Secure and Efficient Two-Party Signing Protocol for the Identity-Based Signature Scheme in the IEEE P1363 Standard for Public Key Cryptography

Debiao He, Yudi Zhang, Ding Wang, and Kim-Kwang Raymond Choo, *Senior Member, IEEE*

Abstract—Mobile device and application (app) security are increasing important, partly due to the constant and fast-paced cyberthreat evolution. To ensure the security of communication (e.g., data-in-transit), a number of identity-based signature protocols have been designed to facilitate authorization identification and validation of messages. However, in many of these protocols, a user's private key may leak when a new signature is generated since the private keys are stored on the device. Seeking to improve the security of the private key, we propose a novel two-party distributed signing protocol for the IEEE P1363 standard in this paper. This protocol requires two devices to separately store one part of the user's private key, and allows these two devices to generate a valid signature without revealing the entire private key of the user. We formally prove the security of the protocol in the random oracle model. Then, we implement the protocol using the MIRACL library and evaluate the protocol on two Android devices. Compared with the protocol of Lindell (CRYPTO'17) that uses zero-knowledge for its security, we demonstrate that our protocol is more suitable for deployment in the mobile environment.

Index Terms—Two-party signature, mobile device, secure and efficient

1 Introduction

Mobile device and application (app) popularity is perhaps best illustrated by their market penetration rate. According to a research released by the China Internet Network Information Center in December 2016 [1], for example, the number of mobile users in China is estimated to be 695 million, among them the number of Internet users has increased by 95.1 percent compared to the corresponding period of last year. A year later in December 2017, the number of mobile users in China has increased by 2.4 percent compared to the year before [2]. This is not surprising, partly due to many technological trends (e.g., Internet of Things - IoT, mobile cloud, and mobile social networking) and advances in mobile technologies [3], [4], [5], [6], [7]. For example, a recent Gartner research report [8] posited that:

> In the next stage of mobility, the mobile user will be increasingly enveloped in a growing "device mesh" the expanding set of endpoints people use to access applications and information or to interact with people, social communities, governments and businesses. This will also involve an increasing number of devices (such as wearables, drones and the Internet of Things [IoT]), enabling technologies (such as 5G, wireless power and Bluetooth) and interface modalities (such as augmented and virtual reality and natural-language processing [NLP]).

A typical mobile Internet architecture is shown in Fig. 1, where mobile (Internet-connected) devices can collect, transmit, process and store a wide range of data (e.g., environmental data, and personally identifiable information such as health data). To ensure the security of such significant amount of data, we have to also improve the security of mobile devices and apps.

One of the challenges in designing security solutions, particularly cryptographic solutions, is the diversity in mobile device hardware and software (e.g., operating system and apps) and their computing capabilities (e.g., some mobile devices have limited computation and storage capabilities) [9]. In a typical network, communications between mobile and other devices are subject to a broad range of attacks,

- *D. He is with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China and also with the State Key Laboratory of Cryptology, Beijing 100878, China. E-mail: hedebiao@163.com.*
- *Y. Zhang is with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China. E-mail: zhangyudi007@gmail.com.*
- *D. Wang is with the School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China. E-mail: wangdingg@pku.edu.cn.*
- *K.-K. R. Choo is with the Department of Information Systems and Cyber Security and the Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX 78249. E-mail: raymond.choo@fulbrightmail.org.*

Manuscript received 12 Dec. 2017; revised 4 June 2018; accepted 12 July 2018. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Debiao He.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TDSC.2018.2857775

Fig. 1. A typical mobile internet architecture.

varying from basic attacks to sophisticated advanced persistent threat-type attacks [10], [11], [12]. Therefore, it is important to verify the identity of the mobile device and confirm the validation of the message. To achieve this goal, we generally rely on the technology of digital signature schemes [13], [14]. However, in the most schemes, the user's private key is typically stored as a single file on the mobile device (i.e., single point of attack) [15]. There is also the risk that the private key can be obtained during the signature generation process, or through the use of mobile forensic techniques [16].

Threshold cryptography [17], [18], [19], [20] can potentially be used to improve the security of the private key, by splitting and distributing a single key into a number of shares/parts that can then be stored on different devices (or for multiple users). Threshold cryptography has been intensively studied since the late 1980s [21], [22], [23], [24]. One common approach is the $t–out–of–n$ threshold secret sharing protocol proposed by Shamir [25] and Blakley [26], where a single (private) key is shared among n parties. This allows the recovery of the private key with t or more shares, rather than all shares in the original threshold cryptography-based approach. In other words, to obtain the private key, an adversary has to corrupt at least t parties or devices. In the context of this paper, an adversary must corrupt all the devices in order to reconstruct and obtain the private key. Such an approach can be used in applications where generating a signature requires multiple users to sign the message or a ciphertext should be decrypted by multiple users.

A known limitation in the $t–out–of–n$ threshold secret sharing protocol is that the (entire) private key can be recovered during the signing or encryption phase. As the recovered key is usually stored on the mobile devices (e.g., Android or iOS devices), any party who holds the recovered private key can sign or encrypt message without requiring the participation of other parties. Recently, Lindell [27] designed a fast secure two-party ECDSA signing protocol in CRYPTO 2017. However, the use of zero-knowledge in the security proof may mean that it is not practical/viable for deployment in a mobile environment.

In this paper, we propose a two-party distributed signature protocol for the IEEE P1363 standard. Specifically, in our proposed protocol, the two parties communicate with each other and one of them outputs a signature without revealing the private key. We also implement the proposed protocol on two mobile devices and a personal computer for evaluation. The main contributions of our work are summarized as follows:

1) We propose a fast and secure two-party distributed signing protocol for the identity-based signature scheme in IEEE P1363. The two parties in the protocol can generate a valid signature without recovering the private key.
2) We analyze the security of our proposed protocol. The analysis shows that our protocol can satisfy the security requirements when implemented on two devices. Our scheme is secure if computing the DL problem, CDH problem and k-CAA problem are infeasible.
3) We implement our protocol on two Android devices and a personal computer (PC). The experimental results show that our protocol is efficient and practical in real-world applications.

In Sections 2 and 3, we review related literature on threshold secret sharing protocol and background materials (i.e., bilinear pairings, IEEE standard for identity-based signature protocol, and the underpinning mathematical assumptions), respectively. In Section 4, we present our two-party distributed signing protocol. We then analyzing its security in Section 5, and its performance in Section 6. Finally, we conclude this paper in the last section.

2 Related Work

Shamir [25] and Blakley [26] independently introduced the concept of threshold secret sharing to protect the security of (private) keys, and designed the first two such protocols. However, cheaters can attempt to beat the system by presenting fake shares, resulting in honest shareholders obtaining a fake secret. Shamir's protocol is not capable of detecting such a malicious user in the process of secret reconstruction. Thus, Harn and Lin [28] extended Shamir's protocol to detect and identify cheater utilizing redundant shares. In their approach, it is assumed that there are more than t participants in the secret reconstruction.

