LA^3: A Lightweight Accountable and Anonymous Authentication Scheme for Resource-constrained Devices

Wensheng Zhang[1] and Chuang Wang[2]

[1]Iowa State University, Ames, Iowa 50010, USA,
[2]Microsoft Inc.
wzhang@iastate.edu,chuwang@microsoft.com

Abstract. In order to provide a lightweight accountable and anonymous authentication solution for resource-constrained devices, this paper proposes a variant of group signature scheme called LA^3. The design is based on the assumptions of the DDH, q-SDH, q-DDHI and LRSW problems, as well as the knowledge of exponent assumption. A security model has been formally defined, and proofs have been provided to show that, the LA^3 scheme achieves the security properties of non-frameability, traceability and selfless anonymity in the random oracle model. We have also implemented the LA^3 scheme and compared its performance to that of a few classic group signature schemes. The results show that the LA^3 scheme achieves much higher computational efficiency.

1 Introduction

More and more services have been provided over the Internet or other forms of networks such as mobile ad hoc networks, sensor networks, home networks, and body-area networks. To ensure a service is accessed only by authorized clients, it is a common practice that a client first authenticates itself before its request is served. If service requests may leak private information of clients, it is further demanded that the authentication algorithm keep anonymity for clients; meanwhile, accountability must be achieved to prevent the anonymity mechanisms from being exploited maliciously.

Resource-constrained devices may be used in the client and/or server side to access/provide online services. For example, it has been popular for people to access Internet services pervasively by using mobile phones, smart watches or other tiny terminals, which typically have low computational capacity and prefer to run in low CPU cycles for the sake of power saving. As another example, a network of wireless sensors deployed in battlefield for surveillance may allow soldiers to access them, where the service devices (i.e., sensors or sensor gateways) and/or the client devices (i.e., mobile devices carried by soldiers) could be resource-constrained. In these scenarios, it is desired to deploy lightweight accountable and anonymous authentication algorithms.

Many accountable and anonymous authentication schemes [15,41,48,32,35,29] have been proposed for controlled access to online services. These schemes are mainly built upon classic group signature algorithms [19,4,5,17,11,10,18,16,43,38,40,30,25,3,13,14,27] which provides provable anonymity and traceability. However, the computational costs

introduced by these schemes may be too high for thin devices, as they were not designed for resource-constrained devices. Most of classic group signature primitives employ bilinear pairing, which are usually implemented over Elliptic curves. As shown by the works [44,26,47,50,22,45] on implementing Elliptic curve based cryptography on resource-constrained devices, bilinear pairing is much more complex and computationally expensive than the point addition and multiplication operations over Elliptic curve. Also, recent advances in group signature [25,33,31,34,20,6,12] have been focused on developing the primitives that are secure post-quantum, which are even more costly in terms of computational and communication overheads than the classic group signature primitives. Based on the above observations, we develop LA^3, a lightweight accountable and anonymous authentication scheme, in which each authentication transaction requires only a few efficient operations over cyclic groups and finite fields.

LA^3 scheme assumes three types of entities in the system: a service provider (called verifier) that needs to verify whether a client has the privilege to access its service, a group of clients (called provers) that need to prove their access privileges, and a trusted authority responsible for choosing system parameters and initializing the verifier and provers. Following the protocol of LA^3, a prover and the verifier can interact with each other in an authentication transaction; the scheme can also be used for clients to anonymously generate signatures for messages, and for the verifier to verify the signatures. The prover can keep anonymous to the verifier; but the authority is able to trace out the prover based on the authentication transcript when needed.

The security of LA^3 relies on the hardness assumptions of the Decisional Diffie-Hellman (DDH) [8], q-Strong Diffie-Hellman (q-SDH) [10], q-Decisional Diffie-Hellman Inversion (q-DDHI) [46] and LRSW problems [36], as well as the Knowledge of Exponent Assumption [7]. Intuitively, the LA^3 has the following security properties:

- *Non-frameability*. It is hard for the verifier and any coalition of provers to impersonate any innocent prover (i.e., any prover not belonging to the coalition) in an authentication transaction.
- *Traceability*. It is hard for any coalition of provers to succeed in an authentication transaction without revealing any of their IDs to the authority.
- *Selfless Anonymity*. It is hard for the verifier and any coalition of provers to determine which of two or more innocent provers involves in an authentication transaction.

Note that, LA^3 is a variant of classic group signature schemes, which have full traceability and selfless anonymity. Similar to the classic schemes, LA^3 provides selfless anonymity for provers and the tracing and revocation capabilities for the trusted authority, and it enables verifier local revocation. LA^3 differs from the classic group signature schemes in two main aspects. First, the classic schemes publish public group keys to allow everyone knowing the public key to perform verification, but LA^3 only allows the verifier to do so; this is the cost paid to achieve the simplicity and lightweight. However, this not a serious limitation to our target application scenarios of authentication for resource access [41], where only some resource manager (not all group members) needs to perform verification. Second, LA^3 provides a weaker traceability than classic schemes in that, the authority can trace an authentication transcript to a prover

only if the verifier is not malicious; that is, a malicious verifiable is able to forge a transcript that the authority cannot trace to any prover that it has initialized. However, this is not a problem to our target application scenario, because if an untraceable transaction is found, the verifier must be the only entity responsible for it; hence, this feature can deter a verifier from forging signatures. More over, LA^3 has the feature of non-frameability; that is, even the verifier can forge a transaction, it (even colluding with some clients) cannot frame an innocent client by forging a transaction tracing to the innocent client.

We have implemented the LA^3 scheme, and compared its performance to that of a few classic group signature schemes. The results show that, as operations needed in the authentication process are simpler, the computational efficiency of LA^3 scheme is much higher than that of the compared classic group signature schemes; particularly, it is more than 10 times faster than the group signature scheme proposed by Boneh and Shacham [11], which has wide applications.

The rest of the paper is organized as follows. Section 2 formally defines the problem. Section 3 elaborates our design. Section 4 presents the security properties. Section 5 reports the performance evaluation results. Section 6 summarizes the related work and compares LA^3 scheme with them. Finally, Section 7 concludes the paper.

2 Problem Definition

2.1 Notations and Assumptions

Let $\mathbb{Z}_p$ denote a prime finite field, where $p > 2^\kappa$ and κ is a security parameter; let $\mathbb{G}$ be a multiplicative cyclic group of p elements, with g as a generator. Our proposed design is based on the following hard problems and assumptions:

Decisional Diffie-Hellman (DDH) Problem [8]: given g, g^a, g^b and g^c in $\mathbb{G}$, determine if $c = ab$.

q-Strong Diffie-Hellman (q-SDH) Problem [10]: given g, g^x, g^{x^2}, $\cdots$, g^{x^q} in $\mathbb{G}$ find $(c, g^{1/(c+x)})$ where $c \in \mathbb{Z}_p$.

q-Decisional Diffie-Hellman Inversion (q-DDHI) Problem [46]: Given g, g^x, g^{x^2}, $\cdots$, g^{x^q} and $g^{1/(x+y)}$ in $\mathbb{G}$, determine if $y = 0$.

LRSW Problem [36]: Given g, g^x and g^y in $\mathbb{G}$ and oracle O which on input s returns $(g', (g')^{sy}, (g')^{x+sxy})$ where $g' = g^z$ for some $z \in \mathbb{Z}_p$, compute $(b, t, b^{ty}, b^{x+txy})$ where $b \neq g^0 \in \mathbb{G}$ and t is not one of the s that has been queried.

Knowledge of Exponent Assumption (KEA) [7]. We use the following general form: For any adversary $\mathbf{A}$ that takes input (g_i, g_i^s) for $i = 1, \cdots, n$ and returns group element (C, Y) such that $Y = C^s$, there exists an "extractor" $\bar{\mathbf{A}}$ which, given the same inputs as $\mathbf{A}$, returns $x_i \in \mathbb{Z}_p$ for $i = 1, \cdots, n$ such that $C = \prod_{i=1}^{n} g_i^{x_i}$.

2.2 Scheme Overview

We consider a system that is composed of a verifier, a set of provers, and an off-line trusted authority which is responsible for initializing the verifier and provers as well as tracing the provers. Our proposed scheme includes the following algorithms.

- *System Initialization*, with which the trusted authority initializes system secrets. It is formally denoted as $SystemInit(1^{\kappa}) \rightarrow \mathbb{SS}$, where κ is a security parameter and $\mathbb{SS}$ is a set of system secrets.
- *Verifier Initialization*, with which the authority initializes the *verifier*. It is formally denoted as $VerifierInit\ (\mathbb{SS}) \rightarrow \mathbb{VK}$, where $\mathbb{VK}$ is the verification key.
- *Prover Initialization*, with which the authority initializes a prover. It is formally denoted as $ProverInit\ (\mathbb{SS},\ u) \rightarrow (u, \mathbb{PK}_u, T_u)$, where u is the ID of the new prover, $\mathbb{PK}_u$ is the proof key and T_u is the tracing token. The authority gives the prover u and $\mathbb{PK}_u$, but keeps T_u for tracing or revoking the prover when needed.
- *Authentication Protocol*, with which prover u authenticates itself with the verifier. It includes the following two algorithms:
  - $Prove(\mathbb{PK}_u, \tilde{c}) \rightarrow (\tilde{r})$. This algorithm is used by prover u, who holds proof key $\mathbb{PK}_u$, takes challenge $\tilde{c}$ from the verifier, and outputs response $\tilde{r}$.
  - $Verify(\mathbb{VK}, \mathbb{RT}, \tilde{c}, \tilde{r}) \rightarrow 1/0$. This algorithm is used by the verifier. It holds verification key $\mathbb{VK}$, takes as inputs the revocation token set $\mathbb{RT}$ (i.e., the set of tokens of all revoked provers) and the authentication transcript $(\tilde{c},\ \tilde{r})$, and outputs 1 if the verification is a success or 0 otherwise.
- *Prover Tracing*, with which the authority traces the identity of a prover based on an authentication transcript. It is formally denoted as $Trace(\mathbb{T},\ \tilde{c},\ \tilde{r}), \rightarrow u$, which takes as inputs the set $\mathbb{T} = \{T_u | \forall u\}$ of all provers' tokens and an authentication transcript $(\tilde{c},\ \tilde{r})$, and outputs the ID u of the prover who generated response $\tilde{r}$.

