an evaluation query x_i, then output $(F(sk, x_i), P(sk, x_i))$

(2) If $\mathcal{A}$ sends a constrained key query for S_j, output the constrained key $sk_{S_j} \longleftarrow i\mathcal{O}(\mathcal{C}_{sk,S_j})$

(iii) $\mathcal{A}$ sends a challenge query x^* such that $x^* \neq x_i$ for all $i \leq q_1$ and $x^* \in S_j$ for all $j \leq q_2$. Then, the challenger sets $y_0 = F(sk, x^*)$ and $y_1 \longleftarrow \mathcal{Y}$ and outputs (y_b, pk)

(iv) $\mathcal{A}$ outputs b' and wins if $b = b'$

5.2. Game 2. This game is the same as the previous one except that we introduce an abort condition. When the adversary $\mathcal{A}$ makes the first constrained query S_1, the challenger guesses a challenge query $x' \in S_1$. If the last $q_2 - 1$ queries S_j does not contain x', the experiment aborts. In addition, the experiment aborts if $x' \neq x^*$, where x^* is the challenge query.

(i) The challenger chooses $b \longleftarrow \{0, 1\}$ and then generates (pk, sk) by running the algorithm F.Setup (1^λ).

(ii) The adversary makes evaluation queries or constrained queries: For the first constrained query S_1, the challenger chooses $x' \longleftarrow S_1$ and output $sk_{S_1} \longleftarrow i\mathcal{O}(\mathcal{C}_{sk,S_1})$. For all evaluation queries x_i before the first constrained query, the challenger outputs $(F(sk, x_i), P(sk, x_i))$. For all queries after the first constrained query, the challenger does as follows:

(1) If $\mathcal{A}$ sends an evaluation query x_i such that $x_i = x'$, the experiment aborts. Else, if $x_i \neq x'$, output $(F(sk, x_i), P(sk, x_i))$

Input: a value $x \in X$
Constants: the function description F and P, the secret key sk, the constrained set $S \in X$

1. if $S \in X$, output $(\perp, \perp)$;
2. else, output $y = F(sk, x)$, $\pi = P(sk, x)$.

Figure 9: Circuit $\mathscr{C}_{sk,S}$.

(2) If $\mathscr{A}$ sends a constrained key query for S_j such that $x_i \notin S_j$, the experiment aborts. Else, output $sk_{S_j} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk,S_j})$.

(iii) $\mathscr{A}$ sends a challenge query x^* such that $x^* \neq x_i$ for all $i \leq q_1$, and $x^* \in S_j$ for all $j \leq q_2$. If $x^* \neq x'$, the experiment aborts. Else, the challenger sets $y_0 = F(sk, x^*)$ and $y_1 \longleftarrow \mathscr{Y}$ and outputs (y_b, pk).

(iv) $\mathscr{A}$ outputs b' and wins if $b = b'$.

Lemma 13. *For any PPT adversary $\mathscr{A}$, if $\mathscr{A}$ wins with advantage ϵ in Game 1, then it wins with advantage $\varepsilon/|S_1|$ in Game 2.*

Proof. According the pseudorandomness defined in Section 3, the challenge point belongs to the constrained set. The two experiments are equal if Game 2 does not abort. Since the challenger guesses correctly with probability $1/|S_1|$, if $\mathscr{A}$ wins with advantage ϵ in Game 1, then it wins with advantage $\varepsilon/|S_1|$ in Game 2. □

5.3. Game 2_i. For $0 \leq i \leq q_2$, the experiment is the same as the previous one except that the constrained queries use $sk_{x'}$ instead of sk in the first i experiment. We observe that the Game 2_0 is equal to the Game 2.

(i) The challenger chooses $b \longleftarrow \{0,1\}$ and then generates (pk, sk) by running the algorithm F.Setup (1^λ).

(ii) The adversary makes evaluation queries or constrained queries: For the first constrained query S_1. The challenger chooses $x' \longleftarrow S_1$, computes $sk_{x'} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk_{x'},S_1})$ and $\pi\prime = \text{Puncture}(sk, x')$, and outputs $sk_{S_1} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk_{x'},S_1})$, where the description of the circuit $\mathscr{C}_{sk_{x'},S_1}$ is given in Figure 10. For all evaluation queries x_i before the first constrained query, the challenger outputs $(F(sk, x_i), P(sk, x_i))$. For all queries after the first constrained query, the challenger does as follows:

(1) If $\mathscr{A}$ sends an evaluation query x_i such that $x_i = x'$, the experiment aborts. Else, if $x_i \neq x'$, output $(F(sk, x_i), P(sk, x_i)) = \text{Prove}(sk_{x'}, x_i)$

(2) If $\mathscr{A}$ sends a constrained key query for S_j such that $x_i \notin S_j$, the experiment aborts. Else, if $j \leq i$ output $sk_{S_j} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk_{x'},S_j})$, else output $sk_{S_j} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk,S_j})$, where the description of the circuit $\mathscr{C}_{sk_{x'},S_j}$ is given in Figure 10.

(iii) $\mathscr{A}$ sends a challenge query x^* such that $x^* \neq x_i$ for all $i \leq q_1$ and $x^* \in S_j$ for all $j \leq q_2$. If $x^* \neq x'$, the experiment aborts. Else, the challenger sets $y_0 = F(sk, x^*)$ and $y_1 \longleftarrow \mathscr{Y}$, computes, and outputs (y_b, pk).

(iv) $\mathscr{A}$ outputs b' and wins if $b = b'$.

Lemma 14. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, Game 2_{i-1} and 2_i are computationally indistinguishable.*

Proof. We observe that the difference between Game 2_{i-1} and 2_i respond to the $i\prime$th constrained query. In Game 2_{i-1}, $sk_{S_i} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk,S_i})$, while in Game 2_i, $sk_{S_i} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk_{x'},S_i})$. In order to prove that the two games are indistinguishable, we only need to show that the circuit $\mathscr{C}_{sk,S_i}$ and $\mathscr{C}_{sk_{x'},S_i}$ are functionally identical.

(i) If $x \in S_i$, both circuits output $(\perp, \perp)$

(ii) For any input $x \notin S_i$, $\mathscr{C}_{sk,S_i}(x) = \text{Prove}(sk, x) = \text{Prove}(sk_{x',S_i}, x) = \mathscr{C}_{sk_{x'},S_i}(x)$

Therefore, by the security of $i\mathcal{O}$, the two experiments are indistinguishable. □

5.4. Game 3. This game is the same as game 2_{q_2} except that y_0 is replaced by a random element from $\mathscr{Y}$.

(i) The challenger chooses $b \longleftarrow \{0,1\}$ and then generates (pk, sk) by running the algorithm F.Setup (1^λ).