In a follow-up work, Tian et al. [29] presented a fair (t, n) threshold secret sharing protocol using the cheater detection proposed by Harn and Lin [28]. In their approach, the authors studied the fairness of secret reconstruction and demonstrated that their protocol achieves security and fairness against three attacks. Two of the three attacks are relevant to synchronous network, and one attack is relevant to asynchronous network. However, Harn [30] pointed out that the protocol of Tian et al. [29] can only be deployed in a synchronous network, contrary to the authors' claim. Harn also pointed out that Tian et al.'s protocol is vulnerable to impersonation attacks performed by an external adversary, since this adversary can send a forged valid share to other users after t shares are released.

Tompa and Woll [31] designed a protocol in which the secret message is hidden in the same false secret sequence. The cheaters have an advantage over a honest user in guessing the position of the real secret in sequence. Lee and Laih [32] proposed a V-fairness (t, n) secret sharing protocol, where they defined the number of v (i.e., the number cheaters) to be less than half of t. Thus, all players have an equal chance of revealing the secret key when they release their shares simultaneously. Hwang and Chang [33] proposed an improved protocol, in which they used twice cheater detection method to enforce each player having only two shares. However, the private key can be recovered during the signing or encryption phase.

3 Preliminaries

A summary of notations used in this paper is presented as follows.

- U_i: User
- P_1, P_2: two devices for U_i

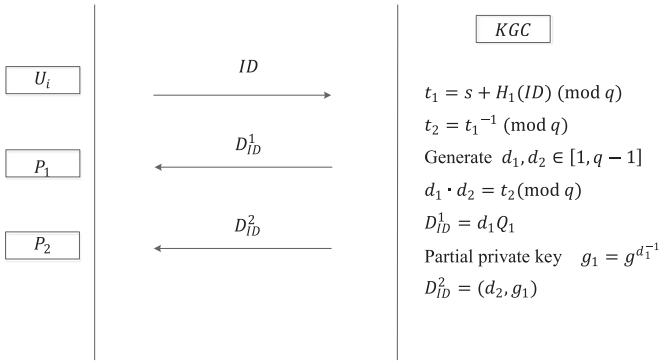


Fig. 2. Key generation.

- q: the order of group
- s: the master key of KGC
- G_1, G_2: additive cyclic groups of order q
- G_T: multiplicative cyclic group of order q
- Q_1, Q_2: the generators of additive cyclic groups G_1, G_2
- D_{ID}: private key of user U_i
- D_{ID}^1, D_{ID}^2: partial private key of two devices P_1, P_2
- H_1, H_2: hash function of $\{0,1\}^* \rightarrow Z_n^*$
- e: bilinear pairing of $G_1 \times G_2 \rightarrow G_T$
- g: $e(Q_1, Q_2)$
- g^u: exponentiation
- $[x, y]$: number set between x and y

3.1 Bilinear Pairing

Let G_1 and G_2 denote additive cyclic groups, G_T denote a multiplication cyclic group, and $e : G_1 \times G_2 \rightarrow G_T$ denotes a bilinear map. Suppose Q_1 and Q_2 are the generators of G_1 and G_2, g is the element that Q_1, and Q_2 maps to G_T. Thus, the map e is a bilinear pairing on the condition that e satisfies the following properties:

- *Bilinear*: Given any two elements $a, b \in Z_q^*$, and $\forall X \in G_1, \forall Y \in G_2$, there is $e(a \cdot X, b \cdot Y) = e(X, Y)^{a \cdot b}$.
- *Non-degenerate*: There exists at least one element X satisfies the inequation $e(X, X) \neq 1$.
- *Efficient Computability*: Given any two elements $\forall X \in G_1, \forall Y \in G_2$, there exists at least one efficient algorithm to compute $e(X, Y)$.

3.2 IEEE Standard for Identity-Based Signature

We briefly review the IEEE standard for identity-based signature scheme (BLMQ signature scheme) [34]. The steps of signature generation are as follows:

1. **Setup**
   Input: a security parameter k
   Output: a public parameter set P
   (a) Establishes the cyclic groups G_1, G_2, G_T and a bilinear map $e : G_1 \times G_2 \rightarrow G_T$.
   (b) Selects two random generators $Q_1 \in G_1$ and $Q_2 \in G_2$.
   (c) Generates a random server secret s in Z_q^*, then calculates $P_{pub} = s \cdot Q_2$ and $g = e(Q_1, Q_2)$.
   (d) Makes the set $P = \{P_{pub}, g, Q_1, Q_2, G_1, G_2, G_3, e\}$.
2. **Extract**
   Input: a user's identity ID, a public parameter set P and the server secret s
   Output: user's private key D_{ID}
   (a) Computes the identity element $h_{ID} = H_1(ID)$ in Z_q^*.
   (b) Computes $D_{ID} = (s + h_{ID})^{-1} Q_1$.
3. **Sign**
   Input: a message m, a public parameter set P and user's private key D_{ID}.
   Output: a signature σ.
   (a) Generates a random number $r \in Z_q^*$, and computes $u = g^r$.
   (b) Computes $h = H_2(m, u)$ and $S = (r + h) \cdot D_{ID}$.
   (c) Outputs signature $\sigma = (h, S)$.
4. **Verify**
   Input: a signature σ, a message m and user's public key h_{ID}.
   Output: $valid$ if the signature is accepted, otherwise output $invalid$.
   (a) Computes $u = \frac{e(S, h_{ID} \cdot Q_2 + P_{pub})}{g^h}$.
   (b) If $h = H_2(m, u)$, then outputs $valid$, otherwise outputs $invalid$.

4 Proposed Two-Party Signing Protocol

In this section, we describe the two-party signing protocol, which consists of the key generation phase and the distributed signature generation phase.

4.1 Key Generation Phase

In the key generation phase (see Fig. 2), KGC generates the user's private keys D_{ID}^1 and D_{ID}^2. Then, KGC distributes D_{ID}^1 and D_{ID}^2 to be stored on two devices P_1 and P_2, respectively.

1. U_i sends a request (i.e., user ID ID) to KGC.
2. Upon receiving the request, KGC computes $t_1 = s + H_1(ID)$ and $t_2 = t_1^{-1}$ (mod q).
3. KGC generates two random numbers $d_1 \in [1, q-1]$, and computes $d_2 = t_2 \cdot {d_1}^{-1}$ (mod q).
4. KGC computes the first partial private key $D_{ID}^1 = d_1 \cdot Q_1$ and sends D_{ID}^1 to P_1.
5. Then, KGC computes a partial private key $g_1 = g^{d_1^{-1}}$ and the second partial private key $D_{ID}^2 = (d_1, g_1)$. Finally, KGC sends D_{ID}^2 to P_2.

Notice that, the user's private key $D_{ID} = d_2 D_{ID}^1$.

4.2 Distributed Signature Generation Phase

In the distributed signature generation phase (see Fig. 3), the user inputs the message m to be signed and receives a legitimate signature $\sigma = (h, S)$ as the output.