2.3 Security Definitions

Our design aims to achieve the following security properties:

Correctness

Definition 1. *The scheme is correct if: the authentication transcript generated by the verifier and a prover u that has not been revoked, must be verified successfully. Formally,*

$$(Prove(\mathbb{PK}_u, \tilde{c}) \rightarrow \tilde{r}) \Rightarrow [(Verify(\mathbb{VK}, \mathbb{RT}, \tilde{c}, \tilde{c}) \rightarrow 1) \vee (T_u \in \mathbb{RT})]. \quad (1)$$

Non-frameability Intuitively, non-frameability defines the property that, the verifier and any set of collusive provers cannot impersonate any innocent prover. Its formal definition is as follows.

Definition 2. *A scheme is (t, q_A, ϵ) non-frameable if no adversary can win the following* Non-frameability Game *with a probability greater than ϵ, in time t with no more than q_A authentication queries on each prover.*

Non-frameability Game The game is between an adversary and a challenger, and composed of the following phases.

- **Phase I: Initialization.** The challenger initializes one verifier and a set of n provers.
- **Phase II: Queries.** The adversary can issue the following types of queries and the challenger should respond accordingly.
  - *Corruption of the verifier:* The adversary issues a corruption query on the verifier. In response, the challenger returns the verification key.
  - *Corruption of a prover u:* The adversary issues a corruption query on a prover u. The challenger responds with the proof key of the prover.
  - *Authentication for a prover v:* The adversary issues an authentication query for a prover v, and provides a challenge $\tilde{c}$. In response, the challenger returns a valid response $\tilde{r}$ on behalf of prover v.
  - *Hash:* The adversary issues a hash query, and the challenge responds with an element of $\mathbb{Z}_p$ randomly and consistently.
- **Phase III: Adversary's Response.** The adversary provides transcript $\tilde{c}'$ and $\tilde{r}'$.

The adversary wins if the response satisfies the following conditions: (i) The authentication is successful; i.e., $Verify(\mathbb{VF}, \emptyset, \tilde{c}', \tilde{r}') = 1$. (ii) The authentication transaction traces to a prover. (iii) If the authentication transcript traced to prover w, then no authentication query has been made for w with challenges $\tilde{c}'$, and no corruption query has been made on prover w.

Traceability Intuitively, traceability defines the property that, authentication transcripts forged by any coalition of collusive provers must trace to one of these provers; note that, the verifier is not part of the coalition. Its formal definition is as follows.

Definition 3. *A scheme is (t, ϵ) traceable if no adversary can win the following* Traceability Game *with a probability greater than ϵ in time t.*

Traceability Game The game is between an adversary and a challenger, and composed of the following phases.

- **Phase I: Initialization.** The challenger initializes one verifier and a set of provers.
- **Phase II: Pre-challenge Queries.** The adversary can issue three types of queries, namely, *corruption of a prover*, *authentication for a prover*, and *hash*. and the challenger responds accordingly as in the *Non-frameability Game*.
- **Phase III: Challenge.** The challenger provides a challenge $\tilde{c}'$.
- **Phase IV: Post-challenge Queries.** This is the same as Phase II.
- **Phase V: Adversary's Response.** The adversary responds with $\tilde{r}'$.

The adversary wins if the response satisfies the following conditions: (i) $Verify(\mathbb{VK}, \emptyset, \tilde{c}', \tilde{r}') = 1$. (ii) The authentication transcript cannot trace to any prover which has been corrupted or has been queried for authentication; i.e., if $\mathbb{P}$ denotes such set of provers, then $Trace(\mathbb{T}, \tilde{c}', \tilde{r}') \rightarrow u$ where $u \in \mathbb{P}$.

Selfless Anonymity Intuitively, selfless anonymity defines the property that, the verifier and any coalition of collusive provers cannot determine which innocent prover involve in an authentication process; thus, the anonymity of prover is achieved. Its formal definition is as follows.

Definition 4. *A scheme is* (t, q_A, ϵ) *selfless anonymous if no adversary can win the following* Selfless Anonymity Game *with a probability greater than* ϵ*, in time* t *with less than* q_A *authentication queries on each prover.*

Selfless Anonymity Game The game is between an adversary and a challenger, and composed of the following phases.

- **Phase I: Initialization.** A verifier and a set of provers.
- **Phase II: Pre-Challenge Queries.** The adversary can issue four types of queries, namely, *corruption of the verifier*, *corruption of a prover*, *authentication for a prover*, and *hash*, which are responded by the challenger as in the *Non-frameability Game*.
- **Phase III: Challenge.** The adversary selects two provers u_0 and u_1 from $\mathbb{P}$ that have not been compromised. The challenger randomly picks i from 0 or 1, and presents a response generated by u_i.
- **Phase IV: Post-Challenge Queries.** The same as Phase II except that corruption queries cannot be made on provers u_0 or u_1.
- **Phase V: Adversary's Response.** The adversary returns $i' \in \{0, 1\}$.

The adversary wins if $i' = i$.

3 Our Construction

The **intuition** of the proposed construction is as follows: The authority randomly picks numbers k_1, d and l and a polynomial $C(x)$ from $\mathbb{Z}_p$. Each prover in the system is associated with a unique set of randomly selected numbers and polynomials including (i) ID u, (ii) numbers λ_u, s_u and e_u, and (iii) polynomial $B_u(x)$; based on the above, prover u is also associated with polynomial $F_u(x)$ such that

$$\lambda_u(k_1 + k_2 s_u + 1)C(x) + B_u(x)d + e_u l + F_u(x) = 0, \text{ where } k_2 = 1 - k_1. \quad (2)$$

The authority initializes each prover u by assigning to it a proof key that encodes λ_u, s_u, $B_u(x)$, e_u and $F_u(x)$, and initializes the verifier by assigning to it a verification key that encodes k_1, d, l and $C(x)$. Note that, some of these numbers and polynomials are not given in plain text, but *encoded* into the exponents of group elements in order to achieve the afore-defined security properties. In every authentication transaction, the verifier provides some challenge that is never reused. The prover generates response based on the challenge and its proof key, to prove its knowledge of the key without exposing its identity or the key. The verifier can determine if the prover has the required key through some test derived from Eq. (2). The scheme is presented in detail as follows.

3.1 System Initialization

The trusted authority initializes the system by selecting c_0, c_1, d, k_1 and l randomly from $\mathbb{Z}_p$ where $p \geq 2^\kappa$, and let $k_2 = 1 - k_1$. Formally, the procedure of system initialization can be specified as $SystemInit(1^\kappa) \rightarrow \mathbb{SS} = \{c_0, c_1, d, k_1, k_2, l\}$.

3.2 Verifier Initialization

The authority provides the following secrets to the verifier: k_1, k_2, l, $C(x)$, and $\hat{d} = g^d$. The procedure of verifier initialization can be formally specified as $VerifierInit(\mathbb{SS}) \rightarrow \mathbb{VK} = \{k_1, k_2, l, c_0, c_1, \hat{d}\}$.

3.3 Prover Initialization

Each prover is given a unique ID u and the following secrets:

- $\hat{k}_{u,0} = g^{\lambda_u}$ and $\hat{k}_{u,1} = \hat{k}_{u,0}^{k_1}$, where $\lambda_u \xleftarrow{R} \mathbb{Z}_p$;
- $s_u \xleftarrow{R} \mathbb{Z}_p$; $b_u \xleftarrow{R} \mathbb{Z}_p$; $\hat{e}_u = g^{e_u}$ where $e_u \xleftarrow{R} \mathbb{Z}_p$;
- $\hat{F}_{u,1} = g^{f_{u,1}}$, $\hat{F}'_{u,1} = g^{f'_{u,1}}$, $\hat{F}_{u,0} = g^{f_{u,0}}$ and $\hat{F}'_{u,0} = g^{f'_{u,0}}$, where $f_{u,1}, f'_{u,1}, f_{u,0}, f'_{u,0} \in \mathbb{Z}_p$ and $F_u(x) = (f_{u,1} + f'_{u,1})x + (f_{u,0} + f'_{u,0})$ satisfies Eq. (2).

Formally, the procedure of initializing prover u is specified as $ProverInit(\mathbb{SS}, u) \rightarrow (\mathbb{PK}_u, T_u)$, where $\mathbb{PK}_u = \{\hat{k}_{u,0}, \hat{k}_{u,1}, s_u, B_u(x), \hat{e}_u, \hat{F}_{u,1}, \hat{F}'_{u,1}, \hat{F}_{u,0}, \hat{F}'_{u,0}\}$ and $T_u = \{\lambda_u, s_u, b_u\}$. Note that T_u is not provided to prover u but is kept by the authority for the purpose of tracing or revoking a prover.

3.4 Authentication Protocol

The authentication protocol runs as follows.

1. When a prover u sends a verification request to the verifier.
2. The verifier randomly picks a challenge $r'_1 \in \mathbb{Z}_p$ and sends it to the prover.
3. The prover works as follows to generate a response.
   (a) It picks $r'_2 \xleftarrow{R} \mathbb{Z}_p$, and computes $r = h(r'_1|r'_2)$, where $h()$ is a hash function mapping arbitrary strings to $\mathbb{Z}_p$.
   (b) It picks $\alpha, \beta, \xi \xleftarrow{R} \mathbb{Z}_p$, and computes and sends the following to the verifier:

$$a_{u,1,r} = 2\alpha + \xi - 1, \;\; a_{u,2,r} = \alpha(s_u + 1) + \xi - 1, \;\; B_{u,r} = \alpha\beta B_u(r), \quad (3)$$

$$\hat{k}_{u,0,r} = \hat{k}_{u,0}^{\beta}, \;\; \hat{k}_{u,1,r} = \hat{k}_{u,1}^{\beta}, \;\; \hat{k}_{u,2,r} = (\hat{k}_{u,0})^{-\xi}, \;\; \hat{k}_{u,3,r} = (\hat{k}_{u,1})^{-\xi}, \quad (4)$$

$$\hat{e}_{u,r} = (\hat{e}_u)^{\alpha\beta}, \;\; \hat{F}_{u,r} = [(\hat{F}_{u,1}\hat{F}'_{u,1})^r(\hat{F}_{u,0}\hat{F}'_{u,0})]^{\alpha\beta}. \quad (5)$$

4. The verifier tests if

$$\hat{k}_{u,1,r} = \hat{k}_{u,0,r}^{k_1}, \;\; \hat{k}_{u,3,r} = \hat{k}_{u,2,r}^{k_1}, \quad (6)$$

and

$$(\hat{k}_{u,0,r}^{k_1 a_{u,1,r} + k_2 a_{u,2,r} + 1}\hat{k}_{u,2,r})^{C(r)}\hat{d}^{B_{u,r}}\hat{e}_{u,r}^{l}\hat{F}_{u,r} = g^0. \quad (7)$$

If the above tests are successful, the verifier will check if the prover has been revoked as detailed in Section 3.6. If the prover is not revoked, the verification succeeds.

