Research Article

PKDIP: Efficient Public-Key-Based Data Integrity Protection for Wireless Image Sensors

Changsheng Wan,^{1,2} Juan Zhang,³ and Jie Huang¹

¹*School of Information Science and Engineering, Southeast University, Nanjing, Jiangsu 210096, China*

²*The Third Research Institute of Ministry of Public Security, Shanghai 201204, China*

³*Accounting Department, Nanjing University, Nanjing, Jiangsu 210093, China*

Correspondence should be addressed to Changsheng Wan; wan.changsheng@163.com

Received 23 April 2015; Accepted 11 June 2015

Academic Editor: Jian-Nong Cao

Copyright © 2015 Changsheng Wan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Due to limited energy of “wireless image sensors (WISs),” existing data integrity protection mechanisms typically employ a hash-function-based signing algorithm to generate “message authentication codes (MACs)” for long image frames. However, hash-function-based signing algorithm requires the WIS and the “end user (EU)” sharing a secret, which leads to a new security issue: Once the EU becomes malicious due to some reasons, it will be able to forge the WIS’s data since it holds the shared secret. Therefore, public-key cryptography is desirable. Unfortunately, public-key cryptographic operations are quite time-consuming for energy-restrained WISs. Facing this dilemma, we present a novel data integrity protection protocol named PKDIP in this paper. Similar to the mechanisms of this field, PKDIP generates MACs for data integrity protection. However, different from other well-known approaches, PKDIP introduces the “Montgomery Modular Multiplication (MontMM)” technique to current public-key-based signing algorithms. Since MontMM is much more efficient than hash functions, PKDIP can reduce the signing cost significantly. Experimental results show PKDIP can even be more efficient than hash-function-based schemes.

1. Introduction

Nowadays, “wireless image sensors (WISs)” are being widely deployed in military, environmental, and other commercial applications [1]. Examples include (i) visually monitoring the environment. WISs have been widely deployed in many situations (e.g., toxic areas, disaster locations, and natural environments such as forests and deserts) to visually detect unusual events or to acquire environmental data over long periods of time. Moreover, a recent report has shown that “Jet Propulsion Laboratory (JPL)” [2] of NASA even deployed WISs to detect surface of Mars. Examples also include (ii) visually monitoring the combat field. To improve the perceptibility of command state and to reliably make correct estimation about menace and track it, many countries have developed some military scientific research on WISs to improve the combat and command ability of army.

In the above applications, WISs are deployed on a remote land, which collect image frames and send them to “end users

(EUs)” over the wireless sensor network (WSN) and Internet. And EUs process those frames to visually monitor a remote area.

(A) *Motivation.* To transport image frames from WIS to EU, data integrity protection modules have been deployed, ensuring that the image frames are not forged by attackers between WIS and EU. In our network model, as shown in Figure 1, a data integrity protection scenario includes three parties: WIS, EU, and a “trusted third-party (TTP),” which are defined as follows.

- (1) The TTP (e.g., a computer controlled by a commander in the command center) is the key server for WIS and EU. Before image-transmitting, the TTP deploys keys to WIS and EU for signing and verification, respectively.
- (2) The WIS is deployed in a remote area (e.g., the monitored combat field). During image-transmitting,

FIGURE 1: Data integrity protection overview.

the WIS acquires environmental image frames and generates a “message authentication code (MAC) [3]” for each frame. Then it sends image frames and their MACs to the EU over the WSN and Internet.

- (3) The EU (e.g., a cloud server controlled by employees in the command center) processes image frames. Upon receiving data from WIS, EU checks MACs to ensure that the image frames are not forged by attackers (e.g., enemies in the monitored combat field).

In the above scenario, the TTP and the EU have enough computational resources, while the WIS has limited energy.

Computation cost is a serious concern for the above data integrity protection system. Due to limited energy, WIS is deeply concerned about the extremely high computation cost arising from generating MACs for long image frames (e.g., a typical RGB mode image after interpolation is around hundreds and thousands of kilobytes in length [1]). Therefore, to increase the lifetime of the WIS, current data integrity protection techniques mainly use hash functions [4–6] for generating MACs, instead of public-key-based signing algorithms [7].

Unfortunately, hash-function-based schemes require that the WIS and the EU share a secret and use this secret for signing and verification, respectively. This will lead to a new security issue as illustrated by the following example: If the EU is under attack and becomes malicious (e.g., a military spy succeeds in logging in EU, or an employee is dismissed), the attacker may get the shared secret key and forge image frames of WIS. And then it can hide the forgery action by generating a new MAC using the shared key. In this case, the commander will not be able to know the real situation of the combat field, resulting in wrong commands. Therefore, to avoid such attacks, a public-key-based data integrity protection protocol for WIS is desired, where WIS generates MACs using its private key and EU verifies MACs using the corresponding public key. By doing so, the attacker is unable to forge the WIS’s image frames even though it can log in the EU, because the EU does not know the private key. However, public-key-based cryptographic operations are quite time-consuming. For example, as discussed in Section 2, the computation cost of modular exponentiation operation is around $10^2 \sim 10^3$ times that of hash functions. This will lead to short lifetime of the

WIS. Facing this dilemma, this paper aims to design a public-key-based data integrity protection protocol for WISs, where the signing cost is low and the verification cost is at a practical level.

A data integrity protection protocol for WISs should satisfy the following requirements.

- (1) *Correctness*. It should be ensured that no one can forge WIS’s image frames. Specifically, it should be guaranteed that even when the EU is under attack, the attacker cannot forge image frames.