1. $P_1 \rightarrow P_2$: $\{request\}$
   P_1 sends a signature request to P_2.
2. $P_2 \rightarrow P_1$: $\{\mu_1, \mu_2\}$
   (a) Upon receiving the request from P_1, P_2 chooses two random numbers $k_1, k_2 \in [1, q-1]$.
   (b) P_2 computes $\mu_1 = g_1^{k_1}$ and $\mu_2 = g^{k_2}$.
   (c) P_2 sends $\{\mu_1, \mu_2\}$ to P_1.
3. $P_1 \rightarrow P_2$: $\{h'\}$.
   (a) Upon receiving the message, P_1 chooses two random numbers $k_3, k_4 \in [1, q-1]$.
   (b) P_1 computes $\mu = \mu_1^{k_3} \cdot \mu_2 \cdot g^{k_4}$, $h = H_2(m, \mu)$ and $h' = h + k_4$ (mod q).

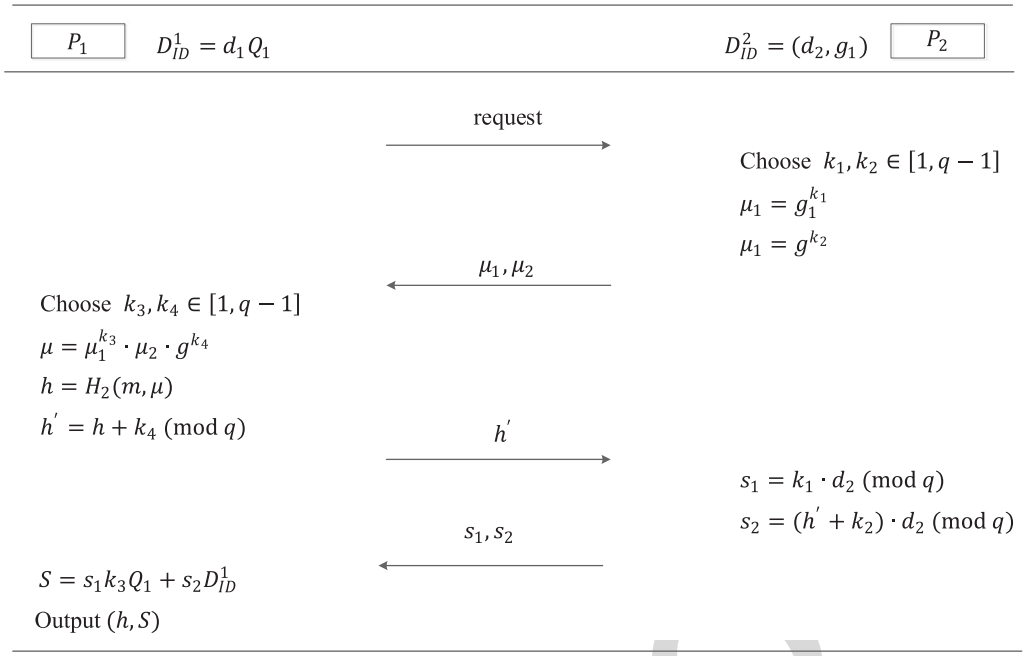


Fig. 3. Distributed signature generation.

(c) P_1 sends $\{h'\}$ to P_2.

4. $P_2 \rightarrow P_1$: $\{s_1, s_2\}$
   (a) Upon receiving the message, P_2 computes $s_1 = k_1 \cdot d_2$ (mod q) and $s_2 = (h' + k_2) \cdot d_2$ (mod q).
   (b) P_2 sends $\{s_1, s_2\}$ to P_1.
5. P_1 outputs the signature
   (a) P_1 receives the message, and computes $S = s_1 k_3 Q_1 + s_2 D_{ID}^1$.
   (b) P_1 outputs the signature (h, S).

4.3 Correctness

It is easy for P_1 to compute $\mu = g^{k_1 \cdot k_3 \cdot d_1^{-1} + k_2 + k_4}$, and

$$\begin{aligned} S &= k_3 \cdot s_1 Q_1 + s_2 D_{ID}^1 \\ &= k_1 \cdot k_3 \cdot d_2 Q_1 + (h + k_4 + k_2) \cdot d_2 D_{ID}^1 \\ &= k_1 \cdot k_3 \cdot d_2 \cdot d_1^{-1} D_{ID} + (h + k_4 + k_2) D_{ID} \\ &= (k_1 \cdot k_3 \cdot d_1^{-1} + k_2 + k_4 + h) D_{ID}. \end{aligned}$$

Therefore, the correctness of the proposed protocol for the identity-based signature scheme in the IEEE P1363 standard is demonstrated.

4.4 Verify

When a verifier receives the signature (h, S) on a message m, the following actions are undertaken.

(1) Computes $\mu^* = \frac{e(S, H_1(ID) Q_2 + P_{pub})}{g^h}$.
(2) Computes $h^* = H_2(m, \mu^*)$.
(3) If and only if $h^* = h$, outputs *valid*, otherwise outputs *invalid*.

5 Security Analysis

5.1 Mathematical Assumptions

We define the mathematical assumptions required in our security proof.

• *Discrete Logarithm (DL) Problem*: Let G be cyclic groups of prime order q. The DL problem in G is to compute $a \in Z_q$ for any given $P, Y = aP \in G$. An probability polynomial time (P.P.T) algorithm $\mathcal{A}$ has advantage ϵ in solving DL in G if

$$\Pr\left[\mathcal{A}(P, Y) = a : a \in Z_q, Y = aP\right] \geqslant \epsilon.$$

We say that DL problem in G is infeasible if all polynomial time (P.P.T) algorithms have a negligible advantage ϵ in solving DL in G.

• *Computational Diffie-Hellman (CDH) Problem*: Let G be cyclic groups of prime order q. The CDH problem in G is, given $P, aP, bP \in G$ for randomly choose $a, b \in Z_q$, to compute $abP \in G$. A P.P.T algorithm $\mathcal{A}$ has advantage ϵ in solving CDH in G if

$$\Pr\left[\mathcal{A}(P, aP, bP) = abP : a, b \in Z_q\right] \geqslant \epsilon$$

We say that CDH problem in G is infeasible if all P.P.T algorithms have a negligible advantage ϵ in solving co-CDH in G.

• *Collusion Attack Algorithm with k traitor (k-CAA) Problem*: Let G be cyclic groups of prime order q. For an integer k, the number $x, h_1, h_2, ..., h_k \in Z_q$ and a generator P of G. A k-CAA problem instance is: given $P, Q = xP \in G$, and k pairs $(h_1, (x + h_1)^{-1} P), \ldots, (h_k, (x + h_k)^{-1} P)$, finding a pair $(h, (x + h)^{-1} P)$ for some $h \notin h_1, h_2, ..., h_k$. A P.P.T algorithm $\mathcal{A}$ has advantage ϵ in solving k-CAA problem in G if

$$\Pr\left[\mathcal{A}\left(P, xP, \left(h_1, (x + h_1)^{-1} P\right), \ldots, \left(h_k, (x + h_k)^{-1} P\right)\right) = \left(h, (x + h)^{-1} P\right) : x \in Z_q^*, h \notin h_1, h_2, ..., h_k\right] \geqslant \epsilon.$$

We say that k-CAA problem in G is infeasible if any polynomial time algorithm have a negligible advantage ϵ in solve k-CAA problem in G.