(ii) The adversary makes evaluation queries or constrained queries: For the first constrained query S_1, the challenger chooses $x' \longleftarrow S_1$ and output $sk_{S_1} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk,S_1})$. For all evaluation queries x_i before the first constrained query, the challenger outputs $(F(sk, x_i), P(sk, x_i))$. For all queries after the first constrained query, the challenger does as follows:

(1) If $\mathscr{A}$ sends an evaluation query x_i such that $x_i = x'$, the experiment aborts. Else, if $x_i \neq x'$, output $(F(sk, x_i), P(sk, x_i))$

(2) If $\mathscr{A}$ sends a constrained key query for S_j such that $x_i \notin S_j$, the experiment aborts. Else, output $sk_{S_j} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk,S_j})$

(iii) $\mathscr{A}$ sends a challenge query x^* such that $x^* \neq x_i$ for all $i \leq q_1$ and $x^* \in S_j$ for all $j \leq q_2$. If $x^* \neq x'$, the experiment aborts. Else, the challenger sets $y_0 \longleftarrow \mathscr{Y}$ and $y_1 \longleftarrow \mathscr{Y}$ and outputs (y_b, pk).

(iv) $\mathscr{A}$ outputs b' and wins if $b = b'$.

Input: a value $x \in X$
Constants: the function description F, the secret key $sk_{x'}$, the constrained set $S_j \in X$
1. if $x \in S_j$, output $(\perp, \perp)$;
2. else, output $(y, \pi) = \text{Prove}(sk_{x'}, x)$.

Figure 10: Circuit $\mathscr{C}_{sk_{x'}, S_j}$.

Lemma 15. *Assuming the puncturable VRFs are secure, Game* 2_{q_2} *and Game 3 are computationally indistinguishable.*

Proof. We prove that if there exists an adversary $\mathscr{A}$ that distinguishes the Game 2_{q_2} and Game 3, then there exists another adversary $\mathscr{B}$ that breaks the security of puncturable VRFs.

$\mathscr{B}$ can simulate perfect experiment for $\mathscr{A}$. For each evaluation query x before the first constrained key query, $\mathscr{B}$ sends x to the puncturable VRFs' challenger and returns (y, π) to $\mathscr{A}$. When $\mathscr{A}$ queries the constrained key S_1, $\mathscr{B}$ chooses $x' \in S_1$, sends x' to the challenger, and receives $(sk_{x'}, pk, \text{and } y)$. Then, $\mathscr{B}$ uses $sk_{x'}$ to respond the remaining queries. On receiving the challenge input x^*, $\mathscr{B}$ checks $x' = x^*$ and outputs y. $\mathscr{B}$ outputs the response of $\mathscr{A}$. We observe that if y is chosen randomly, then $\mathscr{B}$ simulates Game 3, else it simulates Game 2_{q_2}. Therefore, Game 2_{q_2} and Game 3 are computationally indistinguishable.

We observe that both y_0 and y_1 are chosen randomly from $\mathscr{Y}$. Therefore, for any PPT adversary $\mathscr{A}$, it has negligible advantage in Game 3. This completes the proof of Theorem 2. □

6. Conclusion

In this work, we construct a novel constrained VRF for polynomial size set and give the proof of security under a new secure definition which is called semiadaptive security. Meanwhile, our construction is based on bilinear maps, which avoid the application of multilinear maps. Although it does not satisfy full adaptive security, it has solved some problems compared with selective security, which allows the adversary to query the evaluation oracle before it outputs the challenge point. To construct a fully adaptive security constrained VRFs is our future work.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61871430 and 61971458), Science and Technology Project of Henan Educational Department (no. 20A520012), and Special Project of Key Scientific Research Projects of Henan Higher Education Institutions (no. 19zx010).

References

[1] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," in *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, pp. 464–479, Singer Island, FL, USA, October 1984.

[2] D. Boneh and B. Waters, "Constrained pseudorandom functions and their applications," in *Advances in Cryptology—ASIACRYPT 2013—Proceedings of the 19th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 280–300, Bengaluru, India, December 2013.

[3] D. Boneh and M. Zhandry, "Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation," in *Advances in Cryptology—CRYPTO 2014—Proceedings of the 34th Annual Cryptology Conference*, pp. 480–499, Santa Barbara, CA, USA, August 2014.

[4] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proceedings of the 40th Annual Symposium on the Foundations of Computer Science*, pp. 120–130, IEEE, New York, NY, USA, October 1999.

[5] M. Liskov, "Updatable zero-knowledge databases," in *Advances in Cryptology—ASIACRYPT 2005, Proceedings of the 11th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 174–198, Chennai, India, December 2005.

[6] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya, "Compact e-cash and simulatable VRFs revisited," in *Pairing-Based Cryptography—Pairing 2009, Proceedings of the Third International Conference*, pp. 114–131, Palo Alto, CA, USA, August 2009.

[7] G. Fuchsbauer, "Constrained verifiable random functions," in *Security and Cryptography for Networks—Proceedings of the 9th International Conference, SCN 2014*, pp. 95–114, Amalfi, Italy, September 2014.

[8] S. Micali and R. L. Rivest, "Micropayments revisited," in *Topics in Cryptology—CT-RSA 2002, Proceedings of the the Cryptographer's Track at the RSA Conference*, pp. 149–163, San Jose, CA, USA, February 2002.

[9] S. Hohenberger, V. Koppula, and B. Waters, "Adaptively secure puncturable pseudorandom functions in the standard model," in *Advances in Cryptology—ASIACRYPT 2015—Proceedings of the 21st International Conference on the Theory and Application of Cryptology and Information Security*, pp. 79–102, Auckland, New Zealand, November-December 2015.

[10] B. Barak, O. Goldreich, R. Impagliazzo et al., "On the (Im) possibility of obfuscating programs," in *Advances in Cryptology—CRYPTO 2001*, pp. 1–18, Springer, Berlin, Germany, 2001.

[11] A. Lysyanskaya, "Unique signatures and verifiable random functions from the DH-DDH separation," in *Advances in Cryptology—CRYPTO 2002, Proceedings of the 22nd Annual*

International Cryptology Conference, pp. 597–612, Santa Barbara, CA, USA, August 2002.

[12] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *Public Key Cryptography—PKC 2005, Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography*, pp. 416–431, Les Diablerets, Switzerland, January 2005.

[13] S. Hohenberger and B. Waters, "Constructing verifiable random functions with large input spaces," in *Advances in Cryptology—EUROCRYPT 2010, Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 656–672, French Riviera, France, May-June 2010.

[14] M. Abdalla, D. Catalano, and D. Fiore, "Verifiable random functions: relations to identity-based key encapsulation and new constructions," *Journal of Cryptology*, vol. 27, no. 3, pp. 544–593, 2014.

[15] G. Fuchsbauer, M. Konstantinov, K. Pietrzak, and V. Rao, "Adaptive security of constrained PRFs," in *Advances in Cryptology—ASIACRYPT 2014*, Springer, Berlin, Germany, 2014.

[16] D. Hofheinz, A. Kamath, V. Koppula, and B. Waters, "Adaptively secure constrained pseudorandom functions," in *Financial Cryptography and Data Security*, Springer, Cham, Switzerland, 2019.

[17] A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias, "Delegatable pseudorandom functions and applications," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, pp. 669–684, CCS'13, Berlin, Germany, November 2013.

[18] E. Boyle, S. Goldwasser, and I. Ivan, "Functional signatures and pseudorandom functions," in *Public-Key Cryptography—PKC 2014—Proceedings of the 17th International Conference on Practice and Theory in Public-Key Cryptography*, pp. 501–519, Buenos Aires, Argentina, March 2014.

