IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 13, 2019, accepted June 4, 2019, date of publication June 19, 2019, date of current version July 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2923697

Certificateless Short Aggregate Signature Scheme for Mobile Devices

LUNZHI DENG[1,2,3], YIXIAN YANG[1], AND YULING CHEN[1]
[1]Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China
[2]College of Computer Science and Technology, Guizhou University, Guiyang 550025, China
[3]School of Mathematical Sciences, Guizhou Normal University, Guiyang 550001, China

Corresponding author: Lunzhi Deng (denglunzhi@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61562012, in part by the Innovation Group Major Research Projects of Department of Education of Guizhou Province under Grant KY[2016]026, and in part by the Science and Technology Plan Project of Guizhou Province under Grant [2019]1434.

ABSTRACT Today, data is exploding. A large amount of data needs to be processed in a timely and an efficient manner. Aggregate signatures are an efficient and secure way to handle large numbers of digital signatures. In an aggregate signature scheme, n signatures on n messages from n users can be combined into a single signature, which can make anyone believes that the n messages were indeed signed by the n corresponding users. In the recent decade, numerous certificateless aggregate signature (CLAS) schemes have been introduced. In most CLAS schemes currently known, the number of hash-to-point operations and the size of signature increase linearly with the number of signers, so they are not suitable for computing restricted devices, such as mobile devices. In this paper, a new CLAS scheme is constructed, which requires only two pairing operations and the signature contains only one point and state information. It is more efficient than the previous ones and suit for the mobile devices.

INDEX TERMS Certificateless cryptography, short signature, aggregate signature, pairing, random oracles model.

I. INTRODUCTION

The continuous advancement of hardware, software and communication technologies has driven the development of mobile communication devices such as personal digital assistants (PDAs) and smart phones. Market research firm Newzoo recently released the "2018 Global Mobile Market Report", which shows that by the end of 2018, the number of global smartphone users will reach 3.3 billion, and China will be the largest smartphone market - with 783 million users. By 2021, the total number of global smartphone users will increase to 3.8 billion, of which the number of users in the Asia-Pacific region will increase to 2.1 billion.

The development of network technology and communication equipment has brought great convenience to people's lives. The Internet has become an indispensable part of people's lives. According to the 43rd "Statistical Report on the Development of China's Internet Network", as of December 2018, the number of online users in China was 829 million, and the number of online users increased by 55.63 million. Internet penetration reached 59.6%, an increase of 3.8% over 2017. The proportion of Internet users using mobile phones in China is 98.6%. In 2018, the number of new mobile Internet users was 64.33 million, and the number of mobile Internet users reached 187 million, accounting for 98.6% of the size of China's online users. In 2017, this ratio was 97.5%. In 2018, the average online time for online users in China was 27.6 hours per week, 0.6 hours higher than in 2017. This also means that the average online time per person in 2018 is 4 hours.

In traditional public key infrastructure (PKI), the user chooses his own private key and sets the corresponding public key which is bundled with the user by a digital certificate issued by a trusted certification authority (CA). To manage the certificates, a lots of resources is consumed. To reduce costs, Shamir [21] introduced identity-based public key cryptography. In this setting, the user's private key is generated by a trusted private key generator (PKG) through the user's public key which is a string (e.g. user's identity card number or email address), since PKG controls all the private keys, it can harm to any user. To solve the two problems, Al-Riyami *et al.* [1] put forward certificateless public key cryptography. In this notion, a user's full private key includes two parts: a secret value chosen by himself and a partial

The associate editor coordinating the review of this manuscript and approving it for publication was Alessandra De Benedictis.

private key issued by a semi-trusted key generation center (KGC) according to user's identity.

In 2001, Boneh *et al.* [2] presented short signature. The length of the signature is much smaller than that of the general signature. In 2003, Boneh *et al.* [3] put forward aggregate signature, n signatures on n messages from n users can be combine into a single signature. Verifying the resulting signature, anyone believes that the n messages were indeed signed by the n corresponding users. With the rapid growth of various data, how to process data safely and efficiently becomes an urgent problem to be solved. The CLAS scheme has the advantages of certificateless cryptography and aggregate signature, which has attracted the attention of researchers. In most CLAS schemes currently known, the number of hash-to-point operations and the size of the signature increase linearly with the number of signers, which requires the user to have strong computing and storage capabilities. In order to be portable, the size of the mobile device is relatively small, and the computing and storage capabilities are limited. Therefore, these CLAS solutions are not suitable for mobile devices. It is very meaningful to design an efficient and secure CLAS scheme for mobile devices.

A. RELATED WORK

Huang *et al.* [13] put forward the first certificateless short signature (CLSS) scheme and gave the security proofs. However, Shim [20] indicated that the scheme [13] is insecure against the type I adversary. Du and Wen [11] proposed a CLSS scheme and showed the security proofs. However, Choi *et al.* [7] demonstrated that the scheme [11] is vulnerable against the strong Type I adversary and constructed a new CLSS scheme. However, Tian *et al.* [25] pointed out that the scheme [7] is still insecure and a strong Type I adversary can forge a signature. Tso *et al.* [23] presented a CLSS scheme and showed the security proofs based on $k - CAA$ (the collusion attack algorithm with k traitors) problem. Tso *et al.* [24] constructed another CLSS scheme and gave the security proofs in a weak secure model, where the attacker was not allowed to query the secret value of the challenge user. He *et al.* [14] put forward a CLSS scheme requiring hash-to-point operations. Tsai [26] proposed a CLSS scheme requiring only one pairing operation. Deng *et al.* [9] constructed a new CLSS scheme and showed the security proofs in the standard model.

Many CLAS schemes have been given in the past 12 years. Castro and Dahab [4] designed the first CLAS scheme. Gong *et al.* [12] put forward two CLAS schemes and gave the security proofs in a weak model. In order to improve computing efficiency, Zhang and Zhang [30], Zhang *et al.* [29] and Nie *et al.* [19] put forward a CLAS scheme, respectively. In the three schemes, the signers need to share synchronized clocks to generate an aggregate signature. Xiong *et al.* [27] constructed a new CLAS scheme, which requires only three pairing operations. He *et al.* [15] pointed out that the scheme [27] is insecure and presented an improved scheme. However, Li *et al.* [18] showed that a malicious-but-passive

TABLE 1. Notations.