5.2 System Model

Definition 1. *Let $\mathcal{A}$ be a P.P.T adversary, and π be a digital signature protocol $\pi = \{Gen, Sign, Verify\}$. We define $Sign_{\mathcal{A},\pi}(1^n)$ as follows.*

1. *$(pk, sk) \leftarrow Gen(1^n)$. $\mathcal{A}$ runs the $Gen(1^n)$ algorithm, and outputs a random key pair (pk, sk).*
2. *$(m^*, \sigma^*) \leftarrow \mathcal{A}^{Sign_{sk}(\cdot)}$. $\mathcal{A}$ runs the $Sign(\cdot)$ algorithm, and outputs (m^*, σ^*).*
3. *The set of all m that $\mathcal{A}$ queries to its oracle is $\mathcal{M}$. When $m^* \notin \mathcal{M}$ and $Verify_{pk}(m^*, \sigma^*) = 1$, the experiment outputs 1.*

Definition 2 (Existentially Unforgeable under Choose Message Attacks. For any P.P.T adversary). *$\mathcal{A}$, if there exists a negligible function μ for every n that satisfies $Pr[Sign_{\mathcal{A},\pi}(1^n) = 1] \leq \mu(n)$, then the signature protocol π is existentially unforgeable under the chosen message attack.*

Definition 3. *Let $\mathcal{A}$ be a P.P.T adversary, and π be a digital signature protocol $\pi = \{Gen, Sign, Verify\}$. We define $DistSign^b_{\mathcal{A},\Pi}(1^n)$, $b \in \{1, 2\}$ as follows.*

1. *$(pk, sk) \leftarrow \mathcal{A}^{\Pi_{3-b}}(1^n)$.*
2. *$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\Pi_b(\cdot,\cdot)}$.*
3. *The set of all inputs $m \in (sid, m)$ that $\mathcal{A}$ queries to its oracle is $\mathcal{M}$, and the identifier sid should not be previously queried. When $m^* \notin \mathcal{M}$, $Verify_{pk}(m^*, \sigma^*) = 1$ and $Verify$ are as specified in π, the experiment outputs 1.*

Definition 4 (Secure Two-Party Distributed Signature Scheme). *For any P.P.T adversary $\mathcal{A}$ and every $b \in \{1, 2\}$, if there exists a negligible function μ for every n that satisfies $Pr[DistSign^b_{\mathcal{A},\Pi}(1^n) = 1] \leq \mu(n)$, then Π is a secure two-party protocol for distributes signature generation for π.*

Definition 5 (Blindness). *Given two signature records and a signature pair, for any P.P.T adversary $\mathcal{A}$, $\mathcal{A}$ is not able to specify which signature record belongs to the signature pair, then Π is not blind.*

In the above definitions, we should first run the distributed key generation phase and only once, before running the distributed signature phase. A honest device P_{3-b} operates the protocol Π and makes the stateful oracle $\Pi_b(\cdot, \cdot)$. In the two-party signature generation phase $DistSign^b_{\mathcal{A},\Pi}$, $\mathcal{A}$ can control the device $P_b, b \in \{1, 2\}$, choose the messages to be signed, and interact with multiple instances to generate a forged signature concurrently.

$\mathcal{A}$ can win the game if the message m^* for the forged signature is not queried. The *oracle* has two inputs: a session identifier and an input, and it works as follows:

1. On receiving a query (sid, m), if the distributed key generation phase has not completed, then *oracle* returns $\perp$.
2. In the event that a query (sid, m) is received when the distributed key generation phase has commenced, if sid is the first to be queried, then the device P_{3-b} operates the protocol Π with session identifier sid and the input message m which is to be signed. What the device P_{3-b} first sends in the sign phase is the *oracle* replay.
3. On receiving a query (sid, m) when the distributed key generation phase has computed and identifier sid has been queried, the *oracle* sends the message m to the device P_{3-b} and returns the next message output from P_{3-b}.

5.3 Security Proof

We prove the protocol Π is a secure two-party distributed signature protocol and existentially unforgeable under chosen message attacks, and the protocol is also blind, in this section.

Theorem 1. *Suppose that IEEE standard for identity-based (BLMQ) signature scheme is existentially unforgeable under choose message attacks, then our protocol is a secure two-party distributed signature and existentially unforgeable under choose message attacks.*

Proof. If an adversary $\mathcal{A}$ can break the protocol with the probability ϵ, then it can break the protocol with probability $\epsilon + \mu(n)$, where μ is a negligible function. In this proof, we separately prove the security of a corrupted P_1 and a corrupted P_2. For any $\mathcal{A}$ who can attack the protocol and forge a signature in $DistSign^b_{\mathcal{A},\Pi}(1^n)$, we select another adversary $\mathcal{S}$ who can forge a BLMQ signature in $Sign_{\mathcal{S},\pi}(1^n)$ with a negligibly close probability. Then, for every P.P.T adversary $\mathcal{A}$ and $b \in \{1, 2\}$, there exists a negligible function μ for every n and an adversary $\mathcal{S}$, that satisfies,

$$|Pr[Sign_{\mathcal{S},\pi}(1^n) = 1] - Pr[DistSign^b_{\mathcal{A},\Pi}(1^n) = 1]| \leq \mu(n)$$

□

According to Definition 2, there exists a negligible function μ' for every n that satisfies $Pr[Sign_{\mathcal{A},\pi}(1^n) = 1] \leq \mu'(n)$. Combining this with the above equation, we obtain $Pr[DistSign^b_{\mathcal{S},\Pi}(1^n) = 1] \leq \mu(n) + \mu'(n)$. Thus, we prove the equation from two aspects: $b = 1$ and $b = 2$.

Proof of b = 1. The adversary corrupts device P_1. Assume $\mathcal{A}$ can forge a signature in $DistSign^b_{\mathcal{A},\Pi}(1^n)$ and $\mathcal{S}$ can forge a BLMQ signature in $Sign_{\mathcal{S},\pi}(1^n)$. Thus, $\mathcal{A}$ can use the response of $\mathcal{S}$.

1. $\mathcal{S}$ receives $(1^n, H_1(ID))$, in which $H_1(ID)$ is user's public key.
2. $\mathcal{S}$ invokes $\mathcal{A}$ and simulates the instructions of $\mathcal{A}$ in $DistSign$ with the input of (1^n) and the output as described in the following:
   (a) If the distributed key generation phase has not completed, then $\mathcal{S}$ replies $\perp$ to all queries (sid, m).
   (b) On receiving a query (sid, m), if sid is the first to be queried, $\mathcal{S}$ queries the signature in $DistSign$ and receives (h, S), and $\mathcal{S}$ can compute μ in then BLMQ signature scheme, then $\mathcal{A}$ queries $\mathcal{S}$ with identifier sid as below.
   Case 1:
      i. The first message m_1 in (sid, m_1) as $(prove, 1, (\mu_1, \mu_2), (k_1, k_2))$ that $\mathcal{S}$ sends to $\mathcal{F}_{zk}$, if $\mu_1 = g_1^{k_1} = g^{k_1 d_1^{-1}}$ and $\mu_2 = g^{k_2}$, then $\mathcal{A}$ computes $\{\mu, h'\}$, and replies the message $(prove, 2, \mu, k_3)$ to $\mathcal{S}$.

ii. On receiving the message, if the proof is $valid$, then $\mathcal{S}$ sets $s_1 = k_3^{-1} \cdot S$ and replies the message $(prove, 3, s_1, k_3^{-1})$ to $\mathcal{A}$.