[19] N. Chandran, S. Raghuraman, and D. Vinayagamurthy, "Constrained pseudorandom functions: verifiable and delegatable," in *Security and Cryptography for Networks*, Springer, Cham, Switzerland, 2014.

[20] D. Boneh and X. Boyen, "Secure identity based encryption without random oracles," in *Advances in Cryptology—CRYPTO 2004, Proceedings of the 24th Annual International CryptologyConference*, pp. 443–459, Santa Barbara, CA, USA, August 2004.

[21] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, pp. 40–49, FOCS, Berkeley, CA, USA, October, 2013.

[22] M. Chase, S. Meiklejohn, and Q. Déjà, "Using dual systems to revisit *q*-type assumptions," in *Advances in Cryptology—EUROCRYPT 2014—Proceedings of the 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 622–639, Copenhagen, Denmark, May 2014.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 6124165, 10 pages
https://doi.org/10.1155/2019/6124165

Research Article

A Highly Effective Data Preprocessing in Side-Channel Attack Using Empirical Mode Decomposition

ShuaiWei Zhang,[1,2] **XiaoYuan Yang**,[1] **Lin Chen**,[1] **and Weidong Zhong**[1]

[1]*Key Laboratory of Network and Information Security Under the People's Armed Police, Engineering University of People's Armed Police, Xi'an 710086, China*
[2]*Academy of People's Armed Police, Beijing 100012, China*

Correspondence should be addressed to XiaoYuan Yang; yxyangyxyang@163.com

Received 5 February 2019; Revised 24 September 2019; Accepted 8 October 2019; Published 30 October 2019

Guest Editor: Giovanni Agosta

Side-channel attacks on cryptographic chips in embedded systems have been attracting considerable interest from the field of information security in recent years. Many research studies have contributed to improve the side-channel attack efficiency, in which most of the works assume the noise of the encryption signal has a linear stable Gaussian distribution. However, their performances of noise reduction were moderate. Thus, in this paper, we describe a highly effective data-preprocessing technique for noise reduction based on empirical mode decomposition (EMD) and demonstrate its application for a side-channel attack. EMD is a time-frequency analysis method for nonlinear unstable signal processing, which requires no prior knowledge about the cryptographic chip. During the procedure of data preprocessing, the collected traces will be self-adaptably decomposed into sum of several intrinsic mode functions (IMF) based on their own characteristics. And then, meaningful IMF will be reorganized to reduce its noise and increase the efficiency of key recovering through correlation power analysis attack. This technique decreases the total number of traces for key recovering by 17.7%, compared to traditional attack methods, which is verified by attack efficiency analysis of the SM4 block cipher algorithm on the FPGA power consumption analysis platform.

1. Introduction

A safe encryption system must be reliably realized for every section from the initial design to the final implementation. In the last decades of 20th century, cryptologists emphasized their research studies on the safety of mathematic structures and characteristics for encryption algorithms. However, the safety of implementation was omitted until Paul Kocher proposed a side-channel attack technique in 1996 [1]. Keys were successfully acquired through measuring the leaked physical information from the encrypted device or chip while they were running encryption algorithms. Since then, the page for encryption attack and defense has been turned. Side-channel attack embraced a rapid development in the past decades; many new techniques have emerged, such as timing attack [1], power consumption attack [2], electromagnetic attack [3], fault attack [4, 5], template attack [6], combinational attack [7–9], and machine learning attack [10–14], resulting in significant safety hazards to encryption system implementations.

Due to limitations of technology, developing new methods for a side-channel attack has reached the bottleneck. More research studies have turned their focus onto improving the attack efficiency. As we know, the amount of the traces needed for correctly recovering the keys is an important indicator for attack efficiency. The higher the signal-to-noise ratio (SNR), the less the amount of traces is needed, leading to higher efficiency. In detail, not only higher SNR for a single trace is needed, but also the noise introduced between the traces, such as alignment, should also be eliminated. This work takes the abovementioned two requirements into consideration and comes up with a

thorough method for data preprocessing to increase attack efficiency.

The remainder of this paper is arranged as follows. In Section 2, the state of the art for data preprocessing including raising SNR and alignment is briefly reported. Section 3 describes the preliminaries of power analysis and EMD algorithm. The proposed scheme of a highly effective data preprocessing in a side-channel attack using EMD is illustrated in Section 4. Then, the measurement setup and result of demonstrating experiments are shown in Section 5. At last, Section 6 concludes the paper.

2. Related Works

Data preprocessing is normally applied to collected traces in the real side-channel attack scenario by the attacker, to reduce the noise and increase attack efficiency. The existence of noise within the traces is significant, including artificially introduced noise [15] (e.g., unstable clock, random delay, and inaccurate triggering) and intrinsic noise of the signal as two main sources.

In terms of analyzing artificially introduced noise, Clavier et al. utilized the method of sliding window DPA [16] to counter the noise problem introduced by random process interrupts. Charvet and Pelletier implemented the wavelet transform [17] to improve the attack efficiency by analyzing clock frequency randomization of the encrypted circuit and resynchronizing the clock. Homma et al. proposed a high-resolution waveform matching method using a phase-only correlation (POC) function to address the displacement problem between waveforms [18]. And Plos et al. discovered DFA, a powerful, fast, and time-invariant analysis which even works in harsh environments with misaligned traces caused by noise and randomization [19]. While regarding minimizing intrinsic noises, van Woudenberg et al. applied the elastic alignment algorithm onto the encrypted power consumption traces to diminish the effect of intrinsic noise and improve attack efficiency [20]. And Le et al. generalized all the noises as Gaussian white noises, and by using the fourth-order cumulant, the noises have been reduced [21].

Nevertheless, most of the abovementioned data-preprocessing methods for noise reduction approximately assume the side-channel traces and their noises as linear stable signals, performing time-frequency analysis with Fourier transform to decrease noises. Since Fourier transform is a global transform, sectional time-frequency characteristics cannot be highlighted, leading to negligence of meaningful encryption information.

This work proposed a data-preprocessing method based on EMD algorithm, separating and reorganizing meaningful traces at different frequencies in the time domain. It simply depends on characteristics of the input traces without any prior knowledge of the encryption devices, which is completely independent of base function. Experimental results of the block cipher algorithm (SM4) with FPGA power consumption analysis platform are reported to have higher efficiency than the traditional attacks without our scheme.

3. Preliminaries

3.1. Power Analysis. Power analysis includes simple power analysis (SPA), differential power analysis (DPA), and correlation power analysis (CPA).

Simple power analysis directly measures power consumption of encryption devices or chips while they are running encryption algorithms and then corresponds the power consumption data with related execution instructions and operands to recover keys with a small amount or even a single power consumption trace. SPA requires fewer experimental conditions and has simple operations; however, the application is limited because it requests prior knowledge of the target's algorithm, specifically, its branches, conditional statements, and sequence of execution instructions.

Differential power analysis is a side-channel attack method, proposed by Paul Kocher et al. in 1999 with the model based on Hamming weight. The method is built on the fact that power consumption for storing "0" is different from that of "1" in the registers, resulting in leakage of power consumption information. DPA employs the statistical difference technique to recover keys, without having any prior knowledge of the algorithm, at a cost of collecting and analyzing more power consumption traces.