F_p	A prime finite field.
q	A prime number.
z_q^*	A set consisting of positive integers less than q.
$\hat{e}$	A bilinear pairing, where $\hat{e}: G_1 \times G_1 \to G_2$.
P	A generators of the group G_1.
P_{pub}	The public key of system, where $P_{pub} = xP$.
$H_1 \sim H_3$	Three secure hash functions.
ID_i	The identity of i^{th} user, where $ID_i \in \{0,1\}^*$.
D_i	The partial private key i^{th} user, where $D_i = (R_i, d_i)$.
SV_i	The secret value i^{th} user, where $SV_i = (t_i, x_i)$.
PK_i	The public key of i^{th} user, where $PK_i = (T_i, X_i, R_i, S_i)$ and $T_i = t_iP, X_i = x_iP, R_i = r_iP, S_i = s_iP$.
W	A set consisting of n identities, where $W = \{ID_1, \ldots, ID_n\}$.
A	A set consisting of the identities/public keys, where $A = W \bigcup \{PK_i : ID_i \in W\}$.
$\triangle$	A state information, where $\triangle \in \{0,1\}^{160}$.
m_i, σ_i	A message and the signature.
M	A set consisting of n messages, where $M = \{m_1, \ldots, m_n\}$.
σ	An aggregate signature.

KGC can forge a valid signature in the scheme [15]. Zhang *et al.* [31] indicated that the scheme [27] is vulnerable by showing four kinds of concrete attacks, then presented a new CLAS scheme. Cheng *et al.* [6] showed that the scheme [27] is vulnerable against the "honest-but-curious" KGC, and put forward a reformative scheme. Chen *et al.* [5] constructed a CLAS scheme with a constant number of pairing operations. However, Shen *et al.* [22] pointed out that the scheme [5] is vulnerable. Deng *et al.* [8] proposed a new CLAS scheme with a constant number of pairing operations. Deng *et al.* [10] constructed another CLAS scheme and gave the security proofs based on RSA and discrete logarithm problem. Kumar *et al.* [17] proposed a CLAS scheme and gave the security proofs. However, Xie *et al.* [28] indicated that the scheme [17] is vulnerable and presented a new CLAS scheme without pairing operations.

B. MOTIVATION AND CONTRIBUTIONS

In all known CLAS schemes, the number of hash-to-point operations increase linearly with the number of signers. In most CLAS schemes, the size of signature increase linearly with the number of signers. These schemes require that the devices in the network have strong computing and storage capabilities. Due to the size constraints, the computing and storage capabilities of mobile devices are limited. So these schemes are not suit for mobile devices. Therefore, it is quite significant to constructed an efficient and secure CLAS scheme for mobile devices.

In this paper, a new CLAS scheme is constructed that has the following features:

- It is proved to be secure under the assumption that it is hard to solve the computational Diffie-Hellman problem.
- It requires only 2 pairing operations, independent of the number of signers.
- The size of signature is a point and a state information, independent of the number of signers.

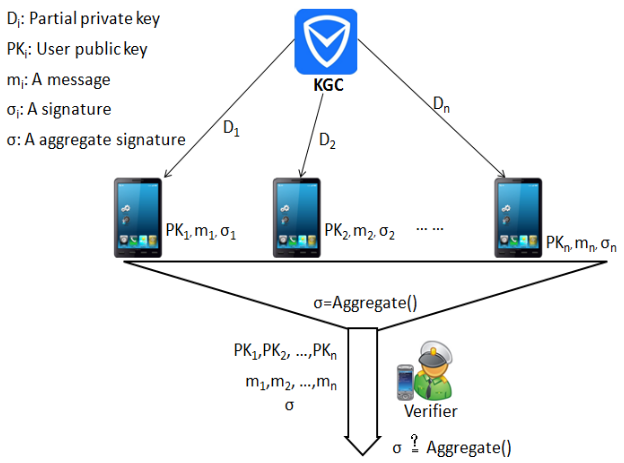

FIGURE 1. CLAS Scheme.

II. PRELIMINARIES

In this section, the definition of bilinear pairing and computational Diffie-Hellman problem is first given, then the system model and security requirements of the CLAS scheme are introduced. The notations used throughout the paper are listed in Table 1.

A. BILINEAR PAIRING

Let $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be a map with the following properties. Where $G_1 = (P)$ is an additive group with prime order q and G_2 be a multiplicative group with the same order.

- Bilinearity: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_q$.
- Non-degeneracy: There exist $P, Q \in G_1$ such that $\hat{e}(P, Q) \neq 1_{G_2}$.
- Computability: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in G_1$.

Definition 1: Computation Diffie-Hellman (CDH) problem. For $P \in G_1$, given a tuple (aP, bP), compute abP.

B. SYSTEM MODEL

A certificateless aggregate signature scheme (Fig.1) consists of the following seven algorithms:

- Setup: On input a security parameter ν, this algorithm outputs *params* (system parameters) and *msk* (master secret key).
- PPK-Extract: On input an identity $ID_i \in \{0, 1\}^*$, this algorithm outputs the partial private key D_i.
- SV-Set: On input an identity ID_i, this algorithm outputs the secret value SV_i.
- UPK-Set: On input an identity ID_i, this algorithm outputs the public key PK_i.
- Sign: On input a tuple $(\triangle, m, SV_i, D_i)$, this algorithm outputs a signature σ.
- Aggregate: On input a tuple $(\triangle, \sigma_1, \cdots, \sigma_n, M, A)$, this algorithm outputs an aggregate signature σ.
- Agg-verify: On input a tuple (σ, M, A), this algorithm outputs 1 or 0.

Remark: In order to shorten the length of the signature, all users must use the same state information $\triangle$ when signing. Where $\triangle$ be the current time parameter, or other information which all users have shared.

C. SECURITY MODEL

Definition 2: A CLAS scheme is unforgeable (UNF-CLAS) if the advantage of any adversary is negligible in the following two games.

Game I. The first game is carried out between a challenger $\mathscr{C}$ and a Type I adversary $\mathscr{A}_1$.

Initialization. $\mathscr{C}$ runs the Setup algorithm to obtain *msk* and *params*. $\mathscr{C}$ keeps *msk* secret and gives *params* to $\mathscr{A}_1$.

