Short Traceable Signatures Based on Bilinear Pairings

Seung Geol Choi[1], Kunsoo Park[2], and Moti Yung[1]

[1] Dept. of Computer Science, Columbia University, USA
{sc2506, moti}@cs.columbia.edu
[2] Dept. of Computer Science and Engineering, Seoul National University, Korea
kpark@theory.snu.ac.kr

Abstract. We propose a short traceable signature scheme based on bilinear pairings. Traceable signatures, introduced by Kiayias, Tsiounis and Yung (KTY), support an extended set of fairness mechanisms (mechanisms for anonymity management and revocation) when compared with the traditional group signatures. Designing short signatures based on the power of pairing has been a current activity of cryptographic research, and is especially needed for long constructions like that of traceable signatures. The size of a signature in our scheme is less than one third of the size in the KTY scheme and about 40% of the size of the pairing based traceable signature (which has been the shortest till today). The security of our scheme is based on the Strong Diffie-Hellman assumption and the Decision Linear Diffie-Hellman assumption. We prove the security of our system in random oracle model using the security model given by KTY.

Keywords: traceable signatures, group signatures, anonymity, cryptographic protocols, bilinear pairings.

1 Introduction

Group signatures, introduced by Chaum and van Heyst [11], provide anonymity for signers. Any member of the group can sign messages, but the resulting signature keeps the identity of the signer secret. Because unconditional anonymity may be a very dangerous tool against public safety, in case of dispute about the signed message group signatures allow the group manager to open the signature and identify its originator. In this respect, group signatures can be said to incorporate a *fairness* mechanism.

Traceable signatures, introduced by Kiayias, Tsiounis and Yung [13], support an extended set of fairness mechanisms (mechanisms for anonymity management and revocation) when compared with the traditional group signature schemes. Consider the following scenario: a certain member of the group is suspected of illegal activity. Its identity was revealed by opening a signature value. It is then necessary to detect the signatures issued by this member so that his/her transactions are traced. The only solution with existing group signature schemes is to have the group manager open all signatures. However, this solution have two

H. Yoshiura et al. (Eds.): IWSEC 2006, LNCS 4266, pp. 88–103, 2006.

problems. First, since all signatures are opened the solution violates the privacy of all group members. Second, since only the group manager can open signatures it impairs scalability. We need some parallel mechanism for scalability. Traceable signatures support three different types of traceability :

1. signature opening : as in group signature, reveal the signer of a given signature
2. user tracing : check whether a signature was issued by a given user; it can be applied to all signatures by designated tracing agents running in parallel
3. signature claiming : the signer of a signature provably claims a given signature that he/she has signed.

Therefore, the above scenario can be solved with user tracing if we use traceable signatures.

Kiayias, Tsiounis and Yung (hereafter KTY) gave a formal security model through three security requirements: misidentification, anonymity, framing. If the adversary is able to generate a signature whose originator is not traced by the group manager, the scheme is not secure against misidentification. Given a signature and two members of which one is its originator, if the adversary can identify its originator no better than randomly, the scheme has anonymity. The adversary succeeds in framing an honest user, if he generates a signature that is wrongly traced to an innocent user. The KTY scheme [13] is secure under the Strong RSA and Decisional Diffie-Hellman assumptions. We note that in concurrent independent work, Bellare et al. [7,8] gave a formal security model for group signatures with a security model which is nearly the same as that of KTY [13].

Traceable signature with the above scalable anonymity is suitable for various applications and extends the reach of e-commerce while allowing users extended anonymity. In a typical web based commerce, it may be desired that the service provider does not know the user, yet that there is a mechanism that a tracing authority (either as a law enforcement mechanisms for illegal activity or as a routine mechanisms like billing by a bank) will be able at a different layer to expose users selectively. The notion of "anonymous non-repudiation" that indeed combines both anonymity at the service provider level, but identification at another layer has high potential in many application domains.

Recent pairing based signatures: Boneh et al. [6] noticed that bilinear maps can shorten signature schemes; this started a line of research of employing pairings in order to shorten signatures. Boneh et al. [5] devised a short group signature scheme using bilinear pairings. The size of a signature is under 200 bytes that offer approximately the same level of security as a regular RSA signature of the same length. The scheme used the Strong Diffie-Hellman (SDH) and Decision Linear Diffie-Hellman (DLDH) assumptions. Nguyen and Safavi-Naini. [16] also introduced a group signature scheme using bilinear pairings. The size of its signature is slightly bigger than that of [5], but the scheme has stronger anonymity. They also introduced a traceable signature scheme using El Gamal public key encryption under the assumptions above. Boyen and Waters [9] gave the group signature scheme without random oracles but the scheme is not practical in that the size

of a signature grows logarithmically in the number of group members. Ateniese et al. [2] devised practical group signature scheme without random oracles. Their scheme is based on the Strong LRSW, q-EDH, and Strong SXDH assumptions.

Our Result: We extend the result of [5] to construct a traceable signature scheme with the length of signatures 362 byte long (just about the size of three RSA signatures), which is shorter than those of [13] and [16], which are 1200 byte and 900 byte long, respectively. In contrast to the previous schemes that need two separate parts for tracing and claiming [13,16], we use one part for the two procedures, which is possible with the help of bilinear pairing, and therefore we get shorter signature size. In spite of its shorter length, the security level of our scheme is the same as of the schemes using bilinear pairings. We used the SDH and DLDH assumptions given by [5].

2 Preliminaries

2.1 Bilinear Pairings

We first review a few concepts related to bilinear pairings. Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic additive groups generated by P_1 and P_2, respectively, both with prime order p, and $\mathbb{G}_T$ be a cyclic multiplicative group of order p. Suppose there is an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ such that $\psi(P_2) = P_1$. A bilinear pairing is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties.

- Non-degeneracy : $e(P_1, P_2) \neq 1$
- Bilinearity : For all $Q_1 \in \mathbb{G}_1, Q_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(aQ_1, bQ_2) = e(Q_1, Q_2)^{ab}$.
- Computability : For all $Q_1 \in \mathbb{G}_1, Q_2 \in \mathbb{G}_2$, there is an efficient algorithm to compute $e(Q_1, Q_2)$.

We assume that p is about 2^{170}. $\mathbb{G}_1$ and $\mathbb{G}_2$ are assumed to be subgroups with order p of an elliptic curve group where possibly $\mathbb{G}_1 = \mathbb{G}_2$. $\mathbb{G}_T$ is a subgroup with order p of a finite field of size about 2^{1024}. We note that the bilinear groups of Rubin and Siverberg [18] or Miyagi et al. [15] can be used. We denote the generation algorithm of bilinear pairings by $\mathcal{G}_{BP}$ throughout this paper.

2.2 SDH Representation

Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic additive groups of prime order p where possibly $\mathbb{G}_1 = \mathbb{G}_2$. Let P_1 be a generator of $\mathbb{G}_1$ and P_2 a generator of $\mathbb{G}_2$. The q-Strong Diffie Hellman (q-SDH) problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows [5]: Given $(q+2)$-tuple $(P_1, P_2, \gamma P_2, \ldots, \gamma^q P_2)$ as input, output a pair $(\frac{1}{\gamma+x} P_1, x)$ where $x \in \mathbb{Z}_p^*$. The advantage of an algorithm $\mathcal{A}$ for the q-SDH problem is defined as follows:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{q-SDH}}(k) = \Pr\left[\begin{array}{l} \mathcal{A}(\mathcal{G}, \gamma P_2, \ldots, \gamma^q P_2) = \left(\frac{1}{\gamma+x} P_1, x\right) \wedge x \in \mathbb{Z}_p^* \quad \text{where} \\ \quad \mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e) \leftarrow \mathcal{G}_{BP}(1^k) \\ \quad \gamma \xleftarrow{R} \mathbb{Z}_p^* \end{array}\right]$$

We define the q-Strong Diffie-Hellman assumption. This assumption was used by Boneh and Boyen [4] to construct short signatures.

