A Batch-Authenticated and Key Agreement Framework for P2P-Based Online Social Networks

Lo-Yao Yeh, *Member, IEEE*, Yu-Lun Huang, *Member, IEEE*, Anthony D. Joseph, *Member, IEEE*, Shiuhpyng Winston Shieh, *Senior Member, IEEE*, and Woei-Jiunn Tsaur, *Member, IEEE*

***Abstract*—Online social networks (OSNs) such as Facebook and MySpace are flourishing because more and more people are using OSNs to share their interests with friends. Because security and privacy issues on OSNs are major concerns, we propose a security framework for simultaneously authenticating multiple users to improve the efficiency and security of peer-to-peer (P2P)-based OSNs. In the proposed framework, three batch authentication protocols are proposed, adopting the one-way hash function, ElGamal proxy encryption, and certificates as the underlying cryptosystems. The hash-based authentication protocol requires lower computational cost and is suitable for resource-limited devices. The proxy-based protocol is based on asymmetric encryption and can be used to exchange more information among users. The certificate-based protocol guarantees nonrepudiation of transactions by signatures. Without a centralized authentication server, the proposed framework can therefore facilitate the extension of an OSN with batched verifications. In this paper, we formally prove that the proposed batch authentication protocols are secure against both passive adversaries and impersonator attacks, can offer implicit key authentication, and require fewer messages to authenticate multiple users. We also show that our protocols can meet important security requirements, including mutual authentication, reputation, community authenticity, nonrepudiation, and flexibility. With these effective security features, our framework is appropriate for use in P2P-based OSNs.**

***Index Terms*—Authentication protocol, batch authentication, Online social networks (OSNs), Peer to peer (P2P).**

Manuscript received May 20, 2011; revised October 14, 2011 and December 5, 2011; accepted February 2, 2012. Date of publication February 23, 2012; date of current version May 9, 2012. This work was supported in part by the Team for Research in Ubiquitous Secure Technology Center, University of California, Berkeley, the National Science Council under Grant NSC 100-2218-E-492-026 and Grant NSC 100-2219-E-212-001, the Networked Communications Program, the Industrial Technology Research Institute, the Institute for Information Industry, the Chung Shan Institute of Science and Technology, Chunghwa Telecomm., Bureau of Investigation, D-Link, the International Collaboration for Advancing Security Technology, and the Taiwan Information Security Center, Ministry of Education, Taiwan. The review of this paper was coordinated by Prof. J. Misic.

L.-Y. Yeh is with the Network and Information Security Division, National Center for High-Performance Computing, Hsinchu 30076, Taiwan, and also with the Department of Information Management, National Chi Nan University, Nantou 545, Taiwan (e-mail: lyyeh@nchc.narl.org.tw).

Y.-L. Huang is with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan.

A. D. Joseph is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA.

S. W. Shieh is with Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan.

W.-J. Tsaur (Corresponding author) is with the Department of Information Management, Da-Yeh University, Changhua 51591, Taiwan (e-mail: wjtsaur@mail.dyu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2012.2188821

I. Introduction

ONLINE social networks (OSNs) such as Facebook, MySpace, and Twitter are increasingly popular services. People can share information and pictures with old acquaintances, as well as relationships with friends. It is estimated that half a billion registered users interact with friends over OSNs [1], [2]. However, the weak authentication and registration process of current OSNs may lead to some security attacks [3]. With the rapid growth of OSNs, more valuable information is stored on OSNs. Hence, the privacy and security issues inherent to OSNs have attracted much attention.

Peer-to-peer (P2P) technology is considered in the design of next-generation OSNs. As described in [6], a P2P-based OSN consists of the following three levels: 1) the social network level represents members and their relationships; 2) the application service level implements the P2P-based application infrastructure; and 3) the communication and transport level provides transport services over networks such as the Internet or mobile ad hoc networks. Relying on the cooperation between a number of independent parties who are also OSN users, a decentralized P2P architecture can be adopted with merits, including strong privacy protection, better scalability, and a lowered requirement for continuous Internet connectivity [1], [4]. Furthermore, a P2P architecture can take advantage of real social networks and geographic proximity to support local services.[1]

P2P-based OSNs is a relatively new trend. Only a few studies [3], [5] have designed their security protocols based on the P2P architecture. In 2009, Buchegger *et al.* [5] examined the feasibility of P2P-based OSNs and advocated the use of encryption techniques to ensure privacy issues. In [6], the authors described a decentralized and privacy-preserving OSN application called Safebook, but no specific protocol was defined. Ge *et al.* [3] proposed a message delivery scheme (VisualSec) based on a fully distributed method that attempts to provide authentication and key generation services. However, identity authentication was not fully addressed, and only the out-of-band (OOB) authentication method is mentioned in VisualSec. Another alternative is to integrate the existing authentication protocols from P2P networks into the P2P-based OSNs. In 2003, Shardul *et al.* [8] divided P2P users into several groups (called *troupes*). Taking zero knowledge and CLIQUES as bases, each troupe required a central computing authority for authentication. Lee and Kim [9] also proposed an adaptive

[1]This paper focuses only on the security issues of P2P-based OSNs and not other challenging problems of P2P networks such as global search and content distribution [7].

authentication protocol based on the reputation of P2P systems. The reputation of each peer affects the type of certificates to be used in authenticating a user. In 2006, Nguyen [10] adopted identity-based cryptography and preshare keys to design a simplified P2P device authentication protocol. However, these existing protocols suffer from the following weaknesses.

1) Most of the current security protocols [1], [5], [6] for P2P-based OSNs lack specific procedures.
2) In current security protocols for P2P-based OSNs, each user has to be authenticated by OOB methods, which may impede the extension speed of social networks.
3) Most of the existing protocols support only one-to-one authentication.
4) The existing protocols do not consider the restrictions of underlying devices such as computing power and memory limitations.

This paper proposes a framework to take advantage of the P2P architecture, including geographic proximity. Under the proposed framework, three batch authentication protocols are designed, using different cryptographic primitives, for different devices in P2P-based OSNs. The novel contributions of this paper are listed as follows.

- The proposed framework reduces the communication cost required for authenticating users.
- Due to their different security properties, the proposed protocols can be realized on a variety of devices such as personal digital assistants (PDAs), mobile phones, and laptops.
- By incorporating different trust levels, the proposed protocols allow a user with a high trust level to help authenticate other users and achieve the extensibility of a social network.
- The proposed protocols support a one-to-many authentication, which is the basis of batch authentication, to simultaneously authenticate multiple users. To the best of our knowledge, this paper is the first study that offers one-to-many batch authentication in P2P-based OSNs.

The proposed protocols are proved to be capable of mutually authenticating communication peers and remain secure against passive adversaries.

This paper is organized as follows. We briefly introduce the cryptographic background in Section II. Section III presents the assumptions and notations used in the proposed protocols. The details of the proposed protocols are described in Section IV. Then, we give the proof of security analysis and a performance comparison in Sections V-B and VI, respectively. Finally, this paper is concluded in Section VII.

II. ElGamal Proxy Encryption

By applying proxy encryption, a ciphertext C that is encrypted by one user is transformed to $\hat{C}$, which is decrypted by another user. In 2007, Huang *et al.* [12] realized proxy encryption using an ElGamal cryptosystem. In the ElGamal cryptosystem, two public parameters p and q are shared by all users, where p is a prime that is equal to $2q+1$, and g is the generator in $\mathbb{Z}_p^*$.

Then, a sender U_A randomly chooses a secret key x in $\mathbb{Z}_p^*$ and makes $\beta = g^x$ as his/her public key. U_A generates two cipher values (C_1, C_2), where $C_1 = g^r \bmod p$, $C_{2_A} = \xi(\beta^r) = \xi((g^x)^r) \bmod p$, $r \in \mathbb{Z}_q^*$ is a randomly selected number, and ξ represents the message. Assuming that the receiver U_B possesses his/her secret key y and the proxy U_P holds the transformation key $(y-x)$, the sender U_A sends the cipher value (C_1, C_{2_A}) to U_P, and the proxy U_P implements the following two tasks:

- transforms the cipher value C_{2_A} into $C_{2_P} = C_{2_A} \cdot C_1^{(y-x)}$;
- sends (C_1, C_{2_P}) to the receiver U_B.

Upon receipt of (C_1, C_{2_P}), U_B decrypts the cipher with his/her secret key y and derives the message ξ.

The proxy U_P not only alters the cipher (C_1, C_{2_A}) but also distributes a tailor-made component to U_B. When adopting such an encryption method to our protocols, a minor modification is required for authenticating multiple users in one verification procedure. In our protocols, the proxy distributes the tailor-made parameters to each peer, and the authentication requester relies on these parameters to decrypt the cipher, cooperatively made by authenticating peers. If the message contents are received and verified, these peers are considered authenticated.

III. Batch Authentication Framework

A. Background

In P2P-based OSNs, each user may act as a client or a server. Based on the uniqueness of such characteristics, we propose three novel protocols for authenticating P2P-based OSN users in batch. In the proposed protocols, a user can simultaneously authenticate several users through a trusted friend.

With regard to the authentication of the trusted friend, similar to [1], [5], and [6], the OOB method is assumed in our protocols. For example, a face-to-face preauthentication method [11], [13] through a location-limited channel can be used for negotiating a shared secret key between two friends. Once a user (e.g., U_X) is authenticated by another user (U_R), U_X becomes a friend of U_R and further helps U_R to scale up his/her social network. Note that the proposed protocols are different from other group key management protocols [15] due to the following reasons.

1) *Different goals*. In conventional group key management protocols, a group is formed for a temporary purpose. The group is dismissed when that temporary purpose is served, and then, the group members lose relationships with other group members. Our social-based batch authentication protocols are designed for authenticating friends in OSNs. Once a group member is authenticated, he/she can help friends for another batch authentication. Such authentication protocols help extend the social network of a user.
2) *Different behaviors*. In most group key management protocols, group members are authenticated by the group leader "one by one." That is, n authentication messages are required to authenticate n group members. Then, these members share *one* common group key for the group communication. In our batch authentication

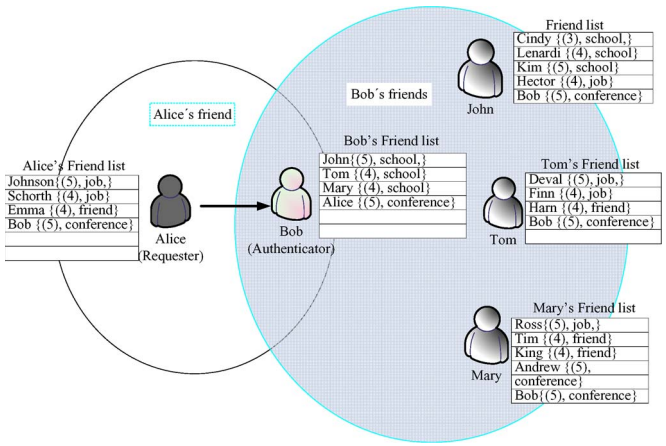


Fig. 1. Scenario of a batch authentication protocol.

protocols, users are simultaneously authenticated by the requester. That is, *one* authentication message is required to authenticate n session peers. Then, the requester negotiates one secret key with each user instead of sharing one group key among all users.

In Fig. 1, for example, assume that Alice and Bob have performed a face-to-face preauthentication through a location-limited channel (e.g., in a conference) in advance. They exchanged a secret shared key and shared friends lists, trust levels, and social relationships with each other after the preauthentication.

If Alice (who acts as the U_R) wants to communicate with users John, Tom, and Mary from Bob's friends list, she can ask Bob (who behaves as the U_A) to help the authentication. After successful authentication, Alice can exchange shared keys with John, Tom, and Mary and can add them into her friends list to broaden her social network.

B. Requirements and Notations

To support devices with different computing power, we propose three batch authentication protocols that adopt hash function, proxy encryption, and certification [16] as their underlying cryptography methods. The hash-based batch authentication protocol adopts a lightweight hash function and is suitable for resource-limited devices such as PDAs and cell phones. For devices with stronger computing power, the proxy-based batch authentication protocol can be applied to ensure confidentiality. The certificate-based batch authentication protocol can be used to protect sensitive and nonrepudiation transactions. These batch authentication protocols are explained in Section IV.

Considering the features of OSNs [1], [5], [6] and P2P architecture [9], we summarize the requirements of an authentication protocol for P2P-based OSNs as follows.