Query. $\mathscr{A}_1$ executes a polynomially bounded number of queries.

- Hash-Query: $\mathscr{A}_1$ can query the values of the hash functions for any input.
- UPK-Query: $\mathscr{A}_1$ inputs an identity ID_i, $\mathscr{C}$ returns a value PK_i.
- UPK-Replace: $\mathscr{A}_1$ inputs a tuple (PK_i', ID_i), $\mathscr{C}$ replaces PK_i with PK_i'.
- PPK-Query: $\mathscr{A}_1$ inputs an identity ID_i, $\mathscr{C}$ returns a value D_i. $\mathscr{A}_1$ cannot do it if R_i or S_i has been replaced.
- SV-Query: $\mathscr{A}_1$ inputs an identity ID_i, $\mathscr{C}$ returns a value SV_i. $\mathscr{A}_1$ cannot do it if T_i or X_i has been replaced.
- Sig-Query: $\mathscr{A}_1$ inputs a tuple (m_i, ID_i, PK_i), $\mathscr{C}$ returns a signature σ_i.

Forge. $\mathscr{A}_1$ outputs a tuple $(\sigma^*, \triangle^*, M^*, A^*)$ and wins if the following conditions hold:

1) Agg-verify $(\sigma^*, \triangle^*, M^*, A^*) = 1$.
2) There is an identity $ID_j^* \in W^*$ for which $\mathscr{A}_1$ did not perform PPK-Query or replace R_j or S_j.
3) $\mathscr{A}_1$ did not perform Sig-Query for $(\triangle^*, m_j^*, ID_j^*, PK_j^*)$.

The advantage of $\mathscr{A}_1$ is defined as:

$$Adv_{\mathscr{A}_1}^{EUF-CLAS} = \Pr[\mathscr{A}_1 \ wins].$$

Game II. The game is performed between a challenger $\mathscr{C}$ and a Type II adversary $\mathscr{A}_2$.

Initialization. $\mathscr{A}_2$ runs the Setup algorithm to get *msk* and *params*, then gives them to $\mathscr{C}$.

Query. $\mathscr{A}_2$ performs a polynomially bounded number of queries as those in Game I.

Forge. $\mathscr{A}_2$ outputs a tuple $(\sigma^*, \triangle^*, M^*, A^*)$ and wins if the following conditions hold:

1) Agg-verify $(\sigma^*, \triangle^*, M^*, A^*) = 1$.
2) There is one user $ID_j^* \in W^*$ for which $\mathscr{A}_2$ did not perform SV-Query or replace T_j or X_j.
3) $\mathscr{A}_2$ did not perform Sig-Query for $(\triangle^*, m_j^*, ID_j^*, PK_j^*)$.

The advantage of $\mathscr{A}_2$ is defined as:

$$Adv_{\mathscr{A}_2}^{EUF-CLAS} = \Pr[\mathscr{A}_2 \ wins].$$

III. NEW SCHEME

In this section, a new CLAS scheme is proposed as follows.

- Setup: Given the security parameter ν, KGC does as follows.

1) Chooses a bilinear pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$. Where G_1 and G_2 are two group with prime order $q > 2^\nu$, P is a generator of G_1.
2) Chooses four cryptographic hash functions $H_1, H_2, H_3 : \{0, 1\}^* \rightarrow Z_q^*$, $H_4 : \{0, 1\}^{160} \rightarrow G_1$.
3) Chooses $\lambda \in Z_q^*$, computes $P_{pub} = \lambda P$, sets master secret key $msk = \{\lambda\}$.
4) Broadcasts the public parameters: $params = \{G_1, G_2, q, \hat{e}, P, P_{pub}, H_1 \sim H_4\}$.

- PPK-Extract: For a user $ID_i \in \{0, 1\}^*$, KGC randomly chooses two different numbers $r_i, s_i \in Z_q^*$, computes $R_i = r_iP$, $S_i = s_iP$, $l_i = H_1(ID_i, R_i, S_i)$, $d_i = r_i + l_i\lambda \mod q$ and $c_i = s_i + l_i\lambda \mod q$, then sends $D_i = (d_i, c_i, R_i, S_i)$ to the user ID_i via a secure channel.
- SV-Set: The user ID_i randomly chooses two different numbers $t_i, x_i \in Z_q^*$.
- UPK-Set: The user ID_i computes $T_i = t_iP$, $X_i = x_iP$ and sets $PK_i = (T_i, X_i, R_i, S_i)$.
- Sign: For a message $m_i \in \{0, 1\}^*$, the signer ID_i first chooses a one-time-use state information $\triangle$, then performs the following steps:
  1) Computes $h_i = H_2(m_i, ID_i, PK_i, \triangle)$, $k_i = H_3(m_i, ID_i, PK_i, \triangle)$, $Q = H_4(\triangle)$.
  2) Computes $\sigma_i = (h_id_i + c_i + k_it_i + x_i)Q$ and outputs σ_i as the signature.
- Aggregate: On input a tuple $(\triangle, \sigma_1, \cdots, \sigma_n, M, A)$, anyone computes $\sigma = \sum_{i=1}^{n} \sigma_i$ and outputs a signature $(\sigma, \triangle, M, A)$.
- Agg-verify: On receive a tuple $(\sigma, \triangle, M, A)$, the verifier performs the following steps:
  1) Computes $l_i = H_1(ID_i, R_i, S_i)$ for $i = 1, 2, \cdots, n$.
  2) Computes $h_i = H_2(m_i, ID_i, PK_i, \triangle)$, $k_i = H_3(m_i, ID_i, PK_i, \triangle)$ for $i = 1, 2, \cdots, n$.
  3) Computes $Q = H_4(\triangle)$.
  4) Computes $V = \sum_{i=1}(h_iR_i + k_iT_i + l_i(h_i + 1)P_{pub} + S_i + X_i$.
  5) Checks whether $\hat{e}(\sigma, P) = \hat{e}(V, Q)$. If the equation holds, accepts the signature. Otherwise, rejects.
- On correctness

$$\begin{aligned}
&\hat{e}(\sigma, P)\\
&= \hat{e}(\sum_{i=1}^{n}(h_id_i + c_i + k_it_i + x_i)Q, \quad P)\\
&= \hat{e}(\sum_{i=1}^{n}(h_i(r_i + l_i\lambda) + s_i + l_i\lambda + k_it_i + x_i)P, \quad Q)\\
&= \hat{e}(\sum_{i=1}^{n}(h_iR_i + l_i(h_i + 1)P_{pub} + k_iT_i + S_i + X_i), \quad Q)\\
&= \hat{e}(V, Q)
\end{aligned}$$

IV. SECURITY

In this section, the proposed CLAS scheme is proved to be unforgeable in the random oracle model(ROM).

