Received December 30, 2018, accepted January 12, 2019, date of publication January 21, 2019, date of current version February 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2894062

Light-Weight and Privacy-Preserving Authentication Protocol for Mobile Payments in the Context of IoT

YANAN CHEN[1,2], **WEIXIANG XU**[1], **LI PENG**[3], **AND HAO ZHANG**[3]

[1]MOE Key Laboratory for Transportation Complex Systems Theory and Technology, School of Traffic and Transportation, Beijing Jiaotong University, Beijing 100044, China
[2]Basic Course Teaching Department, Jiangxi University of Science and Technology, Nanchang 330013, China
[3]School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China

Corresponding author: Weixiang Xu (wxxu@bjtu.edu.cn)

This work was supported in part by the National Science Foundation of China under Grant 61370026, Grant U1401257, Grant 61672002, Grant 61672135, Grant 61602096, Grant 41701120, and Grant 61502087, in part by the Science and Technology Project of Guangdong Province under Grant 2016A010101002, in part by the 13th Five-Year Plan of National Cryptography Development Fund for the Cryptographic Theory of China under Grant MMJJ20170204, and in part by the Fundamental Research Funds for the Central Universities under Grant ZYGX2016J091 and Grant ZYGX2015J072.

ABSTRACT The widespread use of smart devices attracts much attention on the research for a mobile payment protocol in the context of the Internet of Things (IoT). However, payment trust and user privacy still raise critical concerns to the application of mobile payments since existing authentication protocols for mobile payments either suffer from the heavy workload on a resource-limited smart device or cannot provide user anonymity in the mobile payment. To address these challenges elegantly, this paper presents a lightweight and privacy-preserving authentication protocol for mobile payment in the context of IoT. First, we put forward a *unidirectional* certificateless proxy re-signature scheme, which is of independent interest. Based on this signature scheme, this paper, then, gives a new mobile payment protocol that for the first time not only achieves anonymity and unforgeability but also leaves low resource consumption on smart devices. In the proposed protocol, the efficiency is notably improved by placing the most computational cost on Pay Platform (usually with abundant computational power) instead of lightweight mobile devices. Moreover, by considering that the Pay Platform and Merchant Server needs to perform computation for each transaction, the idea of batch-verification has been adopted to mitigate the overhead for millions of users at the Pay Platform and Merchant Server to address the scalability issue. Through the formal security analysis presented in this paper, the proposed protocol is proved to be secure under the extended CDH problem. In addition, the performance evaluation shows that the proposed protocol is feasible and efficient for the resource-limited smart devices in the IoT.

INDEX TERMS Anonymity, authentication, certificateless cryptosystem, proxy re-signature, mobile payments, IoT.

I. INTRODUCTION

With the development and pervasiveness of mobile communication technology [1], [2], mobile smart devices (i.e., smartphones or laptops) are widely used in daily life. This leads to an increasing number of requirements for various online services. As an important part of online services, mobile payments also get a lot of attentions so that many applications of mobile payments are developed, such as, Ali pay [3], Apple pay [4], WeChat pay [5] and so on. Nowadays, no matter where a user is, s/he could use these online transaction applications to buy many products and online services. However, when an online transaction is started, the messages used to ensure validity of transaction often contain user's private identity information that is revealed to merchants. Considering the unreliability and greediness of merchants, they may sell goods that user doesn't need or just sell user's identity to third parties for commercial profit. On the other hand, a merchant must possess the ability to verify the legality and validity of a transaction message, so s/he could assure the goods or services are provided to correct user. In addition, the verification on transaction message can prevent user make the allegation that s/he didn't buy the goods or services. To meet these security requirements, many protocols for mobile payments are proposed based on cryptographic primitives. Their protocols achieve the most important secure requirements mentioned above, i.e., user anonymity and unforgeability. When a protocol provides user anonymity, any merchants and adversaries can't link a transaction message to a user's identity. And unforgeability means the source of a message can be detected and any parties can't forge other parities' transaction message without being detected.

In addition to security requirements, efficiency also needs to be concerned in a mobile payment protocol. The rapid development of Internet of Things (IoT) [6]–[8] inevitably changes peoples lifestyle, so online payments via a variety of smart devices need to be considered. For example, smart meters could pay for electricity automatically, smart headphones could pay for digital music online when needed, and so on. All these devices including widely used smartphones face a common problem that their computation power and storage space are limited. Thus, when a payment protocol implements, the involved computation and necessary storage space should be low for the resource-constrained devices. However, in traditional transaction protocols, a public key infrastructure (PKI) is introduced to issue certificates for public key of the user. Particularly, the validity of the public key can be verified based on the certificates issued by a certificate authority. It is easy to see that PKI caused a lot of communication and storage costs when the revocation, storage, and distribution of certificates are done. So there exists a contradiction between PKI and mobile smart devices which only have limited computation power and storage space especially in the context of IoT. So how to design a mobile transaction protocol that not only suffer from certificates for public key, but also consumes few resources including computation power, communication traffic and storage space is still a challenge.

A. CONTRIBUTIONS

To solve the above challenge, we propose a new mobile payment scheme that achieves anonymity, unforgeability and low resource consumption simultaneously. In a nutshell, the major contributions of this paper are three-fold: (1) We propose the first unidirectional certificateless proxy re-signature scheme. It is of independent interest; (2) A mobile payment protocol with user anonymity is presented based on our proposed scheme. In particular, Pay Platform is introduced as a trusted proxy on behalf of users to interact with Merchant Server securely. Therefore, it is more secure for users because they need not send or receive messages to merchants directly. And resource consumption on the user side is reduced because the main computation is performed on Pay Platform. In addition, certificateless public key encryption technique and proxy re-signature scheme are introduced to achieve anonymity, and signature for every transaction information is used to achieve unforgeability. Moreover, the computation, communication and storage space costs are acceptable for resource limited mobile smart devices in the context of IoT; (3) By considering that the Pay Platform and Merchant Server needs to perform computation for each transaction, the overhead for millions of users at the Pay Platform and Merchant Server should be drastically reduced to address the scalability issue. It is easy to observe the signature verification dominate computation time at the Pay Platform and Merchant Server side. Inspired by [14] and [15], the idea of batch-verification have been utilized to accelerate the signature verification such that multiple signatures from different users (signers) on distinct messages can be verified quickly. Moreover, the signatures from the same user can be further batched to achieve higher efficiency. (4) We implement the presented protocol and compare it with other existing mobile payment protocol. The result of comparison shows our protocol is feasible and efficient in the context of IoT.

B. RELATED WORKS

1) CERTIFICATELESS PROXY RE-SIGNATURES

Digital signature [16], which enables authenticating messages or documents in a manner that repudiation is disallowed, has been widely utilized to secure software distribution, e-commerce, e-government, and a host of other scenarios. Proxy re-signature, which was initialized in 1998 by Blaze *et al.* as the extension of standard digital signature [17]–[19], enables a semi-trusted proxy to convert a signature from delegatee into a signature from delegator on the same message by employing the re-signing key. However, the proxy is not able to sign any message on behalf of either delegator or delegatee. Featured with conversion property, proxy re-signature has been applied in plenty of applications including certificate management simplification and group signature formation. In the conventional proxy re-signature scheme, the public keys of the delegator and delegatee need to be certified by the digital certificate (a digital signature issued by a trusted third party in essence) prior to the verification of signature itself. To mitigate the heavy costs incurred by the digital certificates, identity-based proxy re-signature [20] has been introduced such that the public key of the signer can be effortlessly calculated from his/her publicly known identity. Nevertheless, one notorious and inherent disadvantage of identity-based proxy re-signature is called "key escrow" [21] where the private key of the delegatee/delegator is generated by a fully trusted private key generator. To solve both the certificates management and key escrow problem, proxy re-signature has naturally been studied in the certificateless cryptography [22], [34], [35], which is usually considered as an intermediate between traditional [23] and identity-based [24] public key cryptography. The only know certificateless proxy re-signature [25] is bi-directional such that the proxy can perform the conversion in two-directions. According to [18]–[20], an array of practical applications motivate the construction of proxy re-signature with unidirectional property. As far as we know, the construction of certificateless *unidirectional* proxy re-signature still remains open.

2) MOBILE PAYMENT PROTOCOLS

With the popularization of smartphones, research about secure mobile payments [26] gets wide-spread attention. In 2010, Kamijo *et al.* [27] proposed a SMS-based face-to-face mobile payment protocol which supports anonymity, security, and usability simultaneously. In their protocol, unique information, such as the location and the time, for the payment transaction is used to ensure the security of the transaction, but their protocol only works well for face-to-face payment. Then Sureshkumar *et al.* [28] proposed an efficient mobile transaction protocol that achieves remote payment. They adopt symmetric key operations as well as hash functions to realize untraceability, unlinkability and atomicity, and use two gateways to enhance the efficiency of the whole system. However, Sureshkumar *et al.'s* protocol cannot provide non-repudiation, which is very important in remote payment. Afterwards, Yang and Lin [29] presented a new mobile payment protocol that provides the unforgeability and anonymity. Although the costs for transaction in their protocol are small, the costs for certificates which are used to guarantee the validity and legality of the public keys are very high for the resource-limited devices in the context of IoT.