- (2) *Strong Resistance to Replay Attacks*. It should be ensured that WIS’s data transmission is not fraudulently repeated by an attacker. This requirement is very important for many scenarios. For example, if the attacker keeps replaying an old frame of image to the EU, the EU will not be able to know the visual change of the monitored field (i.e., the EU is blinded).

- (3) *Strong Resistance to Sequence Errors*. It should be ensured that the sequence of WIS’s image frames is not disordered. This requirement has vital significance too. For example, if the first frame contains a standing soldier, and the second frame contains a lying soldier, the EU may assess the situation that this soldier is killed. But if the attacker changes the sequence of these two frames, the EU may assess the situation that this soldier stands up after falling down.

- (4) *The Signing Cost of the WIS Is Low*. Due to limited energy, in WIS, it should be ensured that the signing cost is low, especially when processing long image frames with the length of hundreds and thousands of kilobytes (KBs).

- (5) *The Verification Cost of the EU Is at a Practical Level*. The EU is typically a cloud server with thousands of computers. And hence it has powerful computational resources. However, it is too wasteful for thousands of computers to verify just one image frame, because the EU may have to process frames from multiple WISs. Therefore, it should be guaranteed that it is efficient for only a few computers to do the verification work.

- (6) *The Communication Cost Is at a Practical Level*. Transmitting data to the EU will consume the WIS’s energy too. So, for WIS, it should be ensured that the communication cost is at a practical level.

Obviously, designing a data integrity protection protocol for WIS is a nontrivial task, because the resource-restrained WIS is not competent to sign and transmit long image frames. Current hash-function-based schemes can satisfy requirements (2), (3), (4), (5), and (6) but cannot perfectly satisfy requirement (1) (e.g., if the EU is under attack due to some reasons, the attacker will be able to forge image frames from the WIS). On the other hand, public-key-based schemes can satisfy requirements (1), (2), (3), (5), and (6) but cannot satisfy requirement (4). More importantly, when considering this research issue, we observe that none of the existing cryptographic primitives can be directly applied to achieve the goals discussed above. The detailed analysis to

arrive at these conclusions will be given in Section 2. This becomes a more severe issue given the trend that more and more WISs are being deployed.

(B) *Contributions*. Motivated by the above observation, this paper makes three main contributions:

- (1) We first identify the characteristics of data integrity protection for WISs and then present a comprehensive set of requirements for the protocol of this kind. We show some security and efficiency problems of current data integrity protection schemes.
- (2) We propose a novel approach to ensure data integrity for WISs called PKDIP, which is built on public-key cryptography, and hence it can fulfill requirement (1). However, different from current public-key-based mechanisms, PKDIP mainly uses the Montgomery Modular Multiplication (MontMM) algorithm [8] for processing image frames and only uses few scalar multiplication operations for each frame. Since MontMM is much more efficient than hash functions, the signing cost of PKDIP can even be much lower than those of hash-function-based schemes.
- (3) In addition to the security analysis which demonstrates that PKDIP indeed enforces its security guarantees, this paper also analyzes the efficiency of PKDIP, showing it can even be much more efficient than current hash-function-based schemes.

(C) *Organization*. The remainder of this paper is organized as follows. In the next section, we survey the related work. Section 3 describes PKDIP in detail, followed by the security analysis and efficiency evaluation in Sections 4 and 5, respectively. Finally, we draw our conclusions in Section 6.

2. Related Work

Due to the openness of wireless channel, attackers can easily intercept data from WIS and establish efficient attacks on the data stream. Therefore, security has vital significance for WIS.

There are a lot of security challenges for WIS [9–19], including authentication, authorization, key distribution, data integrity, data confidentiality, data aggregation, and source tracing. Among all those challenges, data integrity [17–19] is the most important one, which ensures that data transmitted over the hostile network is not forged by attackers. Without data integrity protection, EUs may get a wrong result when processing the received data [17].

All current data integrity protection protocols for WISs can be classified into three types: traditional hash-function-based protocols [20–24], watermarking-based protocols [25–27], and public-key-based protocols [28, 29].

(1) *Traditional Hash-Function-Based Protocols*. Traditional hash-function-based protocols use a shared key and a hash function for generating MACs. Perrig et al. [20, 21] introduced the SPINS protocol, which uses the μ TESLA technique for data integrity. In μ TESLA, the key server creates a one-way key chain (i.e., a sequence of shared keys), and each key of

the chain is used within a time interval for generating MACs (i.e., they mainly focus on the key management topic of data integrity). Albath and Madria [22] introduced the PADS protocol, which shows how to support both data integrity and confidentiality together. TinySec [23] investigates data integrity problems in the link-layer. LEAP [24] is a key management solution too.

(2) *Watermarking-Based Protocols*. Similar to traditional hash-function-based protocols, watermarking-based protocols use a shared key and a hash function for generating MACs. They are different from traditional hash-function-based protocols only in the transmitting method of MACs. In watermarking-based protocols [25–27], MACs replace the least significant bits of image frames. By doing so, the communication cost is saved, and the lifetime of WIS is increased.

(3) *Public-Key-Based Protocols*. Due to the high computation cost, public-key cryptography has been deprecated from being used in WSN as a data integrity protection algorithm. In the previous work [28–30], a lot of experiments have been done to test the time costs of cryptographic operations. The results show computation costs of public-key-based algorithms (e.g., scalar multiplication and modular exponentiation) are around $10^2 \sim 10^3$ times those of hash-function-based algorithms (e.g., SHA1).