- R1: Strong authentication. Mutual authentication should be guaranteed to guard against possible attacks.
- R2: Reputation. Reputation ensures accountability in P2P-based OSNs.
- R3: Community authenticity. Each user should efficiently verify the source of messages. For example, a session key can be established with the use of keyed message authentication code.
- R4: Nonrepudiation. Nonrepudiation must be guaranteed so that dispute for an invalid commercial transaction can be settled by a court.
- R5: Flexibility. An authentication protocol should be adopted into any OSN, either web- or P2P-based networks.
- R6: Batch authentication. Because a user may be far away from other users in a P2P network, authenticating multiple users in one transaction may reduce the communication cost.
- R7: Comprehensive hardware support. An authentication protocol should support a variety of devices with different computing powers.

The notations used throughout this paper are listed in Table I.

C. Sketch of the Proposed Batch Authentication

The proposed batch authentication protocols, which are composed of three roles, a requester U_R, an authenticator U_A, and a user group $\hat{U}$, are operated based on the following assumptions.

1) The requester U_R and authenticator U_A have negotiated a shared key by face-to-face preauthentication through a location-limited channel [11], [13].
2) The authenticator U_A is trusted by all his/her friends who are involved in the batch authentication.
3) If two users U_X and U_Y are friends, they have shared a secret key K_{XY}.

In the proposed protocol, U_A helps U_R authenticate the user group $\hat{U}$, in which all users are friends of U_A. After successful authentication, U_R establishes a shared key K_{Ri} with each user U_i in the group ($U_i \in \hat{U}$). We briefly explain our design concept by the following two cases. The detailed explanations of each message are introduced in Section IV.

In the first case, we introduce a user group with only one user U_1 ($\hat{U} = \{U_1\}$), as shown in Fig. 2(a). The message flow is given as follows.

1) $U_R \rightarrow U_A : AQ_{R,A}$.
2) $U_A \rightarrow U_1 : CR_{A,1}$.
3) $U_1 \rightarrow U_R : CR_{1,R}$.
4) $U_R \rightarrow U_1 : MR_{R,1}$.

The requester U_R initiates a request to the authenticator U_A. Then, U_A helps contribute some parameters to U_R and U_1 at Steps 2 and 3. Finally, U_R replies a message ($MR_{R,1}$) at Step 4 for mutual authentication.

The second case scales up the user group to n users ($\hat{U} = \{U_1, U_2, \ldots, U_n\}$, and $|\hat{U}| = n$),[2] as shown in Fig. 2(b). The message flow is given as follows.

1) $U_R \rightarrow U_A : AQ_{R,A}$.
2) $U_A \rightarrow U_1 : CR_{A,1}$.
3) $U_{i-1} \rightarrow U_i : CR_{i-1,i}$, where $2 \leq i \leq |\hat{U}|$.
4) $U_{|\hat{U}|} \rightarrow U_R : CR_{|\hat{U}|,R}$.
5) $U_R \rightarrow U_i : MR_{R,i}$, where $1 \leq i \leq |\hat{U}|$.

[2]We assume that the n users are online in the course of the batch verification or the batch authentication will fail. Note that the assumption can hold if U_A can inform the users in $\hat{U}$ before the batch verification is executed.

TABLE I
PARAMETERS AND NOTATIONS

Symbol	Description
q, p	Large primes such that $p = 2q + 1$.
g	The primitive root of prime q.
RK_i	The private key of U_i.
PK_i	The public key of U_i. PK_i is used for ElGamal encryption such that $PK_i = g^{RK_i} \bmod p$.
$E_{K_{RA}}$	The symmetric encryption function with secret key K_{RA}.
B_n	Representing n positive integers that are pairwise relatively primes used in CRT.
$H()$	Collision-resistant one-way hash functions with length of l, $H() : \{0,1\}^* \rightarrow \{0,1\}^l$
$H(r, msg)$	A length extension hash function with r-times hash operations for message msg. For instance, $H(r, msg) = H(msg) \parallel H^2(msg) ... \parallel H^r(msg)$, where r is determined by the required lengths.
MAC	The message authentication code generated by a keyed hash function.
Requester (U_R)	A user who requests batch authentication.
Authenticator (U_A)	A user who assists U_R for the batch authentication.
$\hat{G}$	The set of all participants involved in the batch authentication. $\hat{G}$={U_R, U_A, U_1, U_2, …, U_n}
$\lvert\hat{G}\rvert$	The number of all participants involved in the batch authentication
$\hat{U}$	A user group to be authenticated $\hat{U}$=$\hat{G} - \{U_A, U_R\}$={U_1, U_2, …, U_n }
$\lvert\hat{U}\rvert$	The number of user group.
UID	The set of $\hat{U}$'s identities in this batch authentication session $UID = \{ID_1, ID_2, \ldots, ID_n\}$
N_i	A nonce picked by U_i.
w_i, t_i	The random numbers that have the same bit lengths as $H()$.
S	A random number serving as a seed of ElGamal proxy encryption key.
T_i	The U_i's certificate.
$\hat{T}$	The set of $\hat{U}$'s certificates $\hat{T}$={$T_1, T_2, \ldots, T_n$}
KP_R	The set of key parameters sent from U_R to $\hat{U}$ for key agreement. $KP_R = \{g^{m_1}, g^{m_2}, \ldots, g^{m_n}\}$.
$KP_{\hat{U}}$	The set of key parameters sent from $\hat{U}$ to U_R. $KP_{\hat{U}} = \{g^{n_1}, g^{n_2}, \ldots, g^{n_n}\}$.
$QR_{R,A}$	The authentication request message transmitted from U_R to U_A.
CR	The chain-reply messages passed through users in a user group.
MR	The reply messages for mutual authentication.

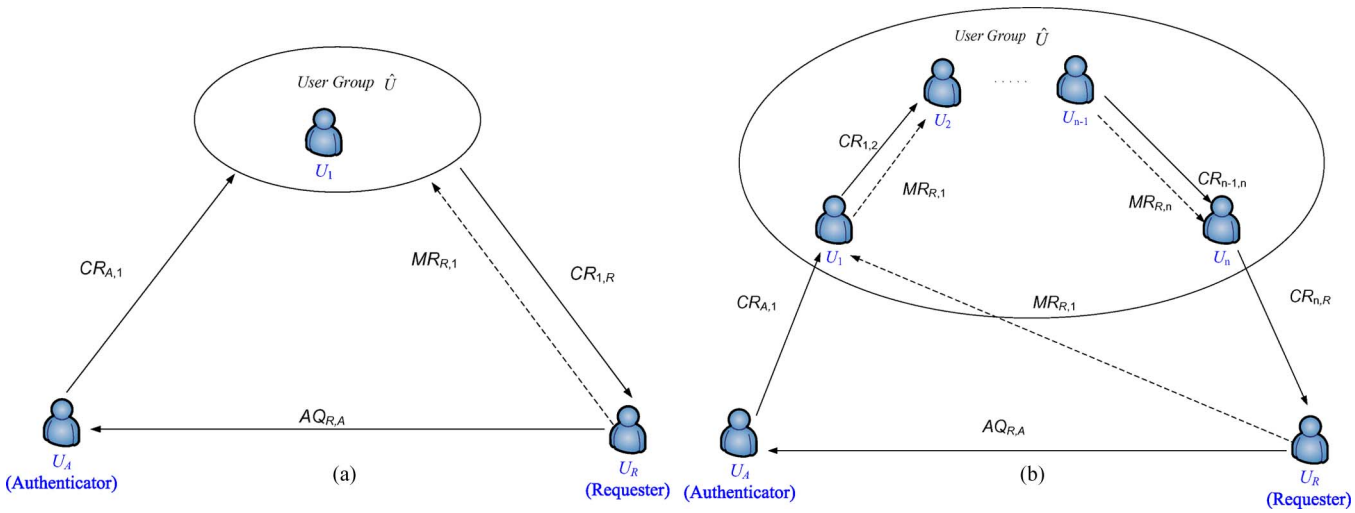


Fig. 2. Message flows of batch authentication for (a) only one member and (b) several members in case n = 3.

Similarly, U_R sends a request to U_A. The authentication requests (chain reply $CR_{i,i+1}$) are then passed through $U_1, U_2, \ldots$ to U_n at Steps 2 and 3*. At Step 4, $U_{|\hat{U}|}$ sends back the chain reply to U_R. For mutual authentication, U_R sends $MR_{R,i}$ to users $U_i \in \hat{U}$.

IV. PROPOSED PROTOCOLS

This section explains the proposed batch protocols with different underlying cryptographic functions, including hash, proxy encryption, and certificates. Considering the security and performance issues, we recommend the proxy-based protocol as the default protocol in general situations.

A. Hash-Based Protocol

The hash-based batch authentication protocol, with the following message flow, is illustrated in Fig. 3.

1) $U_R \rightarrow U_A : AQ_{R,A} = \{ID_R, K_{RA} \oplus N_R, UID \oplus H(r, (N_R + 1)), KP_R \oplus H(r, (N_R + 2)), MAC_R\}$.
2) $U_A \rightarrow U_1 : CR_{A,1} = \{X, MAC_A\}$.
3) $U_{i-1} \rightarrow U_i : CR_{i-1,i} = \{M_i, X, KP_{\hat{U}}, MAC_i\}$, where $2 \leq i \leq |\hat{U}|$.
4) $U_{|\hat{U}|} \rightarrow U_R : CR_{|\hat{U}|,R} = \{M_{|\hat{U}|}, X, KP_{\hat{U}}, MAC_{|\hat{U}|}\}$.
5) $U_R \rightarrow U_i : MR_{R,i} = \{Y, MAC'_R\}$, where $1 \leq i \leq |\hat{U}|$.

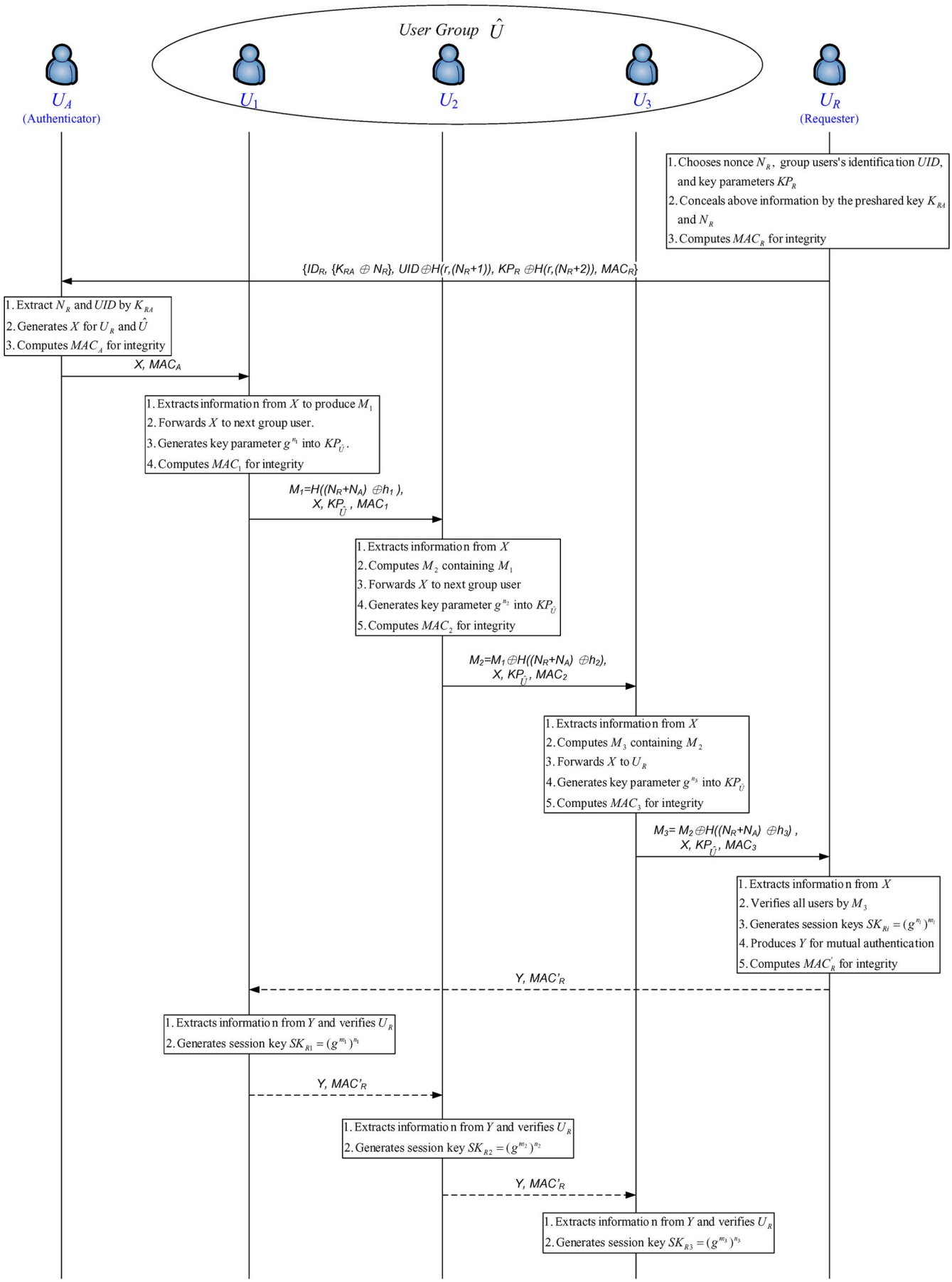


Fig. 3. Message flows of the hash-based batch authentication protocol. The illustration shows an example of a user group of size 3 ($|\hat{U}| = 3$).