By considering the great benefits brought from the cloud computing [11], [12], Qin *et al.* [13] and Yeh [9] proposed secure mobile payment protocols based on certificateless cryptographic primitives respectively. The protocol proposed by Qin *et al.* provides anonymity, unforgeability and certificate-free property. Liao *et al.* [30] found that the verification of Qin *et al.*'s protocol [13] is insecure that users could collude with the untrusted cloud server to cheat Merchant Server. Then they improved Qin *et al.*'s protocol to realize secure verification. However, both Qin *et al.*'s and Yang *et al.*' protocols will produce multiple pseudo identities to hide the real user identity, so a lot of storage spaces are consumed on the resource-limited users. Most recently, Yeh [9] proposed a transaction protocol based on certificateless cryptographic primitives. In Yeh's protocol, an efficient certificateless signature which does not need any certificate to ensure the legality of public key and private key pairs is adopted to achieve secure transaction. In a nutshell, Yeh's protocol has made great progress in the mobile payment protocol that we can complete the transaction protocol at anytime and anywhere efficiently in smartphones.

II. PRELIMINARIES

A. BILINEAR MAPS

Here we use G_1 and G_2 to denote two cyclic additive groups with order q. And P is a generator of G_1. If $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map, it should satisfy the following conditions:

1) Bilinearity, that is, for $\forall x, y \in Z_q$, the equation $e(xP, yP) = e(P, P)^{xy}$ should be hold;
2) Non-degeneracy, that is, $e(P, P) \neq 1$.

B. FRAMEWORK OF CERTIFICATELESS PROXY RE-SIGNATURE

The unidirectional certificateless proxy re-signature scheme consists of the following eight algorithms:

- Setup: On input the security parameter k, the algorithm generates the master secret key msk, the master public key PK_{pub} and the system parameters $params$.
- Partial-Private-Key-Extract: On input the system parameters $params$ and an identity ID of the user, the algorithm generates the user's partial private key D_{ID}.
- Set-Secret-Value: On input the system parameters $params$ and an identity ID of the user, the algorithm generates the user's secret value x_{ID}.
- Set-Public-Key: On input the system parameters $params$ and the user's secret value x_{ID}, the algorithm generates the user's public key P_{ID}.
- ReKey: On input the system parameters $params$, the delegatee's identity ID_i and public key P_i, as well as the delegator's secret key (D_j, x_j) associated with the identity ID_j and public key P_j, the algorithm generates the re-signature key $rk_{i,j}$.
- Sign: On input the system parameters $params$, a message m the user's secret key (D_{ID}, x_{ID}) associated with the identity ID and public key P_{ID}, the algorithm generates two kinds of signatures σ on message m.
- ReSign: On input the re-signature key $rk_{i,j}$, the delegatee's public key P_i and a signature σ_i on message m with the identity ID_i, the algorithm generates the re-signature σ_j on message m with the identity ID_j.
- Verify: On input the system parameters $params$ and the user's public key P_{ID}, the algorithm checks the validity of signature σ on message m under the identity ID. If σ is valid, the algorithm outputs 1; $\perp$, otherwise.

C. SYSTEM MODEL OF OUR TRANSACTION PROTOCOL

The considered system consists of four types of entities: the trusted system authority (TSA), the user app, the merchant server, and the Pay Platform [9].

- **Trusted System Authority**: TSA is a trusted third party organization that provides registration services for User's App and Pay Platform. At the same time, TSA also distributes system params and partial private keys for registered users to ensure the whole scheme successfully works.
- **User's App**: Any software that requires a payment function is called User's App, such as, Ali pay [3], Apple pay [4], WeChat pay [5] and so on. This application needs to be registered with the TSA to obtain the corresponding system params and partial private key. Besides, it also generates its own user secret value and public key. Then User's App completes the signature using its full private key, which consists of partial private key.
- **Pay Platform**: Pay Platform is an application offered by a trusted party, of course, it also needs to register

with the TSA to obtain system params and private key. Simultaneously, in order to protect the user's information of the transaction, Pay Platform will provide re-sign service, that is, the Pay Platform transforms signature of User's App into signature of Pay Platform.

- **Merchant Server**: Merchant Server is the entity which provides the goods or services, verifies the correctness of the transaction information to ensure the goods or services are provided to the corresponding user.

D. OBJECTIVES OF OUR TRANSACTION PROTOCOL

To resist the potential threats in the process of transaction, a secure transaction should should meet the following requirements. (1) User Anonymity: The real identities of users cannot be revealed by anyone except Pay Platform. (2) Unforgeability: All the transaction information cannot be forged by anyone, that is, every receiver can verify the correctness of the received information.

III. BUILDING BLOCKS OF OUR TRANSACTION SCHEME

A. OUR UNIDIRECTIONAL CL-PRS SCHEME

1) *Setup*: With a security data k and a prime number q, KGC generates two group G_1 and G_2 with order q, and then chooses a generator P of G_1 as well as a bilinear pairing $e : G_1 \times G_1 \rightarrow G_2$. Next, KGC selects a secret key $s \in Z_q^*$ and calculates the public key $PK_{pub} = s \cdot P$. After that, KGC chooses three secure hash function H_1: $\{0, 1\}^* \times G_1 \rightarrow Z_q^*$, H_2: $\{0, 1\}^* \times G_1^3 \rightarrow Z_q^*$, H_3: $\{0, 1\}^* \rightarrow G_1$. Eventually KGC publishes $\{G_1, G_2, e, q, P, PK_{pub}, H_1, H_2, H_3\}$ and preserves s secretly.
2) *Partial-Private-Key-Extract*: With params, s and user u_i's identity ID_i, KGC selects a random number $r_i \in Z_q^*$, and computes $R_i = r_i \cdot P$, $h_i = H_1(ID_i, R_i)$, $s_i = r_i + h_i \cdot s \bmod q$. After that, KGC sends the partial private key $D_i = (s_i, R_i)$ to u_i and u_i verifies D_i by checking whether $s_i \cdot P = R_i + h_i \cdot PK_{pub}$.
3) *Set-Secret-Value*: u_i selects a random number $x_i \in Z_q^*$ as his/her secret value.
4) *Set-Public-Key*: With params and x_i, u_i calculates $P_i = x_i \cdot P$ and sets P_i as his/her public key.
5) *Re-Key*: With the delegatee's identity ID_i and public key P_i, as well as the delegator's secret key (D_j, x_j) associated with identity ID_j and public key P_j, the delegator computes $rk_{i,j}^1 = (k_jx_j + s_j)^{-1} \cdot (R_i + h_i \cdot PK_{pub} + k_iP_i)$, $rk_{i,j}^2 = R_j$, where $k_i = H_2(ID_i, P_i, R_i, PK_{pub})$ and $k_j = H_2(ID_j, P_j, R_j, PK_{pub})$. Finally, this algorithm outputs $rk_{i,j} = (rk_{i,j}^1, rk_{i,j}^2)$ as re-signature key.[1]
6) *Sign*: With params, user secret key (D_i, x_i), user public key P_i, identity ID_i and message m, u_i is able to generate two kinds of signatures as follows:
   - Level 1: $\sigma_i = (\sigma_{i1}, \sigma_{i2}) = ((k_ix_i + s_i)H_3(m), R_i)$, where $k_i = H_2(ID_i, P_i, R_i, PK_{pub})$.
   - Level 2: $\sigma_i = (\sigma_{i1}, \sigma_{i2}, \sigma_{i3}, \sigma_{i4}) = (t_i(k_ix_i + s_i)H_3(m), t_i(R_i + h_iPK_{pub} + k_iP_i), t_iP, R_i)$, where $h_i = H_1(ID_i, R_i)$, $k_i = H_2(ID_i, P_i, R_i, PK_{pub})$ and t_i is randomly chosen from Z_q^*.
7) *Re-Sign*: With a level 1 signature $\sigma_i = (\sigma_{i1}, \sigma_{i2})$ on message m under the identity ID_i and user public key P_i, a re-signature key $rk_{i,j}$, this algorithm is able to transform the signature σ_i into a level 2 signature σ_j on the same message m under the identity ID_j and user public key (P_j, R_j) as follows.
   - Checks whether $e(P, \sigma_{i1}) = e(H_3(m), \sigma_{i2} + h_iPK_{pub} + k_iP_i)$ holds or not, if this equation holds, performs the following steps; otherwise, outputs failure.
   - Outputs $\sigma_j = (\sigma_{j1}, \sigma_{j2}, \sigma_{j3}, \sigma_{j4}) = (t_i \cdot \sigma_{i1}, t_i \cdot (\sigma_{i2} + h_iPK_{pub} + k_iP_i), t_i \cdot rk_{i,j}^1, rk_{i,j}^2) = (t_i(k_ix_i + s_i)H_3(m), t_i(R_i + h_iPK_{pub} + k_iP_i), t_i(k_jx_j + s_j)^{-1}(R_i + h_i \cdot PK_{pub} + k_iP_i), R_j) = (t_j \cdot (k_jx_j + s_j)H_3(m), t_j(R_j + h_jPK_{pub} + k_jP_j), t_jP, R_j)$, where t_i is randomly chosen from Z_q^* and $t_j = t_i \cdot (k_ix_i + s_i)/(k_jx_j + s_j)$.