Theorem 1: In ROM, the scheme is unforgeable against the Type I adversary if the CDH problem is hard.

Proof: Suppose that the challenger $\mathscr{C}$ receives a tuple (P, aP, bP) with the purpose of computing abP, $\mathscr{C}$ will act as $\mathscr{A}_1$'s challenger in the Game I.

Initialization. $\mathscr{C}$ runs the Setup program with the parameter ν, then gives $\mathscr{A}_1$ the $params = \{G_1, G_2, q, \hat{e}, P, P_{pub} = \lambda P, H_1 \sim H_4\}$.

Queries. $\mathscr{A}_1$ will perform UPK-Query before an identity ID_i is used in any other queries. Several lists are set to store the queries and answers. All the lists are initially empty.

- UPK-Query: $\mathscr{C}$ maintains a list L_U of tuple $(ID_i, t_i, x_i, r_i, s_i)$. $\mathscr{A}_1$ inputs an identity ID_i, $\mathscr{C}$ responds as follows:
  1) If $i = f$, randomly picks $t_f, x_f, s_f \in Z_q^*$, sets $ID_f = ID^\diamond$ and $PK^\diamond = (t_fP, x_fP, aP, s_fP)$. then adds $(ID_f, t_f, x_f, *, s_f)$ to the list L_U.
  2) If $i \neq f$, randomly picks $t_i, x_i, r_i, s_i \in Z_q^*$ and returns $PK_i = (t_iP, x_iP, r_iP, s_iP)$, then adds $(ID_i, t_i, x_i, r_i, s_i)$ to the list L_U.
- H_1-Query: $\mathscr{C}$ maintains a list L_1 of tuple (α_i, l_i). $\mathscr{A}_1$ issues a query $H_1(\alpha_i)$, $\mathscr{C}$ randomly picks $l_i \in Z_q^*$, sets $H_1(\alpha_i) = l_i$ and adds (α_i, l_i) to the list L_1.
- H_2-Query: $\mathscr{C}$ maintains a list L_2 of tuple (β_i, h_i). $\mathscr{A}_1$ issues a query $H_2(\beta_i)$, $\mathscr{C}$ randomly picks $h_i \in Z_q^*$, sets $H_1(\beta_i) = h_i$ and adds (β_i, h_i) to the list L_2.
- H_3-Query: $\mathscr{C}$ maintains a list L_3 of tuple (β_i, k_i). When $\mathscr{A}_1$ issues a query $H_3(\beta_i)$, $\mathscr{C}$ randomly picks $k_i \in Z_q^*$, sets $H_3(\beta_i) = k_i$ and adds (β_i, k_i) to the list L_3.
- H_4-Query: $\mathscr{C}$ maintains a list L_4 of tuple $(\triangle_i, \mu_i, \delta_i)$. $\mathscr{A}_1$ issues a query $H_4(\triangle_i)$, $\mathscr{C}$ generates a random coin $\mu_i \in \{0, 1\}$ such that $\Pr[\mu_i = 0] = \frac{1}{q_R+1}$, and randomly picks $\delta_i \in Z_q^*$. If $\mu_i = 0$, sets $H_4(\triangle_i) = \delta_ibP$. Otherwise, sets $H_4(\triangle_i) = \delta_iP$, then adds $(\triangle_i, \mu_i, \delta_i)$ to the list L_4.
- UPK-Replace: $\mathscr{C}$ maintains a list L_R of tuple (ID_i, PK_i, PK_i'). $\mathscr{A}_1$ inputs a tuple (ID_i, PK_i'), $\mathscr{C}$ replaces PK_i with PK_i' and adds (ID_i, PK_i, PK_i') to the list L_R.
- PPK-Query: $\mathscr{C}$ maintains a list L_D of tuple (ID_i, D_i). $\mathscr{A}_1$ inputs an identity ID_i. If $ID_i = ID^\diamond$, $\mathscr{C}$ fails and stops. Otherwise, $\mathscr{C}$ finds $(ID_i, t_i, x_i, r_i, s_i)$ in the list L_U, outputs the D_i by calling the PPK-Extract algorithm, then adds (ID_i, D_i) to the list L_D.
- SV-Query: $\mathscr{C}$ maintains a list L_E of tuple (ID_i, t_i, x_i). $\mathscr{A}_1$ inputs an identity ID_i. $\mathscr{C}$ finds $(ID_i, t_i, x_i, r_i, s_i)$ in the list L_U, responds with $SV_i = (t_i, x_i)$ and adds (ID_i, t_i, x_i) to the list L_E.
- Sig-Query: $\mathscr{A}_1$ inputs a tuple (m_i, ID_i, PK_i), $\mathscr{C}$ does as follow:
  1) Chooses a state information $\triangle_i$.
  2) If $ID_i \neq ID^\diamond$ and $ID_i \notin L_R$, $\mathscr{C}$ gives a signature by calling the Sign algorithm.
  3) If $ID_i = ID^\diamond$ or $ID_i \in L_R$, $\mathscr{C}$ finds $(\triangle_i, \mu_i, \delta_i)$ in the list L_4. If $\mu_i = 0$, $\mathscr{C}$ fails and stops. Otherwise, $\mathscr{C}$ does as follow:
     a) Computes $l_i = H_1(ID_i, R_i, S_i)$, $h_i = H_2(m_i, ID_i, PK_i, \triangle_i)$, and $k_i = H_3(m_i, ID_i, PK_i, \triangle_i)$.

b) Computes $\sigma_i = \delta_i(h_i R_i + l_i(h_i+1)P_{pub} + k_i T_i + S_i + X_i)$.