Correlation power analysis is published by Brier et al. on CHES conference in 2004 [22]. Its main background for attack is having sufficient plaintext and able to acquire corresponding power consumption traces with different plaintexts. Intermediate state can be calculated by the known plaintext and assumed encryption key after exhausting part of the key. According to the Hamming weight model, the corresponding power consumption is proportional to the Hamming weight of the intermediate state. Function (1) is used to calculate the correlation coefficient of power consumption and its Hamming weight:

$$W = aH(W) + b, \tag{1}$$

where a is the fixed scalar parameter, $H(W)$ is the Hamming weight of the intermediate state, b is the noise independent of the signal, and W is the power consumption of the intermediate state. Finally, correct key is obtained as the correlation coefficient reaches maximum.

3.2. EMD Algorithm. Fourier transform is a signal processing method widely used for linear stable signals in the field of signal analyzing and processing. Overall spectrum of the signal can be obtained through Fourier transform; characteristic spectrum of the target signal is analyzed to help filter out noises from the overall spectrum to achieve noise reduction. But, Fourier transform is not suitable for the side-channel attack because the physical information generated during the processing of encryption devices or chips, such as power consumption and electromagnetic fields are nonlinear unstable signals, and signal frequencies related to key are not fixed as well.

Hereafter, signal processing methods, such as short-time Fourier transform, bilinear time-frequency distribution, and wavelet transform, have been raised to describe nonlinear unstable signals in different aspects, which greatly make up for the shortcomings of Fourier transform. And yet, these methods still belong to global analysis; their analysis capabilities of these methods are directly influenced by the selection of base functions, specifically, the more compatible between the signal and the selected base function, the better analysis result there will be.

While in reality, signal profile varies from one to another; it is extremely difficult to find one base function compatible to all the signals. Under the circumstances, in 1998, Huang et al. from NASA put forward an empirical mode decomposition algorithm [23], a new adaptive time-frequency signal analysis method, which extracts intrinsic mode functions from the signal depends on its own characteristics. This method can be applied to effectively analyze nonlinear unstable encryption signals, which is considered to be a breakthrough in the traditional linear stable time-frequency signal analysis method, as we mentioned before.

4. Our Scheme

In this section, detailed description of the proposed power consumption data-preprocessing method based on the EMD algorithm is given below, together with the power consumption analysis of the SM4 block cipher algorithm, applying chosen plaintext attack introduced in [24, 25].

4.1. Data Preprocessing Based on EMD. Since EMD algorithm assumes that any signal consisted of several finite intrinsic mode functions (IMF), the single signal $x_1(t)$ of the first acquired encryption power consumption trace can be decomposed into intrinsic mode functions by steps (1) to (5):

(1) Finding out all the maximum points of $x_1(t)$ and obtaining the envelope of them ($e_+(t)$) by fitting with the cubic spline function. Similarly, search for all the minimum points of $x_1(t)$ and also obtain the envelope of them ($e_-(t)$) by fitting with the cubic spline function. $m_1(t)$ is the average of two envelopes:

$$m_1(t) = \frac{e_+(t) + e_-(t)}{2}. \tag{2}$$

(2) Subtracting $m_1(t)$ from $x_1(t)$ and removing low frequency signal, $h_1^1(t)$ can be acquired:

$$h_1^1(t) = x_1(t) - m_1(t). \tag{3}$$

(3) In general, $h_1^1(t)$ is an unbalanced signal, which does not satisfy the definition of the intrinsic mode function. Thus, repeat the first two steps for K times, until $h_1^k(t)$ satisfies the definition of the intrinsic mode function. Component of the first-order intrinsic mode function:

$$c_1(t) = \mathrm{imf}_1(t) = h_1^k(t). \tag{4}$$

(4) Subtracting $c_1(t)$ from $x_1(t)$ and removing high-frequency signal, $r_1(t)$ can be acquired:

$$r_1(t) = x_1(t) - c_1(t). \tag{5}$$

(5) Repeat the similar steps of achieving $c_1(t)$ from $h_1^1(t)$ to calculate the component of the second-order intrinsic mode function $c_2(t)$ from $r_1(t)$. These processes are continued Until two conditions followed: firstly, component of the n^{th} order intrinsic mode function $c_n(t)$ or its residue $r_n(t)$ is less than the predefined value; secondly, the residue $r_n(t)$ is a monotonic function or a constant. $x_1(t)$ will be decomposed by EMD into

$$x_1(t) = \sum_{i=1}^{n} c_i(t) + r_n(t), \tag{6}$$

where $c_1(t), c_2(t), \ldots, c_n(t)$ are components of the 1^{st} to n^{th} order intrinsic mode functions, respectively. Workflow of finding IMF is shown in Algorithm 1.

Moreover, to perform noise reduction on $x_1(t)$, reconstruct components of the 1^{st} to n^{th} order intrinsic mode functions depending on their correlations with the encryption device. Those ones having higher correlations are selected to generate the new trace $x_1(t)'$:

$$x_1(t)' = \sum_{i=select} c_i(t) + r_n(t), \tag{7}$$

in which select is the set of components having high correlation with the encryption device and $x_1(t)'$ is the first power consumption trace preprocessed by the EMD algorithm.

Finally, iterate the steps from (1) to (6) for all the collected power consumption traces to have $x_1(t)', x_2(t)', \ldots, x_m(t)'$.

4.2. Correlation Power Analysis in the Chosen Plaintext Attack. SM4 block cipher algorithm is one of the standard commercial encryption algorithm designed independently in China. Selected as an analysis object, SM4 algorithm is analyzed in this section with correlation power analysis in the chosen plaintext attack, combining the proposed EMD data-preprocessing method.

Structure diagram of the SM4 algorithm is illustrated in Figure 1. Linear transformation L spreads round key into multiple digits of round output, connecting round input and key with every bit of round output, which makes the output become intermediate data of the attack.

However, during power consumption analysis attack, all the 32 bits of round output will be used as intermediate data, leading to a $[0, 2^{32}-1]$ round key search domain. The

Input: $x_1(t)$
Output: $\sum_{i=1}^{n} c_i(t)$ and $r_n(t)$
(1) find all the maximal value of $x_1(t)$
(2) $e_+(t) \longleftarrow$ Spline $[v_{\max 1}, v_{\max 2}, \ldots, v_{\max p}]$
(3) find all the minimal value of $x_1(t)$
(4) $e_-(t) \longleftarrow$ Spline $[v_{\min 1}, v_{\min 2}, \ldots, v_{\min p}]$
(5) $m_1(t) = (e_+(t) + e_-(t))/2$
(6) $h_1^1(t) = x_1(t) - m_1(t)$
(7) $c_1(t) = \mathrm{imf}_1(t) = h_1^k(t)$
(8) $r_1(t) = x_1(t) - c_1(t)$
(9) repeat 8 Compute IMF from $\mathrm{imf}_2(t)$ to $\mathrm{imf}_n(t)$ $[c_2(t), c_3(t), \ldots, c_n(t)]$
(10) then $x_1(t) = \sum_{i=1}^{n} c_i(t) + r_n(t)$
(11) return $\sum_{i=1}^{n} c_i(t)$ and $r_n(t)$

Algorithm 1: Workflow of finding IMF.