   It is easy to see σ_j is a valid signature at level 2 on message m under the identity ID_j and user public key P_j.
8) *Verify*: With params, a signature σ_i on message m under identity ID_i and user public key P_i, this algorithm is performed to check the validity of signature:
   - Level 1: If $e(P, \sigma_{i1}) = e(H_3(m), \sigma_{i2} + h_iPK_{pub} + k_iP_i)$ holds, accept this signature; otherwise, outputs failure.
   - Level 2: If $e(P, \sigma_{i1}) = e(H_3(m), \sigma_{i2} + h_iPK_{pub} + k_iP_i)$ and $e(P, \sigma_{i2}) = e(\sigma_{i3}, \sigma_{i4} + h_iPK_{pub} + k_iP_i)$ hold, accept this signature; otherwise, outputs failure.

B. SECURITY ANALYSIS

Lemma 1: With a Type I adversary $\mathcal{A}_1$ breaking the security of the proposed CL-PRS scheme in the **EUF-CL-PRS-CMA-I** game, an algorithm is able to be constructed to solve the extCDH problem efficiently.

Lemma 2: With a Type II adversary $\mathcal{A}_2$ breaking the security of the proposed CL-PRS scheme in the **EUF-CL-PRS-CMA-II** game, an algorithm is able to be constructed to solve the extCDH problem efficiently.

IV. OUR LIGHTWEIGHT AND ANONYMOUS TRANSACTION SCHEME

Similar to [9], four entities including the KGC, the Merchant Server, the Android Pay platform and the user with Android App are involved in our transaction scheme, which consists of system initialization and transaction process phases.

1) *System Initialization.* A trusted system authority (shorten as TSA) provided by the Google Play Services is responsible to initialize the system and serve as

[1]It is worth noting that R_i used in the generation of re-signature key is included in every signature on behalf of delegatee and can be obtained by the delegator from any signature in the name of delegatee. That is to say, the delegatee does not need to be involved in the *Re-Key* algorithm, which makes this algorithm non-interactive.

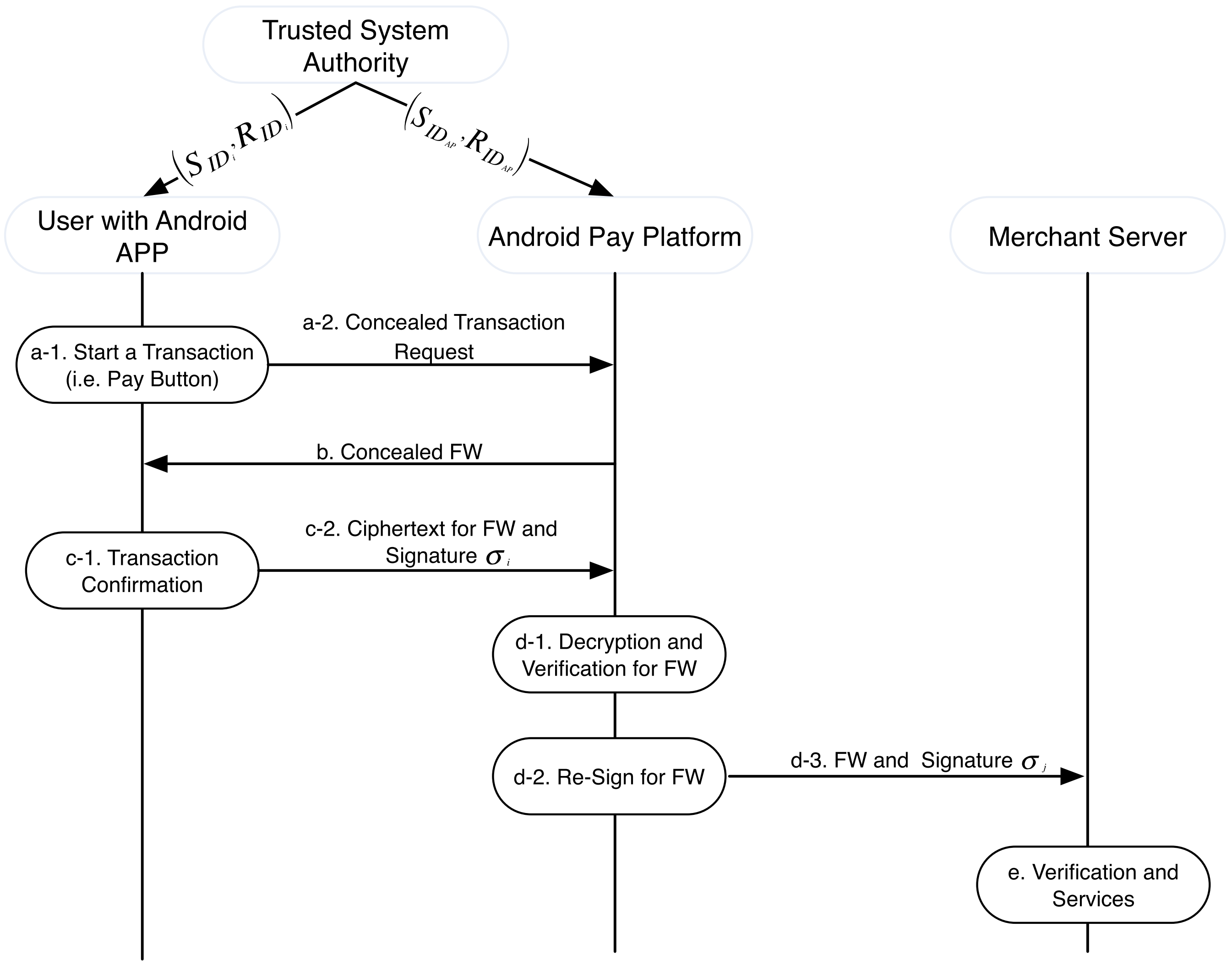
Trusted System Authority
(S_{ID_i}, R_{ID_i})
$(S_{ID_{AP}}, R_{ID_{AP}})$
User with Android APP
Android Pay Platform
Merchant Server
a-1. Start a Transaction (i.e. Pay Button)
a-2. Concealed Transaction Request
b. Concealed FW
c-1. Transaction Confirmation
c-2. Ciphertext for FW and Signature σ_i
d-1. Decryption and Verification for FW
d-2. Re-Sign for FW
d-3. FW and Signature σ_j
e. Verification and Services

FIGURE 1. **Proposed transaction scheme for Android Pay platform.**

KGC for all entities in the system. With a security parameter k and a prime number q, TSA generates two group G_1 and G_2 with order q, and then chooses a generator P of G_1 as well as a bilinear pairing $e : G_1 \times G_1 \rightarrow G_2$. Next, TSA selects a secret key $s \in Z_q^*$ and calculates the public key $PK_{pub} = s \cdot P$. After that, TSA chooses three secure hash function H_1: $\{0, 1\}^* \times G_1 \rightarrow Z_q^*$, H_2: $\{0, 1\}^* \times G_1^3 \rightarrow Z_q^*$, H_3: $\{0, 1\}^* \rightarrow G_1$ and publishes $\{G_1, G_2, e, q, P, PK_{pub}, H_1, H_2, H_3\}$ and preserves s secretly.

Before performing the mobile payment, the user with identity ID_i needs to register with TSA using his/her Android app as follows.

- The user with identity ID_i selects a random number $x_{ID_i} \in Z_q^*$ as his/her user secret value and calculates $P_{ID_i} = x_{ID_i} \cdot P$ as his/her corresponding user public key.
- TSA selects a random number $r_{ID_i} \in Z_q^*$, and computes $R_{ID_i} = r_{ID_i} \cdot P$, $h_{ID_i} = H_1(ID_{ID_i}, R_{ID_i})$, $s_{ID_i} = r_{ID_i} + h_{ID_i} \cdot s \bmod q$. After that, TSA sends the partial private key $D_{ID_i} = (s_{ID_i}, R_{ID_i})$ to ID_i and ID_i verifies D_{ID_i} by checking whether $s_{ID_i} \cdot P = R_{ID_i} + h_{ID_i} \cdot PK_{pub}$.

Similarly, the Android Pay Platform with identity ID_{AP} needs to register with TSA as follows before providing the payment services for the requested users.

- The Android Pay Platform with identity ID_{AP} selects a random number $x_{ID_{AP}} \in Z_n^*$ as his/her user secret value and calculates $P_{ID_{AP}} = x_{ID_{AP}} \cdot P$ as his/her corresponding user public key.
- TSA selects a random number $r_{ID_{AP}} \in Z_q^*$, and computes $R_{ID_{AP}} = r_{ID_{AP}} \cdot P$, $h_{ID_{AP}} = H_1(ID_{ID_{AP}}, R_{ID_{AP}})$, $s_{ID_{AP}} = r_{ID_{AP}} + h_{ID_{AP}} \cdot s \bmod q$. After that, TSA sends the partial private key $D_{ID_{AP}} = (s_{ID_{AP}}, R_{ID_{AP}})$ to ID_{AP} and ID_{AP} verifies $D_{ID_{AP}}$ by checking whether $s_{ID_{AP}} \cdot P = R_{ID_{AP}} + h_{ID_{AP}} \cdot PK_{pub}$.