Forge. $\mathscr{A}_1$ outputs a tuple $(\sigma^*, \triangle^*, M^*, A^*)$ that fulfills the requirements as defined in the Game I.

Solve CDH problem. By the proposed scheme, it follows that $\sigma^* = \sum_{i=1}^{n} \sigma_i^*$, σ_i^* is a signature on the message m_i^* with the state information $\triangle^*$ under ID_i^*/PK_i^* for $i = 1, \cdots, n$. There is an identity $ID_j^* \in W^*(1 \le j \le n)$ for which $\mathscr{A}_1$ did not perform PPK-Query or replace R_j or S_j. which implies that σ_j^* is a forge signature on the message m_j^*. After replaying $\mathscr{A}_1$ with the same random tape but different h_j^* returned by query $H_2(m_j^*, ID_j^*, PK_j^*, \triangle^*)$, $\mathscr{C}$ gets two valid signatures σ^* and $\sigma^{*\prime}$, where $\sigma^* = \sum_{i=1}^{n} \sigma_i^*$ and $\sigma^{*\prime} = \sum_{i=1}^{n} \sigma_i^{*\prime}$, $\sigma_j^* \neq \sigma_j^{*\prime}$ and $\sigma_i^* = \sigma_i^{*\prime}$ for $i \neq j$. If $ID_j^* = ID^\diamond$ and $\mu^* = 0$, then $\sigma_j^* = (h_j^*(a + l_j^*\lambda) + (s_j^* + l_j^*\lambda) + k_j^* t_j^* + x_j^*)\delta^* bP$, $\sigma_j^{*\prime} = (h_j^{*\prime}(a + l_j^*\lambda) + (s_j^* + l_j^*\lambda) + k_j^* t_j^* + x_j^*)\delta^* bP$. $\mathscr{C}$ finds $(ID_j^*, R_j^*, S_j^*, l_j^*)$ in the list L_1 and $(\triangle^*, \mu^*, \delta^*)$ in the list L_4, then solves CDH problem by computing: $abP = \delta^{*-1}(h_j^{*\prime} - h_j^*)^{-1}(\sigma^{*\prime} - \sigma^*) - l_j^*\lambda bP$.

Probability. Let $q_{H_i}(i = 1, 2, 3, 4)$, q_U, q_R, q_D and q_S be the number of hush function H_i-Query $(i = 1, 2, 3, 4)$, UPK-Query, UPK-Replace, PPK-Query, Sig-Query respectively. Due to $\mathscr{A}_1$ cannot perform PPK-Query for ID_i if R_i or S_i has been replaced, it is an reasonable assumption that $L_D \bigcap L_R = \emptyset$. Several notations are defined as follows:

π_1: $\mathscr{C}$ does not fail in PPK-Query.

π_2: $\mathscr{C}$ does not fail in Sig-Query.

π_3: $ID_j^* = ID^\diamond$ and $\mu^* = 0$.

It is easy to get following results:

$\Pr[\pi_1] = \frac{q_U - q_D}{q_U}$.

$\Pr[\pi_2|\pi_1] = (\frac{q_U-(q_R+1)}{q_U} + \frac{q_R+1}{q_U} \cdot (1 - \frac{1}{q_R+1}))^{q_S} = (1 - \frac{1}{q_U})^{q_S}$,

$\Pr[\pi_3|\pi_1 \wedge \pi_2] = \frac{1}{q_U - q_D - q_R} \cdot \frac{1}{q_R+1}$.

$$\begin{aligned}
&\Pr[\mathscr{C}\ success] \\
&= \Pr[\pi_1 \wedge \pi_2 \wedge \pi_3] \\
&= \Pr[\pi_1] \cdot \Pr[\pi_2|\pi_1] \cdot \Pr[\pi_3|\pi_1 \wedge \pi_2] \\
&= \frac{q_U - q_D}{q_U} \cdot (1 - \frac{1}{q_U})^{q_S} \cdot \frac{1}{q_U - q_D - q_R} \cdot \frac{1}{q_R + 1} \\
&\ge \frac{1}{q_U(q_R+1)} \cdot e^{-\frac{q_S}{q_U}}
\end{aligned}$$

Therefore, if $\mathscr{A}_1$ can forge a aggregate signature with probability ϵ, then $\mathscr{C}$ can solve the CDH problem with probability $\frac{\epsilon}{q_U(q_R+1)} \cdot e^{-\frac{q_S}{q_U}}$.

Theorem 2: In ROM, the scheme is unforgeable against the Type II adversary if the CDH problem is hard.

Proof: Suppose that the challenger $\mathscr{C}$ receives a tuple (P, aP, bP) with the purpose of computing abP, $\mathscr{C}$ will act as $\mathscr{A}_2$'s challenger in the Game II.

Initialization. $\mathscr{C}$ runs the Setup program with the parameter ν, then gives $\mathscr{A}_2$ the $params = \{G_1, G_2, q, \hat{e}, P, P_{pub} = \lambda P, H_1, H_2, H_3, H_4\}$ and $msk = \{\lambda\}$.

Queries. $\mathscr{A}_2$ will perform UPK-Query before an identity ID_i is used in any other queries. Several lists are set to store the queries and answers. All the lists are initially empty.

- UPK-Query: $\mathscr{C}$ maintains a list L_U of tuple $(ID_i, t_i, x_i, r_i, s_i)$. $\mathscr{A}_2$ inputs a user ID_i, $\mathscr{C}$ responds as follows:
  1) If $i = f$, randomly picks $x_f, r_f, s_f \in Z_q^*$, sets $ID_f = ID^\diamond$ and $PK^\diamond = (aP, x_f P, r_f P, s_f P)$, then adds $(ID_f, *, x_f, r_f, s_f)$ to the list L_U.
  2) If $i \neq f$, randomly picks $t_i, x_i, r_i, s_i \in Z_q^*$ and returns $PK_i = (t_i P, x_i P, r_i P, s_i P)$, then adds $(ID_i, t_i, x_i, r_i, s_i)$ to the list L_U.
- H_i-Query(i = 1, 2, 3, 4): Same as those in the proof of Theorem 1.
- UPK-Replace: Same as that in the proof of Theorem 1.
- PPK-Query: $\mathscr{C}$ maintains a list L_D of tuple (ID_i, D_i). $\mathscr{A}_2$ inputs an identity ID_i. $\mathscr{C}$ finds $(ID_i, t_i, x_i, r_i, s_i)$ in the list L_U, outputs the D_i by calling the PPK-Extract algorithm, then adds (ID_i, D_i) to the list L_D.
- SV-Query: $\mathscr{C}$ maintains list L_E of tuple (ID_i, t_i, x_i). $\mathscr{A}_2$ inputs a user ID_i. If $ID_i = ID^\diamond$, $\mathscr{C}$ fails and stops. Otherwise, $\mathscr{C}$ finds $(ID_i, t_i, x_i, r_i, s_i)$ in list L_U, responds with $SV_i = (t_i, x_i)$ and adds (ID_i, t_i, x_i) to list L_E.
- Sig-Query: Same as that in the proof of Theorem 1.