Definition 1. q-Strong Diffie-Hellman (q-SDH) Assumption. *For every PPT algorithm $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{q-SDH}}(k)$ is negligible in k.*

We next define an SDH representation. The representation is similar to a discrete-log representation of an arbitrary power in the KTY scheme [13].

Definition 2. SDH representation. *For $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$ and (Q, R) where $Q \in \mathbb{G}_1, R = \gamma P_2 \in \mathbb{G}_2$ with unknown γ, an SDH representation is a tuple (A, x, t) with $A \in \mathbb{G}_1$ and $x, t \in \mathbb{Z}_p^*$ such that $A = \frac{1}{\gamma + t}(xP_1 + Q)$. Note that the tuple satisfies $e(A, tP_2 + R) = e(xP_1 + Q, P_2)$.*

In this work we will be interested in the following computational problem.

Definition 3. One more SDH representation problem. *Given K SDH representations for $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$ and (Q, R), one more SDH representation problem is to find another SDH representation.*

Lemma 1. *Under the q-SDH assumption, it is infeasible for a PPT algorithm to solve one more SDH representation problem with $K = q$.*

2.3 Linear Encryption

Let $\mathbb{G}_1$ be a cyclic additive group of prime order p, and let X, Y, Z be generators of $\mathbb{G}_1$. The Decision Linear Diffie-Hellman problem in $\mathbb{G}_1$ is defined as follows [5]: Given $X, Y, Z, aX, bY, cZ \in \mathbb{G}_1$ as input, output `yes` if $a+b = c$ and no otherwise. The advantage of an algorithm $\mathcal{A}$ for the Decision Linear Diffie-Hellman problem is defined as follows:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{DLDH}}(k) = \left| p_k^{(0)} - p_k^{(1)} \right|$$

where

$$p_k^{(i)} = \Pr\left[\begin{array}{l} \mathcal{A}(\mathcal{H}, aX, bY, cZ) = \texttt{yes} \text{ where} \\ \quad \mathcal{H} = (p, \mathbb{G}_1, X, Y, Z) \leftarrow \mathcal{G}_{DL}(1^k) \\ \quad a, b \xleftarrow{R} \mathbb{Z}_p^* \\ \quad c \leftarrow a + b \text{ if } i = 0 \\ \quad c \xleftarrow{R} \mathbb{Z}_p^* \text{ otherwise} \end{array}\right].$$

We define the Decision Linear Diffie-Hellman assumption. This assumption was used by Boneh et al. [5] to construct short group signatures.

Definition 4. Decision Linear Diffie-Hellman (DLDH) assumption. *For every PPT algorithm $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{DLDH}}(k)$ is negligible in k.*

The DLDH assumption gives rise to the Linear encryption (LE) scheme [5]. This scheme is semantically secure against chosen-plaintext attacks, assuming the DLDH assumption holds.

Definition 5. Linear encryption. *With $M \in \mathbb{G}_1$, along with arbitrary generators X, Y and Z of $\mathbb{G}_1$, linear encryption LE with public key X, Y and Z is as follows:*

$$LE(M) = (r_1 X, r_2 Y, M + (r_1 + r_2)Z)$$

where $r_1, r_2 \in \mathbb{Z}_p$ is randomly chosen.

Lemma 2. *[5] Under the DLDH assumption, linear encryption is secure against chosen-plaintext attacks.*

3 Zero-Knowledge Protocol for an SDH Representation

We assume that $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e), Q \in \mathbb{G}_1, R = \gamma P_2 \in \mathbb{G}_2$ with unknown γ are given as specified in Section 2. Let X, Y, Z be generators of $\mathbb{G}_1$, and W a generator of $\mathbb{G}_2$. A zero-knowledge protocol for an SDH representation (A, x, t) is as follows.

Protocol 1. The prover chooses exponents $r_1, r_2, r_3 \xleftarrow{R} \mathbb{Z}_p$, $d_1 \leftarrow tr_1, d_2 \leftarrow tr_2$, and then computes the following values:

$$T_1 \leftarrow r_1 X, \quad T_2 \leftarrow r_2 Y, \quad T_3 \leftarrow A + (r_1 + r_2)Z,$$
$$T_4 \leftarrow r_3 W, \quad T_5 \leftarrow e(P_1, T_4)^x \quad .$$

Note that T_1, T_2 and T_3 constitute a Linear encryption of the value A. Note also that if we precalculate the value $e(P_1, W)$, we can compute $T_5 = e(P_1, W)^{r_3 x}$ with just one exponentiation on $\mathbb{G}_T$ avoiding the expensive pairing calculation of $e(P_1, T_4)^x$. Now, the prover and the verifier execute a proof of knowledge of values $(r_1, r_2, d_1, d_2, t, x)$ which satisfy the following equations:

$$r_1 X = T_1, \quad r_2 Y = T_2,$$
$$tT_1 - d_1 X = O, \quad tT_2 - d_2 Y = O,$$
$$e(P_1, T_4)^x = T_5,$$
$$e(T_3, P_2)^t \cdot e(Z, P_2)^{-d_1 - d_2} \cdot e(Z, R)^{-r_1 - r_2} \cdot e(P_1, P_2)^{-x} = e(Q, P_2)/e(T_3, R) \quad .$$

This proof is a typical 3-move honest verifier zero-knowledge proof for discrete logarithm relation set. For the first move, the prover randomly chooses $b_{r_1}, b_{r_2}, b_{d_1}, b_{d_2}, b_t, b_x$ from $\mathbb{Z}_p$, and then computes the following values:

$$B_1 \leftarrow b_{r_1} X, \quad B_2 \leftarrow b_{r_2} Y,$$
$$B_3 \leftarrow b_t T_1 - b_{d_1} X, \quad B_4 \leftarrow b_t T_2 - b_{d_2} Y,$$
$$B_5 \leftarrow e(P_1, T_4)^{b_x},$$
$$B_6 \leftarrow e(T_3, P_2)^{b_t} \cdot e(Z, P_2)^{-b_{d_1} - b_{d_2}} \cdot e(Z, R)^{-b_{r_1} - b_{r_2}} \cdot e(P_1, P_2)^{-b_x},$$

He sends $(T_1, \ldots, T_5, B_1, \ldots, B_6)$ to the verifier, who sends a random challenge value $c \xleftarrow{R} \mathbb{Z}_p$ to the prover as a second move. The prover computes and sends back the values in response to the verifier as the last move:

$$s_{r_1} \leftarrow b_{r_1} + cr_1, \quad s_{r_2} \leftarrow b_{r_2} + cr_2, \quad s_{d_1} \leftarrow b_{d_1} + cd_1,$$
$$s_{d_2} \leftarrow b_{d_2} + cd_2, \quad s_x \leftarrow b_x + cx, \quad s_t \leftarrow b_t + ct \quad .$$

The verifier checks if the following equations hold; if they hold the verifier accepts, otherwise he rejects.