2) *Transaction Process*

a) When user ID_i starts a new transaction by clicking a purchase button in the app, the app sends a concealed request to Android Pay platform.

b) The Android Pay platform ID_{AP} sends a concealed *Full Wallet (FW)* containing an unique ID_T for this transaction, the buyer's shipping address, the buyer's email, and the cart items, back to ID_i with the app.

c) ID_i checks the correctness of concealed wallet object from the Android Pay platform. If the verification holds, the app presents a confirmation page containing all necessary information for this transaction. Note that ID_T will be revised if u_i alters the details of purchase. With params, ID_i, ID_{AP}, PK_i, (s_i, R_i), x_i and FW, the app calculates $\sigma_i = (\sigma_{i1}, \sigma_{i2}) = ((k_i x_i + s_i)H_3(FW), R_i)$, where $k_i = H_2(ID_i, P_i, R_i, PK_{pub})$. Then it chooses a random number $a_i \in z_p^*$ and computes $C_1 = rP$ as well as $C_2 = H(r(P_{ID_{AP}} + R_{ID_{AP}} + H_1(ID_{AP}, R_{ID_{AP}})PK_{pub})) \bigoplus (FW||\sigma_i)$ where $r = H(ID_i, P_{ID_i}, R_{ID_i}, a_i P)$. Finally, ID_i sends C_1, C_2 to Android Pay platform.

d) After receiving (C_1, C_2), the Android Pay platform gets $FW||\sigma_i$ by computing $FW||\sigma_i = H((x_{ID_{AP}} + s_{ID_{AP}})C_1) \bigoplus C_2$, and verifies FW by checking whether $e(P, \sigma_{i1}) = e(H_3(FW), \sigma_{i2} + h_i PK_{pub} + k_i P_i)$ holds or not. If this equation holds, it computes $\sigma_j = (\sigma_{j1}, \sigma_{j2}, \sigma_{j3}, \sigma_{j4}) = (t_i \cdot \sigma_{i1}, t_i \cdot (\sigma_{i2} + h_i PK_{pub} + k_i P_i), t_i \cdot rk^1_{i,j}, rk^2_{i,j})$, where $rk^1_{i,j} = (k_j x_j + s_j)^{-1}(R_i + h_i \cdot PK_{pub} + k_i P_i)$, $rk^2_{i,j} = R_j$, t_i is randomly chosen from Z_q^* and $t_j = t_i \cdot (k_i x_i + s_i)/(k_j x_j + s_j)$. Then it sends message FW, σ_j securely to the merchant server.

e) After receiving (FW, σ_j), the merchant server checks the validity of it as follows: with params, ID_i, P_i, FW, σ_j, the server checks whether $e(P, \sigma_{j1}) = e(H_3(FW), \sigma_{j2} + h_j PK_{pub} + k_j P_j)$ and $e(P, \sigma_{j2}) = e(\sigma_{j3}, \sigma_{j4} + h_j PK_{pub} + k_j P_j)$.
After the verification phase ends, the merchant records the transaction information and provides corresponding services.

A. BATCH VERIFICATION

We introduce a batch verification [14] to reduce the computation overhead in the verification phase. Here, we take the verification phase on Pay Platform as an example, and it is similar for Merchant Server.

BatchVerify: When obtaining the signatures $\sigma_i = (\sigma_{i1}, \sigma_{i2})$ on distinct *Full Wallet* FW_i for $i = 1, \ldots, n$, Pay Platform first check each user public key P_i is valid. If so, Pay Platform randomly selects a vector $\theta = (\theta_1, \ldots, \theta_n)$, where $\theta_i \in Z_q$ is of ℓ bits. Then Pay Platform checks that $e(P, \sum_{i=1}^n \theta_i \sigma_{i1}) = \prod_{i=1}^n e(H_3(FW_i), \sigma_{i2} + h_i PK_{pub} + k_i P_i)^{\theta_i}$. If the result is correct, Pay Platform performs the remainder operations to complete the transaction; otherwise outputs 0 and terminates the transaction. Moreover, when receiving multiple signatures from a single user with identity ID_i, Pay Platform only needs to compute $e(P, \sum_{k=1}^n \theta_k \sigma_{k1}) = e(\sum_{k=1}^n \theta_k H_3(FW_k), \sigma_{i2} + h_i PK_{pub} + k_i P_i)$ which only needs two pairings.

Theorem 1: The above algorithm is a batch verification algorithm for the proposed scheme.

Proof: It is easy to observe that $Verify(\sigma_1, FW_1, P_1) = \cdots = Verify(\sigma_n, FW_n, P_n) = 1$ implies that *BatchVerify* $((\sigma_1, FW_1, P_1), \ldots, (\sigma_n, FW_n, P_n) = 1$. This is derived from the verification equation of the proposed scheme:

$$\prod_{i=1}^n e(P, \sigma_{i1})^{\theta_i} = \prod_{i=1}^n e(H_3(FW_i), \sigma_{i2} + h_i PK_{pub} + k_i P_i)^{\theta_i}$$
$$\Leftrightarrow e(P, \sum_{i=1}^n \theta_i \sigma_{i1}) = \prod_{i=1}^n e(H_3(FW_i), \sigma_{i2} + h_i PK_{pub} + k_i P_i)^{\theta_i}$$

The technique to prove the small exponents test in [15] is used to accomplish this proof as follows. We define $\sigma_{1i} = a_i P$, $H(FW) = b_i P$, $(\sigma_{i2} + h_i PK_{pub} + k_i P_i) = c_i P$ for some $a_i, b_i, c_i \in Z_q$. Now, the above equation can be written as

$$\prod_{i=1}^n e(P, \theta_i \sigma_{i1}) = \prod_{i=1}^n e(H_3(FW_i), \sigma_{i2} + h_i PK_{pub} + k_i P_i)^{\theta_i}$$
$$\Rightarrow e(P, P)^{\sum_{i=1}^n \theta_i a_i} = e(P, P)^{\sum_{i=1}^n \theta_i b_i c_i}$$
$$\Rightarrow \sum_{i=1}^n \theta_i a_i - \sum_{i=1}^n \theta_i b_i c_i \equiv 0 \pmod q.$$

After setting $\beta_i = a_i - b_i c_i$, it is equal to

$$\sum_{i=1}^n \theta_i \beta_i \equiv 0 \quad (mod\ q).$$

Suppose that $BatchVerify((\sigma_1, FW_1, P_1), \ldots, (\sigma_n, FW_n, P_n) = 1$, but the equation *BatchVerify* $(\sigma_i, FW_i, P_i) = 0$ holds for at least one i. Without loss of generality, suppose that it is true for $i = 1$, that is, $\beta_1 \neq 0$. And we can easily get an inverse ξ_1 of β_1 such that $\beta_1 \xi_1 \equiv 1 (\text{mod}\ q)$ since q is a prime. So we can get:

$$\theta_1 \equiv -\xi_1 \sum_{i=2}^n \theta_i \beta_i \quad (mod\ q).$$

Given (σ_i, FW_i, P_i) for $i = 1, \ldots, n$, Ev is an event that $BatchVerify(\sigma_1, FW_1, P_1) = 0$ holds but $BatchVerify((\sigma_1, FW_1, P_1), \ldots, (\sigma_n, FW_n, P_n)$ also equals 1, namely, the batch verification is broken. We define $\Gamma = (\theta_1, \ldots, \theta_n)$ and denote the last $n-1$ values of Γ as $\Gamma' = \theta_2, \ldots, \theta_n$ with the number as $|\Gamma|$. The above Equation can be comprehended that given a fixed vector Γ', only one unique value of θ_1 will make Ev happen. That is to say, the probability of Ev is $Pr[Ev|\Gamma'] = 2^{-\ell}$ with a randomly picked θ_1. So if θ_1 is selected at random and all possible choices of Γ' are considered, the probability that event Ev appears is $Pr[Ev] \leq \sum_{i=1}^{2^{\ell(n-1)}} (2^{-\ell} \cdot 2^{-\ell(n-1)}) = 2^{-\ell}$.

TABLE 1. Function comparison of different protocols.

Protocol	Anonymity	Unforgeability	Certificateless	Storage overhead
[9]	✓	✓	✓	low
[13]	✓	×	✓	high
Our Protocol	✓	✓	✓	low

TABLE 2. Computation efficiency comparison of different protocols.

Protocol	User's App	Pay Platform	Merchant Server	Pay Platform with batch verification	Merchant Server with batch verification
[9]	$5T'_{G_1}$	nT_{G_1}	$4mT_{G_1}$	–	–
[13]	$2T'_{G_1}+T'_{G_2}$	–	$2mT_{G_1}+mT_{G_2}$	–	–
Our protocol	$4T'_{G_1}$	$2nP+7nT_{G_1}$	$4mP+2mT_{G_1}$	$(n+1)p$ $+3nT_{G_1}+nT_{G_2}$	$(m+1)p$ $+mT_{G_1}+mT_{G_2}$

‡ P represents the time cost of a pairing computation, T_{G_1} represents the time cost of an exponentiation operation in G_1 and T_{G_2} represents the time cost of an exponentiation operation in G_2 with the particular hardware environment of Pay Platform and Merchant Server. P',T'_{G_1},T'_{G_2} represent the time cost of the corresponding operation on the particular hardware environment of User's App. n represents the number of transactions received by Pay Platform. m represents the number of transactions received by Merchant Server.

B. SECURITY STRENGTH OF OUR TRANSACTION PROTOCOL

Theorem 2 (User Anonymity): Anonymity in our protocol means that except for the user and the Pay Platform, any outsider (including the TSA) is unable to link a transaction message to a particular identity.