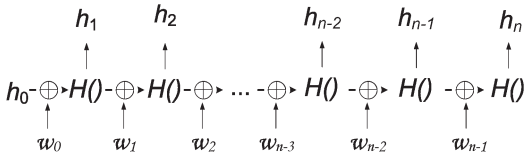


Fig. 4. Proposed one-way hash chain.

The hash-based batch authentication protocol is detailed as follows.

Step 1) U_R sends $AQ_{R,A}$ to U_A. $AQ_{R,A}$ is composed of U_R's identification (ID_R), a nonce (N_R), the user group identification $(UID = \{ID_1, ID_2, \ldots, ID_{|\hat{U}|}\})$, and the parameters of key agreement $(KP_R = \{g^{m_1}, g^{m_2}, \ldots, g^{m_{|\hat{U}|}}\}$, where $m_i \in Z_p^*)$. The nonce is protected by a secret key K_{RA} that is shared by U_R and U_A. The group identification and key parameters are protected by the nonce. In addition, a message authentication code $MAC_R = H(ID_R, \{K_{RA} \oplus N_R\}, UID \oplus H(r, (N_R+1)), KP_R \oplus H(r, (N_R+2)), (N_R+3))$ is attended to ensure the integrity of message.

Step 2) Upon the receipt of $AQ_{R,A}$, U_A derives N_R by performing $K_{RA} \oplus \{K_{RA} \oplus N_R\}$ and checks the validity of MAC_R. If $AQ_{R,A}$ is correct, the following steps are implemented.

- U_A randomly generates an initial value h_0 and a sequence of random numbers w_i (for $0 \le i \le |\hat{U}| - 1$). Then, U_A constructs and maintains a chain of one-way hash values ($h_i = H(h_{i-1} \oplus w_{i-1})$ for $1 \le i \le |\hat{U}|$), as shown in Fig. 4.
- U_A derives the user group identification UID and the key parameters KP_R by N_R.
- U_A computes V_0 for U_R, where

$$V_0 = \Bigg\{ ID_A, H\left(r, (K_{RA} \| t_0)\right) \oplus \left(h_0 \| H(K_{RA}) \| N_A \| \bigcup_{j=0}^{|\hat{U}|-1} w_j \right), t_0 \Bigg\}.$$

Note that the unequal-bit-length problem can be solved by the specific length extension hash function $H(r, msg)$ and V_0 should be regarded as a single element from the view of calculation. As mentioned in the previous section, K_{RA} is the shared key between U_R and U_A, N_A and t_0 are random challenges from U_A, and $\cup_{j=0}^{|\hat{U}|-1} w_j$ is a concatenation of $w_0, w_1, \ldots, w_{|\hat{U}|-1}$.

- U_A also computes V_i for $U_i \in U$, $(1 \le i \le |\hat{U}|)$, where

$$V_i = \{ID_A, H\left(r, (K_{Ai} \| t_i)\right) \oplus \left(h_i \| H(K_{Ai}) \| N_R + N_A \| g^{m_i}\right), t_i\}.$$

In V_i $(i \neq 0)$, g^{m_i} is used for negotiating session keys K_{Ri} between U_R and U_i in the end of the batch authentication.

- To eliminate the bandwidth requirements, we adopt the Chinese reminder theory (CRT) [17] to accommodate messages in a single message. Let $B_0, B_1, B_2, \ldots, B_{|\hat{U}|}$ be $|\hat{U}| + 1$ positive integers that are pairwise relative primes and $A_0, A_1, A_2, \ldots, A_{|\hat{U}|}$ be the multiplicative inverses of $B_0, B_1, B_2, \ldots, B_{|\hat{U}|}$. U_A computes a common solution X for the following congruous equations:

$$X \equiv V_0 \bmod B_0 (\text{for } U_R)$$
$$X \equiv V_i \bmod B_i \left(\text{for } U_i \in U,\ 1 \le i \le |\hat{U}|\right).$$

By the CRT, we have $X = (\sum_{i=0}^{|\hat{U}|} L/B_i \times V_i \times A_i) \bmod L$, where

$$L = \prod_{i=0}^{|\hat{U}|} B_i$$
$$A_i \times (L/B_i) \bmod B_i \equiv 1.$$

- U_A calculates $MAC_A = H(X, N_R + N_A)$ and sends the chain reply $CR_{A,1} = \{X, MAC_A\}$ to the first user in the group (U_1).

Step 3) After receiving $CR_{A,1} = \{X, MAC_A\}$, the following steps are implemented.

- U_1 retrieves V_1 by calculating $X \bmod B_1$. Next, U_1 obtains $H(r, (K_{A1} \| t_1)) \oplus (h_1 \| H(K_{A1}) \| N_R + N_A \| g^{m_1})$ and t_1 from V_1. U_1 then uses the shared keys K_{A1} and t_1 to derive h_i, $H(K_{A1})$, $N_R + N_A$, and g^{m_1}.
- The validity of V_1 and $CR_{A,1}$ can be verified by $H(K_{A1})$ and MAC_A, respectively. The request is dropped when any invalidity is detected. Then, U_1 computes $M_1 = H((N_R + N_A) \oplus h_1)$ and adds a key parameter g^{n_1} to $KP_{\hat{U}}$.
- U_1 generates $MAC_1 = H(M_1, X, KP_{\hat{U}}, (N_R + N_A))$ and sends message $CR_{1,2} = \{M_1, X, KP_{\hat{U}}, MAC_1\}$ to the next group user U_2.

For $U_i \in U (2 < i \le |\hat{U}|)$, the following steps repeat until the chain reply passes through all group users.

- U_i gets $CR_{i-1,i} = \{M_{i-1}, X, KP_{\hat{U}}, MAC_{i-1}\}$ from U_{i-1}. U_i extracts V_i by $X \bmod B_i$. Similarly, U_i can obtain h_i, $H(K_{Ai})$, $N_R + N_A$, g^{m_i} from V_i by the shared key K_{Ai} and random challenge t_i.
- The validity of V_i and $CR_{i-1,i}$ can be verified by $H(K_{Ai})$ and MAC_{i-1}, respectively. When any invalidity is detected, the request is dropped, and U_i reports the failure to U_A. Then, U_i computes $M_i = M_{i-1} \oplus H((N_R + N_A) \oplus h_i)$ and adds a key parameter g^{n_i} to $KP_{\hat{U}}$.
- U_i generates $MAC_i = H(M_i, X, KP_{\hat{U}}, (N_R + N_A))$ and sends $CR_{i,i+1} = \{M_i, X, KP_{\hat{U}}, MAC_i\}$ to the next group user U_{i+1}.

Step 4) Upon the receipt of the last chain reply $CR_{|\hat{U}|,R} = \{M_{|\hat{U}|}, X, KP_{\hat{U}}, MAC_{|\hat{U}|}\}$, the following steps are implemented.
- U_R computes X and B_0 and obtains V_0. With the shared key K_{RA} and random challenge t_0, U_R derives h_0, $H(K_{RA})$, N_A, and $\cup_{j=0}^{|\hat{U}|-1} w_j$ from V_0.
- Similarly, the authenticity of V_0 and $MAC_{|\hat{U}|}$ can be verified by $H(K_{RA})$ and $MAC_{|\hat{U}|}$. If validated, U_R derives h_i $(1 \le i \le |\hat{U}|)$ by h_0 and $\cup_{j=0}^{|\hat{U}|-1} w_j$.
- U_R also computes $M'_{|\hat{U}|} = H((N_R + N_A) \oplus h_1) \oplus H((N_R + N_A) \oplus h_2) \oplus \cdots \oplus H((N_R + N_A) \oplus h_{|\hat{U}|})$ and compares it with $M_{|\hat{U}|}$. If matched, the user group $\hat{U}$ is authenticated. Otherwise, at lease one of the users fails the authentication, and the session terminates.
- After the successful batch authentication, U_R computes session keys $SK_{Ri} = (g^{n_i})^{m_i}$ for U_i $(1 \le i \le |\hat{U}|)$.
- For mutual authentication, U_R calculates replies $S_i = H((N_R + N_A + 1) \oplus h_i) \bmod B_i$. Again, by applying the CRT [17], we can find a common solution for

$$\begin{aligned} Y &\equiv S_1 \bmod B_1 \\ Y &\equiv S_2 \bmod B_2 \\ &\vdots \\ Y &\equiv S_{|\hat{U}|} \bmod B_{|\hat{U}|}. \end{aligned}$$

Then, U_R generates $MAC'_R = H(Y, (N_R + N_A))$ and sends $MR_{R,i} = \{Y, MAC'_R\}$ to U_i $(1 \le i \le |\hat{U}|)$. In the case that U_R cannot directly reach U_i, U_A can be involved to help forward the messages.

Step 5) After receiving $MR_{R,i}$ from U_R (or U_{i-1}), the following steps are implemented.
- U_i first checks the validity of MAC'_R.
- The session is dropped if MAC'_R fails the check; otherwise, U_i computes $S_i = Y \bmod B_i$ and checks the equality of S'_i, where $S'_i = H((N_R + N_A + 1) \oplus h_i)$ $(1 \le i \le |\hat{U}|)$. If the equality holds, U_R is authenticated; otherwise the session is terminated.
- After the successful batch authentication, U_i computes the session key $SK_{Ri} = (g^{m_i})^{n_i}$. Subsequent communications between U_R and U_i can be protected by SK_{Ri}.

B. Proxy-Based Protocol

By using the ElGamal proxy encryption scheme [12], the proxy-based protocol can carry trust levels [9] for entities that are involved in the batch authentication. In this protocol, we set two public parameters, p and g, where p is a prime of the form $2q + 1$, and g is a generator in $\mathbb{Z}_p^*$. Similar to the hash-based protocol, the proxy-based batch authentication protocol comprises the following five messages, as illustrated in Fig. 5.

1) $U_R \to U_A : AQ_{R,A} = \{ID_R, \{C1, C2_R\}, K_{RA} \oplus N_R, UID \oplus H(r, (N_R + 1)), MAC_R\}$.
2) $U_A \to U_1 : CR_{A,1} = \{C1, C2_A, X, MAC_A\}$.
3) $U_{i-1} \to U_i : CR_{i-1,i} = \{C1, C2_i, X, KP_{\hat{U}}, MAC_i\}$, where $2 \le i \le |\hat{U}|$.
4) $U_{|\hat{U}|} \to U_R : CR_{|\hat{U}|,R} = \{C1, C2_{|\hat{U}|}, X, KP_{\hat{U}}, MAC_{|\hat{U}|}\}$.
5) $U_R \to U_i : MR_{R,i} = \{C2', MAC'_R\}$, where $1 \le i \le |\hat{U}|$.

Step 1) The requester U_R sets the shared key K_{RA} as the seed of the ElGamal proxy encryption key and then starts the batch authentication protocol as follows.
- U_R sends the authentication request $AQ_{R,A} = \{ID_R, \{C1, C2_R\}, K_{RA} \oplus N_R, UID \oplus H(r, (N_R + 1)), MAC_R\}$ to U_A.