Hence, the authentication protocol can be formally expressed as

$$Prove(\mathbb{PK}_u, \tilde{c}) \rightarrow \tilde{r}, \tag{8}$$

$$Verify(\mathbb{VK}, \tilde{c}, \tilde{r}) \rightarrow 1/0, \tag{9}$$

where $\tilde{c} = \{r'_1\}$ and $\tilde{r} = \{r'_2, a_{u,1,r}, a_{u,2,r}, \hat{k}_{u,0,r}, \hat{k}_{u,1,r}, \hat{k}_{u,2,r}, \hat{k}_{u,3,r}, B_{u,r}, \hat{e}_{u,r}, \hat{F}_{u,r}\}$.

3.5 Tracing Algorithm

The authority keeps $T_i = \{s_i, \lambda_i, b_i\}$ for each prover i that it has initialized. Given $\hat{k}_{u,0,r}$, $a_{u,1,r}$, $a_{u,2,r}$ and $B_{u,r}$ responded by a prover during authentication, the authority traces the prover as follows. For each prover ID i,

$$\alpha' = (a_{u,2,r} - a_{u,1,r})/(s_i - 1). \tag{10}$$

Then, if

$$\hat{k}_{u,0,r}^{\alpha' B_i(r)/(\lambda_i B_{u,r})} = g, \tag{11}$$

the prover involved in the authentication transaction is traced to prover i.

Formally, the tracing procedure can be formally expressed as $Trace(\mathbb{T}, \tilde{c}) \rightarrow i$, where $\mathbb{T} = \{T_i | \forall \text{ prover } i\}$.

3.6 Revocation

For each revoked prover v, revocation token T_v is provided to the verifier. After a prover has passed the test expressed in equation (7), formula (10) is computed and then equation (11) is tested to find if the prover in the verification procedure has been revoked.

4 Security Proofs

This section report the main results of security analysis. Due to space limit, we leave the detailed proofs to the appendixes in the full version of this paper [49].

4.1 Correctness

The correctness of the design is stated in Theorem 1 and proved in the following.

Theorem 1. *The LA^3 scheme is correct.*

Proof. Consider that the verifier provides a challenge r'_1, and then a prover u honestly executes $Prove(\mathbb{PK}_u, r'_1)$ and produces r'_2, $r = h(r'_1|r'_2)$, $\hat{k}_{u,0,r}$, $\hat{k}_{u,1,r}$, $\hat{k}_{u,2,r}$, $\hat{k}_{u,3,r}$, $a_{u,1,r}$, $a_{u,2,r}$, $B_{u,r}$, $\hat{e}_{u,r}$, and $\hat{F}_{u,r}$. It holds that

$$[(\hat{k}_{u,0,r})^{k_1 a_{u,1,r} + k_2 a_{u,2,r} + 1} \hat{k}_{u,2,r}]^{C(r)} = g^{\lambda_u \alpha \beta (k_1 + k_2 s_u + 1) C(r)}, \tag{12}$$

and

$$\hat{d}^{B_{u,r}} = g^{d\alpha\beta B_u(r)}, \hat{e}_{u,r}^{l} = g^{\alpha\beta e_u l}, \hat{F}_{u,r} = g^{\alpha\beta F_u(r)}. \tag{13}$$

Hence, due to Equation (2),

$$(g^{\alpha\beta})^{\lambda_u(k_1+k_2 s_u+1)C(r)+B_u(r)d+e_u l+F_u(r)} = g^0. \quad (14)$$

Next, the verifier runs the revocation algorithm as follows. For each revocation token $T_v = (s_v, \lambda_v, b_v) \in \mathbb{RT}$ and $\gamma = \hat{k}_{u,0,r}^{\alpha' B_v(r)/(\lambda_v B_{u,r})} = g^{\frac{\lambda_u(s_u-1)(v\cdot r+b_v}{\lambda_v(s_v-1)(u\cdot r+b_u)}}$, if $T_v = T_u = (s_u, \lambda_u, b_u)$ then $\gamma = g$; in this case, the verifier identifies prover u as a revoked prover and outputs 0. Otherwise (i.e., $T_v \neq T_u$), since λ_u, λ_v, s_u, s_v, b_u and b_v are all randomly picked from $\mathbb{Z}_p$, the probability for $\gamma = g$ is $1/p$ which is negligible; so the verifier outputs 1. To summarize,

$$(Prove(\mathbb{PK}_u, \tilde{c}) \rightarrow \tilde{r}) \Rightarrow [(Verify(\mathbb{VK}, \mathbb{RT}, \tilde{c}, \tilde{c}) \rightarrow 1) \vee (T_u \in \mathbb{RT})].$$

4.2 Non-frameability

Theorem 2. *If the q-SDH problem is (t, ϵ)-hard, i.e., it cannot be solved with a probability greater than ϵ in time t, then the LA^3 scheme is $(t', q+1, n\epsilon)$ non-frameable where $t' = \Theta(1) \cdot t$ and n is the total number of provers.*

Proof. See Appendix 1 in [49]. The main idea is that, if an adversary $\mathcal{A}$ wins the non-frameability game in time t, an algorithm $\mathcal{B}$ can be constructed to solve the q-SDH problem in $\Theta(1) \cdot t$.

4.3 Traceability

To facilitate the proof of traceability, we first extend the LRSW problem and show its equivalence to the original LRSW problem.

Definition 5. *Extended LRSW Problem: given g, g^k, g^{c_1}, $g^{c_0} \in \mathbb{G}$ (k, c_1, $c_0 \in \mathbb{Z}_p$), and oracle O which on input s returns g', $(g')^k$, $(g')^{[k\cdot(s-1)+2]c_1}$ and $(g')^{[k\cdot(s-1)+2]c_0}$, where $g' = g^z$ for some $z \in \mathbb{Z}_p$, to compute g'', t, r, $(g'')^k$ and $(g'')^{[k\cdot(t-1)+2](c_1 r+c_0)}$, where $g'' \neq g^0 \in \mathbb{G}$ and t is different from any s that has been queried.*

Lemma 1. *If the LRSW problem is (t, ϵ)-hard, then the extended LRSW problem is (t', ϵ)-hard where $t' = \Theta(1) \cdot t$.*

Proof. See Appendix 2 in [49].

The following theorem states that LA^3 scheme meets the requirement of traceability.

Theorem 3. *If the LRSW problem and the DDH problem are (t, ϵ)-hard, the proposed authentication scheme is (t', ϵ) traceable where $t' = \Theta(1) \cdot t$.*

Proof. See Appendix 3 in [49]. The proof considers two cases: (1) if $\mathcal{A}$ wins by providing a response that traces to a faked s_w of some faked prover w, an algorithm $\mathcal{B}$ can be constructed to solve the extended LRSW problem; (2) if $\mathcal{A}$ wins by providing a response that traces to an existing s_u but a faked $b_w \neq b_u$, an algorithm $\mathcal{B}$ can be constructed to solve the DDH problem. In both cases, we use the KEA3 assumption.

4.4 Selfless Anonymity

The following theorem states that LA^3 meets the requirement of selfless anonymity.

Theorem 4. *Suppose the q-DDHI (q-Decisional Diffie-Hellman Inversion) problem is (t, ϵ)-hard; i.e., no any algorithm can solve the problem with advantage greater than ϵ in time t. Then the LA^3 scheme is $(t', m, 0.5 + 2n^2\epsilon)$ selflessly anonymous where $t' = \Theta(1) \cdot t$, $m = \lfloor q/2 \rfloor + 1$ and n is the total number of provers.*

Proof. See Appendix 4 in [49]. The proof shows that, if there is adversary algorithm $\mathcal{A}$ winning the selfless anonymity game with advantage ϵ' in time t', an algorithm $\mathcal{B}$ can be constructed to solve the q-DDHI problem with advantage $0.5\epsilon'/n^2$ in time $\Theta(1) \cdot t'$. This is equivalent to that, if q-DDHI problem is (t, ϵ)-hard, there is no algorithm can win the selfless anonymity game with advantage $2n^2\epsilon$ in time $\Theta(1) \cdot t$.

5 Implementation and Evaluation

5.1 Implementation

We have implemented the LA^3 scheme based on an elliptic cyclic group. Specifically, we use the elliptic cryptographic primitives contributed by FlexiProvider [28]. When using the ECC libraries, we adopt the recommended elliptic curve parameters specified by secp160r1 [42].

We compare the performance of LA^3 with that of the group signature scheme proposed by Boneh and Shacham [11], denoted as the BS scheme in this paper. We implemented BS based on java paring-based cryptographic (jPBC) library to enable the operations on the bilinear maps [2,23]. We adopted the type A curve in our implementation, which is the fastest curve to compute the pairing operations based on the benchmark results from jPBC [1]. Note that, the above settings make both LA^3 and BS to have the same 80-bit level of security.

5.2 Performance Comparison with the BS Scheme [11]

We measured the performance of LA^3 and BS on a laptop computer with 1.83 GHz Genuine Intel (R) processor and 3 GB of RAM. The experimental results reported below are the averaged results of over 100 experimental runs.