Proof: In our transaction protocol, the anonymity for user is achieved by the certificateless encryption and proxy re-signature. On the one hand, user ID_i sends the encrypted signature and transaction information ($C1$, $C2$) to Pay Platform ID_{AP}. Then Pay Platform uses its full private key $(x_{ID_{AP}}, s_{ID_{AP}}, R_{ID_{AP}})$ to decrypt the encrypted message and use user's ID_i as well as user's public key P_i to verify the correctness of the transaction. So, anyone other than ID_i and Pay Platform can not know the identity of ID_i at this stage.

On the other hand, after verifying the transaction information, Pay Platform use its full private key $(x_{ID_{AP}}, s_{ID_{AP}}, R_{ID_{AP}})$ to re-sign the signature σ_i of ID_i to σ_j. Then it sends σ_j and corresponding transaction information to Merchant Server. Finally, Merchant Server verifies the received message under the Pay Platform's public key and identity. So, the message cannot be linked to identity of ID_i by the Merchant Server in this stage. In general, the user identity associated to transaction will not be revealed in the whole process.

Theorem 3 (Unforgeability): No PPT adversary can forge transaction information without being detected.

Proof: It should be noted that the signature and re-signature in our transaction protocol are proved secure in Lemma 1 and Lemma 2. So here we assume unforgeability under signature and re-signature is guaranteed. In our transaction protocol, TSA only generate partial private key for users and Pay Platform so that TSA cannot forge a correct signature without full private key. Although Pay Platform can get user's signature and transaction information, it dosen't possess user's full private key. And Merchant Server only receives re-signed transaction from Pay Platform, so it cannot know the user's identity as well as full private key. In conclusion, no adversary including TSA, Pay Platform and Merchant Server can forge transaction information with non-negligible advantage.

C. COMPARISON WITH PREVIOUS PROTOCOLS

In this section, a performance evaluation is shown by comparing our proposed protocol with some other protocols in the aspect of function and computation efficiency. The comparison result of the function is shown in Table 1, which includes: Anonymity, Undeniable, Certificateless and Storage overhead (the storage space required by the user' App). As for the comparison of computation efficiency, our experiment was performed on a computer equipped with an Intel i3-380M processor running at 2.53GHz and 8GB memory, and the simulation platform of the User's App client is set as an Intel PXA270 624-MHz processor and 1GB memory. PXA270 is a very powerful embedded processor and has a very rich expansion interface ,simultaneously, its energy consumption is also very low. So we chose it as the processor of User's App client for our experiment. For the overall security of our protocol, our hash function will use SHA-3. We implemented our protocol in VC++ 6.0 with PBC library [38], and set the size of G_1 and Z_q to 64B (512bits), as well as, we set the size of G_2 to 128B (1024bits). To offer the security with the equivalent level and achieve the comparable level of security to 2048 bits RSA in our scheme, we used the elliptic curve $y^2 = x^3 + x$ defined on $\mathbb{F}_{q^2}$ providing ECC group. With the above setting, we can get the result shown in table 3. We divide those protocols which we proposed and previous

TABLE 3. Performance of algorithm (Millisecond).

P	P'	T_{G_1}	T'_{G_1}	T_{G_2}	T'_{G_2}
96.2	587.7	53.85	312.5	30.6	189.29

TABLE 4. **Storage overhead of different protocols.**

Protocol	User's signature	Pay Platform signature	Merchant signature	Ciphertext	Storage overhead
[9]	$2\|G_1\| + Z_q$	$2\|G_1\| + \|Z_q\|$	$-$	$3\|G_1\| + \|FW\|$	$\|G_1\| + 2\|Z_q\|$
[13]	$\|G_1\|$	$-$	$\|G_1\|$	$-$	$3n\|G_1\| + \|Z_q\|$
Our Protocol	$2\|G_1\|$	$4\|G_1\|$	$-$	$-$	$\|G_1\| + 2\|Z_q\|$

‡ $|G_1|$ denote the length of an element in G_1, $|Z_q|$ denote the length of an element in Z_q and $|FW|$ denote the length of specific transaction information.

protocol into three parts: User's App computation efficiency, Pay Platform computation efficiency and Merchant Server computation efficiency. We calculate their respective computation efficiency. The computation efficiency result is shown in table 2, and in table 4 we give the storage overhead of different protocols.

From Fig. 2, we can note that the computation cost of User's App phase in our protocol is nearly to other protocols. It means that our protocol is also suitable for using in lightweight devices. Our extensive computation cost is in the Pay Platform phase and it is easy to observe the signature verification dominate computation time at the Pay Platform and Merchant Server side. We use batch verification [14], [15] in our protocol to accelerate the signature verification. Theoretic analysis and experiment evaluation demonstrate that our batch verification improves the computational efficiency at the Pay Platform and Merchant Server significantly. The result is shown in Fig.3 and Fig. 4. From Fig. 5, we can see our protocol consumes very little on the terminal storage space compared with the protocol in [13] and the storage cost can't be ignored for lightweight devices. Overall, the protocol in [13] consumes a lot of storage space and fails to achieve the desirable security properties. Different from existing work, our proposed protocol for the first time achieve a trade-off between the security and performance requirements. Taking account of the untrusted network and limited computing power of smart device in the context of IoT, our protocol is more suitable than existing transaction protocols.

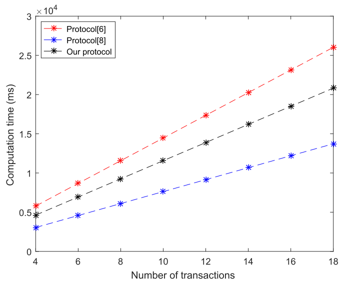

FIGURE 2. **Computational of User's App.**

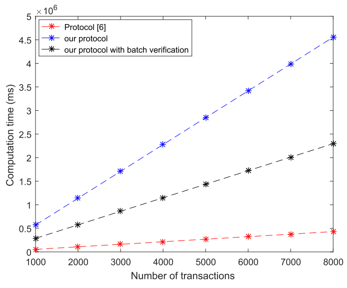

FIGURE 3. **Computation of payment platform.**

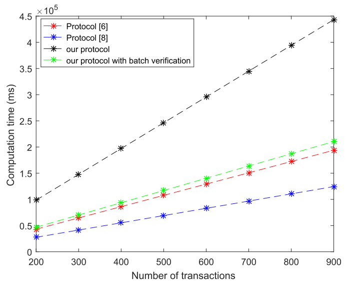

FIGURE 4. **Computation of Merchant sever.**

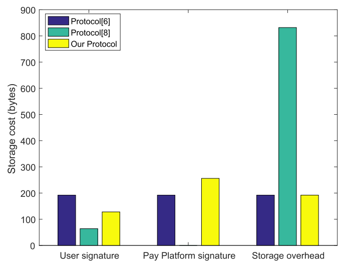

FIGURE 5. **Storage overhead of different protocols.**

V. SUMMARY

In this paper, we have presented a lightweight and anonymous authentication protocol for mobile payment by using a new certificateless *unidirectional* signature scheme. To the best of our knowledge, this is the first transaction protocol that achieves user anonymity, unforgeability, certificateless and low resource cost on resource-limited smart device. Furthermore, the newly proposed certificateless *unidirectional* signature scheme, which is proven secure under the extended CDH assumption by using random oracle model, is also of independent interest. According to the results of our experiments, we can observe that our mobile payment transaction is very efficient and highly practical.

APPENDIX A
ADVERSARY MODEL FOR CLS

The security of a CLS scheme is modeled via the following two games between a challenger and an adversary $\mathcal{A}_1$ or $\mathcal{A}_2$.

Game 1: Game 1 is a secure game between a challenger $\mathcal{C}$ and a Type I adversary $\mathcal{A}_1$ interacting within a CLS game.

1) Initialization: $\mathcal{C}$ executes *Setup* algorithm to get a secret key s and the public key PK_{pub}. Then $\mathcal{C}$ sends PK_{pub} to $\mathcal{A}_1$ and retains s.
2) Query: $\mathcal{A}_1$ adaptively performs *Hash, Request-Public-Key, Extract-Partial-Secret, Extract-Secret, Replace-Public-Key and Sign* queries.
3) Output: $\mathcal{A}_1$ outputs a forged signature σ_i. $\mathcal{A}_1$ wins this game if 1) $\mathcal{A}_1$ has never carried out *Extract-Partial-Secret*, 2) $\mathcal{A}_1$ has never carried out *Sign*, 3) *Verify* algorithm outputs true when it takes the current public key of party ID_i, and this public key may be replaced by $\mathcal{A}_1$.

Game 2: Game 2 is a secure game between a challenger $\mathcal{C}$ and a Type II adversary $\mathcal{A}_2$ interacting within a CLS game.

1) Initialization and Query: These two phases are the same with Initialization and Query respectively in Game 1 except that $\mathcal{A}_1$ has been changed to $\mathcal{A}_2$.
2) Output: $\mathcal{A}_2$ outputs a forged signature σ_i. $\mathcal{A}_2$ wins this game if 1) $\mathcal{A}_2$ has never carried out *Extract-Secret*, 2) $\mathcal{A}_2$ has never carried out *Sign*, 3) *Verify* algorithm outputs true when it takes the origianl public key of party ID_i.