Step 2) Upon the receipt of $AQ_{R,A}$, the following steps are implemented.
- U_A first derives N_R by the shared key K_{RA} and extracts the UID by N_R.
- Next, U_A verifies MAC_R and checks whether each U_i's trust level that is maintained by himself is higher than the predefined trust threshold. If one of the verifications fail, U_A drops this session. Otherwise, U_A computes V_0 for U_R and V_i for $U_i \in U$ as

$$V_0 = \left\{ ID_A, E_{K_{RA}} \left(N_A, \sum_{j=1}^{|\hat{U}|} K_{Aj}, H(K_{RA}) \right) \right\}$$

$$V_i = \left\{ ID_A, E_{K_{Ai}} \left(K_{RA} + N_R + N_A + \sum_{\substack{j=1 \\ j \ne i}}^{|\hat{U}|} K_{Aj}, H(K_{Ai}) \right) \right\}.$$

- Similarly, by applying the CRT [17], we can accommodate all replies in a single message as

$$\begin{aligned} X &\equiv V_0 \bmod B_0 \text{ (for } U_R) \\ X &\equiv V_1 \bmod B_1 \text{(for } U_1) \\ &\vdots \\ X &\equiv V_i \bmod B_i \text{(for } U_i \in U). \end{aligned}$$

As mentioned in Section IV-A, by the CRT, we obtain $X = (\sum_{i=0}^{|\hat{U}|} L/B_i * V_i * A_i) \bmod L$.
- Based on the ElGamal proxy encryption scheme, U_A calculates

$$\begin{aligned} C2_A &= (C2_R \times C1^{N_R}) \bmod p \\ &= \left(\xi \beta^r \times g^{r(N_R)} \right) \bmod p \\ &= \left(\xi g^{(K_{RA})r} \times g^{r(N_R)} \right) \bmod p \\ &= \left(\xi g^{r(K_{RA} + N_R)} \right) \bmod p \end{aligned}.$$

  - U_A generates the message authentication code to protect the integrity of the message, where $MAC_A = H(C1, C2_A, X, (K_{RA} + N_R + N_A + \sum_{j=1}^{|\hat{U}|} K_{Aj}))$.

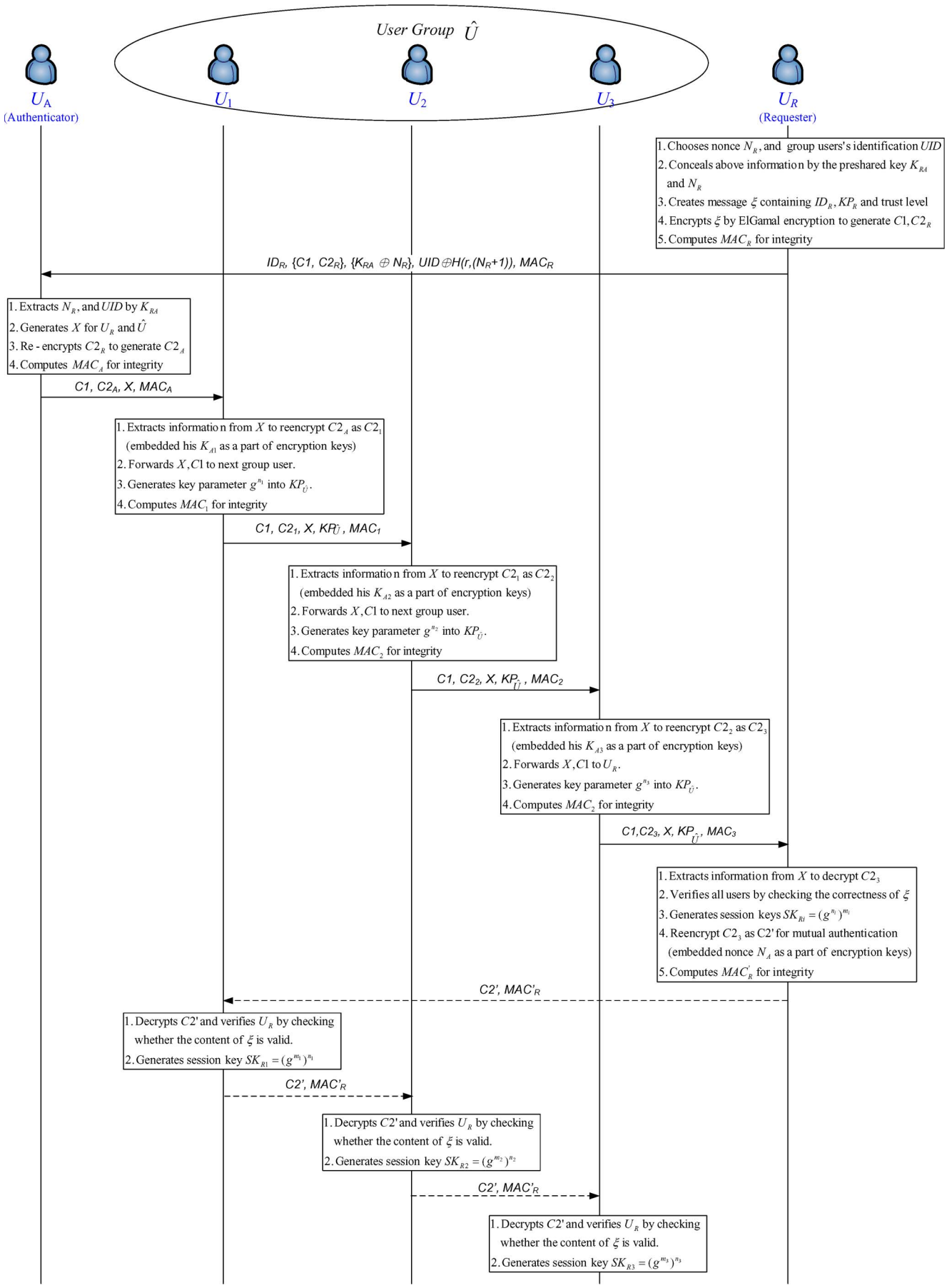


Fig. 5. Message flow of the proxy-based batch authentication protocol. The illustration shows an example of a user group of size 3 ($|\hat{U}| = 3$).

- U_A sends $CR_{A,1} = \{C1, C2_A, X, MAC_A\}$ to U_1.

Step 3) After receiving $CR_{A,1}$, the following steps are implemented.

- U_1 extracts $V_1 = X \bmod B_1$ and decrypts $E_{K_{A_i}}(K_{RA} + N_R + N_A + \sum_{j=2}^{|\hat{U}|} K_{Aj}, H(K_{A1}))$ by K_{A1}.
- U_1 verifies the integrity of V_1 and $CR_{A,1}$ by checking $H(K_{A1})$ and MAC_A, respectively. The request is dropped when any invalidity is detected.
- U_1 calculates

$$\begin{aligned} C2_1 &= \left(C2_A \times C1^{(K_{A1})}\right) \bmod p \\ &= \left(\xi g^{r(K_{RA}+N_R)} \times g^{r(K_{A1})}\right) \bmod p. \\ &= \left(\xi g^{r(K_{RA}+N_R+K_{A1})}\right) \bmod p \end{aligned}$$

- U_1 selects the parameter of the session key $KP_{\hat{U}} = \{g^{n_1}\}$.
- Because K_{A1} is shared with U_A and U_1, only legitimate U_1 can decrypt V_1, add K_{A1} with $K_{RA}+N_R+N_A+\sum_{j=2}^{|\hat{U}|} K_{Aj}$, and compute the message authentication code $MAC_1 = H(C1, C2_1, X, KP_{\hat{U}}, K_{RA} + N_R + N_A + \sum_{j=1}^{|\hat{U}|} K_{Aj})$.
- U_1 sends $CR_{1,2} = \{C1, C2_1, X, KP_{\hat{U}}, MAC_1\}$ to U_2.

For $U_i \in U$ $(2 < i \le |\hat{U}|)$, the following steps repeat until the chain reply passes through all group users.

- Upon the receipt of $CR_{i-1,i}$, U_i derives $V_i = X \bmod B_i$ and decrypts $E_{K_{A_i}}(K_{RA} + N_R + N_A + \sum_{\substack{j=1 \\ j\neq i}}^{|\hat{U}|} K_{Aj}, H(K_{Ai}))$ by K_{Ai}.
- U_i checks the validity of $H(K_{Ai})$ and MAC_{i-1}. The session is dropped if any invalidity is detected; otherwise, U_i computes

$$\begin{aligned} C2_i &= \left(C2_{i-1} \times C1^{(K_{Ai})}\right) \bmod p \\ &= \left(\xi g^{r\left(K_{RA}+N_R+\sum_{j=1}^{i-1} K_{Aj}\right)} g^{r(K_{Ai})}\right) \bmod p \\ &= \left(\xi g^{r\left(K_{RA}+N_R+\sum_{j=1}^{i} K_{Aj}\right)}\right) \bmod p. \end{aligned}$$

- U_i selects a parameter of session key g^{n_i} and attaches it to $KP_{\hat{U}}$. Then, U_i generates $MAC_i = H(C1, C2_i, X, KP_{\hat{U}}, K_{RA} + N_R + N_A + \sum_{j=1}^{|\hat{U}|} K_{Aj})$.
- U_i sends $CR_{i,i+1} = \{C1, C2_i, X, KP_{\hat{U}}, MAC_i\}$ to the next user U_{i+1}.

Step 4) After receiving $CR_{|\hat{U}|,R}$, the following steps are implemented.

- U_R computes $V_0 = X \bmod B_0$ and decrypts V_0 by K_{RA} to obtain $(N_A, \sum_{j=1}^{|U|} K_{Aj}, H(K_{RA}))$.
- U_R checks the validity of V_0 by $H(K_{RA})$ and $MAC_{|\hat{U}|}$. If valid, U_R computes

$$\begin{aligned} \xi' &= C2_{|\hat{U}|} \times \left(C1^{\left(K_{RA}+N_R+\sum_{j=1}^{|\hat{U}|} K_{Aj}\right)}\right)^{-1} \\ &= \left(\xi g^{r\left(K_{RA}+N_R+\sum_{j=1}^{|\hat{U}|} K_{Aj}\right)}\right) \\ &\quad \times \left(g^{r\left(K_{RA}+N_R+\sum_{j=1}^{|\hat{U}|} K_{Aj}\right)}\right)^{-1} \bmod p. \end{aligned}$$

- If ξ' is identical to ξ, then all the group users $U_i \in \hat{U}$ are authenticated; otherwise, at least one group user fails the batch authentication.
- Once $\hat{U}$ is authenticated, U_R can extract the key parameters of session key g^{n_i} from $KP_{\hat{U}}$ and negotiate session keys with $U_i \in U$. The session keys can be obtained by $SK_{Ri} = (g^{n_i})^{m_i}$.
- For mutual authentication and key agreement, U_R computes $C2' = C2_{|\hat{U}|} \times C1^{N_A} \bmod p$ and $MAC'_R = H(C2', K_{RA} + N_R + N_A + \sum_{j=1}^{|\hat{U}|} K_{Aj})$. Then, the message $\{C2', MAC'_R\}$ is sent to U_i $(1 \le i \le |\hat{U}|)$. In the case that U_R cannot directly reach U_i, U_A can be involved to help forward the messages.

Step 5) After receiving $MR_{R,i}$ from U_R, the following steps are implemented.

- U_i verifies the authenticity of MAC'_R and computes

$$\begin{aligned} \xi'' &= C2' \times \left(C1^{\left(K_{RA}+N_R+N_A+\sum_{j=1}^{|\hat{U}|} K_{Aj}\right)}\right)^{-1} \bmod p \\ &= \left(\xi g^{r\left(K_{RA}+N_R+N_A+\sum_{j=1}^{|\hat{U}|} K_{Aj}\right)}\right) \\ &\quad \times \left(g^{r\left(K_{RA}+N_R+N_A+\sum_{j=1}^{|\hat{U}|} K_{Aj}\right)}\right)^{-1} \bmod p. \end{aligned}$$

- U_i also checks whether ID_R is included in ξ''. If yes, U_R is authenticated.