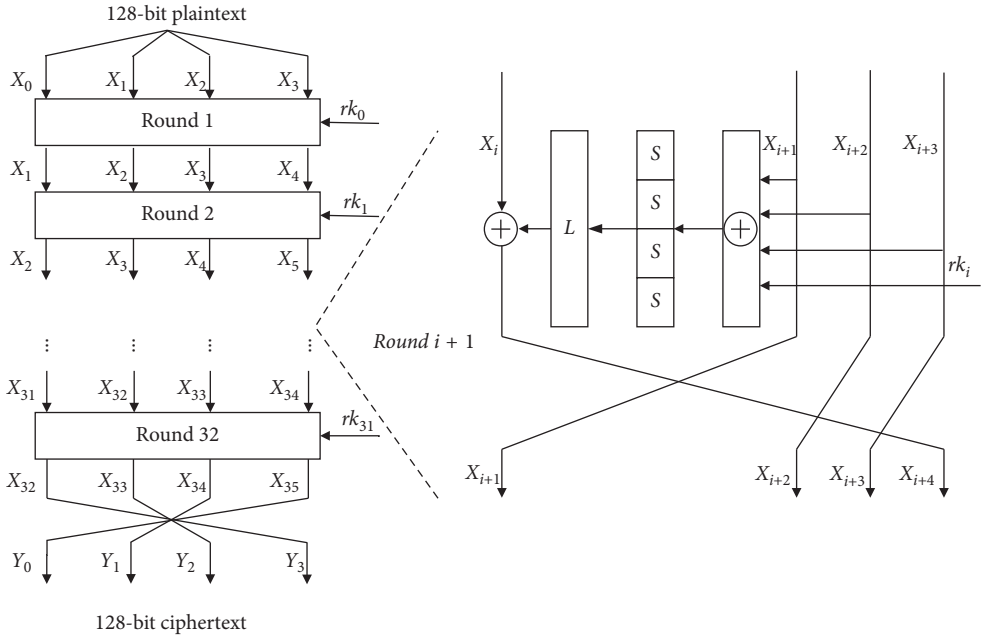


Figure 1: Structure diagram of the SM4 algorithm.

minimum number of power consumption traces needed to be collected and processed is 2^{32}, which makes the idea of using round output as attack intermediate data and performing power consumption analysis attack impractical, considering the attack complexity and data processing difficulty. Consequently, the chosen plaintext attack method introduced in [24, 25] is employed in this work for power consumption analysis on SM4 encryption key. The analysis procedures are demonstrated as follows.

First of all, $(X_0, X_1, X_2, X_3) = (00000000h, 00000000h, 00000000h, XX000000h)$ is the particularly chosen plaintext, in which X represents variable data. Performing encryption operation and collecting power consumption signal data, the intermediate calculation expression is

$$X_4 = F(X_0, X_1, X_2, X_3, rk_0) = T(XX000000h \oplus rk_0). \tag{8}$$

In function (8), since T transformation is the series of τ transformation and L transformation, τ transformation needs to calculated beforehand, which is depicted in

$$B_0 = \tau(A_0) = \tau(XX000000h \oplus rk_0) = \tau(a_{0,0} \| a_{0,1} \| a_{0,2} \| a_{0,3}) = (b_{0,0}, b_{0,1}, b_{0,2}, b_{0,3}), \tag{9}$$

$b_{0,1}$, $b_{0,2}$, and $b_{0,3}$ are fixed constants, but $b_{0,0}$ is a variable. Then, calculate L transformation, which is demonstrated in Figure 2 with X representing variable data and other characters standing for fixed data.

In Figure 3,

$$c_{0,0} = b_{0,0} \oplus (b_{0,0} \ll 2) \oplus (b_{0,1} \gg 6) \oplus (b_{0,1} \ll 2) \oplus (b_{0,2} \gg 6) \oplus (b_{0,2} \ll 2) \oplus (b_{0,3} \gg 6) \oplus b_{0,3}, \tag{10}$$

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$	
XXXXXXXX	ABCDEFGH	I J KLMNOP	QRSTUVWY	B_0
XXXXXXAB	CDEFGH I J	KLMNOPQR	STUVWYXX	$B_0 << 2$
CDEFGH I J	KLMNOPQR	STUVWYXX	XXXXXXAB	$B_0 << 10$
KLMNOPQR	STUVWYXX	XXXXXXAB	CDEFGH I J	$B_0 << 18$
⊕QRSTUVWY	XXXXXXXX	ABCDEFGH	I J KLMNOP	$B_0 << 24$
$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	

Figure 2: Process diagram of L linear transformation for the SM4 algorithm.

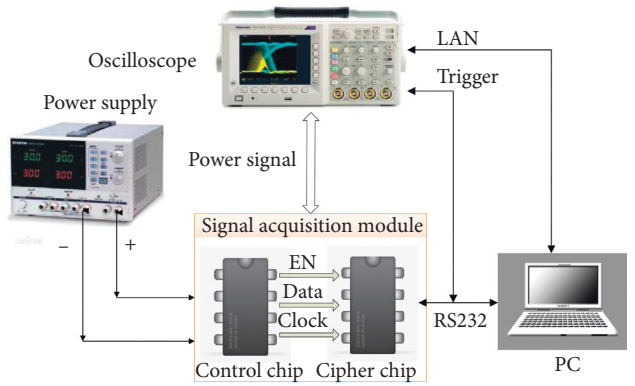


Figure 3: Architecture diagram of the power analysis system.

in which $(b_{0,1} \gg 6) \oplus (b_{0,1} \ll 2) \oplus (b_{0,2} \gg 6) \oplus (b_{0,2} \ll 2) \oplus (b_{0,3} \gg 6) \oplus b_{0,3}$ are fixed values and marked as $\text{Mask}_{0,0}$; similarly, $\text{Mask}_{0,i}$ stands for the fixed value operation result of $c_{0,i}$ in L transformation. $c_{0,i}$ can be written as $b_{0,0} \oplus (b_{0,0} \ll 2) \oplus \text{Mask}_{0,0}$, and the intermediate data calculation expression of the attack changes into equation (11). During the attack, higher byte $(rk_{0,0})$ of round key in the first round can be acquired with the power consumption model, which is built based on Hamming weight and Hamming distance of $X_{4,0}$ and the principle of power consumption analysis attack.

$$X_{4,0} = c_{0,0} = b_{0,0} \oplus \left(b_{0,0} \ll 2\right) \oplus \text{Mask}_{0,0}. \tag{11}$$

5. Measurement Setup and Experimental Results

5.1. Measurement Setup. Power analysis system based on FPGA was designed and implemented to verify the proposed EMD data-preprocessing method in our lab. The overall architecture mainly includes a PC for analyzing power consumption data, an oscilloscope, signal acquisition module, power supply, and other miscellaneous equipment, such as a serial cable used for interconnection, MMCX power consumption signal transport cable, electromagnetic probe, nternet cable, power cable, and so on. Detailed structure is shown in Figure 3.

The platform of the power analysis system is organized by Labview on host PC. Oscilloscope and signal acquisition module are controlled through internet and serial cables to collect power consumption traces generated by the encryption chip. Then, the recorded data are transferred back to host PC and analyzed by the signal processing module using the side-channel attack method to recover the key. An overview of all the equipment used and their models are listed in Table 1.