Computational efficiency BS spends about 1611 milliseconds to generate a group signature and about 1807 milliseconds to verify a group signature. In contrast, LA^3 spends only about 66 milliseconds at the prover and the verifier side, respectively, for each authentication transaction. LA^3 outperforms BS because BS needs pairing operations while LA^3 does not, and also the exponential and pairing operations over groups support bilinear mapping are much more expensive than the exponential operation over elliptic cyclic group. Particularly, for bilinear map $e : G \times G \rightarrow G_T$ that we use on type A curve, an exponentiation computation on group G takes 115 milliseconds and a pairing computation takes 150 milliseconds, while an exponential computation on elliptic curve secp160r1 only takes 12 milliseconds. To check whether a prover has been

revoked or not, both BS and LA^3 require only the verifier to check a revocation list. Table 1 compares the revocation cost between the two schemes: LA^3 is more efficient than BS; particularly, LA^3 needs only 4.4% of the time needed by BS.

Table 1. Checking Time (millisecond) vs. Number of Revoked Provers

Number of Revoked Provers	10	20	30	40	50	60	70	80
Time for Checking Revoked Provers (BS Scheme)	2965	5878	8859	11754	14772	17660	20910	23932
Time for Checking Revoked Provers (LA^3 Scheme)	132	262	401	522	655	785	926	1061

Bandwidth consumption In terms of bandwidth consumption for each verification transaction, BS needs to transmit 2 elements from the bilinear group and 5 elements from $\mathbb{Z}_p$. Based on the type A curve that we use in the experiment, each element from $\mathbb{Z}_p$ takes 20 bytes and each element from the bilinear group takes 128 bytes. Hence, the total signature size is 356 bytes. As the element from the bilinear group can be denoted as compressed version, which is 65 bytes, the total length of a group signature in BS is 230 bytes in the compressed format. Note that the length of group signature in BS scheme can be shorter if other curves such as type D curve is used [1]. However, this benefit is paid by the overhead of degraded computational efficiency. In order to show the computational efficiency of the proposed scheme, we choose type A curve when implementing BS scheme as type A curve is the fastest curve for pairing operation.

With LA^3, the prover needs to submit 6 points from the elliptic curve and 4 elements from $\mathbb{Z}_p$. As the secp160r1 elliptic curve we used in the experiment is 160-bit elliptic curve, each element from $\mathbb{Z}_p$ takes 20 bytes and each point from elliptic curve can be represented by 41 bytes. Hence, the total bandwidth consumption of LA^3 is 326 bytes. Note that, each point from elliptic curve can also be represented by the compressed version, which takes 21 bytes. Hence, by using the compressed representation, the bandwidth consumption of LA^3 is 206 bytes.

5.3 Performance Comparison with Other VLR Group Signature Schemes

As LA^3 is a verifier-local revocation (VLR) group signature scheme, we also analyze the computational costs of other VLR group signature schemes such as [38,39,40] and compare their costs with that of LA^3 in Table 2. As we can see, LA^3 is much more efficient than all these schemes because (i) LA^3 needs no pairing operation and smaller number of exponential operations, and (ii) the exponential operation over a regular multiplicative group is much more efficient than that in a group support pairing operations.

6 Related Work

Group signature schemes Since Chaum and van Heyst introduced the group signature concept in 1991 [21], researchers have proposed a large number of group signature schemes [19,4,5,17,11,10,18,16,43,38,40,30,25,37,24,3,13,14,27].

Table 2. Comparison of Computational Costs among Group Signature Schemes [11,38,39,40]

Schemes	Signing	Verification
LA^3	$6E$	$(6+\|RL\|)E$
BS Scheme [11]	$8E$+$2P$	$6E$+$(3+2\|RL\|)P$
Scheme [38]	$10E$+$1P$	$6E$+$(2+\|RL\|)P$
Scheme [39]	$6E$+$1P$	$3E$+$(2+2\|RL\|)P$
Scheme [40]	$(6+8\|RL\|)E$	$(9+8\|RL\|)E+3\|RL\|P$

E and P: exponential and pairing operations, respectively; $|RL|$: number of revoked users.

Based on the cryptographic assumptions they rely on, the classic group signature schemes can be briefly summarized as follows. Some schemes [4,5,17,16,43] are based on the strong RSA and decisional Diffie Hellman assumptions. An exemplified scheme was proposed by Ateniese *et al.* [4], which was improved to support membership revocation in [5,17] at the cost of degraded efficiency. Camenisch and Groth [16] then proposed a scheme with an order of magnitude better efficiency and support of revocation. Later on, Boneh *et al.* designed short group signature schemes based on bilinear maps [11,10]. The security of the design relies on the strong Diffie-Hellman and the Decision Linear assumptions. Since then, various other group signatures built on bilinear maps have been proposed in [18,3,38,13,14,27,40,37,24].

Based on the approaches for revocation, the classic group signature schemes can also be summarized as follows. In early group signature schemes, when revocation occurs, the authority needs to change the group public key, and redistributes the secret keys of all non-revoked members [19,4]. An improved revocation mechanism was to broadcast a message to all signers and verifiers such that non-revoked members can update their secret keys while revoked users cannot [17,10,16]. Recently, schemes [37,24], in which the revocation overhead is constant independently of the number of revoked members, have been proposed. However, the feature of constant revocation overhead in scheme [37] is achieved in the cost of large storage overhead, as the size of its public key is $O(\sqrt{N})$, where N is the total number of group members. Scheme [24] shifts the quadratic computational cost $O(NR)$ to the group manager, where N and R are the total number of group members and number of revoked members; both the signer and verifier then need to update some information computed by the group manager for generating and verifying signature respectively. With a different approach, revocation can only affect the verifier [5,11,38,39,40], provers do not need to know the revocation, and the authority needs not to be contacted by provers for revocation; however, the cost for verifying a signature scheme has to be linearly dependent on the number of revoked users. Such approach is called verifier local revocation (VLR).

Comparing to the above group signature schemes, our proposed LA^3 is more computationally efficient than any of them, and LA^3 is a VLR scheme in which revocation is transparent to provers. As scheme [11] (called BS scheme hereafter) is the most similar to LA^3 and it has many applications [41,48,32,35], we provide the following more detailed comparison: LA^3 is similar to the BS scheme in that, they both provide selfless anonymity for provers (signers) and the tracing and revocation capabilities for the trusted authority, and they are both VLR schemes. They are different in that, the BS scheme can publish public group keys to allow everyone knowing the public key to perform verification. LA^3, on the other hand, only allows the so-called verifier, which is

typically the group manager in practical systems, to verify signatures. However, LA^3 has the feature of non-frameability; that is, it is hard for any coalition of group members (including the group manager) to impersonate an innocent group member. Therefore, we expect LA^3 will still be useful in many practical scenarios due to the following reasons: First, in many cases (e.g., authentication for resources access [41]), only the group manager needs to perform verification. Second, with LA^3, only group manager can forge fake signature that cannot trace to any valid group member; hence, if an untraceable signature is found, it can be determined that the group manager must be responsible for it. This feature can deter a group manager from forging signatures.

More recently, research on group signature [25,33,31,34,20,6,12] has been focused on developing schemes that remain secure post-quantum.

Accountable and anonymous authentication schemes As group signature schemes provide anonymity and traceability, numerous accountable and anonymous authentication schemes have been proposed based on the application of group signature schemes. For example, EPID [15], PEACE [41] and schemes [48,32,35] are proposed on top of the short group signature schemes [11,10] devised by Boneh *et al*. These schemes need to conduct pairing and exponentiation operations over bilinear maps, which are computational extensive. Hence, considerable delay and computational overheads may be introduced when the prover and/or the verifier are resource-constrained. Compared to these schemes, the LA^3 scheme is more efficient and thus provides a lightweight alternative for accountable and anonymous authentication.

7 Conclusions

The paper presents LA^3, a lightweight accountable and anonymous authentication scheme for resource-constrained devices. The proposed design is based on the hardness of the DDH, q-SDH, q-DDHI and LRSW problems, as well as the knowledge of exponent assumption. We have proved that the LA^3 scheme has the security properties of non-frameability, traceability and selfless anonymity in the random oracle model. The LA^3 scheme has also been implemented and compared to several classic group signature schemes. The results showed that the LA^3 achieves much higher computational efficiency, which makes it more applicable to resource-constrained devices.

References

1. Benchmark.
   http://libeccio.dia.unisa.it /projects /jpbc /benchmark.html
2. Java pairing-based cryptography library. http://gas.dia.unisa.it/projects/jpbc/
3. Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385 (2005), `http://eprint.iacr.org/`
4. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: CRYPTO. pp. 255–270 (2000)
5. Ateniese, G., Song, D.X., Tsudik, G.: Quasi-efficient revocation in group signatures. In: Financial Cryptography. pp. 183–197 (2002)
6. Bansarkhani, R., Misoczki, R.: G-merkle: A hash-based group signature scheme from standard assumptions. IACR Cryptology ePrint Archive (2018)

7. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. Proceedings of CRYPTO'04 pp. 273–289 (2004)
8. Boneh, D.: The decision-diffie-hellman problem. ANTS-III pp. 48–63 (1998)
9. Boneh, D., Boyen, X.: Short signatures without random oracles. Proceedings of Eurocrypt'04 pp. 56–73 (May 2004)
10. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: CRYPTO. pp. 41–55 (2004)
11. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: CCS. pp. 168–177 (2004)
12. Boneh, S. Eskandarian, and B. Fisch, D.: Post-quantum epid group signatures from symmetric primitives. IACR Cryptology ePrint Archive (2018)
13. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: EUROCRYPT. pp. 427–444 (2006)
14. Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Public Key Cryptography. pp. 1–15 (2007)
15. Brickell, E., Li, J.: Enhanced privacy id from bilinear pairing for hardware authentication and attestation. In: SOCIALCOM 2010 (2010)
16. Camenisch, J., Groth, J.: Group signatures: Better efficiency and new theoretical aspects. In: SCN. pp. 120–133 (2004)
17. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: CRYPTO. pp. 61–76 (2002)
18. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: CRYPTO. pp. 56–72 (2004)
19. Camenisch, J., Michels, M.: A group signature scheme based on an rsa-variant. Tech. rep. (1998)
20. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha, M.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. ACM CCS pp. 1825–1842 (2017)
21. Chaum, D., van Heyst, E.: Group signatures. In: EUROCRYPT. pp. 257–265 (1991)
22. Cheng, Z.: Implementing pairing-based cryptosystems in usb tokens. IACR Cryptology ePrint Archive (2014)
23. De Caro, A., Iovino, V.: jpbc: Java pairing based cryptography. In: Computers and Communications (ISCC), 2011 IEEE Symposium on (2011)
24. Fan, C.I., Hsu, R.H., Manulis, M.: Group signature with constant revocation costs for signers and verifiers. In: Cryptology and Network Security, Lecture Notes in Computer Science, vol. 7092, pp. 214–233. Springer Berlin / Heidelberg (2011)
25. Gordon, S.D., Katz, J., Vaikuntanathan, V.: A group signature scheme from lattice assumptions. In: ASIACRYPT. pp. 395–412 (2010)
26. Gouvea, C., Lopez, J.: Software implementation of pairing-based cryptography on sensor networks using the msp430 microcontroller. INDOCRYPT, Lecture Notes in Computer Science 5922 (2009)
27. Groth, J.: Fully anonymous group signatures without random oracles. In: ASIACRYPT. pp. 164–180 (2007)
28. Group, F.R.: Flexiprovider. `http://www.cdc.informatik.tu-darmstadt.de/flexiprovider/`
29. He, D., Bu, J., Zhu, S., Yin, M., Gao, Y., Wang, H., Chan, S., Chen, C.: Distributed privacy-preserving access control in a single-owner multi-user sensor network. In: INFOCOM 2011 (2011)
30. Libert, B., Yung, M.: Dynamic fully forward-secure group signatures. In: ASIACCS. pp. 70–81 (2010)

31. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang, B.: Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. ASIACRYPT pp. 373–403 (2016)
32. Lin, X., Sun, X., Ho, P.H., Shen, X.: Gsis: A secure and privacy-preserving protocol for vehicular communications. Vehicular Technology, IEEE Transactions on 56(6), 3442–3456 (2007)
33. Ling, K. Nguyen, and H. Wang, S.: Group signatures from lattices: Simpler, tighter, shorter, ring-based. PKC pp. 427–449 (2015)
34. Ling, K. Nguyen, H. Wang, and Y. Xu, S.: Lattice-based group signatures: Achieving full dynamicity with ease. ACNS pp. 293–312 (2017)
35. Lu, R., Lin, X., Zhu, H., Ho, P.H., Shen, X.: Ecpp: Efficient conditional privacy preservation protocol for secure vehicular communications. In: INFOCOM 2008 (2008)
36. Lysyanskaya, A., Rivest, R., Sahai, A., Wolf, S.: Pseudonym systems. SAC pp. 684–199 (1999)
37. Nakanishi, T., Fujii, H., Hira, Y., Funabiki, N.: Revocable group signature schemes with constant costs for signing and verifying. In: Public Key Cryptography C PKC 2009, Lecture Notes in Computer Science, vol. 5443, pp. 463–480. Springer Berlin / Heidelberg (2009)
38. Nakanishi, T., Funabiki, N.: Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In: ASIACRYPT. pp. 533–548 (2005)
39. Nakanishi, T., Funabiki, N.: A short verifier-local revocation group signature scheme with backward unlinkability. Lecture Notes in Computer Science, vol. 4266, pp. 17–32. Springer Berlin / Heidelberg (2006)
40. Nakanishi, T., Funabiki, N.: A short anonymously revocable group signature scheme from decision linear assumption. In: ASIACCS. pp. 337–340 (2008)
41. Ren, K., Lou, W.: A sophisticated privacy-enhanced yet accountable security framework for metropolitan wireless mesh networks. In: ICDCS 2008 (2008)
42. Research, C.: Sec 2: Recommended elliptic curve domain parameters. In: Standards for Efficient Cryptography (2000), `http://www.secg.org/download/aid-386/sec2-final.pdf`
43. Song, D.X.: Practical forward secure group signature schemes. In: CCS. pp. 225–234 (2001)
44. Szczechowiak, L. Oliveira, M. Scott, M. Collier, R. Dahab, P.: Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. Wireless Sensor Networks, Lecture Notes in Computer Science 4913 (2008)
45. Unterluggauer, T., Wenger, E.: Efficient pairings and ecc for embedded systems. IACR Cryptology ePrint Archive (2014)
46. Vercautern, F.: Main computational assumptions in cryptography. http://www.ecrypt.eu.org/documents/D.MAYA.3.pdf (2010)
47. Xiong, D. Wong, and X. Deng, X.: Tinypairing: A fast and lightweight pairing-based cryptographic library for wireless sensor networks. IEEE Wireless Communication and Networking Conference (2010)
48. Zhang, J., Ma, L., Su, W., Wang, Y.: Privacy-preserving authentication based on short group signature in vehicular networks. In: Proceedings of the The First International Symposium on Data, Privacy, and E-Commerce (2007)
49. Zhang, W., Wang, C.: La^3: A lightweight accountable and anonymous authentication scheme for resource-constrained devices (full version). Technical Report in Computer Science Department at ISU (2018), `http://www.cs.iastate.edu/~wzhang/la3full.pdf`
50. Zhu, D. Ma, S. Wang, R. Feng, Y.: Efficient identity-based encryption without pairings and key escrow for mobile devices. International Conference on Wireless Algorithms, Systems, and Applications pp. 42–53 (2013)

Appendix 1 Proof of Theorem 2 (Non-frameability)

This proof shows that, if an adversary $\mathcal{A}$ wins the non-frameability game in time t, an algorithm $\mathcal{B}$ can be constructed to solve the q-SDH problem in $\Theta(1) \cdot t$.

Suppose $\mathcal{B}$ is given (q+1)-tuple: $(\tilde{g},\ \tilde{g}^{\gamma},\ \cdots,\ \tilde{g}^{\gamma^q}) \in \mathbb{G}^{q+1}$. Using the technique in the proof of Lemma 3.2 in [9], we can obtain certain $g \in \mathbb{G}$ and q-1 SDH pairs $(x_1, A_1), \cdots, (x_q, A_q)$, where $x_i \xleftarrow{R} \mathbb{Z}_p$ and $A_i = g^{1/(\gamma+x_i)}$ for $i \in \{1, \cdots, q-1\}$. Then, $\mathcal{B}$, acting as the challenger, plays with $\mathcal{A}$ in the non-frameability game as follows.

Phase I: Initialization. $\mathcal{B}$ runs $SystemInit(1^{\kappa})$ to obtain $\mathbb{SS} = \{k_1, k_2, d, l, C(x)\}$, and runs $VerifierInit(\mathbb{SS})$ to obtain $\mathbb{VK} = \{k_1, k_2, l, C(x), \hat{d}\}$. Let n be the total number of provers and $w \xleftarrow{R} \{1, \cdots, n\}$. For each prover $u \in \{1, \cdots, n\} \setminus \{w\}$, $\mathcal{B}$ runs $ProverInit(\mathbb{SS}, u)$ to obtain $\mathbb{PK}_u = \{\hat{k}_{u,0}, \hat{k}_{u,1}, s_u, B_u(x), \hat{e}_u, \hat{F}_u(x)\}$. To initialize prover w, $\mathcal{B}$ picks s_w and $q-1$ distinct numbers, denoted as $r''_1, \cdots, r''_{q-1}$, randomly from $\mathbb{Z}_p$. A set Ω is created to contain 3-tuples $(*, r''_i, x_i)$ $(i = 1, \cdots, q)$ where $*$ represents an empty placeholder and each x_i is a part of the SDH pair (x_i, A_i).

Phase II: Queries. $\mathcal{B}$ responds to queries issued by $\mathcal{A}$ as follows.

Corruption of the verifier: $\mathcal{B}$ returns $\mathbb{VK}$.

Corruption of a prover u: If $u = w$, failure is declared and the game exits. Otherwise, $\mathcal{B}$ returns $\mathbb{PK}_u$.

Authentication for prover v: If $q-1$ authentication queries have been served on prover v, failure is declared and the game exits. If $v \neq w$, $\mathcal{B}$ responds according to the authentication protocol in Section 3.4. If $v = w$ and the pending query is the i-th ($i \leq q$) authentication query on w, $\mathcal{B}$ responds as follows: The challenge provided by the adversary is denoted as $\tilde{c} = \{r'_i\}$. In Ω, the 3-tuple $(*, r''_i, x_i)$ is replaced with (r'_i, r''_i, x_i). $b_{w,i}, \alpha, \xi \xleftarrow{R} \mathbb{Z}_p$. $h(r'_i|r''_i) = x_i$ and $b_{w,i} = (\gamma + x_i)\alpha\beta$ for some β are assumed. Hence, $g^{1/(\gamma+x_i)} = g^{\frac{\alpha\beta}{b_{w,i}}}$, and therefore $g^{\beta} = (g^{1/(\gamma+x_i)})^{b_{w,i}/\alpha}$. Then, $\mathcal{B}$ generates response $\tilde{r}$ as $\{r''_i, a_{w,1,i}, a_{w,2,i}, b_{w,i}, \hat{k}_{w,0,i}, \hat{k}_{w,1,i}, \hat{k}_{w,2,i}, \hat{k}_{w,3,i}, \hat{e}_{w,i}, \hat{f}_{w,i}\}$, where $\hat{e}_{w,i} \xleftarrow{R} \mathbb{G}$, $a_{w,1,i} = 2\alpha + \xi - 1$, $a_{w,2,i} = \alpha(s_w + 1) + \xi - 1$, $\hat{k}_{w,0,i} = g^{\beta}$, $\hat{k}_{w,1,i} = \hat{k}_{w,0,i}^{k_1}$, $\hat{k}_{w,2,i} = (\hat{k}_{w,0,i})^{-\xi}$, $\hat{k}_{w,3,i} = (\hat{k}_{w,1,i})^{-\xi}$, and $\hat{F}_{u,i} = [\hat{k}_{w,0,i}^{\alpha(1+k_1+k_2 s_w)C(x_i)} \hat{d}^{b_{w,r}} \hat{e}_{w,i}^{l}]^{-1}$.