$$s_{r_1} X \stackrel{?}{=} cT_1 + B_1 \tag{1}$$

$$s_{r_2} Y \stackrel{?}{=} cT_2 + B_2 \tag{2}$$

$$s_t T_1 - s_{d_1} X \stackrel{?}{=} B_3 \tag{3}$$

$$s_t T_2 - s_{d_2} Y \stackrel{?}{=} B_4 \tag{4}$$

$$e(P_1, T_4)^{s_x} \stackrel{?}{=} T_5^c \cdot B_5 \tag{5}$$

$$e(T_3, P_2)^{s_t} \cdot e(Z, P_2)^{-s_{d_1} - s_{d_2}} \cdot e(Z, R)^{-s_{r_1} - s_{r_2}} \cdot e(P_1, P_2)^{-s_x} \stackrel{?}{=} (e(Q, P_2)/e(T_3, R))^c \cdot B_6 \quad . \tag{6}$$

Lemma 3. *Protocol 1 is complete.*

Lemma 4. *There exists a simulator for the transcripts of Protocol 1 for an honest verifier under the DLDH assumption.*

Lemma 5. *There exists an extractor for Protocol 1.*

Proof. We allow an extractor to rewind a prover in the protocol to the point just before the prover is given a challenge c. Then, the extractor can obtain two protocol transcripts :

$$(T_1, \ldots, T_5, B_1, \ldots, B_6, c, s_{r_1}, s_{r_2}, s_{d_1}, s_{d_2}, s_x, s_t)$$
$$(T_1, \ldots, T_5, B_1, \ldots, B_6, c^*, s^*_{r_1}, s^*_{r_2}, s^*_{d_1}, s^*_{d_2}, s^*_x, s^*_t) \ .$$

First observe that, from (1), $B_1 = s_{r_1} X - cT_1 = s^*_{r_1} X - c^* T_1$ from which we obtain $(c - c^*)T_1 = (s_{r_1} - s^*_{r_1})X$ and it follows that $\tilde{r_1} = (s_{r_1} - s^*_{r_1})(c - c^*)^{-1} (\mathrm{mod}\ p)$ satisfies $\tilde{r_1} X = T_1$. In a similar fashion we obtain from (2) $\tilde{r_2} = (s_{r_2} - s^*_{r_2})(c - c^*)^{-1} (\mathrm{mod}\ p)$ which satisfies $\tilde{r_2} Y = T_2$.

Next we have, from (5), $B_5 = e(P_1, T4)^{s_x} / T_5^c = e(P_1, T_4)^{s^*_x} / T_5^{c^*}$ from which we obtain $e(P_1, T_4)^{s_x - s^*_x} = T_5^{c - c^*}$ and it follows that $\tilde{x} = (s_x - s^*_x)(c - c^*)^{-1} (\mathrm{mod}\ p)$ satisfies $e(P_1, T_4)^{\tilde{x}} = T_5$.

Next we have, from (3), $B_3 = s_t T_1 - s_{d_1} X = s^*_t T_1 - s^*_{d_1} X$ from which we obtain $(s_t - s^*_t)T_1 = (s_{d_1} - s^*_{d_1})X$. Since $\tilde{r_1} X = T_1$, we have $(s_{d_1} - s^*_{d_1}) = \tilde{r_1}(s_t - s^*_t) (\mathrm{mod}\ p)$. Similarly, we have from (4) $(s_{d_2} - s^*_{d_2}) = \tilde{r_2}(s_t - s^*_t) (\mathrm{mod}\ p)$.

Finally, dividing two instances of (6), we obtain

$$\begin{aligned}
&(e(Q, P_2)/e(T_3, R))^{(c - c^*)} \\
&= e(T_3, P_2)^{(s_t - s^*_t)} \cdot e(Z, P_2)^{-(s_{d_1} - s^*_{d_1}) - (s_{d_2} - s^*_{d_2})} \cdot \\
&\quad e(Z, R)^{-(s_{r_1} - s^*_{r_1}) - (s_{r_2} - s^*_{r_2})} \cdot e(P_1, P_2)^{-(s_x - s^*_x)} \\
&= e(T_3, P_2)^{(s_t - s^*_t)} \cdot e(Z, P_2)^{-\tilde{r_1}(s_t - s^*_t) - \tilde{r_2}(s_t - s^*_t)} \cdot \\
&\quad e(Z, R)^{-(s_{r_1} - s^*_{r_1}) - (s_{r_2} - s^*_{r_2})} \cdot e(P_1, P_2)^{-(s_x - s^*_x)} \quad .
\end{aligned}$$

Taking $(c - c^*)$-th roots, we have

$$\begin{aligned}
&e(Q, P_2)/e(T_3, R) \\
&= e(T_3, P_2)^{\tilde{t}} \cdot e(Z, P_2)^{\tilde{t}(-\tilde{r_1}-\tilde{r_2})} \cdot e(Z, R)^{-\tilde{r_1}-\tilde{r_2}} \cdot e(P_1, P_2)^{-\tilde{x}} \\
&= e(T_3, \tilde{t}P_2) \cdot e(-(\tilde{r_1} + \tilde{r_2})Z, \tilde{t}P_2) \cdot e(-(\tilde{r_1} + \tilde{r_2})Z, R) \cdot e(\tilde{x}P_1, P_2)^{-1},
\end{aligned}$$

where $\tilde{t} = (s_t - s_t^*)(c - c^*)^{-1} (\mathrm{mod}\ p)$. This can be rearranged as

$$e(\tilde{x}P_1 + Q, P_2) = e(T_3 - (\tilde{r_1} + \tilde{r_2})Z, \tilde{t}P_2 + R) \quad .$$

Thus the extractor obtains an SDH representation $(T_3 - (\tilde{r_1} + \tilde{r_2})Z, \tilde{x}, \tilde{t})$. □

By the three lemmas above, the following holds.

Theorem 1. *Protocol 1 is an honest-verifier zero-knowledge proof of knowledge for an SDH representation under the DLDH assumption.*

4 The Traceable Signature Scheme

This section describes our traceable signature scheme. With Theorem 1, we obtain from Protocol 1 a signature scheme secure in the random oracle model by applying the Fiat-Shamir heuristic [1,12]. In our construction, in order to reduce the length of a signature we use a variant of the Fiat-Shamir heuristic used by Ateniese et al. [3], where the challenge c is included in the signature instead of $B_1, \ldots, B_6$. We verify the validity of the signature by checking whether the output of the random oracle is equal to the challenge c.

A traceable signature scheme consists of nine operations Setup, Join/Iss, Sign, Verify, Open, Reveal, Trace, Claim, Claim_Verify. The operations are executed by the active participants of the system, which are identified by the group manager, tracing agents, and a set of users.

Setup(1^k). For a given security parameter 1^k, the system is setup as follows:

$$\begin{aligned}
&\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e) \leftarrow \mathcal{G}_{BP}(1^k), \\
&\gamma \xleftarrow{R} \mathbb{Z}_p^*, \quad Q \xleftarrow{R} \mathbb{G}_1, \quad R \leftarrow \gamma P_2, \quad W \xleftarrow{R} \mathbb{G}_2 \setminus \{1_{\mathbb{G}_2}\}, \\
&Z \xleftarrow{R} \mathbb{G}_1 \setminus \{1_{\mathbb{G}_1}\}, \quad \xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p^*, \quad X \leftarrow \xi_1^{-1} Z, \quad Y \leftarrow \xi_2^{-1} Z \quad .
\end{aligned}$$

The system public key $\mathcal{Y}$ is $(\mathcal{G}, Q, R, W, X, Y, Z)$. The group manager's private key $\mathcal{S}$ is (γ, ξ_1, ξ_2). The scheme also selects an hash function $H : \{0,1\}^* \to \mathbb{Z}_p$, which is to be considered as a random oracle here.