3. PKDIP: The Protocol

In this section, we will give the system model of PKDIP and then present a bilinear-map-based data integrity protection construction, which can reduce the signing cost significantly.

3.1. Preliminaries

Definition 1. Let G and G_T be two cyclic groups with the same prime order q (i.e., $|G| = |G_T| = q$). Let g be the generator of G . Then, a bilinear map group is defined as $\hat{e} : G \times G \rightarrow G_T$, where $\hat{e}$ has the following properties.

- (1) *Bilinearity*. $\forall a, b \in \mathbb{Z}_q, \hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab} = \hat{e}(g^{ab}, g) = \hat{e}(g, g^{ab}) = \hat{e}(g^b, g^a)$. $\forall A, B \in G, \hat{e}(AB, g) = \hat{e}(A, g)\hat{e}(B, g)$.
- (2) *Nondegeneracy*. $\exists A, B \in G$ such that $\hat{e}(A, B) \neq 1_{G_T}$.
- (3) *Computability*. $\forall A, B \in G$, it is efficient to calculate $\hat{e}(A, B)$.

The above $\hat{e}$ can be constructed by the Weil or Tate pairings [31, 32] on elliptic curves.

3.2. *System Model*. The system model of PKDIP is shown in Figure 2, which includes three phases as illustrated below.

3.2.1. *The Preconfiguration Phase*. During this phase, the TTP generates a public-private key pair and deploys it on

FIGURE 2: System model of PKDIP.

the WIS. Then it distributes the public key to the EU. To ensure the EU and the WIS get the correct keys and the keys are not leaked to attackers, the key distributing channels should provide integrity/confidentiality/replay-resistance protection. The key generating algorithm is shown below.

$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$. This algorithm is run by the TTP for initializing cryptographic parameters of PKDIP. It takes as input the security parameter (i.e., λ) and outputs a public key (i.e., pk) and a private key (i.e., sk). In this algorithm, λ is the length of sk , which determines the security level of PKDIP.

After the preconfiguration phase, the WIS holds (pk, sk) for signing image frames, whereas the EU holds pk for verifying image frames.

3.2.2. The Signing Phase. During this phase, the WIS splits the frame of image (M_i) into n data blocks $m_{i1}, m_{i2}, \dots, m_{in} \in Z_q$, and it generates a MAC for it using the following signing algorithm.

$\tau_i \leftarrow \text{GenSig}(wID, Seq_i, m_{i1}, m_{i2}, \dots, m_{in}, sk)$. This algorithm is run by the WIS for generating a MAC for the frame of image (M_i). It takes as input $(wID, Seq_i, m_{i1}, m_{i2}, \dots, m_{in}, sk)$ and outputs a MAC τ_i , where wID is used for identifying the WIS, and Seq_i is a sequence number used for identifying this frame of image. In addition, a sequence number is a nondecreasing number, and the EU uses it for avoiding replay attacks and sequence errors. This is illustrated by the following two examples: (i) If the sequence number of the received frame is the same as a previous one, the EU will know this is a replayed frame and discard it. (ii) If the sequence number of the received frame is less than the previous one, the EU will know this is a disordered frame and re-sort it.

Then, the WIS sends $F_i = (wID, Seq_i, m_{i1}, m_{i2}, \dots, m_{in}, \tau_i)$ to the EU.

3.2.3. The Verification Phase. Upon receiving F_i from the WIS, the EU first checks Seq_i to make sure the received image frame is neither a replayed nor a disordered one and then verifies the image frame using the following verification algorithm.

$\{True, False\} \leftarrow \text{VerSig}(wID, Seq_i, m_{i1}, m_{i2}, \dots, m_{in}, \tau_i, pk)$. This algorithm is run by the EU to make sure that the frame

of image is not forged by the attacker. It takes as input $(wID, Seq_i, m_{i1}, m_{i2}, \dots, m_{in}, \tau_i, pk)$ and outputs *True* for a correct frame or *False* for a forged frame.

From the above *GenSig* algorithm, it can be seen that WIS generates the MAC using its private key (i.e., sk). In Section 4, we will show that, without knowing sk , the adversary cannot generate correct MACs for forged image frames. From the above *VerSig* algorithm, it can be seen that the EU verifies the MAC using the WIS's public key (i.e., pk), without knowing sk . Therefore, even if the adversary can log in the EU, it cannot forge image frames and generate correct MACs. In other words, this system model can fulfill requirement (1) of Section 1.

In addition, from the above verification phase, it can be seen that the EU can avoid replay attacks and sequence errors by checking Seq_i . Therefore, the system model can fulfill requirements (2) and (3) of Section 1.

3.3. Construction. The construction of PKDIP is a tuple $(\text{KeyGen}, \text{GenSig}, \text{VerSig})$ of probabilistic polynomial time algorithms.

KeyGen. The TTP first creates a bilinear map group $\hat{e} : G \times G \rightarrow G_T$ with the prime order q and the generator $g \in G$. Then it randomly generates a set $\{sk_1, \dots, sk_n \in Z_q, u \in G\}$ and computes $\{pk_1 = g^{sk_1}, \dots, pk_n = g^{sk_n} \in G, sk_r = \sum_{j=1}^n sk_j \in Z_q, pk_r = g^{sk_r} \in G\}$. Finally, it gets the private key $sk = \{sk_r, sk_1, \dots, sk_n \in Z_q\}$ and the public key $pk = \{pk_r, pk_1, \dots, pk_n, u\}$.