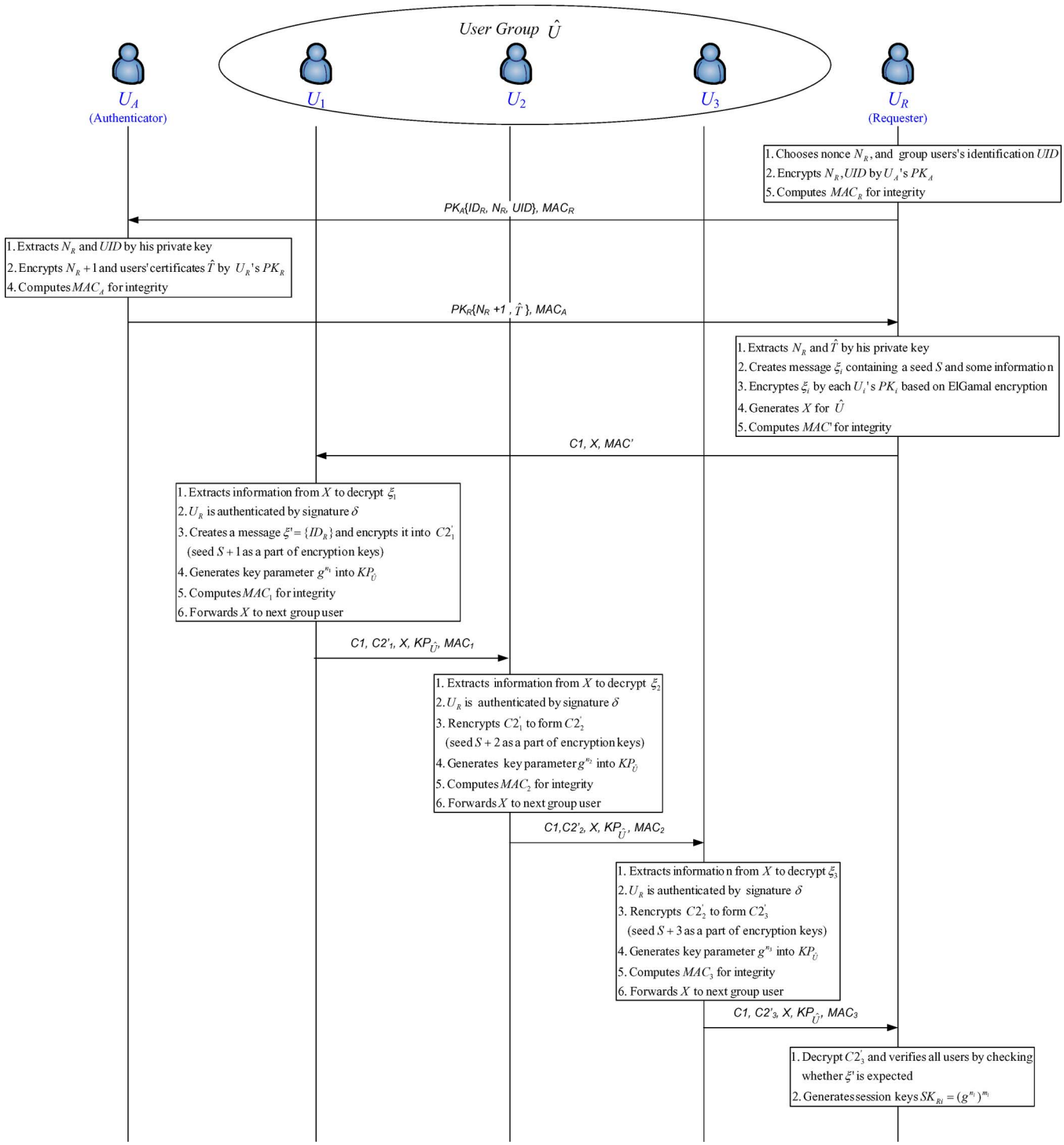


Fig. 6. Message flow of the certificate-based batch authentication protocol. The illustration shows an example of a user group of size 3 ($|\hat{U}|=3$).

- Then, U_i generates the session key $SK_{Ri}=(g^{m_i})^{n_i}$ to protect the communication between U_R and U_i.

C. Certificate-Based Protocol

The certificate-based protocol is proposed to guarantee the nonrepudiation of a transaction. In this protocol, we adopt the Shamir–Tauman online/offline signature [16] to enhance the security property. The authenticator U_A, behaving as a local trusted certificate authority, helps deliver and verify certificates for the group users ($U_i \in U$). The message flow of the proposed certificate-based batch authentication protocol is summarized as follows (see Fig. 6).

1) $U_R \rightarrow U_A : AQ_{R,A}=\{PK_A\{ID_R, N_R, UID\}, MAC_R\}$.
2) $U_A \rightarrow U_R : AR_{A,R}=\{PK_R\{N_R+1, \hat{T}\}, MAC_A\}$.
3) $U_R \rightarrow U_1 : CR_{R,1}=\{C1, X, MAC_A\}$.

4) $U_{i-1} \rightarrow U_i : CR_{i-1,i} = \{C1, C2'_i, X, KP_{\hat{U}}, MAC_i\}$, where $2 \le i \le |\hat{U}|$.
5) $U_{|\hat{U}|} \rightarrow U_R : CR_{|\hat{U}|,R} = \{C1, C2'_{|\hat{U}|}, X, KP_{\hat{U}}, MAC_{|\hat{U}|}\}$.

The certificate-based protocol is described through the following steps.

Step 1) U_R sends $AQ_{R,A}$ to U_A, where $AQ_{R,A} = \{PK_A\{ID_R, N_R, UID = \{ID_1, ID_2, \ldots, ID_{|\hat{U}|}\}\}, MAC_R\}$. In this message, the requester identity ID_R, nonce N_R, and user group identification UID are protected by U_A's public key (PK_A). The message authentication code MAC_R is derived by $H(PK_A\{ID_R, N_R, UID\}, K_{RA} + N_R)$ and used to guarantee the validity of the message.

Step 2) When U_A receives the $AQ_{R,A}$, the following steps are implemented.
- U_A decrypts $AQ_{R,A}$ and obtains ID_R, N_R, and UID.
- U_A checks the validity of MAC_R. If illegal, U_A drops this connection; otherwise, U_A confirms each U_i's trust level and then encrypts $\{N_R + 1, \hat{T}\}$ by U_R's public key (PK_R), where $\hat{T} = \{T_1, T_2, \ldots, T_{|U|}\}$. If any U_i's trust level is lower than the predefined threshold, U_A should note the information to U_R.
- U_A derives $MAC_A = H(PK_R\{N_R + 1, \hat{T}\}, K_{RA} + N_R)$ to protect the validity of the message.
- U_A sends $AR_{A,R}$ to U_R, where $AR_{A,R} = \{PK_R\{N_R + 1, \hat{T}\}, MAC_A\}$.

Step 3) After obtaining $AR_{A,R}$, the following steps are implemented.
- U_R checks the validity of the message by MAC_A and $N_R + 1$. If invalid, U_R immediately drops the session; otherwise, U_R decrypts the message by his/her private key RK_R and obtains $\hat{T}$. Similar to the proxy-based batch authentication protocol, U_R computes the following values:
  1) $C1 = g^r$, where r is a randomly selected secret parameter $r \in \mathbb{Z}_q^*$;
  2) $V_i = C2_i$, where $C2_i = \xi_i(PK_i)^r = \xi_i(g^{RK_i})^r$, and $\xi_i = \{\text{trust level}, S, \delta = Sign(RK_R, C1), T_R, g^{m_i}\}$, where S is a random number used as the seed of the ElGamal proxy encryption key, and δ is a signature generated by the Shamir–Tauman online/offline signature protocol [16].
- In addition, by adopting the CRT [17], we can merge all messages to $U_i \in U$ in a single message. Letting $B_1, B_2, \ldots, B_{|\hat{U}|}$ be positive integers that are pairwise relative primes and $V_1, V_2, \ldots, V_{|U|}$ be positive integers, we intend to obtain the solution for the following congruous equations:

$$X \equiv V_i \bmod B_i \ \left(\text{for } U_i \in U,\ 1 \le i \le |\hat{U}|\right).$$

We can derive the common solution X for $X = (\sum_{i=1}^{|\hat{U}|} L/B_i \times V_i \times A_i) \bmod L$, where $1 \equiv A_i \times (L/B_i) \bmod B_i$, and $\{V_i, B_i | i \in \{1, \ldots, |\hat{U}|\}\}$. Let $X = V_i \bmod B_i$ for $1 \le i \le |\hat{U}|$. Then, the common solution X should be within the range of $[1, L-1]$, where $L = \prod_{i=1}^{|\hat{U}|} B_i$.
- U_R generates $MAC_R = H(C1, X, S)$ and sends the message $CR_{R,1} = \{C1, X, MAC_R\}$ to U_1.

Step 4) Upon the receipt of $CR_{R,1}$, the following steps are implemented.
- U_1 extracts $V_1 = X \bmod B_1$ to obtain $C2_1 = \xi_1(PK_1)^r = \xi_1(g^{RK_1})^r$.
- U_1 decrypts $C2_1$ by his/her own private key RK_1 and obtains ξ_1, where

$$\begin{aligned} C2_1(C1^{RK_1})^{-1} \bmod p &= \xi_1(PK_1)^r(C1^{RK_1})^{-1} \bmod p \\ &= \xi_1(g^{RK_1})^r \left((g^r)^{RK_1}\right)^{-1} \bmod p \\ &= \xi_1 = \{\text{trust level}, S, \delta, T_R, g^{m_1}\}. \end{aligned}$$

- U_1 derives S from ξ_1 and checks the validity of MAC_R. The session is dropped if MAC_R is invalidated; otherwise, U_1 continues to verify the signature δ that is signed by U_R. U_R is authenticated if the signature is valid.
- If U_R is authenticated, U_1 selects a random number n_1, generates the parameter of session key g^{n_1}, and adds the parameter to $KP_{\hat{U}} = \{g^{n_1}\}$. Note that U_1 can now obtain the session key $SK_{R1} = (g^{m_1})^{n_1}$ and use it to protect the session.
- U_1 generates $C2'_1 = \xi'(PK_R)^{S+1} = \xi'(g^{RK_R})^{S+1}$, where $\xi' = \{ID_R\}$.
- U_1 appends the authentication message code $MAC_1 = H(C1, C2'_1, X, KP_{\hat{U}}, S)$ to the message $CR_{1,2} = \{C1, C2'_1, X, KP_{\hat{U}}, MAC_1\}$ and sends the message to U_2.

For $U_i \in U$ $(2 < i \le |\hat{U}|)$, the following steps repeat until the chain reply passes through all group users.
- After receiving the $CR_{i-1,i} = \{C1, C2'_{i-1}, X, KP_{\hat{U}}, MAC_{i-1}\}$
  1) U_i derives $V_i = X \bmod B_i$ to obtain $C2_i = \xi_i(PK_i)^r = \xi_i(g^{RK_i})^r$.
  2) U_i decrypts $C2_i$ by his/her private key RK_i and checks the validity of MAC_{i-1} and δ (U_R's signature).
  3) If U_R is legitimate, U_i adds the parameter g^{n_i} to $KP_{\hat{U}}$ and generates a session key $SK_{Ri} = (g^{m_i})^{n_i}$.
- After extracting S from ξ_i, U_i computes

$$\begin{aligned} C2'_i &= C2'_{i-1}(PK_R)^{(S+i)} \bmod p \\ &= \xi'(g^{RK_R})^{\left(\sum_{j=1}^{i-1}(S+j)\right)}(g^{RK_R})^{(S+i))} \bmod p \\ &= \xi'(g^{RK_R})^{\left(\sum_{j=1}^{i}(S+j)\right)} \bmod p. \end{aligned}$$

- U_i generates $MAC_i = H(C1, C2'_i, X, KP_{\hat{U}} = \{g^{n_1}, \ldots, g^{n_i}\}, S)$.
- U_i delivers the message $CR_{i,i+1} = \{C1, C2'_i, X, KP_{\hat{U}}, MAC_i\}$ to the next user U_{i+1}.

Step 5) After receiving $CR_{|\hat{U}|,R}$, U_R checks $MAC_{|\hat{U}|}$ by S. If validated, then U_R can authenticate all group users $U_i \in U$ as follows.

- U_R computes $C1' = g^{\left(\sum_{j=1}^{|\hat{U}|}(S+j)\right)} \bmod p$.
- U_R derives ξ' by computing

$$\begin{aligned} & C2'_{|\hat{U}|}\left(C1'^{RK_R}\right)^{-1} \bmod p \\ &= \xi'(g^{RK_R})^{\left(\sum_{j=1}^{|\hat{U}|}(S+j)\right)} \\ &\quad \times \left(g^{\left(\sum_{j=1}^{|\hat{U}|}(S+j)\right)RK_R}\right)^{-1} \bmod p \\ &= \xi'. \end{aligned}$$

- U_R checks the validity of $\xi' = \{ID_R\}$. If so, $U_i \in \hat{U}$ are authenticated.
- U_R then generates the session keys $SK_{Ri} = (g^{n_i})^{m_i}$ to protect the sessions with group users $U_i \in U$.

Different from the other two protocols, in the certificate-based protocol, U_R knows whether U_i is authenticated by checking the value of $\sum_{j=1}^{|\hat{U}|}(S+j)$. In a group of three users, for example, U_R obtains $\sum_{j=1}^{3}(S+j) = 3S+6$ if all group users are authenticated. Therefore, if a smaller value, for example, $2S+4$, is received, it is implied that some user (U_2) has failed the authentication.