Table 1: Implementation environment.

Experimental equipment and instruments	Model
PC	Lenovo Thinkpad X240s
Oscilloscope	Tektronix MSO5204B, 2GHz
Power supply	DH1719A-3
FPGA	ALTERA Cyclone IV EP4CE115F2317N

5.2. Experiments and Results. In this section, we show the results of the proposed EMD data-preprocessing method compared with the state-of-the-art attack method as in [24, 25] through SM4 power consumption attack experiments.

Firstly, SM4 block cipher encryption algorithm was implemented in FPGA, obtaining m power consumption traces with the power consumption analysis method as explained in Section 4.1. All traces are presented in Figure 4.

In Figure 4, depending on the algorithm design, one complete encryption is composed of 33 setup rounds (1 data input round and 32 encryption key generation rounds) and 33 encryption rounds (1 data input round and 32 encryption iteration rounds). It is obvious that the key generation domain is from around 1500 to 4800 unit-time-points, while encryption iteration domain falls at around 5500 to 8800 unit-time-points.

Each power consumption trace undergoes the process of the proposed EMD data-preprocessing method. One of the processed trace's intrinsic mode components are illustrated in Figure 5.

There are 14 power consumption traces in Figure 5, among which the first trace is the original encrypted single power consumption trace $x(t)$, traces 2–13 are 1^{st}–12^{th} order intrinsic mode components $[c_1(t), c_2(t), \ldots, c_{12}(t)]$ of $x(t)$, respectively, and the last one is the residue $r_{12}(t)$. It can be concluded that the frequency of the intrinsic mode component trace decreases if the order of it increases, which matches with the theory in Section 4.1. Besides, it is shown in the figure that, among the traces of the 1^{st} order intrinsic mode component IMF1, the high-frequency noise discontinuously occurs with time. Thus, the encrypted signals are proven to be nonlinear unstable signals.

Based on experiences, lower order of intrinsic mode components correspond to high-frequency noises. Thus, after a large amount of simulation verifications, the 1^{st} and 2^{nd} order intrinsic mode components are removed for noise reduction, and the 3^{rd} to 12^{th} order intrinsic mode components are recovered by function (7). Comparison of the single power consumption trace in the first 5 rounds (1 data input round and 4 encryption iteration rounds) before and after EMD decomposition are shown in Figures 6 and 7.

High-frequency noises exist at the peaks of the trace after each round of encryption (Figure 6); however, after

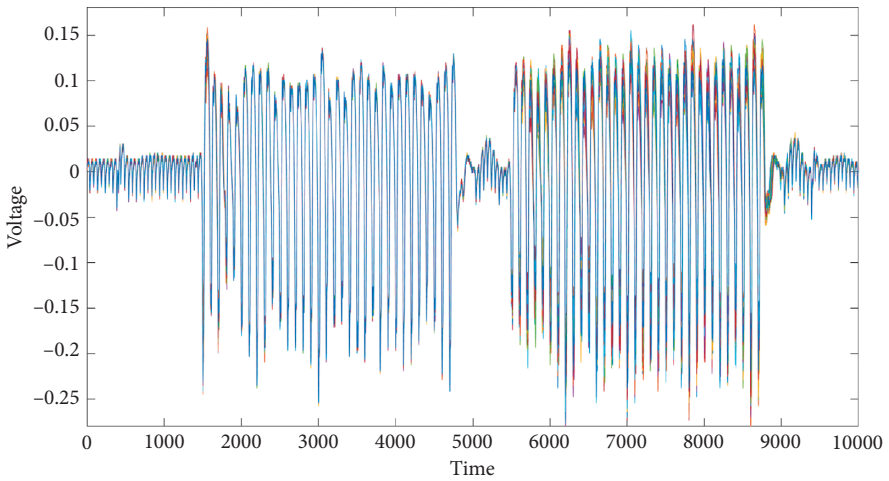


Figure 4: Power consumption traces encrypted once with SM4.

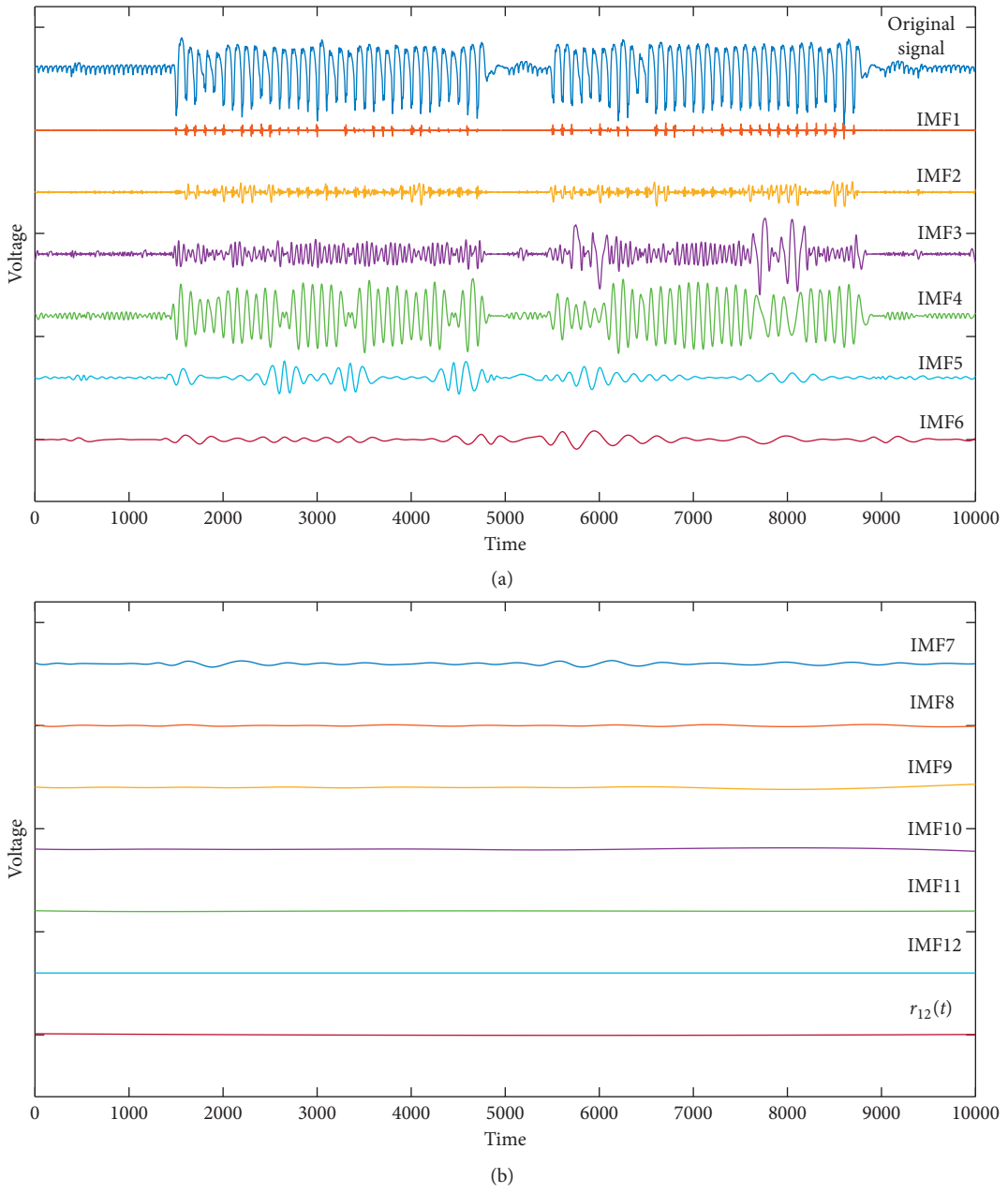


Figure 5: Intrinsic mode components of one power consumption trace processed with the EMD data-preprocessing method.