Join$(\mathcal{Y})$/Iss$(\mathcal{Y}, \mathcal{S})$. By executing Join operation, a user joins this system. On the other part, when received a join request, the group manager gives a certificate to the requester by executing Iss operation. The Join/Iss operation is performed in a secure channel. The details are as follows.

1. A user i generates non-adaptive random x_i (see 4.2 in [16]) and sends $x_i P_1$ to the group manager. We will denote the i-th user's membership secret by $\mathsf{sec}_i = x_i$.
2. The group manager selects $t_i \xleftarrow{R} \mathbb{Z}_p^*$, computes $A_i = \frac{1}{t_i+\gamma}(x_i P_1 + Q)$, and then sends (i, A_i, t_i) to the user i. We will denote the i-th user's membership certificate by $\mathsf{cert}_i = (A_i, t_i)$.
3. The user i checks if (A_i, t_i) satisfies $e(A_i, t_i P_2 + R) = e(x_i P_1 + Q, P_2)$, and then stores $(i, \mathsf{cert}_i, \mathsf{sec}_i)$.
4. We will denote the join transcript between the the group manager and i-th user by $\mathsf{transcript}_i = (C_i = x_i P_1, A_i, t_i)$. The group manager stores $\mathsf{transcript}_i$ in the join transcript table $\mathsf{transcripts}$.

$\mathsf{Sign}(m, \mathcal{Y}, \mathsf{cert}_i, \mathsf{sec}_i)$. A member i of the group can sign a message m using this operation with his certificate $\mathsf{cert_i} = (A_i, t_i)$ and his secret $\mathsf{sec}_i = x_i$.

1. Compute the values $T_1, \ldots, T_5, B_1, \ldots, B_6$ using (A_i, x_i, t_i) according to Protocol 1.
2. Compute c using the hash function :

$$c \leftarrow H(m, T_1, \ldots, T_5, B_1, \ldots, B_6) \quad .$$

3. Compute the values $s_{r_1}, s_{r_2}, s_{d_1}, s_{d_2}, s_x, s_t$ using c according to Protocol 1.
4. The signature is $\sigma = (T_1, \ldots, T_5, c, s_{r_1}, s_{r_2}, s_{d_1}, s_{d_2}, s_x, s_t)$. If each of $T_1, \ldots, T_4$ is 170 bits, T_5 is 1024 bits, and $c, s_{r_1}, \ldots, s_t$ are 170 bits each, the signature size is about $(170 \times 11 + 1024)/8 = 362$ bytes.

$\mathsf{Verify}(m, \sigma, \mathcal{Y})$. The signature σ of message m can be verified with this operation.

1. Parse σ as $(T_1, \ldots, T_5, c, s_{r_1}, s_{r_2}, s_{d_1}, s_{d_2}, s_x, s_t)$.
2. Reconstruct $\tilde{B}_1, \ldots, \tilde{B}_6$ using equations (1)-(6).
3. Check if the following equation holds:

$$c \stackrel{?}{=} H(m, T_1, \ldots, T_5, \tilde{B}_1, \ldots, \tilde{B}_6) \quad .$$

   Return 1 if it holds and return 0 otherwise.

$\mathsf{Open}(\sigma, \mathcal{Y}, \mathcal{S})$. The group manager can find who signed the signature σ of m with this operation.

1. Parse σ as $(T_1, \ldots, T_5, c, s_{r_1}, s_{r_2}, s_{d_1}, s_{d_2}, s_{x_i}, s_{t_i})$.
2. Compute $\tilde{A} = T_3 - (\xi_1 T_1 + \xi_2 T_2)$ using (ξ_1, ξ_2) from $\mathcal{S}$.
3. Look up i in the join transcript table $\mathsf{transcripts}$ such that $A_i = \tilde{A}$ and return i.

$\mathsf{Reveal}(i, \mathsf{transcripts})$. The group manager can obtain the tracing information of user i with this operation. User i is a suspicious user (e.g., detected by an Open operation).

1. Return C_i of $\mathsf{transcript}_i = (C_i = x_i P_1, A_i, t_i)$ in the join transcript table $\mathsf{transcripts}$. Note that only the group manager can access the join transcript table $\mathsf{transcripts}$.

$\mathsf{Trace}(\sigma, C, \mathcal{Y})$. Tracing agents trace the signature generated by a suspicious user. The input C is a tracing information of the suspicious user given by the group manager. By executing this operation, tracing agents check whether σ is generated by the suspicious user. Note that all the signatures can be checked efficiently when many tracing agents execute this operation in parallel.

1. Parse σ as $(T_1, \ldots, T_5, c, s_{r_1}, s_{r_2}, s_{d_1}, s_{d_2}, s_x, s_t)$.
2. Check if the following equation holds: $e(C, T_4) \stackrel{?}{=} T_5$.
   Return 1 if it holds and return 0 otherwise.

Since $C = x_j P_1$ where j is the suspicious user $e(C, T_4)$ can be rewritten as $e(P_1, T_4)^{x_j}$. If the originator of σ is i, we have $T_5 = e(P_1, T_4)^{x_i}$. Therefore, if $i = j$ this procedure will return 1.

$\mathsf{Claim}(\sigma, \mathcal{Y}, \mathsf{sec}_i)$. The originator i of the signature σ can claim that he is its originator with this protocol.

1. Parse σ as $(T_1, \ldots, T_5, c, s_{r_1}, s_{r_2}, s_{d_1}, s_{d_2}, s_x, s_t)$.
2. Generate a proof of knowledge of the value x which satisfies $e(P_1, T_4)^x = T_5$, and return the proof τ. This is possible because if the prover is the real originator of σ, sec_i will be x_i such that $e(P_1, T_4)^{x_i} = T_5$.

$\mathsf{Claim\_Verify}(\sigma, \tau, \mathcal{Y})$. The output of Claim τ is verified with this operation.

1. Parse σ as $(T_1, \ldots, T_5, c, s_{r_1}, s_{r_2}, s_{d_1}, s_{d_2}, s_{x_i}, s_{t_i})$.
2. Verify the proof τ.

5 Correctness of the Traceable Signature Scheme

In this section we will prove the correctness of our scheme in the model of KTY [13].

Definition 6. Correctness of a Traceable Scheme. *A traceable signature scheme with security parameter k is* **correct** *if the following four conditions are satisfied (with overwhelming probability in k). Let $\mathsf{Sign}_{\mathcal{U}}$ be the signing mechanism of user $\mathcal{U}$ and $\mathsf{Claim}_{\mathcal{U}}$ its corresponding claim mechanism and $\mathcal{S}$ the group manager's private key.*

1. **Sign-Correctness:** *For all m,* $\mathsf{Verify}(m, \mathsf{Sign}_{\mathcal{U}}(m)) = 1$.
2. **Open-Correctness:** *For any m,* $\mathsf{Open}(\mathsf{Sign}_{\mathcal{U}}(m), \mathcal{S}) = \mathcal{U}$.
3. **Trace-Correctness:** *For any m,* $\mathsf{Trace}(\mathsf{Sign}_{\mathcal{U}}(m), \mathsf{Reveal}(\mathcal{U})) = 1$*; on the other hand* $\mathsf{Trace}(\sigma, \mathsf{Reveal}(\mathcal{U})) = 0$ *for any* $\sigma \leftarrow \mathsf{Sign}_{\mathcal{U}'}(m)$ *with* $\mathcal{U}' \neq \mathcal{U}$.
4. **Claim-Correctness:** $\mathsf{Claim\_Verify}(m, \sigma, \mathsf{Claim}_{\mathcal{U}}(\sigma)) = 1$ *for all* $m, \sigma \leftarrow \mathsf{Sign}_{\mathcal{U}}(m)$.