V. Correctness and Security Analysis

A. Correctness Analysis

In this section, we demonstrate the formula correctness of our protocols.

1) Hash-Based Protocol:

- *User group authentication.* As mentioned in Section IV-A, Step 3, each U_i computes $M_i = M_{i-1} \oplus H((N_R+N_A) \oplus h_i)$. If $1 \le i \le |\hat{U}|$. Then, $M_{|\hat{U}|} = H((N_R+N_A) \oplus h_1) \oplus H((N_R+N_A) \oplus h_2) \oplus \cdots \oplus H((N_R+N_A) \oplus h_{|\hat{U}|}$. Therefore, in Section IV-A, Step 4, U_R will check whether $M'_{|\hat{U}|} \stackrel{?}{=} M_{|\hat{U}|}$ holds, which is verified as follows:

$$\begin{aligned} M'_{|\hat{U}|} &= H\left((N_R+N_A) \oplus h_1\right) \oplus H\left((N_R+N_A) \oplus h_2\right) \\ &\quad \oplus \ldots \oplus H\left((N_R+N_A) \oplus h_{|\hat{U}|}\right) \\ &= M_{|\hat{U}|}. \end{aligned}$$

- *Requester authentication*. In Section IV-A, Step 4, U_R computes $S_i = H((N_R+N_A+1) \oplus h_i)$ for each U_i. As a result, in Step 5, each U_i can authenticate U_R by checking whether $S'_i \stackrel{?}{=} S_i$ holds as follows:

$$S'_i = H\left((N_R+N_A+1) \oplus h_i\right) = S_i.$$

2) Proxy-Based Protocol:

- *User group authentication*. In Section IV-B Step 3, each U_i contributes its own key K_{Ai} into $C2_i$. If $1 \le i \le |\hat{U}|$, then $C2_i$ evolves into $C2_{|\hat{U}|}$ as follows:

$$\begin{aligned} C2_i &= \left(C2_A \times C1^{(K_{Ai})}\right) \text{ for } 1 \le i \le |\hat{U}| \\ &= \xi g^{r(K_{RA}+N_R)} \times g^{r(K_{A1})} \times g^{r(K_{A2})} \times \cdots \times g^{r(K_{A|\hat{U}|})} \\ &= \xi g^{r(K_{RA}+N_R)} \times g^{r(K_{A1})+r(K_{A2})+\cdots+r(K_{A|\hat{U}|})} \\ &= \xi g^{r\left(K_{RA}+N_R+\sum_{j=1}^{|\hat{U}|}K_{Aj}\right)} \\ &= C2_{|\hat{U}|}. \end{aligned}$$

As a result, in Section IV-B, Step 4, U_R will authenticate the group users by checking whether $\xi' \stackrel{?}{=} \xi$ holds, which follows the ElGamal proxy decryption and is verified as follows:

$$\begin{aligned} \xi' &= C2_{|\hat{U}|} \times \left(C1^{(K_{RA}+N_R+\sum_{j=1}^{|\hat{U}|}K_{Aj})}\right)^{-1} \\ &= \xi g^{r\left(K_{RA}+N_R+\sum_{j=1}^{|\hat{U}|}K_{Aj}\right)} \times \left(g^{r\left(K_{RA}+N_R+\sum_{j=1}^{|\hat{U}|}K_{Aj}\right)}\right)^{-1} \\ &= \xi \bmod p. \end{aligned}$$

- *Requester authentication*. In Section IV-B, Step 4, U_R embeds its N_A into $C2_{|\hat{U}|}$ to form $C2'$ as follows:

$$\begin{aligned} C2' &= C2_{|\hat{U}|} \times C1^{N_A} \\ &= \xi g^{r\left(K_{RA}+N_R+\sum_{j=1}^{|\hat{U}|}K_{Aj}\right)} \times g^{r(N_A)} \\ &= \xi g^{r\left(K_{RA}+N_R+N_A+\sum_{j=1}^{|\hat{U}|}K_{Aj}\right)}. \end{aligned}$$

Therefore, in Section IV-B, Step 5, each U_i also verifies the validity of U_R by checking whether the form and

content of ξ'' are correct. The decryption formula of ξ'' is shown as follows:

$$\begin{aligned}\xi'' &= C2'_{\prime} \times \left(C1^{\left(K_{RA}+N_R+N_A+\sum_{j=1}^{|\hat{U}|} K_{Aj} \right)} \right)^{-1} \\ &= \xi g^{r\left(K_{RA}+N_R+\sum_{j=1}^{|\hat{U}|} K_{Aj} \right)} \\ &\quad \times \left(g^{r\left(K_{RA}+N_R+N_A+\sum_{j=1}^{|\hat{U}|} K_{Aj} \right)} \right)^{-1} \\ &= \xi \bmod p.\end{aligned}$$

3) Certificate-Based Protocol:

- *User group authentication*. In Section IV-C Step 4, U_1 first generates $C2'_1 = \xi'(PK_R)^{S+1} = \xi'(g^{RK_R})^{S+1}$, and then, each U_i for $2 < i \leq |\hat{U}|$ contributes their $(S+i)$ into $C2'_i$. In the end, $C2'_{|\hat{U}|}$ is formed as follows:

$$\begin{aligned}C2'_{|\hat{U}|} &= C2'_1 \times (PK_R)^{(S+2)} \times (PK_R)^{(S+3)} \\ &\quad \times \cdots \times (PK_R)^{(S+|\hat{U}|)} \bmod p \\ &= \xi'(g^{RK_R})^{(S+1)} \times (g^{RK_R})^{(S+2)} \\ &\quad \times \cdots \times (g^{RK_R})^{(S+|\hat{U}|)} \bmod p \\ &= \xi'(g^{RK_R})^{\left(\sum_{j=1}^{|\hat{U}|} (S+j) \right)} \bmod p.\end{aligned}$$

  Finally, in Section IV-C, Step 5, U_R computes $C1' = g^{(\sum_{j=1}^{|\hat{U}|}(S+j))}$ for decryption, which follows the procedure for the ElGamal proxy cryptography. After obtaining ξ', U_R can confirm whether the content of ξ' is legitimate. The correctness of the decryption is presented as follows:

$$\begin{aligned}C2'_{|\hat{U}|} &= C2'_{|\hat{U}|}(C1'^{RK_R})^{-1} \bmod p \\ &= \xi'(g^{RK_R})^{\left(\sum_{j=1}^{|\hat{U}|} (S+j) \right)} \left(g^{\left(\sum_{j=1}^{|\hat{U}|} (S+j) \right) RK_R} \right)^{-1} \bmod p \\ &= \xi'.\end{aligned}$$

- *Requester authentication*. Because the requester authentication is achieved by the verification of U_R's signature, the correctness of signature verification is referenced by the Shamir–Tauman online/offline signature [16].

B. Security Analysis

This section gives security proofs and analyzes the proposed protocols. Due to space limitations, we briefly introduce the security model and follow the concept of formal security proofs to reduce our protocols to well-known hard problems.[3] Similar to [15], we prove the security of proxy- and certificate-based protocols based on the decision Diffie–Hellman (DDH) assumption [19], [20]. Then, we prove that the hash-based batch authentication protocol is secure against passive adversaries, impersonators, and implicit key authentication attacks by applying the concept of the random-oracle model [18].

1) Security Model and Concepts: In 1993, Bellare and Rogaway [18] proposed a theoretical security proof for an authentication and key agreement protocol, called the BR93-Model. In our security analyses, we take the BR93-Model as a foundation for defining the security analysis concept.

Protocol Participants: $\prod_{U_*,R}^{i}$ denotes the user oracle, which plays the role U_* of interacting with R in the ith session, and $\prod_{R,U_*}^{j}$ denotes the server oracle, which plays the role R of interacting with U_* in the jth session. Note that, in our scheme, the server is regarded as the combination of the authenticator and the requester. Let $\mathcal{F}$ be the proposed authentication protocols. During the execution of $\mathcal{F}$, an adversary E who is a probabilistic polynomial-time Turing machine can control the entire network and obtain transmitted data in previous processes. The predefined oracle queries are listed as follows.

- `Execute` $(\prod_{U_*,R}^{i}, \prod_{R,U_*}^{j})$. The query models all kinds of passive attacks, where a passive adversary can eavesdrop on all transmitted data between $\prod_{U_*,R}^{i}$ and $\prod_{R,U_*}^{j}$ in the protocol $\mathcal{F}$.
- `Send` $(\prod_{U_*,R}^{i}, \xi)$. The query models an active attack, where an adversary sends a message ξ to $\prod_{U_*,R}^{i}$. The adversary can get the response message according to the sending message ξ in the protocol $\mathcal{F}$. An adversary can initiate a session by setting $\xi = \lambda$.
- `Send` $(\prod_{R,U_*}^{j}, \xi)$. The query also models active attacks, where an adversary sends a message ξ to $\prod_{R,U_*}^{j}$. The adversary can get the response message according to the sending message ξ in the protocol $\mathcal{F}$.
- `Reveal` $(\prod_{U_*,R}^{i})$. This query models the exposure of the old session key of instance i.
- `Hash`(). This query allows an adversary to ask the hash function value, and the challenger (or the simulator) answers all `Hash`() queries at random, just like real oracles would.
- `Test` $(\prod_{U_*,R}^{i})$. If $\prod_{U_*,R}^{i}$ accepts and shares a session key with the partner oracle $\prod_{R,U_*}^{j}$, the adversary E can launch the query to distinguish a real session key from a random string. The query models the adversary's queries of the test oracle. Depending on a random coin bit, the adversary E is given the real session key or a randomly chosen string.

Security in the model is defined using the game $\mathcal{G}$, which is played between the adversary E and a collection of oracles

[3] More specifically, we will not claim that our analysis is a formal security proof, because the details of query operations and probability calculation are omitted. See [29], [30] for a more formal proof.

$\prod_{U_*,R}^{i} / \prod_{R,U_*}^{j}$. The adversary E runs the game simulation $\mathcal{G}$ with the following setting.

Step 1) The adversary E can perform `Execute`, `Send`, `Reveal`, `Hash`, and `Test` queries in the simulation.

Step 2) At any moment during $\mathcal{G}$, E can choose a fresh session and send a `Test` query to the fresh oracle that is associated with the test session. Depending on a randomly chosen bit $\beta \in \{0,1\}$, the challenger returns either a real session key or a randomly chosen string.

Step 3) E continues issuing the aforementioned `Execute`, `Send`, `Reveal`, and `Hash` queries.

Step 4) Finally, E terminates the game simulation and outputs its guess bit β'.

As we have done, we define the E's guessing advantage as

$$\mathrm{Adv}^E(k) = \left| Pr[\beta = \beta'] - \frac{1}{2} \right|$$

where k is the security parameter.

Security Concept: The aforementioned queries represent the abilities of an adversary in a real environment. The purpose of the adversary is to either pass the authentication procedure or distinguish the session key from a random string with a nonnegligible probability. Our security analysis follows the concept that if an adversary was able to be authenticated or to distinguish the session key, the adversary could solve some well-known computationally hard problems.

2) Security Proof by the DDH Assumption: In the proxy- and certificate-based protocols, U_R adopts the Shamir–Tauman online/offline signature protocol [16] to generate $\delta = Sign(RK_R, C1)$. Therefore, we prove the security of proxy- and certificate-based batch authentication protocols by the DDH assumption [19], [20].

Assumption 1—DDH Problem: There are primes p and q such that $p = 2q + 1$ and a generator $g \in \mathbb{Z}_p^*$ with order q for the subgroup G_q, where G_q is a subgroup of quadratic residues in $\mathbb{Z}_p^*$. For a given $y_a = g^{X_a} \bmod p$ and $y_b = g^{X_b} \bmod p$, where X_a and X_b are randomly chosen from $\mathbb{Z}_p^*$, the following two tuples of random variables $(y_a, y_b, g^{X_a X_b} \bmod p)$ and (y_a, y_b, W), where W is a random value in $\mathbb{Z}_p^*$, are computationally indistinguishable. In other words, there is no efficient adversary algorithm E that satisfies $|Pr[E(g^{X_a}, g^{X_b}, g^{X_a X_b}) = true] - Pr[E(g^{X_a}, g^{X_b}, R)] > (1/Q(|q|))$ for any polynomial Q, where the probability is higher than the random choice of X_a, X_b, and R.