GenSig. The WIS signs the image frame (i.e., M_i) as $\tau_i = H(wID \parallel Seq_i)^{sk_r} u^{\sum_{j=1}^n m_{ij} sk_j} \in G$.

VerSig. The EU verifies the image frame as $\hat{e}(\tau_i, g) \stackrel{?}{=} \prod_{j=1}^n \hat{e}(H(wID \parallel Seq_i) u^{m_{ij}}, pk_j)$. This verification equation holds because

$$\begin{aligned}
 \hat{e}(\tau_i, g) &= \hat{e}\left(H(wID \parallel Seq_i)^{sk_r} u^{\sum_{j=1}^n m_{ij} sk_j}, g\right) \\
 &= \hat{e}\left(\prod_{j=1}^n (H(wID \parallel Seq_i) u^{m_{ij}})^{sk_j}, g\right) \\
 &= \prod_{j=1}^n \hat{e}\left((H(wID \parallel Seq_i) u^{m_{ij}})^{sk_j}, g\right) \\
 &= \prod_{j=1}^n \hat{e}\left(H(wID \parallel Seq_i) u^{m_{ij}}, pk_j\right).
 \end{aligned} \tag{1}$$

From the above *GenSig* algorithm, it can be seen that PKDIP processes $(m_{i1}, \dots, m_{in})$ mainly using the MontMM algorithm. In contrast, hash-function-based schemes (e.g., Kamel's scheme [27]) process $(m_{i1}, \dots, m_{in})$ using hash functions. Since MontMM is much more efficient than hash functions, PKDIP can reduce the signing cost significantly. In Section 5, we will further show, though two scalar multiplication operations are used, the total signing cost of PKDIP can be much lower than those of hash-function-based schemes.

From the above *VerSig* algorithm, it can be seen that PKDIP verifies $(m_{i1}, \dots, m_{in})$ using multiple pairing operations. In Section 5, we will show the verification cost is at a practical level.

4. Security Analysis

We analyze the security of PKDIP with respect to the security requirements given in Section 1. As discussed in Section 3.2, PKDIP can meet requirements (2) and (3) by using the sequence number technique. So, in this section, we mainly focus on how PKDIP meets requirement (1) in the random oracle model [33]. Our security analysis includes three parts: security assumption, adversary model, and security proof. Security assumption defines a well-known mathematical problem that cannot be efficiently solved. Adversary model defines what the adversary can see and what it can do. Security proof shows that the adversary cannot establish an efficient attack on PKDIP. Otherwise, there exists an algorithm, which can use the adversary for breaking the security assumption. In other words, existence of such an adversary contradicts the security assumption. Therefore, if the security assumption holds, the adversary does not exist.

4.1. Security Assumption

4.1.1. Difficult Mathematical Problems on G

(1) The “Computational Diffie-Hellman (CDH)” Problem. For randomly distributed unknown $a, b \in Z_q$, one wants to compute g^{ab} from (g, g^a, g^b) .

(2) The “Decisional Diffie-Hellman (DDH)” Problem. For randomly distributed unknown $a, b, c \in Z_q$, one wants to decide whether $c = ab \bmod q$ from (g, g^a, g^b, g^c) .

(3) The “Gap Diffie-Hellman (GDH)” Group. G is a GDH group if the DDH problem on it can be solved in polynomial time but no probabilistic polynomial time algorithm can solve the CDH problem. Note that, in PKDIP, the DDH problem can be solved by verifying $\tilde{e}(g^c, g) \stackrel{?}{=} \tilde{e}(g^a, g)\tilde{e}(g^b, g)$, because $\tilde{e}(g^c, g) = \tilde{e}(g, g)^c = \tilde{e}(g, g)^{ab} = \tilde{e}(g^a, g)\tilde{e}(g^b, g)$. Therefore, a GDH group is to ensure that the CDH problem holds on G .

4.1.2. Security Assumption of PKDIP. We assume the proposed group G is a GDH group. Specifically, we have the following assumption.

Assumption 2 ((t, ϵ)-GDH group assumption). Given $(g, g^a, g^b) \in G$ for randomly distributed unknown $(a, b) \in Z_q$, there is no t -time algorithm, which has the nonnegligible probability ϵ in computing $g^{ab} \in G$ (i.e., the CDH problem holds on G).

4.2. Adversary Model. To satisfy requirement (1), it must be ensured that no one (including the compromised EU)

can forge the WIS’s image frames. Therefore, the potential adversary is a compromised EU. In addition, since the adversary can obtain a number of image frames and the corresponding signatures, we need to prove that PKDIP is secure in the “chosen-ciphertext attack (CCA)” model [3]. In the CCA model, the adversary ($\mathcal{A}$) can interact with a challenger ($\mathcal{C}$) as follows:

- (1) *Initialization.* $\mathcal{C}$ first generates parameters of $\hat{e}$ (e.g., g and q). Then, it generates the WIS’s public key pk , while the private key sk is unknown. Finally, it sends pk and parameters of $\hat{e}$ to $\mathcal{A}$.
- (2) *Hash Query.* $\mathcal{A}$ issues a series of hash queries, and $\mathcal{C}$ returns the hash values.
- (3) *Signing Query.* $\mathcal{A}$ issues a series of signing queries by sending M_i to $\mathcal{C}$, and $\mathcal{C}$ returns the MAC (i.e., τ_i).
- (4) $\mathcal{A}$ outputs a forged block-MAC pair (M_f, τ_f) , where the query for M_f has never been issued.