Theorem 2. *The traceable signature scheme of Section 4 is correct.*

6 Security Model of the Traceable Signature Scheme

We introduce in this section the security model of KTY [13].

6.1 Oracles

The security definitions will be formulated via experiments in which an adversary's attack capabilities are modelled by providing it with access to certain oracles. Oracles shares the following variables with each other.

- state: It contains the join transcript, certificates and secrets of users which are obtainable in the system's perspective.
- n : It is a counter that stores the number of users joining the system.
- Sigs : It contains signatures generated by $\mathsf{Q}_{\mathsf{sig}}$ oracle.
- Revs : It contains the members revealed by $\mathsf{Q}_{\mathsf{reveal}}$ oracle.
- U^p : It is a set of private users which are not corrupted.
- U^a : It is a set of users in corruption type A. We say that the user i falls in corruption type A if the adversary controls the user i. In this case $(i, \mathsf{cert}_i, \mathsf{sec}_i)$ is leaked to the adversary.
- U^b : It is a set of users in corruption type B. We say that the user i falls in corruption type B if the transcript during the join procedure is exposed to the adversary. In this case (i, cert_i) is leaked to the adversary.

The various oracle specifications are listed below.

- $\mathsf{Q}_{\mathcal{Y}}()$. This oracle returns $(\mathsf{n}, \mathcal{Y})$. It allows the adversary to learn the public-information of the system.
- $\mathsf{Q}_{\mathcal{S}}()$. This oracle returns $\mathcal{S}$. It allows the adversary to corrupt the group manager.
- $\mathsf{Q}_{\mathsf{p-join}}()$. This oracle simulates $\mathsf{Join/Iss}$ protocol in private, increases the user count n by 1, and sets $\mathsf{state} \leftarrow \mathsf{state}||(\mathsf{n}, \mathsf{transcript}_\mathsf{n}, \mathsf{cert}_\mathsf{n}, \mathsf{sec}_\mathsf{n})$. It also adds n into U^p and $\mathsf{transcripts} \leftarrow \mathsf{transcripts} \mid\mid (\mathsf{n}, \mathsf{transcript}_\mathsf{n})$.
- $\mathsf{Q}_{\mathsf{a-join}}()$. This oracle allows the adversary to introduce an adversarially controlled user to the system. The introduced user falls in corruption type A. Firstly, the oracle initiates $\mathsf{Join/Iss}$ protocol with the adversary. In the protocol, the oracles takes the role of the group manager and the adversary the prospective user. When the protocol terminates successfully, the oracle increases n by 1 and sets $\mathsf{state} \leftarrow \mathsf{state}||(\mathsf{n}, \mathsf{transcript}_\mathsf{n}, \mathsf{cert}_\mathsf{n}, \perp)$. It also adds n into U^a and $\mathsf{transcripts} \leftarrow \mathsf{transcripts} \mid\mid (\mathsf{n}, \mathsf{transcript}_\mathsf{n})$.
- $\mathsf{Q}_{\mathsf{t-join}}()$. This oracle is identical to the $\mathsf{Q}_{\mathsf{p-join}}()$ oracle except that at the end it transmits $(\mathsf{cert}_n, \mathsf{sec}_n)$ to the adversary and adds n to U^a (not U^p). As explained in [13], the statistical difference between $\mathsf{Q}_{\mathsf{a-join}}$ and $\mathsf{Q}_{\mathsf{t-join}}$ is negligible. Therefore, we can always replace $\mathsf{Q}_{\mathsf{a-join}}$ with $\mathsf{Q}_{\mathsf{t-join}}$ when it is hard to simulate the behavior of $\mathsf{Q}_{\mathsf{a-join}}$.
- $\mathsf{Q}_{\mathsf{b-join}}()$. This oracle allows the adversary as the group manager to introduce users. Users introduced with this oracle falls in corruption type B. Firstly, the oracle initiates $\mathsf{Join/Iss}$ protocol with the adversary. In the protocol, the oracles takes the role of the prospective user and the adversary the group manager. When the protocol terminates successfully, the oracle increases n by 1 and sets $\mathsf{state} \leftarrow \mathsf{state}||(\mathsf{n}, \perp, \mathsf{cert}_\mathsf{n}, \mathsf{sec}_\mathsf{n})$. It also adds n into U^b. It does not modify the join transcript table $\mathsf{transcripts}$ since this oracle behaves as a user.

- $\mathsf{Q}_{\mathsf{sig}}(i, m)$. This oracle returns a signature of message m by the i-th user. It parses state and if it discovers an entry of the form $(i, \cdot, \mathsf{cert}_i, \mathsf{sec}_i)$ it produces a traceable signature σ using cert_i and sec_i. If no such entry is found or $i \in U^a$, it returns $\perp$. When it successfully produces σ, it sets $\mathsf{Sigs} \leftarrow \mathsf{Sigs}||(i, \sigma)$.
- $\mathsf{Q}_{\mathsf{reveal}}(i)$. This oracle returns the output of $\mathsf{Reveal}(i, \mathsf{transcripts})$. Note that it returns $\perp$ when user i does not exist or $i \in U^b$. It also adds i into Revs.

6.2 Security Definitions of Traceable Signatures

Definition 7. *A traceable signature scheme is said to be* **secure** *if it satisfies security against misidentification, anonymity, and framing attacks.*

Misidentification Attacks. In a misidentification attack the adversary is allowed to control a number of users of the system ($\mathsf{Q}_{\mathsf{a-join}}$, $\mathsf{Q}_{\mathsf{p-join}}$). The adversary is also allowed to observe the operation of the system while users are added and they produce signatures ($\mathsf{Q}_{\mathsf{p-join}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}}$). Finally, the adversary is required to produce a signature that does not open to any of the users controlled by the adversary or that does not trace to any of the users controlled by the adversary.

Experiment $\mathbf{Exp}^{\mathtt{mis}}_{\mathcal{A}}(k)$
$(\mathcal{Y}, \mathcal{S}) \leftarrow \mathsf{Setup}(1^k)$;
$(m, \sigma) \leftarrow A(\mathsf{Q}_{\mathcal{Y}}, \mathsf{Q}_{\mathsf{p-join}}, \mathsf{Q}_{\mathsf{a-join}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}})$;
If $\mathsf{Verify}(m, \sigma, \mathcal{Y}) = 0$ then return 0;
If $\mathsf{Open}(\sigma, \mathcal{Y}, \mathcal{S}) \notin U^a$ or $\bigwedge_{i \in U^a} \mathsf{Trace}(\sigma, \mathsf{Reveal}(i)) = 0$ then return 1;
return 0;

Definition 8. *A traceable signature scheme is secure against misidentification attacks if for any PPT algorithm* $\mathcal{A}$ $\mathbf{Pr}[\mathbf{Exp}^{\mathtt{mis}}_{\mathcal{A}}(k) = 1]$ *is negligible in* k.

Framing Attacks. In a framing attack, the adversary is allowed to act as a group manager. The adversary is also allowed to observe the operation of the system while users are added and they produce signatures. There are two types of successful framing attacks. First, the adversary may construct a signature that opens or traces to an innocent user. Second, it may claim a signature that was generated by another user as its own. Note that in this attack the adversary observes the operations as a group manager which are simulated through $\mathsf{Q}_{\mathcal{S}}$, $\mathsf{Q}_{\mathsf{b-join}}$, and $\mathsf{Q}_{\mathsf{sig}}$ oracles.