Hash function $h()$: When $\mathcal{A}$ queries $h(r_1|r_2)$, $\mathcal{B}$ searches Ω to find (r'_i, r''_i, x_i) such that $r'_i = r_1$ and $r''_i = r_2$. If a match is found, x_i is returned. Otherwise, $y \xleftarrow{R} \mathbb{Z}_p \setminus \{x_1, \cdots, x_q\}$ is returned. Meanwhile, consistency is maintained; that is, hashing on the same pair of (r_1, r_2) always returns the same value.

Phase III: Adversary's Output. Finally, $\mathcal{A}$ outputs the following transaction transcript: $\tilde{c} = \{r'_1\}$ and $\tilde{r} = \{r'_2, a'_1, a'_2, b', \hat{k}'_0, \hat{k}'_1, \hat{k}'_2, \hat{k}'_3, \hat{e}', \hat{F}'\}$.

If $\mathcal{A}$ wins the game, the transcript can trace to a prover w'. As w is randomly placed among the n provers, $w' = w$ with probability $1/n$. In this case, a new SDH pair $(x', g^{1/(\gamma+x')})$ can be derived, where $x' = h(r'_1, r'_2)$ and $(\hat{k}'_0)^{(a'_2 - a'_1)/[(s_w - 1)b']} = g^{1/(\gamma+x')}$, which can be used to further derive $\tilde{g}^{1/(\gamma+x')}$ also using the technique in the proof of Lemma 3.2 in [9]. Hence, the q-SDH problem is solved.

Therefore, if the q-SDH problem cannot be solved with probability greater than ϵ in time t, the LA^3 scheme is $(\Theta(1) \cdot t, q, n\epsilon)$ non-frameable. That is, the probability for

$\mathcal{A}$ to win the Non-frameability Game is not greater than $n\epsilon$ in $\Theta(1) \cdot t$, where the total number of authentication queries towards each prover is no more than q.

Appendix 2 Proof of Lemma 1

This proof shows that, if algorithm $\mathcal{A}$ solves the extended LRSW problem in time t, algorithm $\mathcal{B}$ can be constructed to solve the LRSW problem in $\Theta(1) \cdot t$. Suppose $\mathcal{B}$ is provided with g, g^x, $g^y \in \mathbb{G}$ (where x,$y \in \mathbb{Z}_p$) and an oracle O that can answer queries as specified in the definition of the LRSW problem. $\mathcal{B}$ provides to $\mathcal{A}$ with

$$g, g^k = g^y, g^{c_1} = g^x, g^{c_0} = (g^x)^\alpha,$$

where α is randomly picked from $\mathbb{Z}_p$. Then, $\mathcal{B}$ answers oracle queries form $\mathcal{A}$ as follows. When $\mathcal{A}$ queries with input s, $\mathcal{B}$ queries oracle O with input $s' = (s-1)/2$ and obtains

$$g', (g')^y, (g')^{s'xy+x} = (g')^{(s-1)xy/2+x}.$$

Then it computes and returns to $\mathcal{A}$ the following:

$$g', (g')^k = (g')^y, (g')^{[k(s-1)+2]c_1} = (g')^{x[y(s-1)+2]} = [(g')^{s'xy+x}]^2,$$

$$(g')^{[k(s-1)+2]c_0} = [(g')^{[k(s-1)+2]c_1}]^\alpha.$$

At the end of queries, suppose $\mathcal{A}$ solves the extended LRSW problem by returning

$$g'', t, r, (g'')^k, A = (g'')^{[k(t-1)+2](c_1 r + c_0)}.$$

Then, algorithm $\mathcal{B}$ solves the LRSW problem as follows. It lets $t' = (t-1)/2$ and returns

$$g'',\ t',\ (g'')^y = (g'')^k, (g'')^{t'xy+x} = A^{1/[2(r+\alpha)]}.$$

Appendix 3 Proof of Theorem 3 (Traceability)

This proof shows that, if an adversary $\mathcal{A}$ wins the traceability game in time t, either the DDH problem or the extended LRSW problem can be solved in $\Theta(1) \cdot t$. Specifically, the proof considers two cases: (i) if $\mathcal{A}$ wins by providing a response that traces to a faked s_w of some faked prover w, an algorithm $\mathcal{B}$ can be constructed to solve the extended LRSW problem; (ii) if $\mathcal{A}$ wins by providing a response that traces to an existing s_u but a faked $b_w \neq b_u$, an algorithm $\mathcal{B}$ can be constructed to solve the DDH problem. In both cases, we use the KEA3 assumption.

Case I: $\mathcal{A}$ wins by providing a response that traces to a faked s_w. Given g, g^{k_2}, g^{c_0}, g^{c_1} $\in \mathbb{G}$ (where k_2, c_0, $c_1 \xleftarrow{R} \mathbb{Z}_p$), and oracle O that can answer queries as specified in the extended LRSW problem, we construct algorithm $\mathcal{B}$ that acts as the challenger in the traceability game, as follows.

Phase I: Initialization. $\mathcal{B}$ picks $\hat{d} \xleftarrow{R} \mathbb{G}$, $l \xleftarrow{R} \mathbb{Z}_p$, and let $k_1 = 1 - k_2$.

Phase II: Queries and Responses. When $\mathcal{A}$ issues a corruption query for prover u, $\mathcal{B}$ responds $\mathbb{PK}_u = \{s_u, \hat{k}_{u,0}, \hat{k}_{u,1}, b_u, \hat{e}_u, \hat{F}_{u,1}, \hat{F}'_{u,1}, \hat{F}_{u,0}, \hat{F}'_{u,0}\}$, which is generated

as follows: $\mathcal{B}$ first picks $s_u \xleftarrow{R} \mathbb{Z}_p$ and calls Oracle O on input s_u to obtain g', $(g')^{k_2}$, $(g')^{[(s_u-1)k_2+2]c_1}$ and $(g')^{[(s_u-1)k_2+2]c_0}$. Given that $k_1 = 1 - k_2$, it hence obtains

$$\hat{k}_{u,0} = g', \;\; \hat{k}_{u,1} = g'/(g')^{k_1}. \tag{15}$$

Also, it picks $b_u \xleftarrow{R} \mathbb{Z}_p$, $\hat{e}_u \xleftarrow{R} \mathbb{G}$. Finally, it computes

$$g'_1 = (g')^{(k_1+k_2 s_u+1)c_1} = (g')^{[(s_u-1)k_2+2]c_1}, \tag{16}$$
$$g'_0 = (g')^{(k_1+k_2 s_u+1)c_0} = (g')^{[(s_u-1)k_2+2]c_0}, \tag{17}$$

picks up $x_1, x_2 \xleftarrow{R} \mathbb{Z}_p$, and computes

$$\hat{f}_{u,1} = [g'_1 \hat{d}^{b_{u,1}}]^{-x_1}, \;\; \hat{f}'_{u,1} = [g'_1 \hat{d}^{b_{u,1}}]^{-1-x_1}, \tag{18}$$
$$\hat{f}_{u,0} = ((g'_0 \hat{d}^{b_{u,0}})^{-1}/\hat{e}_u^l)^{x_2}, \;\; \hat{f}_{u,0} = ((g'_0 \hat{d}^{b_{u,0}})^{-1}/\hat{e}_u^l)^{1-x_2}. \tag{19}$$

Note that, the above initialization ensures that

$$\begin{aligned} \hat{k}_{u,0}^{(k_1+k_2 s_u+1)c_1} \hat{d} \hat{f}_{u,1} \hat{f}'_{u,1} &= g^0, \\ \hat{k}_{u,0}^{(k_1+k_2 s_u+1)c_0} \hat{d}^{b_u} \hat{e}_u^l \hat{f}_{u,0} \hat{f}'_{u,0} &= g^0. \end{aligned} \tag{20}$$

When $\mathcal{A}$ issues an authentication query for prover u, $\mathcal{B}$ responds according to the authentication protocol in Section 3.4.

When $\mathcal{A}$ issues a hashing query $h(r_1|r_2)$ with certain $r_1, r_2 \in \mathbb{Z}_p$, $\mathcal{B}$ returns $r \xleftarrow{R} \mathbb{Z}_p$. Consistency is maintained to ensure the same pair of (r_1, r_2) is always hashed to the same value.

Phase III: Challenge. The challenger sends $r'_1 \xleftarrow{R} \mathbb{Z}_p$ to $\mathcal{A}$.

Phase IV: Post-Challenge Queries. This phase is the same as Phase II.

Phase V: Adversary's Response. $\mathcal{A}$ outputs the following response from a real or faked prover w:

$$\begin{aligned} & r'_2, \; a_{w,1,r}, \; a_{w,2,r}, \; B_{w,r}, \; \hat{e}_{w,r}, \; \hat{F}_{w,r}, \\ & \hat{k}_{w,0,r}, \; \hat{k}_{w,1,r} = (\hat{k}_{w,0,r})^{k_1}, \; \hat{k}_{w,2,r} = (\hat{k}_{w,0,r})^{-\xi}, \; \hat{k}_{w,3,r} = (\hat{k}_{w,2,r})^{k_1}, \end{aligned} \tag{21}$$

where $r = h(r'_1|r'_2)$.

If $\mathcal{A}$ wins, $\mathcal{B}$ acts as follows. First, due to the KEA3 assumption, ξ encoded in $\hat{k}_{w,2,r}$ can be found out. Then, $\mathcal{B}$ can compute $\alpha = (a_{w,1,r} - \xi + 1)/2$ and $s_w = (a_{w,2,r} - \xi + 1)/\alpha - 1$.