Case 2:

i. $\mathcal{A}$ does not execute the instructions of P_1 and set $\mu = g^x$ and $h = H_2(m, \mu)$. Then, $\mathcal{A}$ sends the message $(prove, 1, \mu, x)$ to $\mathcal{S}$.

ii. On receiving the message, if the proof is $valid$, then $\mathcal{S}$ generates random (s_1, s_2) and sends them to $\mathcal{A}$.

3. $\mathcal{A}$ computes σ^* and outputs a signature (m^*, σ^*), then $\mathcal{S}$ terminals the simulation and outputs (m^*, σ^*). □

The view of $\mathcal{A}$ in the simulation of key generation phase is different from the real execution of protocol Π. The way a honest P_1 generates μ in the real Π is that: P_1 uses the message from P_2 to generate μ, whereas $\mathcal{A}$ computes $\mu \leftarrow g^x$, in which x is a random number. We consider that $\mathcal{S}$ behaves as P_2 completely in all messages. Since $\{x, k_1, k_2, k_3, k_4\}$ are all chosen randomly, we determine that there is no distinction between $\mu = g^x$ and $\mu = \mu_1^{k_3} \mu_2 g^{k_4}$.

In Case 1 of the distributed signing phase, the simulator P_2 generates s_1 and s_2 using the random numbers $\{k_1, k_2\}$ and its own private key d_2. The output of $\mathcal{A}$ is computationally indistinguishable to the real output signature in a real protocol. In other words, $\mathcal{A}$ outputs a valid signature pair (m^*, σ^*) with the same probability in the simulation and in real $DistSign$. The signatures can be distinguished by verifying whether they match with the public key. When $\mathcal{A}$ receives messages from P_2, it computes the signature S^*. We arrive at the following equation:

$$\begin{aligned} S^* &= k_3 \cdot k_3^{-1} \cdot S = (k_3 k_1^* d_1^{-1} + k_2 + k_4 + h) \cdot D_{ID} \\ &= (k_3 k_1^* d_1^{-1} + k_2 + k_4 + h) \cdot d_1 d_2 \cdot Q_1. \end{aligned}$$

If $\mathcal{A}$ knows the valid signature, then $\mathcal{A}$ can compute d_2 with a non-negligible probability ϵ. That means $\mathcal{A}$ can solve the DL problem with a non-negligible probability ϵ. According to the DL problem, $Pr[d_2 | (k_3 k_1^* d_1^{-1} + k_2 + k_4 + h) \cdot d_1 d_2 \cdot Q_1] \leq \mu(n)$, which contradicts

$$\epsilon \leq Pr[d_2 | (k_3 k_1^* d_1^{-1} + k_2 + k_4 + h) \cdot d_1 d_2 \cdot Q_1] \leq \mu(n).$$

Thus, $\mathcal{A}$ cannot obtain the private key of P_2 even when P_1 corrupted and knows the correct signature.

In Case 2 of the distributed signing phase, since the simulator's private key d_2 and KGC's secret key s are unknown to $\mathcal{A}$, $\mathcal{A}$ selects a random number d_2^* to construct signature $S^* = (x + h) \cdot d_1 \cdot d_2^* \cdot Q_1$. There exists:

$$\begin{aligned} \mu^* &= \frac{e(S^*, H_1(ID) Q_2 + P_{pub})}{g^h} \\ &= \frac{e(S^*, H_1(ID) Q_2 + s Q_2)}{g^h} \\ &= \frac{e(S^*, (H_1(ID) + s) Q_2)}{g^h} \\ &= \frac{e((x + h) d_1 d_2^* Q_1, (H_1(ID) + s) Q_2)}{g^h} \\ &= \frac{g^{(x+h) d_1 d_2^* \cdot (H_1(ID) + s)}}{g^h} = g^{x d_1 d_2^* \cdot (H_1(ID) + s)} \\ \frac{\mu^*}{\mu} &= \frac{g^{x d_1 d_2^* \cdot (H_1(ID) + s)}}{g^x} = g^{d_1 d_2^* \cdot (H_1(ID) + s)}. \end{aligned}$$

$\mathcal{A}$ obtains $g^{d_1 d_2^* \cdot (H_1(ID) + s)}$ and outputs d_2^* as the answer to the CDH problem. Suppose that $\mathcal{A}$ can select the right d_2^* to make the above equations hold in a non-negligible probability ϵ. That means $\mathcal{A}$ can solve the CDH problem in a non-negligible probability ϵ. According to the CDH problem, $Pr[d_2^* | g^{d_1 d_2^* \cdot (H_1(ID) + s)}] \leq \mu(n)$, This contrads

$$\epsilon \leq Pr[d_2^* | g^{d_1 d_2^* \cdot (H_1(ID) + s)}] \leq \mu(n).$$

Thus, the signature cannot be verified based on the unknown d_2 and s.

Proof of b = 2. The adversary corrupts device P_2, and we follow the same steps as the case of $b = 1$. Unlike with $b = 1$, in this case, the signature is computed by $\mathcal{S}$ and replied to $\mathcal{A}$. In other words, $\mathcal{A}$ can forge a signature in $DistSign^b_{\mathcal{A},\Pi}(1^n)$ and $\mathcal{S}$ can forge a BLMQ signature in $Sign_{\mathcal{S},\pi}(1^n)$. Thus, the adversary $\mathcal{A}$ cannot forge a legitimate signature under the circumstances that $\mathcal{S}$ private key is not known to $\mathcal{A}$. □

This simulation is similar to the above with a difference that the last message P_2 to P_1 composes of random numbers. P_1 cannot determine whether the random numbers used in messages from P_2 is consistent. In this simulation, $\mathcal{S}$ computes the signature pair and replies them to $\mathcal{A}$. If $\mathcal{A}$ can forge a legitimate signature σ^*, then $\mathcal{A}$ can carry out the attack without knowing the private key of P_1. We solve this problem by getting $\mathcal{S}$ to abort the simulation at some random points. $\mathcal{S}$ chooses a random number $i \in \{1, ..., p(n) + 1\}$, in which $p(n)$ means the number of queries. If $\mathcal{S}$ chooses correctly, then the simulation is successful.