Experiment $\mathbf{Exp}^{\mathtt{fra}}_{\mathcal{A}}(k)$
$(\mathcal{Y}, \mathcal{S}) \leftarrow \mathsf{Setup}(1^k)$;
$(m, \sigma, \tau) \leftarrow A(\mathsf{Q}_{\mathcal{Y}}, \mathsf{Q}_{\mathcal{S}}, \mathsf{Q}_{\mathsf{b-join}}, \mathsf{Q}_{\mathsf{sig}})$;
If $\mathsf{Verify}(m, \sigma, \mathcal{Y}) = 0$ then return 0;
If $\mathsf{Open}(\sigma, \mathcal{Y}, \mathcal{S}) \in U^b$ or $\bigvee_{i \in U^b} \mathsf{Trace}(\sigma, \mathsf{Reveal}(i)) = 1$ then return 1;
If $\left(\bigvee_{i \in U^b} (i, \sigma) \in \mathsf{Sigs}\right) \wedge (\mathsf{Claim\_Verify}(\sigma, \tau) = 1)$ then return 1;
return 0;

Definition 9. *A traceable signature scheme is secure against framing attacks if for any PPT algorithm* $\mathcal{A}$ $\mathbf{Pr}[\mathbf{Exp}^{\mathtt{fra}}_{\mathcal{A}}(k) = 1]$ *is negligible in* k.

Anonymity Attacks. In an anonymity attack, the adversary operates in two stages called play and guess. In the play stage, the adversary is allowed to join the system through $\mathsf{Q}_{\mathsf{a-join}}$ oracles. The adversary is also allowed to observe the operation of the system while users are added and they produce signatures through $\mathsf{Q}_{\mathsf{p-join}}, \mathsf{Q}_{\mathsf{sig}}$, and $\mathsf{Q}_{\mathsf{reveal}}$ oracles. At end of the play stage, the adversary returns a message and two target users he does not control, and then receives a signature of the message he returned. In the guess stage, the adversary tries to guess which of the two produced the signature.

Experiment $\mathbf{Exp}^{\mathtt{anon}}_{\mathcal{A}}(k)$
$(\mathcal{Y}, \mathcal{S}) \leftarrow \mathsf{Setup}(1^k)$;
$(aux, m, i_0, i_1) \leftarrow A(\mathsf{play} : \mathsf{Q}_{\mathcal{Y}}, \mathsf{Q}_{\mathsf{p-join}}, \mathsf{Q}_{\mathsf{a-join}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}})$;
If $(i_0 \notin U^p) \vee (i_1 \notin U^p) \vee (i_0 \in \mathsf{Revs}) \vee (i_1 \in \mathsf{Revs})$ then return 0;
$b \xleftarrow{R} \{0,1\}, \quad \sigma \leftarrow \mathsf{Sign}(m, \mathcal{Y}, \mathsf{cert}_{i_b}, \mathsf{sec}_{i_b})$;
$b^* \leftarrow A(\mathsf{guess}, \sigma, aux : \mathsf{Q}_{\mathcal{Y}}, \mathsf{Q}_{\mathsf{p-join}}, \mathsf{Q}_{\mathsf{a-join}}, \mathsf{Q}_{\mathsf{sig}}, \mathsf{Q}_{\mathsf{reveal}})$;
If $(i_0 \in \mathsf{Revs}) \vee (i_1 \in \mathsf{Revs})$ then return 0;
If $b = b^*$ then return 1;
return 0;

Definition 10. *A traceable signature scheme is secure against anonymity attacks if for any PPT algorithm* $\mathcal{A}$ $\left|\mathbf{Pr}[\mathbf{Exp}^{\mathtt{anon}}_{\mathcal{A}}(k)] - \frac{1}{2}\right|$ *is negligible in* k.

6.3 Security of Our Scheme

These lemmas show the security properties.

Lemma 6. *Under the q-SDH assumption, our scheme is secure against misidentification attacks provided that the number of joined users is less than or equal to q (an adaptable "assumption parameter" that does not influence the complexity).*

Proof. Let $\mathcal{A}$ be an adversary that violates security against misidentification, We construct an algorithm $\mathcal{B}$ which solves one more representation problem using the attacker $\mathcal{A}$. SDH representations for $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$ and (Q, R) are given as $\mathsf{Refs} = \{(A_l, t_l, x_l)\}_{l=1}^{q}$. $\mathcal{B}$ chooses $W \xleftarrow{R} \mathbb{G}_2, Z \xleftarrow{R} \mathbb{G}_1$ and $\xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p^*$. It then sets $X \leftarrow \xi_1^{-1} Z, Y \leftarrow \xi_2^{-1} Z$. $\mathcal{B}$ simulates oracles allowed to $\mathcal{A}$ as follows.

- $\mathsf{Q}_{\mathcal{Y}}()$. It returns n and $(\mathcal{G}, Q, R, W, X, Y, Z)$.
- $\mathsf{Q}_{\mathsf{p-join}}()$. $\mathcal{B}$ increments n by one, and chooses $A \xleftarrow{R} \mathbb{G}_1$ and $t, x \xleftarrow{R} \mathbb{Z}_p^*$. It sets $\mathsf{cert}_{\mathsf{n}} \leftarrow (A, t), \mathsf{sec}_{\mathsf{n}} \leftarrow x$. It then sets $\mathsf{state} \leftarrow \mathsf{state} \,||(\mathsf{n}, \perp, \mathsf{cert}_{\mathsf{n}}, \mathsf{sec}_{\mathsf{n}})$ and $\mathsf{transcripts} \leftarrow \mathsf{transcripts}||(\mathsf{n}, \perp)$. Also, $\mathcal{B}$ adds n into U^p.

- $\mathsf{Q}_{\mathsf{t-join}}$. $\mathcal{B}$ increments j which is a counter representing how many SDH representations have been consumed and then gets $\mathsf{Refs}_j = (A_j, x_j, t_j)$ from Refs. It gives $(\mathsf{cert}_\mathsf{n} = (A_j, t_j), \mathsf{sec}_\mathsf{n} = x_j)$ to $\mathcal{A}$. It then sets $\mathsf{state} \leftarrow \mathsf{state} \,||(\mathsf{n}, \perp, \mathsf{cert}_\mathsf{n}, \mathsf{sec}_\mathsf{n})$ and $\mathsf{transcripts} \leftarrow \mathsf{transcripts}||(\mathsf{n}, \perp)$. Also, $\mathcal{B}$ adds n into U^a.
- $\mathsf{Q}_{\mathsf{sig}}(i, m)$. If $i \notin U^p$, $\mathcal{B}$ returns "fail" to $\mathcal{A}$. If $i \in U^p$, it simulates Protocol 1 with cert_i from state and get a signature σ, which is is possible because Protocol 1 has a simulator. Also, $\mathcal{B}$ sets $\mathsf{Sig} \leftarrow \mathsf{Sig}||(i, \sigma)$. Note that the value c which is selected by the simulator during simulation must be stored in the hash oracle such that the hash oracle should keep the random oracle property.
- $\mathsf{Q}_{\mathsf{reveal}}(i)$. $\mathcal{B}$ searches from state an entry of the form $(i, \cdot, \cdot, \mathsf{sec}_i = x_i)$, and returns $C_i = x_i P_1$. It also adds i into Revs.