In the proxy- and certificate-based batch authentication protocols, messages are encrypted by the ElGamal proxy cryptography. Based on Assumption 1, [23] has proven that the security of the ElGamal encryption is as hard as the DDH problem. Here, we show how we can reduce our protocols to the ElGamal encryption.

Theorem 5.1: Under Assumption 1, the proposed protocol is secure against passive adversaries.

Proof: In a proof by contradiction, assume that there is an adversary E who can derive the message ξ that was encrypted by the ElGamal encryption $(Z1 = \alpha^r = g^r, Z2 = \xi\beta^r = \xi((g^x)^r) \bmod p)$.

We construct another efficient algorithm E' to obtain the message ξ that was encrypted by the ElGamal proxy encryption. The adversary can perform the `Execute` query mentioned in Section V-B-1 to obtain $(C1, C2_R)$, $(C1, C2_A)$, and $(C1, C2_i)$ for $1 \le i \le |\hat{U}|$ as the inputs of the algorithm E'. Without loss of generality, let $C1 = Z1 = g^t$ and $C2_R = \xi(g^{V_R})^t$, where t, V_R, N_R, and $N_A, V_1, \ldots, V_{n-1}$ are random numbers that were chosen from Z_q^*. We compute the following values:

$$\begin{aligned}
C2_A &= C2_R \times C1^{N_R} \\
&= \xi(g^{V_R})^t \times C1^{N_R} \\
&= \xi g^{t(N_R+V_R)} \\
C2_1 &= C2_A \times C1^{V_1} \\
&= \xi g^{(N_R+V_R)t} \times C1^{V_1} \\
&= \xi g^{t(N_R+V_R+V_1)} \\
&\vdots \\
C2_{|\hat{U}|} &= C2_{|\hat{U}|-1} \times C1^{V_{|\hat{U}|}} \\
&= \xi \left(g^{\left(N_R+V_R+\sum_{j=1}^{|\hat{U}|-1} V_j \right) t} \right) \times C1^{V_{|\hat{U}|}} \\
&= \xi g^{\left(N_R+V_R+\sum_{j=1}^{|\hat{U}|} V_j \right) t} \\
&= \xi\beta^r = Z2.
\end{aligned}$$

Now, the algorithm E' has been constructed. Because E can derive the messages encrypted by the ElGamal encryption, by applying E, E' can obtain the message ξ that was encrypted by the ElGamal proxy encryption, which is a contradiction to Assumption 1. ■

Lemma 5.2: Under Assumption 1, any adversary E cannot generate a valid $C2_i = \xi g^{r(K_{RA}+N_R+\sum_{j=1}^{|\hat{U}|} K_{Aj})} \bmod p$, which can successfully be verified by the requester U_R.

Proof: In our protocol, each user U_i generates $C2_i$ from $C2_{i-1}$. The message $(C1, C2_i, X)$ is broadcast, where $C1 = g^r \bmod p$, $C2_i = \xi g^{r(N_R+K_{RA}+K_{A1}+\cdots+K_{Ai})} \bmod p$, and the adversary E must compute $C2_i$ from these public parameters. Because the Advanced Encryption Standard (AES) algorithm is assumed to be secure [24], the parameter X is well protected. Obviously, the malicious adversary E must efficiently distinguish $(g^r, \xi g^{r(N_R)}, \xi g^{r(K_{RA}+N_R+K_{A1}+\cdots+K_{Ai})} \bmod p)$ from $(g^r, \xi g^{r(N_R)}, W \bmod p)$, where W is a random value in $\mathbb{Z}_q^*$. The problem is obviously a contradiction to Assumption 1. We then prove that no adversary E can pass the authentication under Assumption 1. ■

Lemma 5.3: Assume that, by computing the discrete logarithms modulo a large prime is difficult, any malicious adversary E cannot generate a signature protocol $\delta = Sign(RK_R, C1)$.

Proof: Because the signature $\delta = Sign(RK_R, C1)$ is generated by the Shamir–Tauman online/offline signature protocol, the security proof can directly refer to [16]. We briefly quote the conclusion. If the adversary E, without knowing

the secret key RK_R, can impersonate U_R to generate valid signatures with a nonnegligible probability ϵ, then the adversary E can efficiently compute the discrete logarithms modulo a large prime. Thus, the adversary E who tries to generate $\delta = Sign(RK_R, C1)$, indeed, needs the secret key RK_R. In conclusion, each user U_i can correctly authenticate the requester U_R by signature $\delta = Sign(RK_R, C1)$ under the assumption of discrete logarithms. ■

Theorem 5.4: Under the discrete logarithm assumption and Assumption 1, the proposed proxy- and certificate-based protocols are secure against impersonator attacks.

Proof: In this proof, we first show that the requester U_R can authenticate each user U_i under the Diffie–Hellman problem (DH) assumption. Then, each U_i can also authenticate U_R under the DDH problem assumption or the discrete logarithm problem (DLP) assumption [19].

By Lemma 5.2, we know that only legal requester U_R and users U_i, who possess the secret shared key $K_A i$ with the authenticator U_A, can generate $C2'$ and $C2_i$. Therefore, the protocols based on the ElGamal proxy encryption are secure against impersonator attacks.

By Lemma 5.3, it is clear that only legal requester U_R, who holds RK_R, can generate the signature $\delta = Sign(RK_R, C1)$. Because the adversary has no access to the legal requester's secret key RK_R, he/she cannot generate a valid signature δ. Thus, we prove that the proposed proxy- and certificate-based protocols are secure against impersonator attacks and provide mutual authentication. ■

Theorem 5.5: Under the discrete logarithm assumption and Assumption 1, the proposed proxy- and certificate-based protocols can provide implicit key authentication.

Proof: By Lemma 5.2, we know that any malicious adversary cannot generate $C2_R$ and use it to derive the key agreement parameter g^{m_i} under the assumption of DLP. Although the counterpart of key agreement parameter g^{n_i} is transferred in plaintext, MAC_i can be used to withstand the malicious adversary under the security of AES [24] and the ElGamal proxy encryption. Because "$N_A + N_R + K_{RA} + \sum_{j=1}^{|\hat{U}|} K_{Aj}$" and "S" are well protected by AES and the ElGamal proxy encryption, these values can be used to generate a valid MAC_i. Hence, the proposed proxy- and certificate-based protocols, taking the ElGamal proxy encryption as their basis, are proved to offer the implicit key authentication. ■

Theorem 5.6: Under the discrete logarithm assumption and Assumption 1, the proposed protocols provide forward secrecy.

Proof: In this proof, we first prove that the proxy- and certificate-based protocols guarantee forward secrecy and then show that the hash-based protocol also provides forward secrecy as follows.

- The key agreement handshakes between U_R and U_i are independent of the secret key K_{RA} and K_{Ai}. Without loss of generality, let $U_p = \{U_1, U_2, \ldots, U_n\}$ be the set of users who have established a session key SK_{Rp} with U_R at some past time τ. After closing the session, the session key SK_{Rp} will be tossed. Suppose that an adversary E obtains a secret key K_{Ap} of any user U_p and the secret key SK_{RA} of U_R at time $\tau + 1$. Because the session key SK_{Rp} is isolated from the secret key K_{Ap}, the adversary E cannot obtain the session key SK_{Rp}. In the case that the adversary E obtains g^{m_p} or g^{n_p}, he/she can compute the session key SK_{Rp}. By Lemma 5.2, we know that the adversary E cannot obtain m_p or n_p from g^{m_p} or g^{n_p} under the assumption of discrete logarithms. We hence prove that our protocols guarantee forward secrecy.
- In the hash-based batch authentication protocol, we exploit the DH key agreement to negotiate the session key. Therefore, its security of forward secrecy is the same as previously mentioned. ■

3) Security Proof by Random Oracle Concept: According to the concept of the random-oracle model [18], which assumes that the one-way hash function is a fair random function, we can prove that the proposed hash-based batch authentication protocol is secure against passive adversaries, impersonators, and implicit key authentication attacks.

Theorem 5.7: Under the assumption of the discrete logarithm and the robust one-way hash function, the proposed hash-based protocol is secure against passive adversaries.

Proof: If the adversary cannot acquire any critical meaning information from the messages transmitted in the public channel, the proposed hash-based protocol is secure against passive adversaries. In the hash-based batch authentication protocol, most messages are operated by exclusive ORs (XORs) and the one-way hash function as follows.

- Because the XOR operation is widely adapted to several famous systems [21], [22] to mix with messages that infer that the XOR operation is suited to hide information, we also apply XORs to conceal the information.
- The most important information h_i is protected by the one-way hash function. Under the concept of the random-oracle model [18], the probability is negligible when the adversary tries to extract h_i from $M_{i+2} = H((N_R + N_A) \oplus h_i)$. For MAC_i, the security of the message authentication code is also under the concept of the random-oracle model due to the indistinguishability of the one-way hash function.

Although the parameters of key agreement g^{n_i} are exposed to the public, the assumption of a discrete logarithm ensures that the adversary does not have the nonnegligible advantage to know the value of n_i. As a result, the hash-based batch authentication protocol is secure against a passive attack. ■

Theorem 5.8: As long as the one-way hash function is robust, the proposed hash-based batch authentication protocol is secure against impersonator attacks.

Proof: An impersonator attack is an attack where the adversary acts on behalf of other users to pass the authentication. To impersonate a legal user, the adversary needs to obtain h_i, N_R, and N_A. Under the concept of the random-oracle model, which assumes that the one-way hash function is robust, the probability that the adversary obtains h_i, N_R, and N_A from M_{i+2} is negligible. The adversary may extract $(h_{i-2}, h(K_{Ai}), \sum_{j=1}^{2} N_j, g^{m_i})$ if he/she knows the preshared keys K_{Ai} and t_i between U_i and U_A. Because shared key K_{Ai} is well protected by the one-way hash function, under the concept

TABLE II
PERFORMANCE COMPARISONS OF THE UNDERLYING CRYPTOGRAPHIES

	Kerberos	ElGamal	Proposed batch authentication protocols		
			Hash-	Proxy-	Certificate-
Communication cost	$6n$	$n^2 - n$	$2n + 2$	$2n + 2$	$n + 3$
Computational cost	Low	Medium	Extremely low	Medium	High
Underlying cryptosystem	Symmetric	Asymmetric	One-way hash	Asymmetric	Asymmetric & PKI
Trustworthiness	In Pairs (U & S)	In Pairs (U_R & U_j)	In Pairs (U_R & U_i)	All Entities	All Entities
Trust level	N/A	N/A	N/A	Supported	Supported

n: number of users to be authenticated.

of the random-oracle model, the probability of compromising the shared key from the transmitted messages is negligible. Thus, our hash-based protocol is secure against impersonator attacks. ■

Theorem 5.9: Under the assumption of a discrete logarithm and the robust one-way hash function, the proposed hash-based protocol is secure against implicit key authentication.

Proof: In the hash-based protocol, two parameters of key agreement are exchanged during handshake as follows.

- g^{n_i} is transmitted in plaintext, but its integrity can be guaranteed by MAC_i.
- g^{m_i} is protected by $H(r, (K_{Ai} \| t_i))$. Under the concept of the random-oracle model, only the legal U_i who possesses the shared key K_{Ai} can obtain the g^{m_i}. Based on the discrete logarithm assumption, no one can obtain the value of n_i, except for the legal U_i.

Therefore, the implicit key authentication is provided in our hash-based protocol. ■

VI. COMPARISONS

In this section, we compare our protocols with the Kerberos, ElGamal, and other existing authentication protocols. Because our architecture is similar to Kerberos [25], [26], we first compare our protocols with Kerberos when authenticating multiple users. Table II compares the proposed protocols with Kerberos in terms of the communication cost, computational cost, the underlying cryptosystems, trustworthiness relationship, and the support of trust level. While authenticating n users, the Kerberos protocol requires $6n$ messages to completely authenticate all users. In contrast, the proposed protocols require at most $2n + 2$ messages. To show the advantage of our batch concept, a comparison with the traditional ElGamal encryption is also presented. While authenticating n users, the total number of messages is $2 \times ((n \cdot (n-1))/2) = n^2 - n$. The reason is that the traditional ElGamal encryption requires two messages to achieve mutual authentication. Moreover, if n users try to authenticate the other users, the number of authentication times is $((n \cdot (n-1))/2)$. Fig. 7 illustrates the comparison of communication cost among our framework, Kerberos, and ElGamal encryption. As shown, our scheme outperforms the other schemes. Moreover, the performance advantage increases with an increase in the number of users to be authenticated.