APPENDIX B
SECURITY MODEL OF CERTIFICATELESS PROXY RE-SIGNATURE

The security of a CL-PRS scheme is modeled via the following two games between a challenger and an adversary $\mathcal{A}_1$ or $\mathcal{A}_2$.

Game 1:

1. *Initial*. The challenger runs the Setup algorithm and returns the system parameters *params* and the master public key PK_{pub}. The system parameters *params* is given to $\mathcal{A}_1$, but keep the master public key PK_{pub} secret.

2. *Attack*. In this phase, $\mathcal{A}_1$ can adaptively access all the oracles which are defined as follows:

- Public-Key Queries: $\mathcal{A}_1$ can request a user's public key with the identity ID_i. In response, the challenger outputs the public key P_i with identity ID_i.
- Partial-Private-Key-Extract Queries: $\mathcal{A}_1$ can request a user's partial private key with the identity ID_i. In response, the challenger outputs the partial private key D_i of this user.
- Public-Key-Replace Queries: For any user with the identity ID_i, $\mathcal{A}_1$ can select a new public key P'_i as the new public key of this user. The challenger records this replacement.
- Secret-Value-Extract Queries: $\mathcal{A}_1$ can request a user's secret value with the identity ID_i. In response, the challenger outputs the secret value x_i of this user.
- Re-Sign, Re-Key and Sign Queries: The challenger first queries Secret-Value and Partial-Private-Key-Extract oracles to obtain the partial private key and the secret key and then utilizes both key to answer these queries.

3. *Forgery*. Finally, $\mathcal{A}_1$ outputs a valid forged signature σ^* on message m^* under identity ID^* and the corresponding public key P_{ID^*}. We say that $\mathcal{A}_1$ wins Game 1, if

1) $\mathcal{A}_1$ has never requested the Partial-Private-Key-Extract of the user with the identity ID^*.
2) $\mathcal{A}_1$ has never requested the Sign Queries of $(\sigma^*, m^*, ID^*, P_{ID^*})$.

Game 2:

1. *Initial*. The challenger runs the Setup algorithm and returns the system parameters *params* and the master public key PK_{pub}. The system parameters *params* and the master public key PK_{pub} are given to $\mathcal{A}_2$.

2. *Attack*. In this phase, $\mathcal{A}_2$ can adaptively access all the oracles which are defined as follows:

- Public-Key Queries: $\mathcal{A}_2$ can request a user's public key with the identity $ID_i \neq ID^*$. In response, the challenger outputs the public key P_i of this user.
- Partial-Private-Key-Extract Queries: $\mathcal{A}_2$ can request a user's partial private key with the identity ID_i. In response, the challenger outputs the partial private key D_i of this user.
- Public-Key-Replace Queries: For any user with the identity ID_i, $\mathcal{A}_2$ can select a new public key P'_i as the new public key of this user. The challenger records this replacement.
- Secret-Value-Extract Queries: $\mathcal{A}_2$ can request a user's secret value with the identity $ID_i \neq ID^*$. In response, the challenger outputs the secret value x_i of this user.
- Re-Sign, Re-Key and Sign Queries: The challenger first queries Secret-Value and Partial-Private-Key-Extract oracles to obtain the partial private key and the secret key and then utilizes both key to answer these queries.

3. *Forgery*. $\mathcal{A}_2$ outputs a valid forged signature σ^* on message m^* under identity ID^* and the corresponding public key P_{ID^*}. We say that $\mathcal{A}_2$ wins Game 2, if

1) $\mathcal{A}_2$ has never requested the Secret-Value of the user with the identity ID^*.
2) $\mathcal{A}_2$ has never requested the Public-Key-Replace of the user with the identity ID^*.
3) $\mathcal{A}_2$ has never requested the Sign Queries of $(\sigma^*, m^*, ID^*, P_{ID^*})$.

APPENDIX C
PROOF OF LEMMA 1

Proof: If an adversary $\mathcal{A}_1$ breaking the security of the proposed unidirectional CL-PRS scheme in the **EUF-CL-PRS-CMA-I** game is given, a challenger can be built to solve the CDH problem. With the extCDH instance (aP, bP) as input, the aim of the challenger is to output (Q, abQ), where a, b are randomly chosen from $\mathbb{Z}_q^*$ and Q is chosen from G_1 randomly. To make the security proof reader-friendly, a brief description for this process is presented in Fig. 6.

1) *Initial*. The challenger assigns $PK_{pub} = aP$ and publishes the public parameters $\{G_1, G_2, e, q, P, PK_{pub}, H_1, H_2, H_3\}$ to $\mathcal{A}_1$.

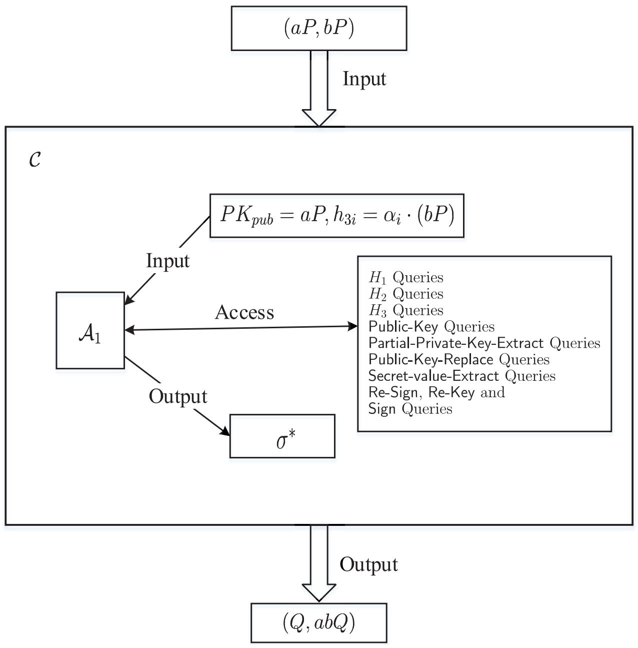

FIGURE 6. Brief security Proof of Lemma 1.

2) *Attack.* In this phase, the challenger maintains four initially-empty lists $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{PK}$ and answers the adaptive queries issued by $\mathcal{A}_1$ as follows.
   - H_1 Queries: Once obtaining the query (ID_i, R_i) from $\mathcal{A}_1$, the challenger searches this item in the list $\mathcal{L}_1$. If this item is found, the challenger responds h_{1i} as the answer; otherwise, the challenger chooses h_{1i} from $\mathbb{Z}_q^*$ at random and inserts $< (ID_i, R_i), h_{1i} >$ in $\mathcal{L}_1$. Finally, the challenger responds h_{1i} as the answer.
   - H_2 Queries: Once obtaining the query (ID_i, P_i, R_i) from $\mathcal{A}_1$, the challenger searches this item in the list $\mathcal{L}_2$. If this item is found, the challenger responds h_{2i} as the answer; otherwise, the challenger chooses h_{2i} from $\mathbb{Z}_q^*$ at random and inserts $< (ID_i, P_i, R_i, PK_{pub}), h_{2i} >$ in $\mathcal{L}_2$. Finally, the challenger responds h_{2i} as the answer.
   - H_3 Queries: Once obtaining the query m_i from $\mathcal{A}_1$, the challenger searches this item in the list $\mathcal{L}_3$. If this item is found, the challenger responds h_{3i} as the answer; otherwise, the challenger chooses α_i from $\mathbb{Z}_q^*$ at random and calculates $h_{3i} = \alpha_i \cdot (bP)$. Finally, the challenger inserts $< m_i, \alpha_i, h_{3i} >$ in $\mathcal{L}_3$ and responds h_{3i} as the answer.
   - Public-Key Queries: Once obtaining the query ID_i from $\mathcal{A}_1$, the challenger searches this item in the list $\mathcal{PK}$. If this item is found, the challenger responds P_i as the answer; otherwise, the challenger chooses x_i from $\mathbb{Z}_q^*$ at random and inserts $< ID_i, x_i, P_i = x_iP >$ in $\mathcal{PK}$. Finally, the challenger responds P_i as the answer.
   - Partial-Private-Key-Extract Queries: Once obtaining the query ID_i from $\mathcal{A}_1$, the challenger searches ID_i in the list $\mathcal{L}_1$. If this item exists, the challenger aborts. Otherwise, the challenger chooses β, s_i from $\mathbb{Z}_q^*$ at random and computes $R_i = s_iP - \beta PK_{pub}$ and inserts $< (ID_i, R_i), \beta >$ in $\mathcal{L}_1$. Finally, (s_i, R_i) is returned as the answer.
   - Public-Key-Replace Queries: Once obtaining the query (ID_i, P_i') from $\mathcal{A}_1$, the challenger searches the list $\mathcal{PK}$ with ID_i and updates the corresponding tuple as $< ID_i, \perp, P_i' >$.
   - Secret-Value-Extract Queries: Once obtaining the query ID_i from $\mathcal{A}_1$, the challenger searches this item in the list $\mathcal{PK}$. If this item is found, the challenger responds x_i as the answer; otherwise, the challenger chooses x_i from $\mathbb{Z}_q^*$ at random and inserts $< ID_i, x_i, P_i = x_iP >$ in $\mathcal{PK}$. Finally, the challenger responds x_i as the answer.
   - Re-Sign, Re-Key and Sign Queries: The challenger first queries Secret-Key-Extract and Partial-Private-Key-Extract oracles to obtain the partial private key and the secret key and then utilizes both key to answer these queries.