Forge. $\mathscr{A}_2$ outputs a tuple $(\sigma^*, \triangle^*, M^*, A^*)$ that fulfills the requirements as defined in the Game II.

Solve CDH problem. By the proposed scheme, it follows that $\sigma^* = \sum_{i=1}^{n} \sigma_i^*$, σ_i^* is a signature on the message m_i^* with the state information $\triangle^*$ under ID_i^*/PK_i^* for $i = 1, \cdots, n$. There is an identity $ID_j^* \in W^*$ for which $\mathscr{A}_2$ did not perform SV-Query or replace T_j or X_j, which implies that σ_j^* is a forge signature on the message m_j^*. After replaying $\mathscr{A}_2$ with the same random tape but different k_j^* returned by query $H_3(m_j^*, ID_j^*, PK_j^*, \triangle^*)$, $\mathscr{C}$ get two valid signatures σ^* and $\sigma^{*\prime}$, where $\sigma^* = \sum_{i=1}^{n} \sigma_i^*$ and $\sigma^{*\prime} = \sum_{i=1}^{n} \sigma_i^{*\prime}$, $\sigma_j^* \neq \sigma_j^{*\prime}$ and $\sigma_i^* = \sigma_i^{*\prime}$ for $i \neq j$. If $ID_j^* = ID^\diamond$ and $\mu^* = 0$, then $\sigma_j^* = (h_j^*(r_j^* + l_j^*\lambda) + (s_j^* + l_j^*\lambda) + k_j^* a + x_j^*)\delta^* bP$, $\sigma_j^{*\prime} = (h_j^*(r_j^* + l_j^*\lambda) + (s_j^* + l_j^*\lambda) + k_j^{*\prime} a + x_j^*)\delta^* bP$. $\mathscr{C}$ finds $(\triangle^*, \mu^*, \delta^*)$ in the list L_4, then solves CDH problem by computing: $abP = (\delta^*(k_j^{*\prime} - k_j^*))^{-1}(\sigma^{*\prime} - \sigma^*)$.

Probability. Let $q_{H_i}(i = 1, 2, 3, 4)$, q_U, q_R, q_E and q_S be the number of H_i-Query (i = 1,2,3), UPK-Query, UPK-Replace, SV-Query, Sig-Query respectively. Due to $\mathscr{A}_2$ cannot perform SV-Query for ID_i if T_i or X_i has been replaced, it is an reasonable assumption that $L_E \bigcap L_R = \emptyset$. Several events are defined as follows:

π_1: $\mathscr{C}$ does not fail during the SV-Query.

π_2: $\mathscr{C}$ does not fail during the Sig-Query.

π_3: $ID_j = ID^*$ and $\mu^* = 0$.

It is easy to get following results:

$\Pr[\pi_1] = \frac{q_U - q_E}{q_U}$,

$\Pr[\pi_2|\pi_1] = (\frac{q_U-(q_R+1)}{q_U} + \frac{q_R+1}{q_U} \cdot (1 - \frac{1}{q_R+1}))^{q_S} = (1 - \frac{1}{q_U})^{q_S}$,

TABLE 2. Comparison of several CLAS schemes.

Scheme	Sign	Agg-verify	Execution time/(n=1000)	Signature size/(n=1000)
Cheng [6]	$4nO_M + nO_H$	$3O_P + 2nO_M + (n+1)O_H$	147.594n+131.721/147725.721	(n+1)$\|G_1\|$ / 64064
Deng [8]	$4nO_M + nO_H$	$3O_P + 3nO_M + (n+1)O_H$	160.999n+131.721/161130.721	(n+1)$\|G_1\|$ / 64064
Li [18]	$5nO_M + nO_H$	$3O_P + 2nO_M + (n+1)O_H$	160.999n+131.721/161130.721	(n+1)$\|G_1\|$ / 64064
Nie [19]	$3nO_M + 2nO_H$	$4O_P + 2nO_M + (n+2)O_H$	167.771n+198.016/167969.016	2$\|G_1\|$+2$\|\triangle\|$ / 168
New scheme	$nO_M + nO_H$	$2O_P + 3nO_M + O_H$	87.202n+99.008/87301.008	$\|G_1\|$+$\|\triangle\|$ / 84

$$\Pr[\pi_3|\pi_1 \wedge \pi_2] = \frac{1}{q_U - q_E - q_R} \cdot \frac{1}{q_R + 1}.$$

$$\begin{aligned}
\Pr[\mathscr{C}\ success] &= \Pr[\pi_1 \wedge \pi_2 \wedge \pi_3] \\
&= \Pr[\pi_1] \cdot \Pr[\pi_2|\pi_1] \cdot \Pr[\pi_3|\pi_1 \wedge \pi_2] \\
&= \frac{q_U - q_E}{q_U} \cdot (1 - \frac{1}{q_U})^{q_S} \cdot \frac{1}{q_U - q_E - q_R} \cdot \frac{1}{q_R + 1} \\
&\geq \frac{1}{q_U(q_R+1)} \cdot e^{-\frac{q_S}{q_U}}
\end{aligned}$$

Therefore, if $\mathscr{A}_2$ can forge a aggregate signature with probability ϵ, then $\mathscr{C}$ can solve the CDH problem with probability $\frac{\epsilon}{q_U(q_R+1)} \cdot e^{-\frac{q_S}{q_U}}$.

V. EFFICIENCY

In this section, performance comparisons are made between the new scheme and four CLAS schemes from the last five years. Several notations are defined as follows.