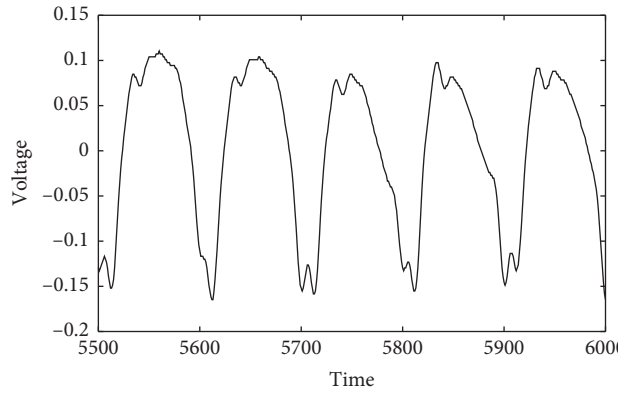


Figure 6: Single power consumption trace before EMD decomposition.

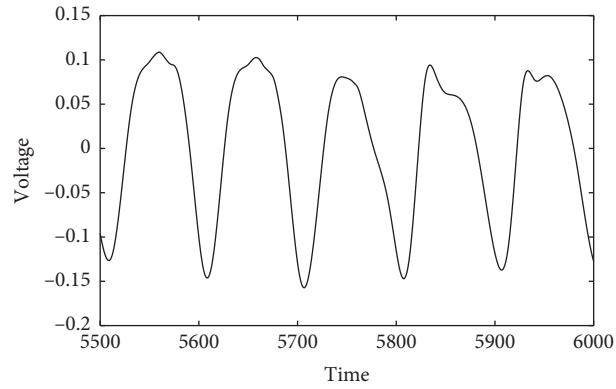


Figure 7: Single power consumption trace after EMD decomposition.

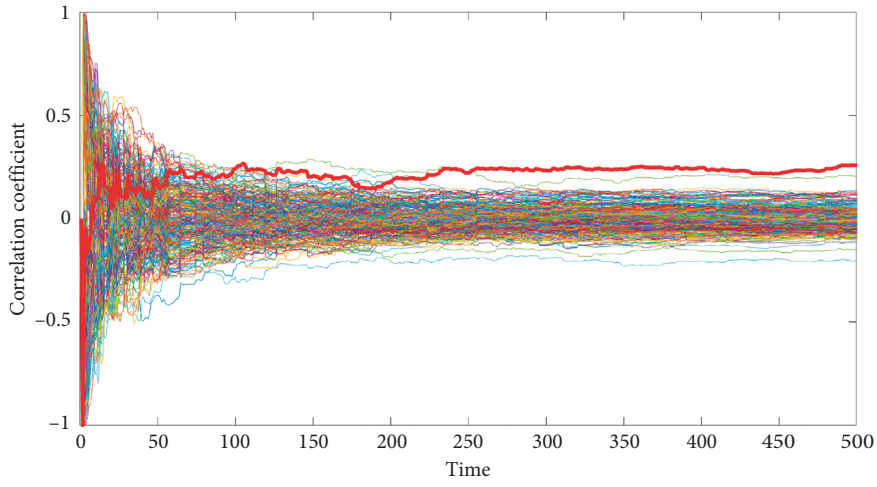


Figure 8: Efficiency of original correlation power analysis.

removing 1st and 2nd order intrinsic mode components with the EMD process, the high-frequency noises at the peaks of trace are absent (Figure 7).

Next, higher bytes ($rk_{0,0}$) of round keys in the first round of SM4 algorithm are recovered from both with and without EMD processed power consumption traces using the chosen plaintext power consumption attack method mentioned in Section 4.2. Analysis of both efficiencies is explicated in Figures 8 and 9.

From above, to separate correct key from incorrect ones, 252 power consumption traces are needed for the data without EMD data preprocessing, while only 195 power consumption traces are required for the data preprocessed by the EMD method. Therefore, the proposed

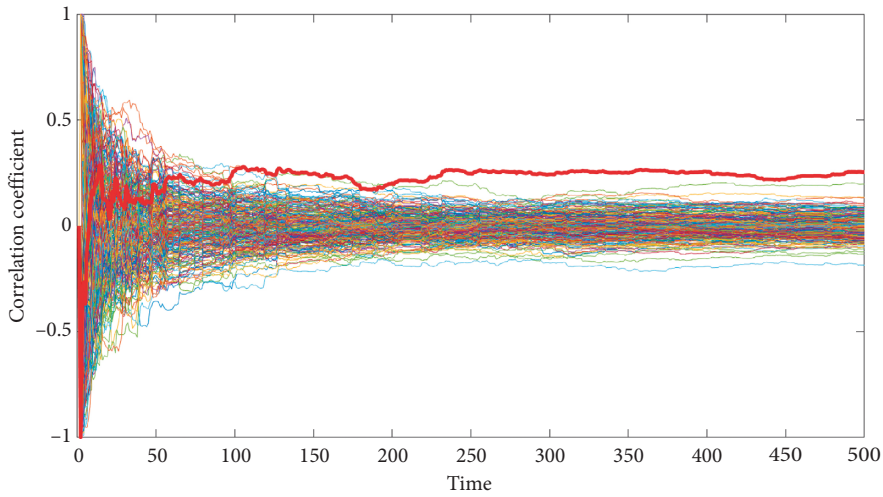


Figure 9: Efficiency of correlation power analysis after the EMD process.

Table 2: Comparison of analysis efficiencies.

	rk_0	rk_1	rk_2	rk_3	Total
Before data preprocessing	894	911	921	922	3654
After data preprocessing	711	753	745	798	3007

Table 3: Comparison between EMD and other methods of analysis efficiencies.

	rk_0	rk_1	rk_2	rk_3	Total
Before data preprocessing	894	911	921	922	3654
After FIR filter	798	811	865	846	3320
After analog filter	786	792	871	893	3342
After EMD	711	753	745	798	3007

EMD data-preprocessing method increases the analysis efficiency, reducing the required power consumption traces by 22.6%.

Applying the abovementioned method to recover all bytes of keys for the first four rounds are recovered as rk_0,rk_1, rk_2, and rk_3; the required amount of power consumption traces is all listed in Table 2. For the power consumption analysis of the data without EMD data preprocessing, in total, 3,654 traces are needed to recover round keys of the first four rounds (i.e., master key). Comparing to the analysis with proposed EMD data preprocessing, only 3,007 power consumption traces are expected, which have reduced the amount by 17.7%, indicating higher efficiency of the proposed method.