1. $\mathcal{S}$ receives $(i^n, H_1(ID))$, in which $H_1(ID)$ is user's public key.
2. Let $p(\cdot)$ denotes the number of queries that $\mathcal{A}$ executes protocol Π. Thus, $\mathcal{S}$ can choose a random number $i \in \{1, ..., p(\cdot) + 1\}$.
3. $\mathcal{S}$ invokes $\mathcal{A}$ and simulates the instructions of $\mathcal{A}$ in $DistSign$ with the input of (1^n). The output is described in the following:
   (a) If the distributed key generation phase has not completed, then $\mathcal{S}$ replies $\perp$ to all queries (sid, m).
   (b) On receiving a query (sid, m), if sid is the first queried, then $\mathcal{S}$ computes and sends the corresponding reply $(proof, sid)$ to $\mathcal{A}$.
   (c) After $\mathcal{S}$ queries the signature in $DistSign$ and returns (h, S), $\mathcal{S}$ can compute the μ in BLMQ signature scheme. Then, $\mathcal{A}$ queries $\mathcal{S}$ with identifier sid as below:
      i $\mathcal{A}$ sends the first message $m_1 = (prove, 1, (\mu_1, \mu_2), (k_1, k_2))$ in (sid, m_1). If $\mu_1 = g_1^{k_1} = g^{k_1 d_1^{-1}}$ and $\mu_2 = g^{k_2}$, then $\mathcal{S}$ sets $\mu = g^{k_1 d_1^{-1} k_3 + k_2}$ and $h = H_2(m, \mu)$, and replies the message $(proof, 2, h)$ to $\mathcal{A}$. Otherwise, $\mathcal{S}$ terminates the simulation.
      ii The second message m_2 in (sid, m_2) is denoted as $(prove, 3, (s_1, s_2), d_2)$. If this is the $i - th$ query of $\mathcal{A}$ to Π, then $\mathcal{S}$ terminates the simulation and returns $\perp$. Otherwise, it continues.
4. Whenever $\mathcal{A}$ stops the query and outputs the (m^*, σ^*), $\mathcal{S}$ terminates the simulation and outputs (m^*, σ^*).

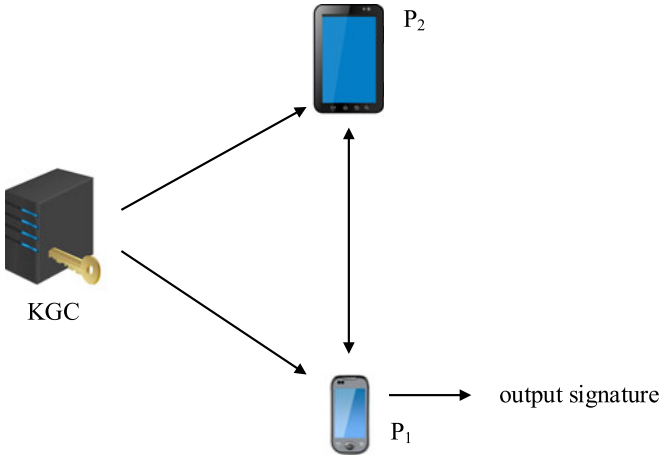


Fig. 4. Model of key generation.

$\mathcal{A}$ can obtain $p(n)$ correct signatures since $i \in \{1, \ldots, p(n)+1\}$. Thus, we have the following equation:

$$\mu^* = \frac{e(S^*, H_1(ID)Q_2 + P_{pub})}{g^h} = \frac{e(S^*, (H_1(ID)+s)Q_2)}{g^h}$$
$$= \frac{e((k_1k_3d_1^{-1}+k_2+h)d_1d_2Q_1, (H_1(ID)+s)Q_2)}{g^h}$$
$$= \frac{g^{(k_1k_3d_1^{-1}+k_2+h)}}{g^h} = g^{k_1k_3d_1^{-1}+k_2}.$$

In this simulation, $\mathcal{A}$ keeps its own random number invariant during all queries, and $\mathcal{S}$ generates different randomly in every query. Thus, for $p(n)$ correct signatures, $\mathcal{A}$ computes g^{k_2} at the beginning of protocol Π, and there exists:

$$\mu_1^* = g^{k_1k_3^1d_1^{-1}+k_2}$$
$$\mu_2^* = g^{k_1k_3^2d_1^{-1}+k_2}$$
$$\cdot$$
$$\mu_n^* = g^{k_1k_3^nd_1^{-1}+k_2}.$$

$\mathcal{A}$ obtains $g^{k_1k_3^id_1^{-1}+k_2}, i \in \{1, ..., p(n)+1\}$ and outputs $k_3d_1^{-1}$ as the answer to k-CAA problem. Suppose that $\mathcal{A}$ can compute $k_3d_1^{-1}$ to make the above equations hold in a non-negligible probability ϵ. That means $\mathcal{A}$ can solve the k-CAA problem in a non-negligible probability ϵ. According to the Collusion Attack Algorithm with k Traitor, $Pr[k_1k_3d_1^{-1}|g^{k_1k_3^id_1^{-1}+k_2}] \leq \mu(n)$. This contradicts

$$\epsilon \leq Pr[k_1k_3d_1^{-1}|g^{k_1k_3^id_1^{-1}+k_2}] \leq \mu(n).$$

Thus, $\mathcal{A}$ cannot know any information about P_1 in the event that P_2 is corrupted.

Theorem 2. *Suppose a P.P.T adversary $\mathcal{A}$ is able to specify which signature record belongs to the signature pair in a negligible function μ for every n, then we can say our protocol is blind.*

Proof. Suppose $\mathcal{A}$ can distinguish the signature records with the probability ϵ. In this proof, we separately prove the blindness of device P_2 and an external adversary.

For device P_2. For every signature pair (m, σ) generated by device P_1, the signature record for P_2 includes $\{\mu_1, \mu_2, h', s_1, s_2, k_1, k_2, d_2\}$. Given two signature records $\{\mu_1', \mu_2', h'', s_1', s_2', k_1', k_2', d_2\}$ and $\{\mu_1'', \mu_2'', h''', s_1'', s_2'', k_1'', k_2'', d_2\}$, and a signature pair (m, h, S). From the signature pair (m, h, S), P_2 can compute:

$$\mu = \frac{e(S, H_1(ID)Q_2 + P_{pub})}{g^h}$$
$$= g^{k_1k_3^1d_1^{-1}+k_2+k_4}.$$

P_2 can compute k_4 on the basis of $h' = h + k_4$. If P_2 can specify which of $\{k_1', k_2'\}$ and $\{k_1'', k_2''\}$ belongs to μ, that means, P_2 can solve DL problem in a non-negligible probability ϵ. According to Discrete Logarithm Problem, $Pr[k_3, d_1|g^{k_1k_3^1d_1^{-1}+k_2}] \leq \mu(n)$, it's contradictory

$$\epsilon \leq Pr[k_3, d_1|g^{k_1k_3^1d_1^{-1}+k_2}] \leq \mu(n).$$

P_2 cannot specify which record belongs to the signature pair.