In the above interactions, we say $\mathcal{A}$ wins the game if the *VerSig* algorithm returns *True* for the forged (M_f, τ_f) .

4.3. Security Proof. In this subsection, we will show that $\mathcal{A}$ cannot efficiently forge the WIS’s image frames. Otherwise, $\mathcal{C}$ can use it for solving the GDH problem. Our proof includes two scenarios. We first consider the $n = 1$ scenario (i.e., the WIS splits the image frame into one block). Then, we extend it to the $n > 1$ scenario.

Scenario 1 (the $n = 1$ scenario). For the $n = 1$ scenario, the signing algorithm becomes $\tau_i = (H(wID \parallel Seq_i)u^{m_{i1}})^{sk_1}$, and we have the following.

Theorem 3. After q_s signing queries, if $\mathcal{A}$ can forge the WIS’s image frame with the probability ϵ in time t , $\mathcal{C}$ can solve the GDH problem with the probability $\epsilon' = (1 - 1/(q_s + 1))^{q_s} * \epsilon * 1/(q_s + 1)$ in time $t' = t + (3q_s + 2)T_{sm}$, where T_{sm} is the time cost of scalar multiplication.

Proof. The proof is given in the Appendix. $\square$

Scenario 2 (the $n > 1$ scenario). For the $n > 1$ scenario, the signing algorithm can be written as $\tau_i = H(wID \parallel Seq_i)^{sk_r} u^{\sum_{j=1}^n m_{ij} sk_j} = \prod_{j=1}^n (H(wID \parallel Seq_i)u^{m_{ij}})^{sk_j} = \prod_{j=1}^n \tau_{ij}$, where $\tau_{ij} = (H(wID \parallel Seq_i)u^{m_{ij}})^{sk_j}$. That is to say, the MAC of the $n > 1$ scenario can be seen as a combination of multiple MACs of the $n = 1$ scenario. Therefore, to forge τ_i , $\mathcal{A}$ has to forge multiple τ_{ij} ’s. Assuming $\mathcal{A}$ can forge τ_{ij} with the probability ϵ , it can forge τ_i with the probability $pr = \epsilon^n < \epsilon$ (keep in mind that $\epsilon < 1$). In other words, it is much harder for $\mathcal{A}$ to forge τ_i than to forge τ_{ij} . So, in the $n > 1$ scenario, $\mathcal{A}$ can forge τ_i only with a negligible probability too.

5. Efficiency Evaluation

In the previous work [28, 29], Watro and Gura et al. have performed a lot of experiments to measure the computation costs of cryptographic operations on multiple kinds of

TABLE 1: Time costs of basic cryptographic operations (unit: μs).

T_p	T_{sm}	T_h	T_{mm}
8120.1	329.4	0.5	0.1

T_p : time cost of Tate pairing; T_{sm} : time cost of scalar multiplication; T_h : time cost of SHA1 per 64-byte data; T_{mm} : time cost of MontMM.

processors. The results have shown, though the computation costs on different processors are different, the time ratio among various cryptographic operations is the same. So, in this paper, we aimed to investigate the time ratio between PKDIP and other hash-function-based schemes (e.g., Kamel's scheme [27]).

All our experiments were conducted on an Intel i7 processor system with the clock frequency 3.40 GHz. To achieve the 80-bit security parameter, we choose the 160-bit elliptic curve group. Hash function used in PKDIP and other hash-function-based schemes is SHA1 [4].

Section 5.1 investigates the time costs of basic cryptographic operations (i.e., SHA1, MontMM, scalar multiplication, and pairing), and Section 5.2 shows the time ratio of PKDIP to Kamel's scheme [27]. Then, in Section 5.3, we discuss the communication cost of PKDIP. Finally, we further discuss the applicability of PKDIP on realistic sensors in Section 5.4.

5.1. Investigation of Basic Cryptographic Operations. We investigated the time costs of basic cryptographic operations using the OPENSSL library [34]. The results are shown in Table 1 (note that when processing a long image frame, SHA1 will split it into multiple 64-byte blocks and perform a set of permutation operations on each block. Therefore, in Table 1, the time cost for SHA1 is the time for each 64-byte data).

From Table 1, it can be seen that (i) the time cost of scalar multiplication is significantly higher than those of SHA1 and MontMM, because $T_{sm}/T_h = 329.4/0.5 = 658.8$ and $T_{sm}/T_{mm} = 329.4/0.1 \approx 3.3 \times 10^3$. (ii) The time cost of SHA1 is much higher than that of MontMM, because $T_h/T_{sm} = 0.5/0.1 = 5$. These two results imply that (i) if a public-key-based scheme only uses the T_{sm} operation for generating MACs, the signing cost will be quite higher than those of hash-function-based schemes. (ii) However, if a public-key-based scheme uses both T_{sm} and T_{mm} operations for generating MACs, the signing cost may be lower than hash-function-based schemes. This is because each data block is processed by one highly efficient T_{mm} operation, and multiple data blocks are processed by only two time-consuming T_{sm} operations.

5.2. Comparison of Computation Costs. The computation costs of WIS and EU are defined as the total time of cryptographic operations in *GenSig* and *VerSig* algorithms, respectively. In Table 2, we mainly compare the computation costs between PKDIP and Kamel's scheme [27], because other hash-function-based schemes [20–26] enjoy similar efficiency to Kamel's scheme [27].