If s_w is different from any s_u that has been queried before, $\mathcal{B}$ computes

$$g'' = \hat{k}'_{0,r}, \;\; (g'')^{k_1} = \hat{k}'_{1,r}, \;\; (g'')^{k_2} = g''/\hat{k}'_{1,r}, \tag{22}$$
$$(g'')^{(k_1+k_2 s'+1)(c_1 r+c_0)} = \frac{1}{(\hat{e}'_r)^l \hat{F}'_r \hat{d}^{b'_r}}. \tag{23}$$

Then, it outputs g'', s', r, $(g'')^{k_1}$, $(g'')^{k_2}$, and $(g'')^{(k_1+k_2 s'+1)(c_1 r+c_0)}$ to solve the extended LRSW problem; due to Lemma 1, it also solves the LRSW problem.

Case II: $\mathcal{A}$ wins by generating a response that traces to an existing s_u but a faked $b_w \neq b_u$. We construct algorithm $\mathcal{B}$, which is given g, g^a, g^b and g^c and asked to find out if $c = ab$. $\mathcal{B}$ acts as the challenger to play with $\mathcal{A}$ in the traceability game similarly as in Case I, except for Phase I and the corruption query of Phase II which are detailed as follows.

In Phase I, $\mathcal{B}$ first picks $k_0, k_1, t \xleftarrow{R} \mathbb{Z}_p$ and let $k_2 = 1 - k_1$. Also, it initializes the verifier by setting

$$c_1 = (c-a)/(b-1),\ \ c_0 = r_0 \cdot c_1,$$
$$\hat{d} = g^{(ab-c)/(b-1)},\ \ l = (ab-c)/(1-b) + t. \tag{24}$$

In Phase II, $\mathcal{B}$ responds to a corruption query for prover u as follows.

- $r_{i,1}, b_i, s_i \xleftarrow{R} \mathbb{Z}_p$ and $r_{i,2} \leftarrow 1 - r_{i,1}$;
- $B_i(x) = x + b_i$, $\hat{e}_i = g^{b_i - r_0}$;
- $\hat{k}_{i,0} = (g^{r_{i,1}}(g^b)^{r_{i,2}})^{1/(k_1+k_2 s_u+1)},\ \hat{k}_{i,1} = (\hat{k}_{i,0})^{k_1}$;
- $\hat{f}_{i,1} = (g^a)^{-r_{i,1}},\ \hat{f}'_{i,1} = (g^c)^{-r_{i,2}},\ \hat{f}_{i,0} = (\hat{f}_{i,1})^{k_0} g^{b_i \cdot t},\ \hat{f}'_{i,0} = (\hat{f}'_{i,1})^{k_0} g^{-k_0 \cdot t}$.

Note that, the above initialization also ensures Eq.(20) to hold.

Assume $\mathcal{A}$ returns a valid response as specified in Equ.(21). As in Case I, ξ, α and s_w can be derived from the response of $\mathcal{A}$. If $s_w = s_u$ for some prover u that has been queried, $\mathcal{B}$ proceeds as follows; otherwise, it aborts.

Also due to KEA3 assumption, v_1 and v_2 can be found such that $\hat{k}_{w,0,r} = (g^{v_1} g^{bv_2})^{1/(k_1+K_2 s_u+1)}$; obviously, $\beta = v_1 + v_2$. Further, let $b_w = \frac{B_{w,r}}{\alpha\beta} - r$.

Next, we construct the following response that prover u can generate with the above r, α, v_1, v_2 and ξ:

$$a_{u,1,r} = a_{w,1,r},\ \ a_{u,2,r} = a_{w,2,r},\ \ B_{u,r} = \alpha\beta(b_u + r),\ \ \hat{e}_{u,r} = g^{\alpha\beta(b_u - k_0)}, \tag{25}$$
$$\hat{k}_{u,0,r} = (g^{v_1} g^{v_2})^{1/(k_1+k_2 s_u+1)} = \hat{k}_{w,0,r},\ \ \hat{k}_{u,1,r} = (\hat{k}_{u,0,r})^{k_1}, \tag{26}$$
$$\hat{k}_{u,2,r} = (\hat{k}_{u,0,r})^{-\xi} = \hat{k}_{w,2,r},\ \ \hat{k}_{u,3,r} = (\hat{k}_{u,2,r})^{k_1}, \tag{27}$$
$$\hat{F}_{u,r} = (\hat{F}_{u,1}\hat{F}'_{u,1})^{\alpha\beta r}(\hat{F}_{u,0}\hat{F}'_{u,0}). \tag{28}$$

Since the response from prover u is valid and the response from the faked prover w is assumed valid,

$$(\hat{k}_{u,0,r})^{(k_1+k_2 s_u+1)(c_1 r + c_0)}(\hat{d})^{\alpha\beta(r+b_u)}(\hat{e}_{u,r})^l = \hat{F}_{u,r}, \tag{29}$$

and

$$(\hat{k}_{w,0,r})^{(k_1+k_2 s_u+1)(c_1 r + c_0)}(\hat{d})^{\alpha\beta(r+b_w)}(\hat{e}_{w,r})^l = \hat{F}_{w,r}. \tag{30}$$

Hence,

$$(\hat{d})^{\alpha\beta(b_w - b_u)}(\hat{e}_{w,r}/\hat{e}_{u,r})^l = (\hat{F}_{w,r}/\hat{F}_{u,r}) \tag{31}$$
$$\Rightarrow (g^{\alpha\beta(b_w = b_u)})^{(ab-c)/(b-1)}(\hat{e}_{w,r}/\hat{e}_{u,r})^{(ab-c)/(1-b)+t} = (\hat{F}_{w,r}/\hat{F}_{u,r}) \tag{32}$$
$$\Rightarrow (\frac{g^{\alpha\beta(b_w = b_u)}}{\hat{e}_{w,r}/\hat{e}_{u,r}})^{(ab-c)/(b-1)} = \frac{\hat{F}_{w,r}/\hat{F}_{u,r}}{(\hat{e}_{w,r}/\hat{e}_{u,r})^t}. \tag{33}$$

Now, we have the following subcases:

- Subcase I: $\frac{\hat{F}_{w,r}/\hat{F}_{u,r}}{(\hat{e}_{w,r}/\hat{e}_{u,r})^t} \neq 1$. In this case, $ab \neq c$, and hence the DDH problem is solved.
- Subcase II: $\frac{\hat{F}_{w,r}/\hat{F}_{u,r}}{(\hat{e}_{w,r}/\hat{e}_{u,r})^t} = 1$.
  - Subcase II-(a): $g^{\alpha\beta(b_w-b_u)} \neq \hat{e}_{w,r}/\hat{e}_{u,r}$. In this case, $ab = c$ holds; hence the DDH problem is solved.
  - Subcase II-(b): $g^{\alpha\beta(b_w-b_u)} = \hat{e}_{w,r}/\hat{e}_{u,r}$. In this case, $g^{\alpha\beta(b_w-b_u)t}$ is equal to $\hat{F}_{w,r}/\hat{F}_{u,r}$ which is known; as $\alpha\beta(b_w - b_u)$ is also known, g^t should be known to the adversary. This however contradicts with the security of ElGamal encryption because of the following reasons. For every prover i corrupted by $\mathcal{A}$, it obtains the following information related to t includes

$$(\hat{f}_{i,1}, \hat{f}'_{i,1} = \hat{f}_{i,1}g^{b_i t});\ \ (\hat{f}_{i,0}, \hat{f}'_{i,0} = \hat{f}_{i,0}g^{k_0 t}), \tag{34}$$

    which are the cipher texts of $g^{b_i t}$ and $g^{k_0 t}$ with private key k_0. If g^t is known, Elgamal encryption is broken and thus the DDH is broken due to the equivalence of the Elgamal encryption and the DDH problem [].

To summarize, if neither the LRSW or the DDH problem can be solved with probability greater than ϵ in time t, the probability for $\mathcal{A}$ to win the Traceability Game cannot be greater than ϵ in $\Theta(1) \cdot t$.

Appendix 4 Proof of Theorem 4 (Selfless Anonymity)

Suppose $\mathcal{B}$ is provided with $(g', (g')^{\gamma}, (g')^{\gamma^2}, \cdots, (g')^{\gamma^q})$ and $(g')^{1/(\gamma+y)}$, and is requested to determine if $y = 0$.

Let x_i $(i = 1, \cdots, q)$ be picked from $\mathbb{Z}_p$ uniformly at random. Using the algorithm presented in the proof of Lemma 3.2 in [9], $\mathcal{B}$ can obtain a certain $g \in \mathbb{G}$ and $q-1$ SDH pairs $(x_i, g^{1/(\gamma+x_i)})$ $(i = 1, \cdots, q-1)$ regarding g and γ, from $(g', (g')^{\gamma}, (g')^{\gamma^2}, \cdots, (g')^{\gamma^q})$.

$\mathcal{B}$ can also obtain g'' from $(g')^{1/(\gamma+y)}$ such that $g'' = g^{1/\gamma}$ if and only if $y = 0$. More specifically, g'' is computed as follows. Let $f(\gamma) = \prod_{i=1}^{q-1}(\gamma + x_i)$. Expand f and write $f(\gamma)$ as $\sum_{i=0}^{q-1} f_i\gamma^i$, where $f_i \in \mathbb{Z}_p$ are the coefficients of f. Suppose $g = (g')^{\theta f(\gamma)}$, where $\theta \in \mathbb{Z}_p$ is randomly selected in the aforementioned course of generating g and the $q-1$ SDH pairs. Then, $\mathcal{B}$ computes

$$g'' = \{\frac{[(g')^{1/(\gamma+y)}]^{f_0}}{\prod_{i=0}^{q-2}(g')^{-f_{i+1}\gamma^i}}\}^{\theta}. \tag{35}$$

Note that, if $y = 0$, g'' can be rewritten as

$$g'' = \{\frac{[(g')^{1/(\gamma)}]^{f_0}}{\prod_{i=0}^{q-2}(g')^{-f_{i+1}\gamma^i}}\}^{\theta} \tag{36}$$

$$= \{\frac{[(g')^{1/(\gamma)}]^{f_0}}{(g')^{-\prod_{i=0}^{q-2} f_{i+1}\gamma^i}}\}^{\theta} \tag{37}$$

$$= \{(g')^{\sum_{i=0}^{q-1} f_i\gamma^i}\}^{\theta/\gamma} \tag{38}$$

$$= \{(g')^{\theta f(\gamma)}\}^{1/\gamma}. \tag{39}$$

In the rest of the proof, we are to show that if $\mathcal{A}$ can win the selfless anonymity game with advantage ϵ, then whether $g'' = g^{1/\gamma}$ (equivalent to whether $y = 0$) can be determined by $\mathcal{B}$ with advantage $0.5\epsilon/n^2$ where n is the number of provers. $\mathcal{B}$ plays with $\mathcal{A}$ in the selfless anonymity game as follows.