5.3. Comparison. To fully demonstrate the high efficiency of the proposed data-preprocessing method based on the EMD algorithm, two traditional solutions of noise reduction filtering have been selected for comparison. Results of the experiments have been analyzed, and the conclusions are illustrated below:

5.3.1. Data-Preprocessing Method Based on the FIR Filter. FIR filter (finite impulse response filter), also known as the nonrecursive filter, is the most fundamental element of the digital signal processing system. Not only does it keep amplitude-frequency characteristics but also has strict linear phase-frequency characteristics with a finite length of unit sampling response, which makes it a stable system. In this section, a FIR filter is designed in Matlab to reduce the high-frequency noise in the encrypted signals.

5.3.2. Data-Preprocessing Method Based on the Mini-Circuit Analog Filter. Analog filtering is achieved by running the signals through filters made of analog circuits before sampling, to improve the signal quality and reduce the workload and difficulty of data preprocessing. A 50 MHz low pass filter is used in the experiments (BLP-50+, Mini-Circuits, ISO9001).

Results of the experiments are shown in Table 3.

In Table 3, as for the power consumption analysis with FIR filter data preprocessing, 3,320 power consumption traces are needed to recover the round keys of the first four rounds (i.e., master key), 9.1% less compared to analysis without the data-preprocessing algorithm. And for the power consumption analysis with the analog filter, 3,342 power consumption traces are required to perform the same task, and it is only 8.5% less. However, the proposed EMD data-preprocessing algorithm uses much less power consumption traces (3,007) than these two methods. Thus, according to the comparison, the EMD data-preprocessing algorithm is proven to have a much higher efficiency for

power consumption analysis. In addition, both data-preprocessing methods of the FIR filter and Mini-Circuit analog filter require the clock frequency of the encrypted chip (50 MHz FPGA in this work) in advance to perform the targeted data preprocessing. However, the proposed method does not require that; this is why EMD is advantageous over other techniques.

6. Conclusion

In this paper, we have explored a general technique to reduce high-frequency noise of power consumption traces produced from our laboratory platform in CPA attack. This technique is a data-preprocessing method based on the EMD algorithm, which is one the most profound time-frequency analysis method for nonlinear unstable signals. It adaptively decomposes the collected encryption traces, based on their own characteristics, into the sum of several intrinsic mode functions without having any prior knowledge of the encryption chip. Then, it reorganizes the trace with meaningful intrinsic mode components selected by simulation verifications to reduce signal noises. Proposed technique is verified by experimentations of the SM4 block cipher algorithm, proving to have a 17.7% reduction on the number of power consumption traces for recovery of the master key, compared to the original CPA method.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key R&D Program of China (grant no. 2017YFB0802000), the National Natural Science Foundation of China (grant nos. U1636114, 61772550, and 61572521), and the National Cryptography Development Fund of China (grant no. MMJJ20170112).

References

[1] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proceedings of the Annual International Cryptology Conference*, pp. 104–113, Springer, Santa Barbara, CA, USA, August 1996.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the Annual International Cryptology Conference*, pp. 388–397, Springer, Santa Barbara, CA, USA, August 1999.

[3] J. J. Quisquater and D. Samyde, "Electromagnetic analysis (EMA): measures and counter-measures for smart cards," in *Smart Card Programming and Security*, pp. 200–210, Springer, Berlin, Heidelberg, 2001.

[4] G. Piret and J. J. Quisquater, "A differential fault attack technique against SPN structures, with application to the AES and Khazad," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 77–88, Springer, Cologne, Germany, September 2003.

[5] C. H. Kim and J. J. Quisquater, "Faults, injection methods, and fault attacks," *IEEE Design & Test of Computers*, vol. 24, no. 6, pp. 544-545, 2007.

[6] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 13–28, Springer, Redwood Shores, CA, USA, August 2002.

[7] F. X. Standaert and C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 411–425, Springer, Washington, DC, USA, August 2008.

[8] W. Schindler, "A combined timing and power attack," in *Proceedings of the International Workshop on Public Key Cryptography*, pp. 263–279, Springer, Paris, France, February 2002.

[9] S. W. Zhang, X. Y. Yang, W. D. Zhong et al., "An improved combinational side-channel attack on S-box in block cipher," *Journal of Internet Technology*, vol. 17, no. 1, pp. 157–166, 2016.

[10] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: a first study," *Journal of Cryptographic Engineering*, vol. 1, no. 4, pp. 293–302, 2011.

[11] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*, pp. 45–68, Springer, Taipei, Taiwan, September 2017.

[12] S. Hou, Y. Zhou, H. Liu, and N. Zhu, "Wavelet support vector machine algorithm in power analysis attacks," *Radioengineering*, vol. 26, no. 3, pp. 890–902, 2017.

[13] L. Lerman, Z. Martinasek, and O. Markowitch, "Robust profiled attacks: should the adversary trust the dataset?," *IET Information Security*, vol. 11, no. 4, pp. 188–194, 2016.

[14] W. Shan, S. Zhang, and Y. He, "Machine learning based side-channel-attack countermeasure with hamming-distance redistribution and its application on advanced encryption standard," *Electronics Letters*, vol. 53, no. 14, pp. 926–928, 2017.

[15] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, Springer Science & Business Media, Berlin, Germany, 2008.

[16] C. Clavier, J. S. Coron, and N. Dabbous, "Differential power analysis in the presence of hardware countermeasures," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 252–263, Springer, Worcester, MA, USA, August 2008.

[17] X. Charvet and H. Pelletier, "Improving the DPA attack using wavelet transform," in *Proceedings of the NIST Physical Security Testing Workshop*, vol. 46, Honolulu, HI, USA, 2005.

[18] N. Homma, S. Nagashima, Y. Imai et al., "High-resolution side-channel attack using phase-based waveform matching," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 187–200, Springer, Yokohama, Japan, October 2006.

[19] T. Plos, M. Hutter, and M. Feldhofer, "Evaluation of side-channel preprocessing techniques on cryptographic-enabled HF and UHF RFID-tag prototypes," in *Proceedings of the Workshop on RFID Security*, pp. 114–127, Budapest, Hungary, 2008.

[20] J. G. J. van Woudenberg, M. F. Witteman, and B. Bakker, "Improving differential power analysis by elastic alignment," in *Proceedings of the Cryptographers' Track at the RSA Conference*, pp. 104–119, Springer, San Francisco, CA, USA, February 2011.

[21] T.-H. Le, J. Clediere, C. Serviere, and J.-L. Lacoume, "Noise reduction in side channel attack using fourth-order cumulant," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 4, pp. 710–720, 2007.

[22] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 16–29, Springer, Cambridge, MA, USA, August 2004.

[23] N. E. Huang, Z. Shen, S. R. Long et al., "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, The Royal Society, vol. 454, no. 1971, pp. 903–995, 1998.

[24] S. Wang, D. Gu, J. Liu et al., "A power analysis on SMS4 using the chosen plaintext method," in *Proceedings of the 2013 9th International Conference on Computational Intelligence and Security (CIS)*, pp. 748–752, IEEE, Leshan, China, December 2013.

[25] Z. Du, Z. Wu, M. Wang, and Js. Rao, "Improved chosen-plaintext power analysis attack against SM4 at the round-output," *Journal on Communications*, vol. 36, no. 10, pp. 85–91, 2015.