For external adversary. For every signature pair (m, σ) generated by device P_1, the signature record for an external adversary includes $\{\mu_1, \mu_2, h', s_1, s_2\}$. Given two signature records $\{\mu_1', \mu_2', h'', s_1', s_2'\}$ and $\{\mu_1'', \mu_2'', h''', s_1'', s_2''\}$, and a signature pair (m, h, S). From the signature pair (m, h, S), the adversary can compute:

$$\mu = \frac{e(S, H_1(ID)Q_2 + P_{pub})}{g^h}$$
$$= g^{k_1k_3^1d_1^{-1}+k_2+k_4}.$$

The adversary can compute k_4 on the basis of $h' = h + k_4$. For $\{\mu_1', \mu_2'\}$, the adversary cannot compute k_1' and k_2' without solving the DL problem in a non-negligible probability ϵ. Thus, the adversary cannot specify which record belongs to the signature pair. □

6 Performance Evaluation

We implemented our protocol using the MIRACL library [35] on two Android devices (i.e., Samsung Galaxy S5 with a Quad-core 2.45 G processor, 2G bytes memory and the Google Android 4.4.2 operating system and Google Nexus 6 with a Quad-core, 2.7 GHz processor, 3G bytes memory and the Google Android 7.1.2 operating system) and a PC (Dell with an i7-6700 3.40 GHz processor, 8G bytes memory and the window 10 operating system). In the evaluation, the PC is the KGC which generates partial private keys for both devices. The two devices interact with each other via bluetooth, as shown in Fig. 4.

The security levels of the different curves are presented in Table 1, and we know that:

1. The security of MNT $k = 6$ curve corresponds to 80 bits cipher length of AES.
2. The security of BN $k = 12$ curve corresponds to 128 bits cipher length of AES.
3. The security of KSS $k = 18$ curve corresponds to 192 bits cipher length of AES.
4. The security of BLS $k = 24$ curve corresponds to 256 bits cipher length of AES.

The implementation consists of three parts, namely: setup, extract, signature and verify. The protocol Π

TABLE 1
Security Level

Type	Symmetric cipher key length (AES)
MNT $k = 6$	80
BN $k = 126$	128
KSS $k = 18$	192
BLS $k = 24$	256

TABLE 2
Distributed Key Generation and Verification

Type \ Process	Setup	Distributed	Verify
MNT $k = 6$	149.16	21.55	130.65
BN k = 12	**137.28**	**59.46**	**214.87**
KSS $k = 18$	2075.37	666.18	2385.73
BLS $k = 24$	19337.18	17052.52	34181.8

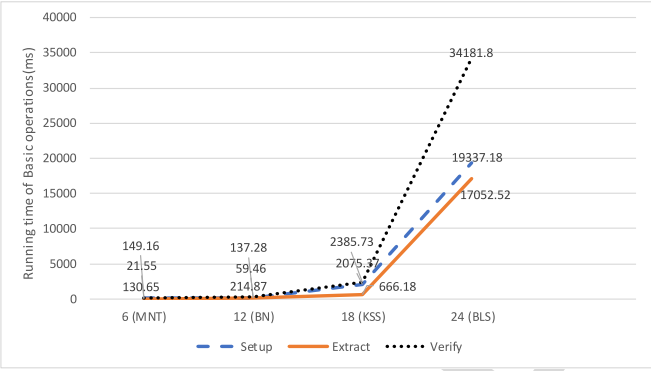


Fig. 5. Runtime of the basic operation.

generates a public parameter set P and the user's private key in the setup phase and the extract phase, respectively. P_1 verifies the validity of the signature after the signature phase. The runtime of each part is shown in Table 2 and Fig. 5.

We analyze the computation cost of the signature phase, whose runtime is displayed in Table 3 and Fig. 6. In the signature phase, step1 refers to the execution of instructions by P_2 after receiving a request from P_1, step2 refers to the execution of instructions by P_1 before sending messages to P_2, step3 refers to the execution of instructions by P_2, and step4 refers to the execution of instructions by P_1 after receiving messages from P_2.

According to the above evaluation, we analyze the communication cost of BN curve, therefore the length of elements in Z_q is 256 bits. In the key generation phase of our protocol, the length of the message that the KGC sends to P_1 is 512 bits, the length of the message that the KGC sends to P_2 is 512 bits. In the distributed signature generation phase, the length of P_2's first message and second message are 1024 bits and 512 bits respectively. The length of P_1's message is 256 bits. Thus, one can observe that our protocol is efficient and practical for real-world deployment.

TABLE 3
Running Times of Signature Phase

Type \ Process	Step1	Step2	Step3	Step4
MNT $k = 6$	35.84	2.3	0.82	6.18
BN k = 12	**103.79**	**16.3**	**1.11**	**9.38**
KSS $k = 18$	1267.29	554.96	5.19	29.65
BLS $k = 24$	33776.69	25916.68	36.46	93.43

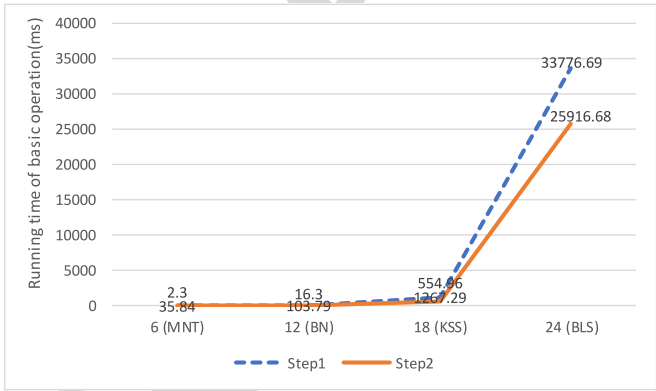


Fig. 6. Runtime of distributed signing.

7 Conclusion

In this paper, we proposed a novel, secure and efficient two-party distributed signature protocol, which builds on the identity-based signature scheme in the IEEE P1363 standard. Specifically, a valid signature can be generated without requiring the entire private key to be reconstructed, and the key cannot be generated from one (lost or stolen) device. This can be extremely attractive to mobile device users, as more users own more than one mobile device. In other words, a misplaced or stolen device will not result in the complete compromise of the user's data. We also proved the security of the protocol, and evaluated its performance on two Android devices.

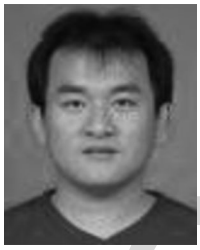
Future work includes designing an app based on the proposed protocol, and recruiting participants to install the app for more extensive evaluation. Another potential research direction is to implement the app on a broader range of mobile and Internet of Things (IoT) devices (e.g., wearable devices such as smart uniforms and smart weapons) for further evaluation.

Acknowledgments

We greatly appreciate the invaluable suggestions provided by the anonymous reviewers and the associate editor. The work was supported in part by the National Natural Science Foundation of China under Grant 61572379, Grant 61501333, and Grant U1536204, in part by the National High-Tech Research and Development Program of China (863 Program) under Grant 2015AA016004, in part by the open fund of State Key Laboratory of Cryptology and in part by the Natural Science Foundation of Hubei Province of China under Grant 2015CFB257. The last author is supported by the Cloud Technology Endowed Professorship.