TABLE 2: Comparison of computation costs (unit: μs).

	PKDIP	Kamel's scheme [27]
T_{WIS}	$2T_{sm} + T_h + \left(\frac{L}{20}\right)T_{mm}$ $= 0.005L + 659.3$	$\left(\frac{L}{64}\right)T_h = 0.008L$
T_{EU}	$\left(\frac{L}{20} + 1\right)T_p + \left(\frac{L}{20}\right)(T_{sm} + T_h)$ $= 422.5L + 8120.1$	$\left(\frac{L}{64}\right)T_h = 0.008L$

T_{WIS} : computation cost on the WIS; T_{EU} : computation cost on the EU; L : frame length in byte.

From Table 2, the following can be seen.

(1) *PKDIP Fulfills Requirement (4) of Section 1.* This is illustrated by the following three scenarios. (i) Scenario 1: for long image frames, the signing cost of PKDIP can be much lower than those of hash-function-based schemes, because $\lim_{L \rightarrow \infty} ((0.005L + 659.3)/0.008L) = 0.625 < 1$. (ii) Scenario 2: for more than 220.0 KB image frames, the signing cost of PKDIP is more efficient than hash-function-based schemes. This conclusion can be arrived at by solving the equation $0.005L + 659.3 = 0.008L \Rightarrow L = 659.3/(0.008 - 0.005) = 220.0$ KB. (iii) Scenario 3: for tens-of-KBs-level image frames, the signing cost of PKDIP and those of hash-function-based schemes are at the same level. For example, setting $L = 50$ KB, we get that the signing costs of PKDIP and hash-function-based schemes are 909.3 μs and 400 μs , respectively (i.e., they are both hundreds of microseconds). In summary, when processing image frames with the length of hundreds and thousands of KBs, PKDIP is much more efficient than hash-function-based schemes on the WIS's side. Additionally, when processing image frames with the length of tens of KBs, PKDIP and hash-function-based schemes enjoy similar efficiency on the WIS's side.

(2) *PKDIP Fulfills Requirement (5) of Section 1.* For example, if $L = 100$ KB, the verification cost of PKDIP is 42 s (i.e., the verification cost is at second level). Currently, the clock frequencies of most computers are at GHz level (i.e., similar to our experimental environment). Therefore, PKDIP is suitable for being deployed on most computers. Moreover, since the EU (i.e., the computing center) is typically composed of thousands of computers and processors, it can distribute the verification work to tens of computers or processors. In this case, the verification cost can even be less than one second.

5.3. Discussions on the Communication Cost of PKDIP. The communication cost of PKDIP is defined as the total length of frames and MACs transmitted between WIS and EU. The frame length can be shortened by image compression algorithms, which is out of scope of this paper. The MAC length of PKDIP (i.e., length of τ_i) is equal to the length of the elliptic curve point, regardless of the length of image frames. For example, if the curve is 160 bits, the MAC length will be $160 \times 2 = 320$ bits. Compared with the frame length, this MAC length can be omitted. Therefore, PKDIP fulfills requirement (6).

Further, to reduce the communication cost, WIS can use the watermarking technique in a similar way as shown in the previous work [25–27]. That is, WIS can replace the least significant bits of M_i with τ_i . By doing so, the communication cost is reduced and hence the energy consumed by the frame-transmitting process is saved.

5.4. Discussions on the Applicability of PKDIP on Realistic Sensors. The CPU speed of most realistic sensors is typically lower than that of the Intel i7 CPU, resulting in higher computation cost. However, as discussed above, the ratio among various cryptographic operations is the same on different CPUs. So, the computation costs of PKDIP can be lower than those of hash-function-based schemes on realistic sensors too.

PKDIP requires the WIS to store $n + 1$ private keys (i.e., $sk_r, sk_1, \dots, sk_n$), resulting in higher storage cost than those of current schemes. From Section 3.3, it can be seen that the total length of those private keys is the same as that of one image frame. This implies the WIS needs to use KB-level memory to store those keys. In contrast, in the current scheme [27], the WIS needs to have KB-level memory to store at least one image frame too. Therefore, though PKDIP requires a little more memories, its storage cost is at the same level as that of current schemes.

From the above discussions, it can be seen that PKDIP is feasible to be deployed on realistic sensors.

6. Conclusion

In this paper, we have identified the characteristics of data integrity protection for wireless image sensors and concluded 6 properties that a secure and efficient data integrity protection scheme for wireless image sensors should satisfy. Moreover, we have proposed a novel public-key-based protocol named PKDIP. The protocol satisfies a set of important requirements which have not been addressed by earlier works. The experimental results show that the proposed approach can even be much more efficient than hash-function-based schemes.

Appendix

Proof of Theorem 3. Assuming $\mathcal{A}$ is a t -time algorithm that can forge the WIS's image frame with the probability ϵ , we show how to construct a t' -time algorithm $\mathcal{C}$ that can solve the GDH problem with the probability ϵ' .

Let the set of transmitted frames be $\{M_i = m_{i1}\}$. Let g be the generator of G . Given $(g, pk = g^{sk}, g^b)$ for randomly distributed unknown $(sk = sk_1, b)$, $\mathcal{C}$ can compute g^{sk*b} as follows.

Initialization. $\mathcal{C}$ starts by giving (g, q, pk) to $\mathcal{A}$. The corresponding private key sk is unknown to $\mathcal{C}$.