3) *Forgery.* In accordance with the forking lemma [37], if a valid forged signature σ^* on message m^* under identity ID^* and public key P_i^* is output by $\mathcal{A}_1$, then the challenger can utilize $\mathcal{A}_1$ as a sub-algorithm to generate two valid signature transcripts as follows.
   - Level 1: $(s_{ID^*}, R_{ID^*}, x_{ID^*}, \sigma_{i1}, \sigma_{i2}, m^*)$ and $(s_{ID^*}, R_{ID^*}, x_{ID^*}, \sigma_{i1}', \sigma_{i2}', m^*)$ under identity ID^* and public key P_i^*. In this way, the challenger is able to solve the extCDH problem by calculating $(h_{i1} - h_{i1}')^{-1}(\sigma_{i1} - \sigma_{i1}') = (h_{i1} - h_{i1}')^{-1}((k_ix_i + r_i + h_i \cdot a) - (k_ix_i + r_i + h_i' \cdot a))\alpha_ibP = \alpha_iabP$. Then, $(\alpha_iP, ab(\alpha_iP))$ is output as the solution, where h_{i1}, h_{i1}' are two different response from H_1 on input (ID_i, R_i) and α_i is the value associated with m^* in table $\mathcal{L}_3$.
   - Level 2: $(s_{ID^*}, R_{ID^*}, x_{ID^*}, \sigma_{i1}, \sigma_{i2}, \sigma_{i3}, \sigma_{i4}, m^*)$ and $(s_{ID^*}, R_{ID^*}, x_{ID^*}, \sigma_{i1}', \sigma_{i2}', \sigma_{i3}, \sigma_{i4}, m^*)$ under identity ID^* and public key P_i^*. In this way, the challenger is able to solve the extCDH problem by calculating $(h_{i1} - h_{i1}')^{-1}(\sigma_{i1} - \sigma_{i1}') = (h_{i1} - h_{i1}')^{-1}t_i((k_ix_i+r_i+h_i \cdot a)-(k_ix_i+r_i+h_i' \cdot a))\alpha_ibP = t_i\alpha_iabP$. Then, $(\alpha_i\sigma_{i3}, ab\alpha_i\sigma_{i3})$ is output as the solution, where h_{i1}, h_{i1}' are two different response from H_1 on input (ID_i, R_i) and α_i is the value associated with m^* in table $\mathcal{L}_3$.

APPENDIX D
PROOF OF LEMMA 2

Proof: If an adversary $\mathcal{A}_2$ breaking the security of the proposed unidirectional CL-PRS scheme in the **EUF-CL-PRS-CMA-II** game is given, a challenger can be built to solve the extCDH problem. With the extCDH instance (aP, bP) as input, the aim of the challenger is to output (Q, abQ), where a, b are randomly chosen from $\mathbb{Z}_q^*$ and Q is chosen from G_1 randomly. To make the security proof reader-friendly, a brief description for this process is presented in Fig. 7.

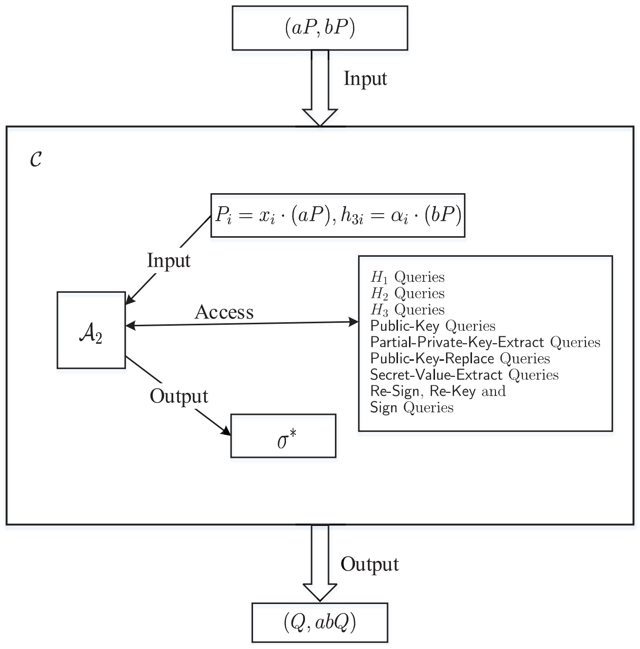
(aP, bP)
Input
$\mathcal{C}$
$P_i = x_i \cdot (aP), h_{3i} = \alpha_i \cdot (bP)$
Input
$\mathcal{A}_2$
Access
H_1 Queries
H_2 Queries
H_3 Queries
Public-Key Queries
Partial-Private-Key-Extract Queries
Public-Key-Replace Queries
Secret-Value-Extract Queries
Re-Sign, Re-Key and Sign Queries
Output
σ^*
Output
(Q, abQ)

FIGURE 7. Brief security Proof of Lemma 2.

1) *Initial.* The challenger selects s from $\mathbb{Z}_q^*$ at random and assigns $PK_{pub} = sP$. Then the challenger publishes the public parameters $\{G_1, G_2, e, q, P, PK_{pub}, H_1, H_2, H_3\}$ to $\mathcal{A}_2$.
2) *Attack.* In this phase, the challenger maintains four initially-empty lists $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{PK}$ and answers the adaptive queries issued by $\mathcal{A}_1$ as follows.
   - H_1 Queries: Once obtaining the query (ID_i, R_i) from $\mathcal{A}_2$, the challenger searches this item in the list $\mathcal{L}_1$. If this item is found, the challenger responds h_{1i} as the answer; otherwise, the challenger chooses h_{1i} from $\mathbb{Z}_q^*$ at random and inserts $< (ID_i, R_i), h_{1i} >$ in $\mathcal{L}_1$. Finally, the challenger responds h_{1i} as the answer.
   - H_2 Queries: Once obtaining the query (ID_i, P_i, R_i) from $\mathcal{A}_2$, the challenger searches this item in the list $\mathcal{L}_2$. If this item is found, the challenger responds h_{2i} as the answer; otherwise, the challenger chooses h_{2i} from $\mathbb{Z}_q^*$ at random and inserts $< (ID_i, P_i, R_i, PK_{pub}), h_{2i} >$ in $\mathcal{L}_2$. Finally, the challenger responds h_{2i} as the answer.
   - H_3 Queries: Once obtaining the query m_i from $\mathcal{A}_2$, the challenger searches this item in the list $\mathcal{L}_3$. If this item is found, the challenger responds h_{3i} as the answer; otherwise, the challenger chooses α_i from $\mathbb{Z}_q^*$ at random and calculates $h_{3i} = \alpha_i \cdot bP$. Finally, the challenger inserts $< m_i, \alpha_i, h_{3i} >$ in $\mathcal{L}_3$ and responds h_{3i} as the answer.
   - Public-Key Queries: Once obtaining the query $ID_i \neq ID^*$ from $\mathcal{A}_2$, the challenger searches this item in the list $\mathcal{PK}$. If this item is found, the challenger responds P_i as the answer; otherwise, the challenger chooses x_i from $\mathbb{Z}_q^*$ at random and inserts $< ID_i, x_i, P_i = x_i \cdot (aP) >$ in $\mathcal{PK}$. Then, the challenger responds P_i as the answer. Once obtaining the query ID^* from $\mathcal{A}_1$, the challenger inserts $< ID^*, \perp, aP >$ in $\mathcal{PK}$ and responds aP as the answer.
   - Partial-Private-Key-Extract Queries: Once obtaining the query ID_i from $\mathcal{A}_2$, the challenger searches ID_i in the list $\mathcal{L}_1$. If this item exists, the challenger aborts. Otherwise, the challenger chooses r_i from $\mathbb{Z}_q^*$ at random and computes $R_i = r_i P$, $h_{i1} = H_1(ID_i, R_i)$, $s_i = r_i + h_{i1}s$ and inserts $< (ID_i, R_i), h_{i1} >$ in $\mathcal{L}_1$. Finally, (s_i, R_i) is returned as the answer.
   - Public-Key-Replace Queries: Once obtaining the query (ID_i, P_i') from $\mathcal{A}_2$, the challenger searches the list $\mathcal{PK}$ with ID_i and updates the corresponding tuple as $< ID_i, \perp, P_i' >$.
   - Secret-Value-Extract Queries: Once obtaining the query $ID_i \neq ID^*$ from $\mathcal{A}_2$, the challenger searches this item in the list $\mathcal{PK}$. If this item is found, the challenger responds x_i as the answer; otherwise, the challenger chooses x_i from $\mathbb{Z}_q^*$ at random and inserts $< ID_i, x_i, P_i = x_i P >$ in $\mathcal{PK}$. Then, the challenger responds x_i as the answer. Once obtaining the query ID^* from $\mathcal{A}_2$, the challenger aborts.
   - Re-Sign, Re-Key and Sign Queries: The challenger first queries Secret-Key-Extract and Partial-Private-Key-Extract oracles to obtain the partial private key and the secret key and then utilizes both key to answer these queries.
3) *Forgery.* In accordance with the forking lemma [37], if a valid forged signature σ^* on message m^* under identity ID^* and public key P_i^* is output by $\mathcal{A}_2$, then the challenger can utilize $\mathcal{A}_2$ as a sub-algorithm to generate two valid signature transcripts as follows.
   - Level 1: $(s_{ID^*}, R_{ID^*}, x_{ID^*}, \sigma_{i1}, \sigma_{i2}, m^*)$ and $(s_{ID^*}, R_{ID^*}, x_{ID^*}, \sigma_{i1}', \sigma_{i2}', m^*)$ under identity ID^* and public key P_i^*. In this way, the challenger is able to solve the extCDH problem by calculating $(k_i - k_i')^{-1}(\sigma_{i1} - \sigma_{i1}') = (k_i - k_i')^{-1}((k_i a + r_i + h_i \cdot s) - (k_i' a + r_i + h_i' \cdot s))\alpha_i bP = \alpha_i abP$. Then, $(\alpha_i P, \alpha_i abP)$ is output as the solution, where k_i, k_i' are two different response from H_2 on input $(ID_i, P_i, R_i, PK_{pub})$ and α_i is the value associated with m^* in table $\mathcal{L}_3$.
   - Level 2: $(s_{ID^*}, R_{ID^*}, x_{ID^*}, \sigma_{i1}, \sigma_{i2}, \sigma_{i3}, \sigma_{i4}, m^*)$ and $(s_{ID^*}, R_{ID^*}, x_{ID^*}, \sigma_{i1}', \sigma_{i2}', \sigma_{i3}, \sigma_{i4}, m^*)$ under identity ID^* and public key P_i^*. In this way, the challenger is able to solve the extCDH problem by calculating $(k_i - k_i')^{-1}(\sigma_{i1} - \sigma_{i1}') = (k_i - k_i')^{-1} t_i((k_i a + r_i + h_i \cdot s) - (k_i' a + r_i + h_i \cdot s))\alpha_i bP = t_i \alpha_i abP$. Then, $(\alpha_i \sigma_{i3}, \alpha_i ab\sigma_{i3})$ is output as the