Because the hash-based protocol takes a one-way hash function as its underlying cryptosystem, its computational costs are extremely low. Because the proposed certificate-based batch

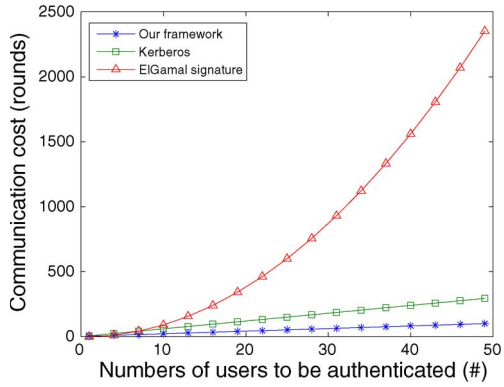


Fig. 7. Communication cost versus the number of users to be authenticated.

authentication protocol adopts the ElGamal proxy encryption scheme and public key infrastructure, its computational cost is the highest, but it requires the fewest messages for authentication. With regard to trustworthiness, both Kerberos and the proposed hash-based protocol build trustworthiness between the requester U_R and individual user U_i. By applying the proxy- and certificate-based protocols, we can build complete trustworthiness among all entities, including U_A, U_R, and $U_i \in U$. In our protocols, the trust level can be used to distinguish the level of trustworthiness. For example, a face-to-face authentication leads to a higher trust level, whereas P2P-based authentication has a lower trust level. In our protocols, only the proxy- and certificate-based protocols provide trust level during authentication. By the provided trust level, entities can predefine a trust level threshold to determine whether to accept the user as a trusted friend.

In Table III, we compare our protocols with other P2P-based authentication protocols with respect to the requirements mentioned in Section III. Most protocols provide mutual authentication among entities, but none of the existing protocols offers batch authentication. The proposed proxy- and certificate-based protocols also support user reputation by the trust level. Moreover, the certificate-based protocol also offers nonrepudiation by signing the messages. Because no centralized server is involved in our design, the proposed protocols can be adapted to any OSN, either web- or P2P-based architecture. Our framework also supports comprehensive hardware, from resource-limited to powerful computing devices. As previously mentioned, the hash-based protocol takes only a one-way hash function as its underlying cryptosystem and, hence, is cost effective. In short, our protocols meet most of the requirements mentioned in Section III.

TABLE III
FEATURE COMPARISONS OF THE P2P-BASED AUTHENTICATION PROTOCOLS

	VistualSec [3]	Safebook* [6]	Groove [27]	FL02 [28]	AAPR [9]	GD03 [8]	N06 [10]	PGP	Kerberos	ElGamal	Our Protocols		
											Hash-	Proxy-	Certificate-
R1. Strong Authen.	△	o	o	o	o	o	o	o	o	o	o	o	o
R2. Reputation	x	o	x	o	o	o	x	△	x	x	x	o	o
R3. Community Authenticity	o	x	o	x	o	o	o	o	o	x	o	o	o
R4. Non-repud.	x	o	x	x	o	x	x	o	x	o	x	x	o
R5. Flexibility	o	o	o	x	x	o	△	o	x	o	o	o	o
R6. Batch Authen.	x	x	x	x	x	x	x	x	x	x	o	o	o
R7. Compre. Hardware Supports	x	x	x	x	x	x	o	x	x	x	o		

O: Supported; X: Not supported; △: Partially supported
*: No specific protocol descriptions

VII. CONCLUSION

In this paper, we have proposed a P2P-based batch authentication framework for OSNs. We have also designed three batch authentication protocols using the one-way hash function, ElGamal proxy encryption, and certificates for different situations and purposes. The hash-based protocol adopts lightweight cryptosystems to reduce the computational costs. To offer higher security properties, the proxy- and certificate-based methods are based on asymmetric encryptions and signature methods to fulfil the security requirements of sensitive transactions.

In this paper, we proved that our protocols are secure against passive adversaries and impersonator attacks and support implicit key authentication. After analyzing the communication and computational costs, we show that our protocols require fewer authentication messages. Notably, the proposed hash-based batch authentication protocol is cost effective when run with resource-limited devices. Compared with other existing P2P-based authentication protocols, the proposed protocols can meet the security requirements of P2P-based OSNs, including mutual authentication, reputation, community authenticity, nonrepudiation, flexibility, and comprehensive hardware support.

REFERENCES

[1] C. Zhang, J. Sun, X. Zhu, and Y. Fang, "Privacy and security for online social networks: Challenges and opportunities," *IEEE Netw.*, vol. 24, no. 4, pp. 13–18, Jul./Aug. 2010.

[2] D. Niyato, P. Wang, W. Saad, and A. Hjorungnes, "Controlled coalitional games for cooperative mobile social networks," *IEEE Trans. on Vehi. Tech.*, vol. 60, no. 4, pp. 1812–1824, May 2011.

[3] M. Ge, K.-Y. Lam, X. Wang, Z. Wan, and B. Jiang, "VisualSec: A secure message delivery scheme for online social networks based on profile images," in *Proc. IEEE GLOBECOM*, 2009, pp. 1–6.

[4] S. Buchegger and A. Datta, "A case for P2P infrastructure for social networks—Opportunities and challenges," in *Proc. WONS*, 2009, pp. 161–168.

[5] S. Buchegger, D. Schioberg, L. H. Vu, and A. Datta, "PeerSoN—P2P social networking: Early experiences and insights," in *Proc. SocialNets*, 2009, pp. 46–52.

[6] L. A. Cutillo and R. Molva, "Safebook: A privacy-preserving online social network leveraging on real-life trust," *IEEE Commun. Mag.*, vol. 47, no. 12, pp. 94–101, Dec. 2009.

[7] U. Lee, J. Sewook, C. Dae-Ki, A. Chang, C. Junho, and M. Gerla, "P2P content distribution to mobile Bluetooth users," *IEEE Trans. Veh. Technol.*, vol. 59, no. 1, pp. 356–367, Jan. 2010.

[8] S. Gokhale and P. Dasgupta, "Distributed authentication for peer-to-peer networks," in *Proc. Appl. Internet Workshops*, Jan. 27–31, 2003, pp. 347–353.

[9] H. Lee and K. Kim, "An adaptive authentication protocol based on reputation for peer-to-peer system," in *Proc. Symp. Crypto. Info. Sec.*, 2003, pp. 661–666.

[10] K. V. Nguyen, "Simplifying peer-to-peer device authentication using identity-based cryptography," in *Proc. ICNS*, 2006, p. 43.

[11] D. Balfanz, D. Smetters, P. Stewart, and H. Wong, "Talking to strangers: Authentication in ad hoc wireless networks," in *Proc. Symp. NDSS*, San Diego, CA, Feb. 2002.

[12] C.-Y. Huang, Y.-P. Chiu, K.-T. Chen, and C.-L. Lei, "Secure multicast in dynamic environments," *Comput. Netw.*, vol. 51, no. 10, pp. 2805–2817, Jul. 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B6VRG-4MM1P6H-5/1/fca29634eb800ee4f1ef830c09d8aefd

[13] M. Cagalj, S. Capkun, and J.-P. Hubaux, "Key agreement in peer-to-peer wireless networks," *Proc. IEEE*, vol. 94, no. 2, pp. 467–478, Feb. 2006.

[14] A. C. Squicciarini, F. Paci, E. Bertino, A. Trombetta, and S. Braghin, "Group-based negotiations in P2P systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 10, pp. 1473–1486, Oct. 2010.

[15] Y.-M. Tseng, "A secure authenticated group key agreement protocol for resource-limited mobile devices," *Comput. J.*, vol. 50, no. 1, pp. 41–52, Jan. 2007.

[16] A. Shamir and Y. Tauman, "Improved online/offline signature schemes," in *Proc. Adv. CRYPTO*, 2001, pp. 355–367.

[17] G.-H. Chiou and W.-T. Chen, "Secure broadcasting using the secure lock," *IEEE Trans. Soft. Eng.*, vol. 15, no. 8, pp. 929–934, Aug. 1989.

[18] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. ACM CCS*, 1993, pp. 62–73.

[19] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Proc. Adv. Cryptol.-EUROCRYPT*, 1997, pp. 256–266.

[20] D. Boneh, "The decision Diffie–Hellman problem," in *Proc. Algorithmic Number Theory*, 1998, pp. 48–63.

[21] M. Bellare, R. Guerin, and P. Rogaway, "XOR MACs: New methods for message authentication using finite pseudorandom functions," in *Proc. Adv. CRYPT0*, 1995, pp. 15–28.

[22] R. Rivest, "The RC5 encryption algorithm," in *Proc. Fast Softw. Encryption*, 1995, pp. 86–96.

[23] Y. Tsiounis and M. Yung, "On the security of ElGamal-based encryption," in *Proc. Public Key Cryptography*, 1998, pp. 117–134.

[24] Advanced Encryption Standard (AES), F. 197, Nov. 2001.

[25] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Commun. Mag.*, vol. 32, no. 9, pp. 33–38, Sep. 1994.

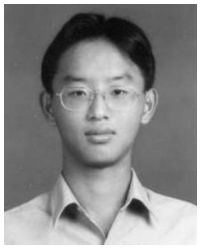
[26] B.-T. Oh, S.-B. Lee, and H.-J. Park, "A peer mutual authentication method using PKI on superpeer-based peer-to-peer systems," in *Proc. ICACT*, 2008, pp. 2221–2225.

[27] G. Networks, A White paper: Groove Security Architecture, 2002. [Online]. Available: http://www.groove.net/products/workspace/security.html

[28] D. Fahrenholtz and W. Lamersdorf, "Transactional security for a distributed reputation management system," in *Proc. ECommerce and Web Technol.*, 2002, pp. 214–223.

[29] C.-I. Fan and Y.-H. Lin, "Provably secure remote truly three-factor authentication scheme with privacy protection on biometrics," *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 4, pp. 933–945, Dec. 2009.

[30] C.-I. Fan, L.-Y. Huang, and P.-H. Ho, "Anonymous multireceiver identity-based encryption," *IEEE Trans. Comput.*, vol. 59, no. 9, pp. 1239–1249, Sep. 2010.



Lo-Yao Yeh (M'12) received the M.S. degree from the National Chi Nan University, Nantou, Taiwan, in 2005 and the Ph.D. degree from the National Chiao Tung University, Hsinchu, Taiwan, in 2010.

He was a Visiting Scholar with the University of California, Berkeley. He is currently an Associate Researcher with the National Center for High-Performance Computing, Hsinchu, Taiwan, and is also an Adjunct Assistant Professor with the Department of Information Management, National Chi Nan University, Nantou, Taiwan. His research interests include cryptography, network security, vehicular networks security, and Botnet.

Yu-Lun Huang (M'04) received the B.S. and Ph.D. degrees in computer science and information engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 1995 and 2001, respectively.

She is currently an Assistant Professor with the Department of Electrical and Control Engineering, National Chiao-Tung University. Her research interests include wireless security, secure testbed design, embedded software, embedded operating systems, risk assessment, secure payment systems, voice over Internet protocol, quality of service, and virtualization security.

Dr. Huang is a member of the Phi Tau Phi Society.

Anthony D. Joseph (M'98) received the Ph.D. degree from Massachusetts Institute of Technology (MIT), Cambridge, in 1998.

He was a Principal Designer with MIT, where he worked on the Rover mobile toolkit for providing application support for intermittent connectivity in a mobile computing environment. He is currently an Associate Professor with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, where he is developing more accurate network modeling and emulation tools and building proxy-based computing environments for mobile code and data. His research interests include computer systems and networking, particularly mobile systems, overlay networks, code and data migration, and network security.

Shiuhpyng Winston Shieh (SM'98) received the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park.

From 2003 to 2004 and from 2005 to 2006, he was a Visiting Professor with the University of California, Berkeley. He is currently a Professor with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, where he is also the Director of the Taiwan Information Security Center. His research interests include reliability and security hybrid mechanisms, network and system security, and software program behavior analysis.

Woei-Jiunn Tsaur (M'10) received the Ph.D. degree in electrical engineering from the National Taiwan University of Science and Technology, Taipei, Taiwan, in 1998.

From 1994 to 2003, he was a Project Manager and Technology Consultant with the R&D Division, Syscom Computer Engineering Company, Taipei, Taiwan, a research center of software development. Since 1999, he has been with the Department of Information Management, Da-Yeh University, Changhua, Taiwan, where he is currently a Full Professor. His research interests include network security, security topics in operating systems, applied cryptography, information security management, and computer networks.