O_P: a pairing operation.

O_M: a scalar multiplication operation in G_1.

O_E: an exponentiation operation in G_2.

O_H: a hash-to-point operation.

$|G_1|$: the size of an element in G_1.

$|\triangle|$: the size of a state information.

Third-party data is used to analyze several CLAS schemes. He *et al.* [16] obtained the time overhead on basic cryptographic operations (Table 2) by using the cryptographic library(MIRACL) and performing the operations on a mobile phone (Samsung Galaxy S5 with the Google Android 4.4.2 operating system, a Quad-core 2.45G processor and 2G bytes memory),

The security level is set to 1024-bit RSA security in the experiments. A Tate pairing $\hat{e} : G_1 \times G_1 \to G_2$ is used, where G_1 is an additive group with q order, which is defined on a super singular curve $E/F_p : y^2 = x^3 + 1$, where the sizes of q and p are 160 bits and 512 bits, respectively.

A simple and intuitive method is adopted to estimate the computation costs. In order to facilitate comparison, it is assumed that there are n signers in an aggregate signature and $n = 1000$. Cheng *et al.*'s scheme [6] requires 3 pairing operations, $6n$ scalar multiplication operation in G_1 and $2n+1$ hash-to-point operations. So the computation time is $32.713 \times 3 + 13.405 \times 6000 + 33.582 \times 2001 = 147725.721$ ms. Deng *et al.*'s scheme [8] requires 3 pairing operations, $7n$ scalar multiplication operation in G_1 and $2n+1$ hash-to-point operations. So the computation time is $32.713 \times 3 + 13.405 \times 7000 + 33.582 \times 2001 = 161130.721$ ms. Li *et al.*'s scheme [18] requires 3 pairing operations, $7n$ scalar multiplication operation in G_1 and $2n+1$ hash-to-point operations. So the computation time is $32.713 \times 3 + 13.405 \times 6000 + 33.582 \times 2001 = 147725.721$ ms. Nie *et al.*'s scheme [19] requires 4 pairing operations, $5n$ scalar multiplication operation in G_1 and $3n+2$ hash-to-point operations. So the computation time is $32.713 \times 4 + 13.405 \times 5000 + 33.582 \times 3002 = 167969.016$ ms. The new scheme [19] requires 2 pairing operations, $4n$ scalar multiplication operation in G_1 and $n+1$ hash-to-point operations. So the computation time is $32.713 \times 2 + 13.405 \times 4000 + 33.582 \times 1001 = 87301.008$ ms.

TABLE 3. Cryptographic operation time (in milliseconds).

O_P	O_M	O_E	O_H
32.713	13.405	2.249	33.582

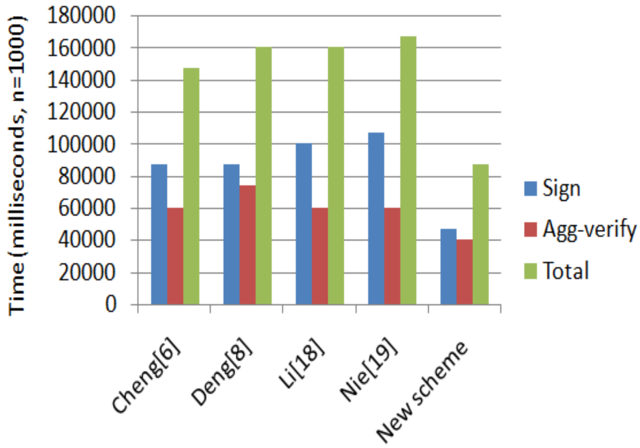

FIGURE 2. Computation costs.

Follow on, the size of signature are computed. In these three schemes [6], [8], [18], each signature contains $n+1$ points in G_1, thus the signature size is $(512 \times 1001)/8 = 64064$ bytes. In the schemes [19], each signature contains 2 points in G_1 and 2 state information, thus the signature size is $(512 \times 2 + 160 \times 2)/8 = 168$ bytes. In the new scheme, each signature contains 1 points in G_1 and 1 state information, thus the signature size is $(512 + 160)/8 = 84$ bytes.

The detailed comparison results of several different CLAS schemes are illustrated in Table 3 (Fig.2).

VI. CONCLUSION

With the widespread use of mobile communication devices, various data are exploding. How to process this data efficiently and safely becomes an urgent problem to be solved. Aggregate signature reduces computation burden and storage burden, which is applied to some actual scenarios, such as electronic trade, electronic monitoring. In most CLAS schemes currently known, the number of hash-to-point operations and the size of signature increase linearly with the number of signers, so they are not suitable for mobile devices. In this paper, a new CLAS scheme is proposed which is unforgeable against the type I/II adversaries in the random oracle model. The scheme requires only 2 pairing operations, the size of signature is one point in G_1 and a state information. They are independent of the number of signers. The scheme reduces computational costs and storage costs, so it is suit for mobile devices.

ACKNOWLEDGMENT

The author is grateful to the anonymous referees for their helpful comments and suggestions.