Let (m, σ_1^*) be a successful output of algorithm $\mathcal{A}$. Using the general forking lemma [14], $\mathcal{B}$ can get another pair (m, σ_2^*) which is also valid. Because Protocol 1 has a knowledge extractor, an SDH representation $(\tilde{A}, \tilde{x}, \tilde{t})$ used in signing m can be extracted.

Now we have two alternative cases: (i) $\mathsf{Open}(\sigma, \mathcal{Y}, \mathcal{S}) \notin U^a$. It means that $\tilde{A}$ is not equal to any A_i for those $i \in U^a$. As a result, we solved one more SDH representation problem. In the second case we have: (ii) $\bigwedge_{i \in U^a} \mathsf{Trace}(\sigma, \mathsf{Reveal}(i)) = 0$. It means that $\tilde{x}$ is not equal to any x_i for those $i \in U^a$. Again we solved one more SDH representation problem. □

Lemma 7. *Under the assumption of infeasibility of discrete logarithm in $\mathbb{G}_1$, our scheme is secure against framing attacks.*

Proof. Let $\mathcal{A}$ be an adversary that violates security against framing attacks. We construct an algorithm $\mathcal{B}$ which solves a problem of discrete logarithm in $\mathbb{G}_1$ using the attacker $\mathcal{A}$. $\mathcal{B}$ is given P_1 and $S = \rho P_1 \in \mathbb{G}_1$ as input, and $\mathcal{B}$ wants to find ρ. We assume $\mathbb{G}_1$ is a group such that $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$ is easily obtained. $\mathcal{B}$ selects $Q \xleftarrow{R} \mathbb{G}_1$, $R \leftarrow \gamma P_2$ with $\gamma \xleftarrow{R} \mathbb{Z}_p^*$, and $W \xleftarrow{R} \mathbb{G}_2$. It also chooses $\xi_1, \xi_2, y \xleftarrow{R} \mathbb{Z}_p^*$ and $Y \leftarrow y P_1$ and then set $Z \leftarrow \xi_2 Y, X \leftarrow \xi_1^{-1} Z$. Then, $\mathcal{Y}$ becomes $(\mathcal{G}, Q, R, W, X, Y, Z)$ and $\mathcal{S}$ becomes (γ, ξ_1, ξ_2). $\mathcal{B}$ simulates oracles allowed to $\mathcal{A}$ as follows.

- $\mathsf{Q}_{\mathcal{Y}}()$. $\mathcal{B}$ returns n and $\mathcal{Y}$.
- $\mathsf{Q}_{\mathcal{S}}()$. $\mathcal{B}$ returns $\mathcal{S}$.
- $\mathsf{Q}_{\mathsf{b-join}}()$. $\mathcal{B}$ increments n by one. In the 1st step of $\mathsf{Join/Iss}$ procedure, $\mathcal{B}$ chooses $x'_\mathsf{n} \xleftarrow{R} \mathbb{Z}_p^*$, and supplies $x'_\mathsf{n} P_1 + S$ as the $x_\mathsf{n} P_1$ value. It receives the tuple $(A_\mathsf{n}, t_\mathsf{n})$ from $\mathcal{A}$, in the 2nd step. When the procedure is finished successfully, it sets $\mathsf{cert}_\mathsf{n} \leftarrow (A_\mathsf{n}, t_\mathsf{n})$, $\mathsf{sec}_\mathsf{n} \leftarrow x'_\mathsf{n} + \rho = x_\mathsf{n}$ where ρ is unknown. It then appends $(\mathsf{n}, \perp, \mathsf{cert}_\mathsf{n}, \mathsf{sec}_\mathsf{n})$ to state. Finally, it adds i into U^b.
- $\mathsf{Q}_{\mathsf{sig}}(i, m)$. If $i \in U^b$, $\mathcal{B}$ extracts $\mathsf{cert}_i = (A_i, t_i)$ and $\mathsf{sec}_i = x'_i + \rho$ from state. Then it chooses $r_1, r_2, r_3 \xleftarrow{R} \mathbb{Z}_p^*$, and sets $T_1, \ldots, T_5$ as follows :

$$T_1 \leftarrow r_1 X, \quad T_2 \leftarrow r_2 Y, \quad T_3 \leftarrow A_i + (r_1 + r_2) Z,$$
$$T_4 \leftarrow r_3 W, \quad T_5 \leftarrow e(x'_i P_1 + S, T_4) = e((x'_i + \rho) P_1, T_4) = e(P_1, T_4)^{x_i} \quad .$$

And then it simulates Protocol 1 and get a signature σ. This is possible because Protocol 1 has a simulator. Also, it sets $\mathsf{Sigs} \leftarrow \mathsf{Sigs}||(i, \sigma)$. Note that the value c which is selected by the simulator during simulation must be stored in the hash oracle such that the hash oracle should keep the random oracle property.

Let $(m, \sigma_1^*, \tau_1^*)$ be a successful output of algorithm $\mathcal{A}$. Now we have three cases: (i) $\mathsf{Open}(\sigma_1^*, \mathcal{Y}, \mathcal{S}) \in U^b$. Using the general forking lemma [14], $\mathcal{B}$ can get another pair (m, σ_2^*) which is also valid. Moreover with the knowledge extractor in Protocol 1, an SDH representation $(\tilde{A}, \tilde{x}, \tilde{t})$ used in signing m can be extracted. Let $i \in U^b$ be the result of Open. Then, $\tilde{x}$ is equal to sec_i, which means $\tilde{x} = x_i' + \rho$. Therefore $\mathcal{B}$ can find ρ. In the second case we have: (ii) $\bigvee_{i \in U^b} \mathsf{Trace}(\sigma_1^*, \mathsf{Reveal}(i)) = 1$. It can also extract an SDH representation $(\tilde{A}, \tilde{x}, \tilde{t})$ using a similar method to the case (i). We get also here $\tilde{x} = \mathsf{sec}_i = x_i' + \rho$. Therefore $\mathcal{B}$ can find ρ. In the final case we have: (iii) $\left(\bigvee_{i \in U^b}(i, \sigma_1^*) \in \mathsf{Sigs}\right) \wedge (\mathsf{Claim\_Verify}(\sigma_1^*, \tau_1^*) = 1)$ Using the general forking lemma [14], $\mathcal{B}$ can get another proof τ_2^* for σ_1^*. Then with a knowledge extractor (actually it is a subpart of the extractor in Protocol 1), a secret $\tilde{x}$ used for claim can be extracted. Let i be such that $(i, \sigma_1^*) \in \mathsf{Sigs}$. Then, we have $\tilde{x} = \mathsf{sec}_i = x_i' + \rho$. Therefore $\mathcal{B}$ can find ρ. □

Lemma 8. *Under the assumption of semantic security of Linear encryption, our scheme is secure against anonymity attacks.*

Proof. Let $\mathcal{A}$ be an adversary that violates security against anonymity. We construct an algorithm $\mathcal{B}$ which breaks the semantic security of Linear encryption using the attacker $\mathcal{A}$. $\mathcal{B}$ is given $X, Y, Z \in \mathbb{G}_1$ as input which are a public key for Linear encryption, and tries to break the semantic security of this encryption scheme.

We assume $\mathbb{G}_1$ is a group such that $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$ is easily obtained. $\mathcal{B}$ selects $Q \xleftarrow{R} \mathbb{G}_1$ and $R \leftarrow \gamma P_2$ where $\gamma \xleftarrow{R} \mathbb{Z}_p^*$. It also selects $W \xleftarrow{R} \mathbb{G}_2$, and set $\mathcal{Y}$ to $(\mathcal{G}, Q, R, W, X, Y, Z)$. $\mathcal{B}$ simulates oracles allowed to $\mathcal{A}$ as follows.