REFERENCES

[1] C. I. N. I. C. Research, 2016. [Online]. Available: http://www.199it.com/archives/560209.html
[2] "China internet network information center research," 2017. [Online]. Available: http://omji24q1y.bkt.clouddn.com/The%2041st%20China%20Statistical%20Report%20on%20Internet%20Development.pdf
[3] C. V. N. Index, "Global mobile data traffic forecast update, 2015–2020 white paper," 2016. [Online]. Available: http://goo.gl/ylTuVx
[4] J. J. V. Díaz, A. B. R. González, and M. R. Wilby, "Bluetooth traffic monitoring systems for travel time estimation on freeways," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 1, pp. 123–132, Jan. 2016.
[5] M. J. Walker, "Hype cycle for emerging technologies, 2017," *Gartner*, vol. G00314560, pp. 1–68, 2017.
[6] D. W. Cearley, B. Burke, S. Searle, and M. J. Walker, "Top 10 strategic technology trends for 2018," *Gartner*, vol. G00327329, pp. 1–34, 2017.
[7] N. Bui, M. Cesana, S. A. Hosseini, Q. Liao, I. Malanchini, and J. Widmer, "A survey of anticipatory mobile networking: Context-based classification, prediction methodologies, and optimization techniques," *IEEE Commun. Surveys Tuts.*, vol. 19, pp. 1790–1821, Jul.-Sep. 2017.
[8] T. H. Nguyen and M. Reinhart, "Hype cycle for mobile device technologies, 2017," *Gartner*, vol. G00314103, pp. 1–62, 2017.
[9] Y. Yamazaki and T. Ohki, "Toward more secure and convenient user authentication in smart device era," *IEICE Trans. Inf. Syst.*, vol. 100, no. 10, pp. 2391–2398, Oct. 2017.
[10] C. J. D'Orazio and K.-K. R. Choo, "Circumventing ios security mechanisms for apt forensic investigations: A security taxonomy for cloud apps," *Future Generation Comput. Syst.*, vol. 79, pp. 247–261, 2018.
[11] C. J. D'Orazio and K.-K. R. Choo, "A technique to circumvent ssl/tls validations on ios devices," *Future Generation Comput. Syst.*, vol. 74, pp. 366–374, 2017.
[12] Q. Do, B. Martini, and K.-K. R. Choo, "Cyber-physical systems information gathering: A smart home case study," *Comput. Netw.*, vol. 138, pp. 1–12, 2018.
[13] N. L. Clarke and A. Mekala, "The application of signature recognition to transparent handwriting verification for mobile devices," *Inf. Manage. Comput. Secur.*, vol. 15, no. 3, pp. 214–225, 2007.
[14] R. Plamondon, G. Pirlo, and D. Impedovo, "Online signature verification," in *Proc. Handbook Document Image Process. Recognit.*, 2014, pp. 917–947.
[15] N. Sae-Bae and N. Memon, "Online signature verification on mobile devices," *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 6, pp. 933–947, Jun. 2014.
[16] D. Quick, B. Martini, and K.-K. R. Choo, *Cloud Storage Forensics*. New York, NY, USA: Syngress Publishing/Elsevier, 2014.
[17] C. Boyd, "Digital multisignatures," *Cryptography Coding*, 1986.
[18] Y. Desmedt, "Threshold cryptosystems," in *Proc. Int. Workshop Theory Appl. Cryptographic Tech.*, 1992, pp. 1–14.
[19] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust threshold dss signatures," in *Proc. Int. Conf. Theory Appl. Cryptographic Tech.*, 1996, pp. 354–371.
[20] V. Shoup, "Practical threshold signatures," in *Proc. Adv. Cryptology EUROCRYPT*, 2000, pp. 207–220.
[21] L. Harn and M. Fuyou, "Multilevel threshold secret sharing based on the chinese remainder theorem," *Inf. Process. Lett.*, vol. 114, no. 9, pp. 504–509, 2014.
[22] J. Kurihara, S. Kiyomoto, K. Fukushima, and T. Tanaka, "A new (k, n)-threshold secret sharing scheme and its extension," in *Proc. Int. Conf. Inf. Secur.*, 2008, pp. 455–470.
[23] M. Stadler, "Publicly verifiable secret sharing," in *Eurocrypt*, vol. 96. New York, NY, USA: Springer, 1996, pp. 190–199.
[24] T. Tassa, "Hierarchical threshold secret sharing," *J. Cryptology*, vol. 20, no. 2, pp. 237–264, 2007.
[25] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[26] G. R. Blakley, et al., "Safeguarding cryptographic keys," in *Proc. Nat. Comput. Conf.*, vol. 48, pp. 313–317, 1979.
[27] Y. Lindell, "Fast secure two-party ecdsa signing," in *Proc. Annu. Int. Cryptology Conf.*, 2017, pp. 613–644.
[28] L. Harn and C. Lin, "Detection and identification of cheaters in (t, n) secret sharing scheme," *Des. Codes Cryptography*, vol. 52, no. 1, pp. 15–24, 2009.
[29] Y. Tian, J. Ma, C. Peng, and Q. Jiang, "Fair (t, n) threshold secret sharing scheme," *IET Inf. Secur.*, vol. 7, no. 2, pp. 106–112, 2013.
[30] L. Harn, "Comments on'fair (t, n) threshold secret sharing scheme'," *IET Inf. Secur.*, vol. 8, no. 6, pp. 303–304, 2014.
[31] M. Tompa and H. Woll, "How to share a secret with cheaters," *J. Cryptology*, vol. 1, no. 3, pp. 133–138, 1989.
[32] C.-S. Laih and Y.-C. Lee, "V-fairness (t, n) secret sharing scheme," *IEE Proc.-Comput. Digital Tech.*, vol. 144, no. 4, pp. 245–248, 1997.
[33] R.-J. Hwang and C.-C. Chang, "Enhancing the efficiency of (v, r, n)-fairness secret sharing scheme," in *Proc. 18th Int. Conf. Adv. Inf. Netw. Appl.*, vol. 1, pp. 208–211, 2004.
[34] P. S. Barreto, B. Libert, N. McCullagh, and J.-J. Quisquater, "Efficient and provably-secure identity-based signatures and signcryption from bilinear maps," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2005, pp. 515–532.
[35] "Miracl library," 2017. [Online]. Available: https://www.miracl.com/



Debiao He received the PhD degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, in 2009. He is currently a professor of the State Key Lab of Software Engineering, Computer School, Wuhan University. His main research interests include cryptography and information security, in particular, cryptographic protocols.

Yudi Zhang received the master's degree from Hubei University of Technology of China, in 2017. He is currently working toward the PhD degree in Computer School, Wuhan University, Wuhan, China. His main research interests include cryptography and information security, in particular, cryptographic protocols.

Ding Wang received the PhD degree in information security from Peking University, in 2017. He is currently supported by the Boya Post- Doctoral Fellowship in Peking University, China. He has authored more than 40 papers at venues like ACM CCS and IEEE TDSC, and his papers get over 800 citations. His research interests mainly focus on password-based authentication and provable security. He received the Top-10 Distinguished Graduate Academic Award from Harbin Engineering University in 2013 and Peking University in 2016.

Kim-Kwang Raymond Choo (SM'15) received the PhD degree in information security from Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed professorship with The University of Texas at San Antonio (UTSA), and has a courtesy appointment with the University of South Australia. In 2016, he was named the Cybersecurity Educator of the Year—APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn), and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He is the recipient of the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, ESORICS 2015 Best Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008. He is also a fellow of the Australian Computer Society, and a senior member of the IEEE.

Queries to the Author

Q1. We have renumber all Definition. Please check for correctness.
Q2. Please provide complete bibliography details for reference [17].