REFERENCES

[1] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Asiacrypt*. Berlin, Germany: Springer, 2003, pp. 452–473.
[2] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proc. ASIACRYPT*, 2001, pp. 514–532.
[3] D. Boneh, C. Gentry, B. Lynn, and B. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2003, pp. 416–432.
[4] R. Castro and R. Dahab, "Efficient certificateless signatures suitable for aggregation," *Cryptol. ePrint Archive* vol. 6, p. 454, Dec. 2007.
[5] Y. C. Chen, R. Tso, M. Mambo, K. Huang, and G. Horng, "Certificateless aggregate signature with efficient verification," *Secur. Commun. Netw.*, vol. 8, no. 13, pp. 2232–2243, 2015.
[6] L. Cheng, Q. Wen, Z. Jin, H. Zhang, and L. Zhou, "Cryptanalysis and improvement of a certificateless aggregate signature scheme," *Inf. Sci.*, vol. 295, pp. 337–346, Feb. 2015.
[7] K. Y. Choi, J. H. Park, and D. H. Lee, "A new provably secure certificateless short signature scheme," *Comput. Math. Appl.*, vol. 61, no. 7, pp. 1760–1768, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0898122111000897
[8] J. Deng, C. Xu, H. Wu, and L. Dong, "A new certificateless signature with enhanced security and aggregation version," *Concurrency Computatation, Pract. Exper.*, vol. 28, pp. 1124–1133, Mar. 2016.
[9] L. Deng, Y. Yang, R. Gao, and Y. Chen, "Certificateless short signature scheme from pairing in the standard model," *Int J Commun Syst*, vol. 25, 2018, Art. no. e3796.
[10] L. Deng, Y. Yang, Y. Chen, and X. Wang, "Aggregate signature without pairing from certificateless cryptography," *J. Internet Technol.*, vol. 19, no. 5, 1479-1486, Sep. 2018.
[11] H. Du and Q. Wen, "Efficient and provably-secure certificateless short signature scheme from bilinear pairings," *Comput. Standards Inter.*, vol. 31, no. 2, pp. 390–394, 2009.
[12] Z. Gong, Y. Long, X. Hong, and K. Chen, "Two certificateless aggregate signatures from bilinear maps," in *Proc. 8th ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw., Parallel/Distrib. Comput.*, Aug. 2007, pp. 188–193.
[13] X. Huang, Y. Mu, W. Susilo, D. Wong, and W. Wu, "Certificateless signature revisited," in *Proc. 12th Australas. Conf. Inf. Secur. Privacy*, 2007, pp. 308–322.
[14] D. He, B. Huang, and J. Chen, "New certificateless short signature scheme," *IET Inf. Secur.*, vol. 7, no. 2, pp. 113–117, Jun. 2013.
[15] D. He, M. Tian, and J. Chen, "Insecurity of an efficient certificateless aggregate signature with constant pairing computations," *Inf. Sci.*, vol. 268, pp. 458–462, Jun. 2014.
[16] D. He, S. Zeadally, N. Kumar, and W. Wu, "Efficient and anonymous mobile user authentication protocol using self-certified public key cryptography for multi-server architectures," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 9, pp. 2052–2064, Sep. 2016.
[17] P. Kumar, S. Kumari, V. Sharma, A. Sangaiah, J. Wei, and X. Li "A certificateless aggregate signature scheme for healthcare wireless sensor network," *Sustain. Comput. Informat. Syst.*, vol. 18, pp. 80–89, Jun. 2018.
[18] J. Li, H. Yuan, Y. Zhang "Cryptanalysis and improvement of three Certificateless aggregate signature schemes," *Fundamenta Informaticae*, vol. 157, pp. 111–123, Jul. 2018.
[19] N. Nie, Y. Li, W. Chen, Y. A. Ding, "NCLAS: A novel and efficient certificateless aggregate signature scheme," *Secur. Commun. Netw.*, vol. 9, no. 16, pp. 3141–3151, 2016.
[20] K. Shim, "Breaking the short certificateless signature scheme," *Inf. Sci.*, vol. 179, no. 3, pp. 303–306, 2009.
[21] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. Workshop Theory Appl. Cryptograph. Techn.*, 1984, pp. 47–53.
[22] H. Shen, J. Chen, J. Shen, and D. He, "Cryptanalysis of a certificateless aggregate signature scheme with efficient verification," *Secur. Commun. Netw.*, vol. 9, pp. 2217–2221, Sep. 2016.
[23] R. Tso, X. Yi, and X. Huang, "Efficient and short certificateless signatures secure against realistic adversaries," *J. Supercomput.*, vol. 55, no. 2, pp. 173–191, Feb. 2011.
[24] R. Tso, X. Huang, and W. Susilo, "Strongly secure certificateless short signatures," *J. Syst. Softw.*, vol. 85, no. 6, pp. 1409–1417, Jun. 2012.
[25] M. Tian, L. Huang, and W. Yang, "On the security of a certificateless short signature scheme," *Malaysian J. Math. Sci.*, vol. 9, pp. 103–113, Jun. 2015. [Online]. Available: http://www.eprint.iacr.org/2011/419.pdf
[26] J.-L. Tsai, "A new efficient certificateless short signature scheme using bilinear pairings," *IEEE Syst. J.*, vol. 11, no. 4, pp. 2395–2402, Dec. 2015.
[27] H. Xiong, Z. Guan, Z. Chen, and F. Li, "An efficient certificateless aggregate signature with constant pairing computations," *Inf. Sci.*, vol. 219, pp. 225–235, Jan. 2013.
[28] Y. Xie, X. Li, S. Zhang, and Y. Li, "*iCLAS*: An improved certificateless aggregate signature scheme for healthcare wireless sensor networks," *IEEE Access*, vol. 7, pp. 15170–15182, 2019.
[29] L. Zhang, B. Qin, Q. Wu, and F. Zhang, "Efficient many-to-one authentication with certificateless aggregate signatures," *Comput. Netw.*, vol. 54, no. 14, pp. 2482–2491, 2010.
[30] L. Zhang and F. Zhang, "A new certificateless aggregate signature scheme," *Comput. Commun.*, vol. 32, no. 6, pp. 1079–1085, 2009.
[31] F. Zhang, L. Shen, and G. Wu, "Notes on the security of certificateless aggregate signature schemes," *Inf. Sci.*. vol. 287, pp. 32–37, Dec. 2014.

LUNZHI DENG received the B.S. and M.S. degrees from Guizhou Normal University, Guiyang, China, in 2002 and 2008, respectively, and the Ph.D. degree from Xiamen University, Xiamen, China, in 2012. He is currently a Professor with the School of Mathematical Sciences, Guizhou Normal University. His current research interests include algebra and information safety.

YIXIAN YANG is currently a Professor with the Guizhou Provincial Key Laboratory of Public Big Data, China. He has been published more than 300 papers in the IEEE Transactions on Aerospace and Electronic Systems, the IEEE Transactions on Communications, the IEEE Transactions on Electromagnetic Compatibility, *Discrete Applied Mathematics*, and other international most authoritative academic journals. He is also a member of the China Science and Technology Commission of the Ministry of Education.

YULING CHEN received the B.S. degree from Taishan University, Tai'an, China, in 2006, and the M.S. degree from Guizhou University, Guiyang, China, in 2009, where she is currently an Associate Professor with the Guizhou Provincial Key Laboratory of Public Big Data. Her current research interests include cryptography and information safety.