Hash Query. To respond to hash queries, $\mathcal{C}$ maintains a list T_H of tuples (M_i, x_i, y_i, z_i) , which is managed below:

- (1) T_H is initially empty.

- (2) When $\mathcal{A}$ issues a hash query for M_i , $\mathcal{C}$ responds as follows: (i) If the query already appears in T_H (e.g., (M_i, x_i, y_i, z_i)), $\mathcal{C}$ responds with $H(wID \parallel Seq_i) = x_i$. (ii) If the query is not in T_H , $\mathcal{C}$ generates a random coin $z_i \in \{0, 1\}$ according to the bivariate probability distribution $pr[z_i = 0] = 1/(q_s + 1)$. (iii) $\mathcal{C}$ randomly generates $y_i \in \mathbb{Z}_q$. If $z_i = 1$, $\mathcal{C}$ computes $x_i = g^{y_i}$. Otherwise, $\mathcal{C}$ computes $x_i = g^b g^{y_i}$. (iv) $\mathcal{C}$ adds (M_i, x_i, y_i, z_i) to T_H and responds with $H(wID \parallel Seq_i) = x_i u^{-m_{i1}}$ to $\mathcal{A}$.

Signing Query. Given M_i , $\mathcal{C}$ responds to this query as follows:

- (1) $\mathcal{C}$ issues the above hash query to obtain $H(wID \parallel Seq_i)$. Let (M_i, x_i, y_i, z_i) be the corresponding tuple in T_H .
- (2) If $z_i = 0$, $\mathcal{C}$ aborts. If $z_i = 1$, $\mathcal{C}$ gives $\tau_i = pk^{y_i}$ to $\mathcal{A}$.

Forgery. $\mathcal{A}$ outputs a block-MAC pair (M_f, τ_f) , where the query for M_f has never been issued.

Solving the GDH Problem. $\mathcal{C}$ solves the GDH problem as follows: (i) $\mathcal{C}$ runs $\mathcal{A}$ to get the forged (M_f, τ_f) , where the query for M_f has never been issued. (ii) $\mathcal{C}$ issues a hash query for M_f to get the tuple (M_f, x_f, y_f, z_f) . If $z_f = 1$, $\mathcal{C}$ aborts. Otherwise, it gets $H(wID \parallel Seq_f) = x_f u^{-m_{f1}} = g^b g^{y_f} u^{-m_{f1}}$. (iii) From the GenSig algorithm, $\mathcal{C}$ gets $\tau_f = (H(wID \parallel Seq_f) u^{m_{f1}})^{sk_1} = (g^b g^{y_f} u^{-m_{f1}} u^{m_{f1}})^{sk_1} = (g^b g^{y_f})^{sk_1}$. From the signing query and forgery process, $\mathcal{C}$ gets τ_f . Combining these two results, $\mathcal{C}$ gets $\tau_f = (g^b g^{y_f})^{sk_1} = g^{sk_1 * b} (g^{sk_1})^{y_f} \Rightarrow g^{sk_1 * b} = \tau_f pk^{-y_f}$. That is to say, given $(g, pk = g^{sk}, g^b)$ for randomly distributed unknown $(sk = sk_1, b)$, $\mathcal{C}$ can compute g^{sk*b} .

Probability for Solving the GDH Problem. The probability for solving the GDH problem is computed from the above simulation process as follows: (i) First, $\mathcal{C}$ has to successfully respond to q_s signing queries from $\mathcal{A}$. The probability is $(1 - 1/(q_s + 1))^{q_s}$. (ii) Second, $\mathcal{C}$ runs $\mathcal{A}$ to get the forged (M_f, τ_f) . The probability of success is ϵ (note that this is the assumption of Theorem 3). (iii) Third, $\mathcal{C}$ issues a hash query for M_f to get the tuple (M_f, x_f, y_f, z_f) . The probability of $z_f = 0$ is $1/(q_s + 1)$. (iv) Finally, the probability for solving the GDH problem is a multiplication of the above three probabilities: $\epsilon' = (1 - 1/(q_s + 1))^{q_s} * \epsilon * 1/(q_s + 1)$.

Time for Solving the GDH Problem. Denote the time cost of scalar multiplication by T_{sm} . We can compute the total time for solving the GDH problem as follows: (i) First, $\mathcal{C}$ has to successfully respond to q_s signing queries from $\mathcal{A}$. The time cost is $3q_s T_{sm}$. (ii) Second, $\mathcal{C}$ runs $\mathcal{A}$ to get the forged (M_f, τ_f) . The time cost of success is t (note that this is the assumption of Theorem 3). (iii) Third, $\mathcal{C}$ issues a hash query for M_f to get the tuple (M_f, x_f, y_f, z_f) . The time cost is $2T_{sm}$. (iv) Finally, the total time for solving the GDH problem is an addition of the above three time costs: $t' = 3q_s T_{sm} + t + 2T_{sm} = t + (3q_s + 2)T_{sm}$. $\square$

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This paper is supported by the NSFC (no. 61101088), the Chinese 863 Plan (no. 2013AA014001), and the Opening Project of Key Lab of Information Network Security of Ministry of Public Security (no. C13607).