Phase I: Initialization. $\mathcal{B}$ initializes the system and the verifier, and thus obtains the set of system-wide parameters, i.e., $\mathbb{SS} = \{k_1, k_2, d, l, C(x)\}$, and the verification key for the verifier, i.e., $\mathbb{VK} = \{k_1, k_2, l, C(x), \hat{d}\}$.

$\mathcal{B}$ randomly picks i_0 and i_1 from $\{1, \cdots, n\}$. For each prover $u \notin \{i_0, i_1\}$, $\mathcal{B}$ initializes prover u and obtains the proof key for it, i.e., $\mathbb{PK}_u = \{\hat{k}_{u,0}, \hat{k}_{u,1}, s_u, B_u(x), \hat{e}_u, \hat{F}_u(x)\}$. For provers i_0 and i_1, $\mathcal{B}$ randomly pick s_{i_0}, s_{i_1}, z_{i_0} and z_{i_1} from $\mathbb{Z}$. It also creates a set Ω which initially contains 3-tuples: $(*, *, x_{j*2-1} + z_{i_0})$ and $(*, *, x_{j*2} + z_{i_1})$ for every $j = 1, \cdots, \lfloor q/2 \rfloor$ and each $*$ represents an empty placeholder.

Phase II: Pre-Challenge Queries and Responses. The adversary can issue the following types of queries which are responded by the challenger.

- *Corruption of the verifier.* When $\mathcal{A}$ issues a corruption query on the verifier, $\mathcal{B}$ returns $\mathbb{VK}$.
- *Corruption of prover u.* If $\mathcal{A}$ requests to corrupt prover $u \in \{1, \cdots, n\} \setminus \{i_0, i_1\}$, $\mathcal{B}$ returns $\mathbb{PK}_u$. Otherwise, failure is declared and the game exits.
- *Authentication for prover v.* If $v \notin \{i_0, i_1\}$, $\mathcal{B}$ responds according to the authentication protocol in Section 3.4.
  Otherwise (i.e., $v \in \{i_0, i_1\}$), if the number of authentication queries on v has exceeded $\lfloor q/2 \rfloor$, failure is declared and the game exits. Now, let the query be the j'-th query on v and the challenge proposed by the adversary be $\tilde{c} = \{r'_v\}$, $\mathcal{B}$ responds as follows. Let $j = 2j' - 1$ if $v = i_0$; or, $j = 2j'$ otherwise. Then $\mathcal{B}$ responds based on $(x_j + z_v, g^{1/(\gamma + x_j)})$, which is a valid SDH pair regarding g and $\gamma - z_v$. Specifically, $\mathcal{B}$ randomly picks r''_v, $B_{v,r}$, α and ξ from $\mathbb{Z}_p$, $\hat{e}_{v,r}$ from $\mathbb{G}$, and sets $r = x_j + z_v$. Meanwhile, in Ω, the 3-tuple $(*, *, r)$ is replaced with (r'_v, r''_v, r). Then, it computes

$$\hat{k}_{v,0,r} = (g^{1/(\gamma + x_j)})^{B_{v,r}/\alpha}, \;\; \hat{k}_{v,1,r} = \hat{k}_{v,0,r}^{k_1}, \tag{40}$$

$$\hat{k}_{v,2,r} = (\hat{k}_{v,0,r})^{-\xi}, \;\; \hat{k}_{v,3,r} = (\hat{k}_{v,1,r})^{-\xi}, \tag{41}$$

$$a_{v,1,r} = 2\alpha + \xi - 1, \;\; a_{v,2,r} = \alpha(s_v + 1) + \xi - 1, \tag{42}$$

$$\hat{F}_{v,r} = [\hat{k}_{v,0,r}^{\alpha(k_1 + k_2 s_v + 1)} \hat{d}^{B_{v,r}} \hat{e}_{v,r}^{l}]^{-1}. \tag{43}$$

  The above is returned as $\tilde{r}$.
- *Hash function $h()$:* When $\mathcal{A}$ queries $h(r_1|r_2)$, $\mathcal{B}$ searches Ω to find if there is a 3-tuple (r'_i, r''_i, x'_i) such that $r'_i = r_1$ and $r''_i = r_2$. If a match is found, x'_i is returned. Otherwise, a number is randomly picked from $\mathbb{Z}_p \setminus \{z_{i_0}, z_{i_1}, x_1 + z_{i_0}, x_2 + z_{i_1}, \cdots, x_{\lfloor q/2 \rfloor * 2 - 1} + z_{i_0}, x_{\lfloor q/2 \rfloor * 2} + z_{i_1}\}$ and returned. Meanwhile, consistency is maintained; hashing on the same pair of (r_1, r_2) always results in the same value.

Phase III: Challenge. The adversary selects two provers u_0 and u_1 that have not been queried. If $\{u_0, u_1\} \neq \{i_0, i_1\}$, failure is declared and the game exits. Otherwise, the challenger randomly picks x from 0 or 1, and presents an authentication transcript $\tilde{r}$ in response to challenges $\tilde{c}$ presented by the adversary, based on (z_{u_x}, g''). Specifically, in

response to r'_1, $\mathcal{B}$ picks r'_2, $B_{u_x,r}$, α and ξ from $\mathbb{Z}_p$, $\hat{e}_{u_x,r}$ from $\mathbb{G}$, and sets $r = z_{u_x}$. Meanwhile, 3-tuple (r'_1, r'_2, r) is injected into Ω. Next, it computes

$$\hat{k}_{u_x,0,r} = (g'')^{B_{u_x,r}/\alpha}, \;\; \hat{k}_{u_x,1,r} = \hat{k}_{u_x,0,r}^{k_1},$$

$$\hat{k}_{u_x,2,r} = (\hat{k}_{u_x,0,r})^{-\xi}, \;\; \hat{k}_{u_x,3,r} = (\hat{k}_{u_x,1,r})^{-\xi}, \tag{44}$$

$$a_{u_x,1,r} = 2\alpha + \xi - 1, \;\; a_{u_x,2,r} = \alpha(s_{u_x} + 1) + \xi - 1, \tag{45}$$

$$\hat{F}_{u_x,r} = [\hat{k}_{u_x,0,r}^{\alpha(k_1+k_2 s_{u_x}+1)} \hat{d}^{B_{u_x,r}} \hat{e}_{u_x,r}^{l}]^{-1}. \tag{46}$$

The above is presented as $\tilde{r}$.

Phase IV: Post-Challenge Queries and Responses. The same as Phase II except that corruption queries cannot be made on the selected provers.

Phase V: Adversary's Response. $\mathcal{A}$ outputs $x' \in \{0, 1\}$ which is a guess of x chosen by $\mathcal{B}$. If $x' = x$, $\mathcal{B}$ returns $y = 0$ with probability $1 - \epsilon/2$; otherwise, it returns $y \neq 0$.

The game may abort if the adversary makes a corruption query on prover i_0 or i_1 in Phase II, or it does not choose $\{u_0, u_1\} = \{i_0, i_1\}$ in Phase III. Suppose the adversary has corrupted k provers in Phase II, the probability that it does not corrupted i_0 or i_1 is C_{n-2}^k / C_n^k, and the probability that it chooses $\{u_0, u_1\} = \{i_0, i_1\}$ in Phase III is $1/C_{n-k}^2$. Hence, the probability for the game to succeed is

$$\frac{C_{n-2}^k}{C_n^k C_{n-k}^2} = \frac{2}{n(n-1)} > 2/n^2.$$

For each successful game, there are two cases: (i) When $y = 0$ is true, $g'' = g^{1/\gamma}$; hence, (z_{u_x}, g'') is a valid SDH pair regarding g and $\gamma - z_{u_x}$. If $\mathcal{A}$ wins with advantage ϵ, the probability for $x' = x$ is $0.5 + \epsilon$ and hence the probability for $\mathcal{B}$ to return $y = 0$ is $(0.5 + \epsilon)(1 - 0.5\epsilon) = 0.5 + \epsilon - 0.25\epsilon - 0.5\epsilon^2 > 0.5 + 0.25\epsilon$. That is, the advantage gained by $\mathcal{B}$ is at least 0.25ϵ. (ii) When $y = 0$ is false, (z_{u_x}, g'') is not a SDH pair regarding g and $\gamma - z_{u_x}$; that is, the response based on (Z_{u_x}, g'') cannot trace to either i_0 or i_1. Hence $\mathcal{A}$ returns x' as 0 or 1 randomly. So, the probability for $x' = x$ is 0.5 and therefore the probability for $\mathcal{B}$ to return $y = 0$ is $0.5(1 - 0.5\epsilon) = 0.5 - 0.25\epsilon$. That is, the probability for $\mathcal{B}$ to return $y \neq 0$ is $0.5 + 0.25\epsilon$; the advantage gained by $\mathcal{B}$ is 0.25ϵ.

To summarize, if $\mathcal{A}$ has probability greater than $0.5+\epsilon$ to win the selfless anonymity game in time t, then $\mathcal{B}$ has probability greater than $0.5 + 0.25\epsilon \cdot (2/n^2) = 0.5 + 0.5\epsilon/n^2$ to solve the q-DDHI problem in time $\Theta(1) \cdot t$. This is equivalent to that, if the q-DDHI problem cannot be solved with probability greater than $0.5 + \epsilon$ in time t, then the proposed scheme cannot win the selfless anonymity game with probability greater than $0.5 + \epsilon'$ in time t', where $\epsilon' = 2n^2\epsilon$ and $t' = \Theta(1) \cdot t$.