- $\mathsf{Q}_{\mathcal{Y}}()$. $\mathcal{B}$ returns n and $\mathcal{Y}$.
- $\mathsf{Q}_{\mathsf{p-join}}()$. $\mathcal{B}$ increments n by one, and chooses $A \xleftarrow{R} \mathbb{G}_1$ and $t, x \xleftarrow{R} \mathbb{Z}_p^*$. It sets $\mathsf{cert}_{\mathsf{n}} \leftarrow (A, t), \mathsf{sec}_{\mathsf{n}} \leftarrow x$. It then sets $\mathsf{state} \leftarrow \mathsf{state}\ ||(\mathsf{n}, \perp, \mathsf{cert}_{\mathsf{n}}, \mathsf{sec}_{\mathsf{n}})$ and $\mathsf{transcripts} \leftarrow \mathsf{transcripts}||(\mathsf{n}, \perp)$. Also, $\mathcal{B}$ adds n into U^p.
- $\mathsf{Q}_{\mathsf{t-join}}()$. $\mathcal{B}$ generates an SDH representation (A_j, x_j, t_j) for $(\mathcal{G}, Q, R)$ which is possible because it knows γ, and then gives $(\mathsf{cert}_{\mathsf{n}} = (A_j, t_j), \mathsf{sec}_{\mathsf{n}} = x_j)$ to $\mathcal{A}$. It sets $\mathsf{state} \leftarrow \mathsf{state}\ ||(\mathsf{n}, \perp, \mathsf{cert}_{\mathsf{n}}, \mathsf{sec}_{\mathsf{n}})$ and $\mathsf{transcripts} \leftarrow \mathsf{transcripts}||(\mathsf{n}, \perp)$. Also, $\mathcal{B}$ adds n into U^a.
- $\mathsf{Q}_{\mathsf{sig}}(i, m)$. If $i \notin U^p$, $\mathcal{B}$ returns "fail" to $\mathcal{A}$. If $i \in U^p$, $\mathcal{B}$ finds $\mathsf{cert}_i = (A_i, t_i)$ and $\mathsf{sec}_i = x_i$ from state, and encrypts A_i using the given public key (X, Y, Z). The resulting cipher-text will be T_1, T_2, T_3, and then it sets $T_4 \leftarrow r_3 W, T_5 \leftarrow e(P_1, T_4)^{x_i}$ where $r_3 \xleftarrow{R} \mathbb{Z}_p^*$. It simulates Protocol 1 to generate a signature. This is possible because Protocol 1 has a simulator.

Table 1. Comparison of number of operations and signature length (in bytes)

		KTY	Nguyen et al.	Ours
sign	exponentiations	19	11	6
	scalar multiplications	0	19	10
	pairing computations	0	1	1
verify	exponentiations	17	6	7
	scalar multiplications	0	14	8
	pairing computations	0	3	3
signature length		1206	917	362

Also, it sets $\mathsf{Sigs} \leftarrow \mathsf{Sigs}||(i, \sigma)$. Note that the value c which is selected by the simulator during simulation must be stored in the hash oracle such that the hash oracle should keep the random oracle property.

– $\mathsf{Q}_{\mathsf{reveal}}(i)$. $\mathcal{B}$ searches $\mathsf{sec}_i = x_i$ from state, and returns $x_i P_1$. It also adds i into Revs.

When $\mathcal{A}$ returns i_0 and i_1 to $\mathcal{B}$ as a challenge after play stage, $\mathcal{B}$ returns A_{i_0} and A_{i_1} as a challenge. $\mathcal{B}$ will be given the cipher-text (T_1', T_2', T_3') of A_{i_b}, where b is unknown. It generates a signature σ' containing (T_1', T_2', T_3') using a simulator for Protocol 1, and the value c must also be stored in a hash oracle. It returns σ' to $\mathcal{A}$. Let b^* be the output of $\mathcal{A}$ after the guess stage. $\mathcal{B}$ returns b^*. If $\mathcal{A}$ breaks anonymity, then it is obvious that $\mathcal{B}$ also breaks semantic security. □

As a result of the lemmas we conclude the following:

Theorem 3. *Under the q-SDH assumption, infeasibility of discrete logarithm in $\mathbb{G}_1$, and the semantic security of Linear encryption, our traceable signature scheme is secure provided that the number of joined users is less than or equal to q .*

7 Efficiency

In this section we compare our scheme with previous schemes in terms of the signature length and the number of important operations such as exponentiations, scalar multiplications and pairing computations. We summarize the result in Table 1. we did not include pre-computable operations such as $e(P_1, P_2)$ in the number of pairing computations. While the numbers of operations are comparable in the three schemes, the signature length of our scheme is much shorter than those of the previous schemes.

8 Conclusion

We presented a traceable signature scheme based on Strong Diffie-Hellman and Decision Linear Diffie-Hellman assumptions. The scheme uses bilinear pairings, and we get a signature under 400 bytes when any of the curves in [6] are used. We have proved the correctness and security of our scheme.

References

1. M. Abdalla, J. An, M. Bellare, and C. Namprepre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. *EUROCRYPT 2002, LNCS, Springer.*
2. G. Ateniese, J. Camenisch, S. Hohenberger, and B. Medeiros. Practical Group Signatures without Random Oracles. *Cryptology ePrint Archive, Report 2005/385, http://eprint.iacr.org/.*
3. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. *Crypto 2000, LNCS, Springer.*
4. D. Boneh, X. Boyen. Short Signatures Without Random Oracles. *EUROCRYPT 2004, LNCS, Springer.*
5. D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. *Crypto 2004, LNCS, Springer.*
6. D. Boneh, B. Lynn, and H.Shacham. Short signatures from Weil pairing. *Asiacrypt 2001, LNCS, Springer.*
7. M. Bellare, D. Micciancio, and B.Warinshci. Foundations of group signatures: Formal definitions, siimplified requirements, and a construction based on general assumptions. *EUROCRYPT 2003, LNCS, Springer.*
8. M. Bellare, H. Shi, C. Zhang. Foundations of group signatures: The case of dynamic groups. *Cryptology ePrint Archive, Report 2004/077, http://eprint.iacr.org/.*
9. X. Boyen and B. Waters. Compact Group Signatures Without Random Oracles. *Cryptology ePrint Archive, Report 2005/381, http://eprint.iacr.org/.*
10. J. Camenisch. Efficient and generalized group signatures. *EUROCRYPT 1997, LNCS, Springer.*
11. D. Chaum and E. van Heyst. Group signatures. *EUROCRYPT 1991, LNCS, Springer.*
12. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. *Crypto 1986, LNCS, Springer.*
13. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable Signatures. *EUROCRYPT 2004, LNCS, Springer.*
14. A. Kiayias and M. Yung. Group signatures: Efficient constructions and anonymity from trapdoor-holders. *Cryptology ePrint Archive, Report 2004/076. http://eprint.iacr.org/.*
15. A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals, E84-A(5):1234-43, May 2001.*
16. L. Nguyen and R. Safavi-Naini. Efficient and Provably Secure Trapdoor-free Group Signatures Schemes from Bilinear Pairings. *Asiacrypt 2004, LNCS, Springer.*
17. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology, 13(3):361-396, 2000.*
18. K. Rubin and A. Silverberg. Supersingular Abelian varieties in cryptology. *Crypto 2002, LNCS, Springer.*
19. C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology, 4(3):161-174, 1991.*