solution, where k_i, k_i' are two different response from H_2 on input $(ID_i, P_i, R_i, PK_{pub})$ and α_i is the value associated with m^* in table $\mathcal{L}_3$.

REFERENCES

[1] E. J. Katz, *Handbook of Mobile Communication Studies*. Cambridge, MA, USA: MIT Press, 2008.

[2] Z. Qin, Y. Wang, H. Cheng, Y. Zhou, Z. Sheng, and V. C. M. Leung, ''Demographic information prediction: A portrait of smartphone application users,'' *IEEE Trans. Emerg. Topics Comput.* vol. 6, no. 3, pp. 432–444, Jul. 2018.

[3] Alipay. [Online]. Available: https://www.alipay.com/

[4] Apple Pay. [Online]. Available: https://www.apple.com/apple-pay/

[5] WeChat Pay. [Online]. Available: https://pay.weixin.qq.com/index.php/public/wechatpay

[6] L. Atzori, A. Iera, and G. Morabito, ''The Internet of Things: A survey,'' *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[7] H. Xiong, Q. Mei, and Y. Zhao, ''Efficient and provably secure certificateless parallel key-insulated signature without pairing for IIoT environments,'' *IEEE Syst. J.*, to be published, doi: 10.1109/JSYST.2018.2890126.

[8] C.-M. Chen, B. Xiang, Y. Liu, and K.-H. Wang, ''A secure authentication protocol for Internet of vehicles,'' *IEEE Access*, doi: 10.1109/ACCESS.2019.2891105.

[9] K.-H. Yeh, ''A secure transaction scheme with certificateless cryptographic primitives for IoT-based mobile payments,'' *IEEE Syst. J.*, doi: 10.1109/JSYST.2017.2668389.

[10] B. C. Hu, D. S. Wong, Z. Zhang, and X. Deng, ''Key replacement attack against a generic construction of certificateless signature,'' in *Proc. ACISP*, vol. 6, 2006, pp. 235–246.

[11] H. Xiong, H. Zhang, and J. Sun, ''Attribute-based privacy-preserving data sharing for dynamic groups in cloud computing,'' *IEEE Syst. J.*, to be published, doi: 10.1109/JSYST.2018.2865221.2018.

[12] H. Xiong and J. Sun, ''Comments on 'verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing,''' *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 4, pp. 461–462, Jul. 2017.

[13] Z. Qin, J. Sun, A. Wahaballa, W. Zheng, H. Xiong, and H. Qin, ''A secure and privacy-preserving mobile wallet with outsourced verification in cloud computing,'' *Comput. Standards Interfaces*, vol. 54, pp. 55–60, Nov. 2017.

[14] J. Camenisch, S. Hohenberger, M. Ø. Pedersen, ''Batch verification of short signatures,'' in *Proc. Int. Conf. Adv. Cryptol.*, vol. 4515, 2007, pp. 246–263.

[15] M. Bellare, J. Garay, and T. Rabin, ''Batch verification of short signatures,'' in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 1998, pp. 236–250.

[16] J. Katz, *Digital Signatures*. Springer, 2010.

[17] M. Blaze, G. Bleumer, and M. Strauss, ''Divertible protocols and atomic proxy cryptography,'' in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 1403. 1998, pp. 127–144.

[18] B. Libert and D. Vergnaud, ''Multi-use unidirectional proxy re-signatures,'' in *Proc. ACM Conf. Comput. Commun. Secur.*, 2008, pp. 511–520.

[19] G. Ateniese and S. Hohenberger, ''Proxy re-signatures: New definitions, algorithms, and applications,'' in *Proc. ACM Conf. Comput. Commun. Secur.*, 2005, pp. 310–319.

[20] J. Shao, G. Wei, Y. Ling, and M. Xie, ''Unidirectional identity-based proxy re-signature,'' in *Proc. IEEE Int. Conf. Commun.*, Jun. 2011, pp. 1–5.

[21] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing* (Prentice Hall Professional Technical Reference). 2002, pp. 182–202.

[22] S. S. Al-Riyami and K. G. Paterson, ''Certificateless public key cryptography,'' in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 2894, no. 2. 2003, pp. 452–473.

[23] W. Diffie and M. E. Hellman, ''New directions in cryptography,'' *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[24] A. Shamir, ''Identity-based cryptosystems and signature schemes,'' in *Proc. Workshop Theory Appl. Cryptograph. Techn.*, 1984, vol. 21, no. 2, pp. 47–53.

[25] D. Guo, W. Ping, Y. Dan, and X. Yang, ''A certificateless proxy re-signature scheme,'' in *Proc. IEEE Int. Conf. Comput. Sci. Inf. Technol.*, Jul. 2010, pp. 157–161.

[26] S. Gordon, L. M. Kristensen, and J. Billington, ''Verification of a revised WAP wireless transaction protocol,'' in *Proc. ICATPN*, 2002, pp. 182–202.

[27] K. Kamijo, T. Aihara, and M. Murase, ''Anonymity-aware face-to-face mobile payment,'' in *Proc. 7th Int. ICST Conf. Mobile Ubiquitous Syst., Comput., Netw., Services (MobiQuitous)*, vol. 73, 2010, pp. 198–209.

[28] V. Sureshkumar, A. Ramalingam, N. Rajamanickam, and R. Amin, ''A lightweight two-gateway based payment protocol ensuring accountability and unlinkable anonymity with dynamic identity,'' *Comput. Elect. Eng.*, vol. 57, pp. 223–240, Jan. 2017.

[29] J.-H. Yang and P.-Y. Lin, ''A mobile payment mechanism with anonymity for cloud computing,'' *J. Syst. Softw.*, vol. 116, pp. 69–74, Jun. 2016.

[30] Y. Liao, Y. He, F. Li, and S. Zhou, ''Analysis of a mobile payment protocol with outsourced verification in cloud server and the improvement,'' *Comput. Standards Interfaces*, vol. 56, pp. 101–106, Feb. 2018, doi: 10.1016/j.csi.2017.09.008.

[31] D. Boneh and M. Franklin, ''Identity-based encryption from the Weil pairing,'' in *Proc. 21st Annu. Int. Cryptol. Conf. (CRYPTO)*, 2001, pp. 213–229.

[32] Z. Zhang, D. S. Wong, J. Xu, and D. Feng, ''Certificateless public-key signature: Security model and efficient construction,'' in *Proc. 4th Int. Conf. Appl. Cryptogr. Netw. Secur. (ACNS)*, 2006, pp. 293–308.

[33] X. Huang, W. Susilo, Y. Mu, and F. Zhang, ''On the security of certificateless signature schemes from Asiacrypt 2003,'' in *Proc. 4th Int. Conf. Cryptol. Netw. Secur. (CANS)*, 2005, pp. 13–25.

[34] H. Xiong, ''Cost-effective scalable and anonymous certificateless remote authentication protocol,'' *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 12, pp. 2327–2339, Dec. 2014.

[35] H. Xiong and Z. Qin, ''Revocable and scalable certificateless remote authentication protocol with anonymity for wireless body area networks,'' *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 7, pp. 1442–1455, Jul. 2015.

[36] J.-S. Coron, ''On the exact security of full domain hash,'' in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 1880. 2000, pp. 229–235.

[37] D. Pointcheval and J. Stern, ''Security arguments for digital signatures and blind signatures,'' *J. Cryptol.*, vol. 13, no. 3, pp. 361–369, 2000.

[38] *The Pairing-Based Cryptography Library (PBC)*. [Online]. Available: https://crypto.stanford.edu/pbc/

Authors' photographs and biographies not available at the time of publication.