References

- [1] Z. Cao, Z. Ji, and M. Hu, "An image sensor node for wireless sensor networks," in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC '05)*, vol. 2, pp. 740–745, IEEE, Las Vegas, Nev, USA, April 2005.
- [2] Sensor Webs, <http://sensorwebs.jpl.nasa.gov/>.
- [3] A. J. Menezes, S. A. Vanstone, and P. C. van Oorschot, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, Fla, USA, 1996.
- [4] D. Eastlake and P. Jones, US Secure Hash Algorithm 1 (SHA1), IETF RFC 3174, <http://www.rfceditor.org/rfc/rfc3174.txt>.
- [5] R. Housley, "A 224-bit one-way Hash function: SHA-224," RFC 3874, IETF, 2004, <http://www.rfceditor.org/rfc/rfc3874.txt>.
- [6] National Institute of Standards and Technology, *Secure Hash Standard, FIPS 180-2*, National Institute of Standards and Technology, Gaithersburg, Md, USA, 2002.
- [7] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '01)*, pp. 514–532, 2001.
- [8] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519–521, 1985.
- [9] X. Ren and H. Yu, "Security mechanisms for wireless sensor networks," *International Journal of Computer Science and Network Security*, vol. 6, pp. 151–161, 2006.
- [10] A. S. K. Pathan, H.-W. Lee, and C. S. Hong, "Security in wireless sensor networks: issues and challenges," in *Proceedings of the 8th International Conference Advanced Communication Technology (ICACT '06)*, vol. 2, pp. 1043–1048, Phoenix Park, Republic of Korea, February 2006.
- [11] F. Hu and N. K. Sharma, "Security considerations in ad hoc sensor networks," *Ad Hoc Networks*, vol. 3, no. 1, pp. 69–89, 2005.
- [12] A.-S. K. Pathan, H.-W. Lee, and C. S. Hong, "Security in wireless sensor networks: issues and challenges," in *Proceedings of the 8th International Conference Advanced Communication Technology (ICACT '06)*, vol. 2, pp. 1043–1048, IEEE, Phoenix Park, Republic of Korea, February 2006.
- [13] T. Zia and A. Zomaya, "A security framework for wireless sensor networks," in *Proceedings of the IEEE Sensors Applications Symposium*, pp. 49–53, Houston, Tex, USA, February 2006.
- [14] J. O. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless sensor network security: a survey," in *Security in Distributed, Grid, and Pervasive Computing*, chapter 17, Auerbach Publications, CRC Press, Boca Raton, Fla, USA, 2006.
- [15] E. Shi and A. Perrig, "Designing secure sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 38–43, 2004.
- [16] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *IEEE Communications Surveys and Tutorials*, vol. 8, no. 2, pp. 2–23, 2006.
- [17] A. Duresi, V. Paruchuri, R. Kannan, and S. S. Lyengar, "Data integrity protocol for sensor networks," *International Journal of Distributed Sensor Networks*, vol. 1, no. 2, pp. 205–214, 2005.
- [18] I. Kamel, "A schema for protecting the integrity of databases," *Computers & Security*, vol. 28, no. 7, pp. 698–709, 2009.
- [19] H. Juma, I. Kamel, and L. Kaya, "On protecting the integrity of sensor data," in *Proceedings of the 15th IEEE International Conference on Electronics, Circuits and Systems*, pp. 902–905, St. Julian's, Malta, September 2008.
- [20] H. Chan, A. Perrig, B. Przydatek, and D. Song, "SIA: secure information aggregation in sensor networks," *Journal of Computer Security*, vol. 15, no. 1, pp. 69–102, 2007.
- [21] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: security protocols for sensor networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pp. 189–199, Rome, Italy, July 2001.
- [22] J. Albath and S. Madria, "Practical algorithm for data security (PADS) in wireless sensor networks," in *Proceedings of the 6th ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE '07)*, pp. 9–16, Beijing, China, June 2007.
- [23] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 162–175, Baltimore, Md, USA, November 2004.
- [24] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 62–72, ACM, Washington, DC, USA, October 2003.
- [25] R. Sion, M. Atallah, and S. Prabhakar, "Resilient rights protection for sensor streams," in *Proceedings of the 30th International Conference on Very Large Data Bases*, pp. 732–743, Toronto, Canada, September 2004.
- [26] H. Guo, Y. Li, and S. Jajodia, "Chaining watermarks for detecting malicious modifications to streaming data," *Information Sciences*, vol. 177, no. 1, pp. 281–298, 2007.
- [27] I. Kamel and H. Juma, "A lightweight data integrity scheme for sensor networks," *Sensors*, vol. 11, no. 4, pp. 4118–4136, 2011.
- [28] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: securing sensor networks with public key technology," in *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, pp. 59–64, Washington, DC, USA, October 2004.
- [29] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," in *Cryptographic Hardware and Embedded Systems—CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11–13, 2004. Proceedings*, vol. 3156 of *Lecture Notes in Computer Science*, pp. 119–132, Springer, Berlin, Germany, 2004.
- [30] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichiitiu, "Analyzing and modeling encryption

overhead for sensor network nodes,” in *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA '03)*, San Diego, Calif, USA, September 2003.

- [31] D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” in *Advances in Cryptology—CRYPTO 2001*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 213–229, Springer, Berlin, Germany, 2001.
- [32] A. Miyaji, M. Nakabayashi, and S. Takano, “New explicit conditions of elliptic curve traces for FR-reduction,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 84, no. 5, pp. 1234–1243, 2001.
- [33] M. Bellare and P. Rogaway, “Random oracles are practical: a paradigm for designing efficient protocols,” in *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS '93)*, pp. 62–73, November 1993.
- [34] Openssl.org, openssl-1.0.1e.tar.gz, 2013, <http://www.openssl.org/source/>.