Real Traceable Signatures

Sherman S.M. Chow

Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu

Abstract. Traceable signature scheme extends a group signature scheme with an enhanced anonymity management mechanism. The group manager can compute a tracing trapdoor which enables anyone to test if a signature is signed by a given misbehaving user, while the only way to do so for group signatures requires revealing the signer of all signatures. Nevertheless, it is not tracing in a strict sense. For all existing schemes, T tracing agents need to recollect all N' signatures ever produced and perform RN' "checks" for R revoked users. This involves a high volume of transfer and computations. Increasing T increases the degree of parallelism for tracing but also the probability of "missing" some signatures in case some of the agents are dishonest.

We propose a new and efficient way of tracing – the tracing trapdoor allows the reconstruction of tags such that each of them can uniquely identify a signature of a misbehaving user. Identifying N signatures out of the total of N' signatures ($N << N'$) just requires the agent to construct N small tags and send them to the signatures holder. N here gives a trade-off between the number of unlinkable signatures a member can produce and the efforts for the agents to trace the signatures. We present schemes with simple design borrowed from anonymous credential systems. Our schemes are proven secure respectively in the random oracle model and in the common reference string model (or in the standard model if there exists a trusted party for system parameters initialization).

Keywords: traceable signatures, efficient tracing, group signatures, anonymity management, bilinear groups, standard model.

1 Introduction

Group signature is one of the important privacy enhancing cryptographic primitives. Each group member can sign a message on behalf of a group such that anyone can verify that the group signature is produced by someone enrolled to the group, but not exactly whom. In other words, one can give a proof of group-membership without revealing the true identity.

Unconditional anonymity may be abused by misbehaving users, and timely identification of "bad" signatures is of the utmost importance. For example, consider the use of group signature in the Vehicle Safety Communications (VSC)

M.J. Jacobson Jr., V. Rijmen, and R. Safavi-Naini (Eds.): SAC 2009, LNCS 5867, pp. 92–107, 2009.

system from the Department of Transportation in the U.S. [19], any wrong traffic information purported by a misbehaving driver or a compromised car should be identified to avoid possibly traffic accident which may cost human life.

Group signatures come with a mechanism which allows the group manager (GM) to "open" a signature and reveal the true signer by the GM's decision. To identity any signatures previously generated by a misbehaving user, the GM is required to open all signatures. This incurs three problems – it penalizes the privacy of all other good users, and imposes a high computational overhead on the GM. Besides, these signatures may be distributed in various locations (e.g. in the VSC scenario) and the GM needs to re-collect all these signatures, which may delay the identification of bad signatures. In view of these shortcomings, Kiayias, Tsiounis and Yung [16] proposed the concept of traceable signatures.

Traceable signature is a group signature with an enhanced anonymity management mechanism. Opening of the signatures is no longer the only option. The GM can compute a user-specific tracing trapdoor which enables anyone to test if a signature is signed by a given user. In this way, the objective of identifying all the signatures produced by a misbehaving user can be achieved, without compromising the privacy of all other good users.

Nevertheless, we found that the latter two problems remain unsolved. The tracing mechanism of the existing schemes [10,13,16,18] actually does not trace the signatures. Instead, it *checks* whether a signature was issued by a given user. The GM may delegate the trapdoor to many tracing agents (TA's) to check in parallel, but the TA's still need to recollect all N' signatures ever produced and perform RN' invocations of the "tracing" algorithm for R revoked users in total. This involves a high volume of transfer and computations. There is also a trade-off between the degrees of parallelism and the trust on the TA's. The more TA's employed, the higher chance that a TA may "miss" some signatures, either intentionally or accidentally.

In this paper, we propose a new and efficient way of tracing – the user-specific tracing trapdoor allows the reconstruction of tags such that each of them can uniquely identify a signature of a misbehaving user. Identifying N signatures just requires the agent to invoke N tag-reconstruction and send these N small tags to the signatures holder, instead of requiring the transfer of $N' >> N$ signatures in the traditional approach. Our new traceable signatures still enjoy the original applications mentioned in [16], namely, transforming an anonymous system to one with "fair privacy", a mix-net application where originators of messages can be opened, and open-bid auctions.

We present schemes with simple design borrowed from existing anonymous credential systems. In particular, [8] has briefly mentioned that their compact e-cash system can be viewed as a bounded group signature scheme supporting efficient tracing. Our schemes are proven secure respectively in the random oracle (RO) model and in the common reference string (CRS) model (or in the standard model if there exists a trusted party for system parameters initialization). The former is more efficient while the latter gives a more modular design and higher security guarantees.

2 Design of Traceable Signatures and Building Blocks

Before we delve into the formal definition of traceable signature and our constructions, we first talk about its high-level design, which motivates the discussion of several building blocks. Since traceable signature is an enhancement of group signature, we start by the latter.

2.1 High Level Designs

Group Signatures. When a user joins a group, the GM gives this new member a signature. The user presents this signature to a verifier to show the membership. However, a verifier who got the same signature can claim the membership too. This means the GM should sign on something that is related to some valuable secrets of the users, such that they would not share it with other easily. To sign on behalf of the group, the user should generate another signature as well. This latter signature can be given by the member's own private key. The signature of the GM and the member's own private key form the credential of a member.

We assume the private key is valuable, and a user does not want to leak this private key to any one, including the GM (for exculpability). It can be "hidden" in the form of a *commitment*. The GM can then give a *signature on the commitment*. The GM may also store part of the communication with this user in an archive. This concludes the joining stage.

A group member wants to preserve anonymity in signing. There should be a *protocol for proof of knowledge (PoK) of a signature*. Another feature of group signature is that it can be opened to reveal the true signer. Thus, it should contain an *encryption* of some information that uniquely identifies a user, such that only a designated party (e.g. the GM) can decrypt it. There should be a way to let any verifier to know that this encryption has been done correctly, so another *zero-knowledge protocol for showing the correctness of the encryption* is needed. All these proofs by the signer should be verifiable by everyone, hence they must be non-interactive. The proof should also be *witness-indistinguishable* such that it is generated equally likely by each possible credential (the witness). This concludes our discussions on the idea of group signature.

Traceable Signatures. The traditional way of tracing only tells if a signature is given by a particular user. We know from the existing schemes that a function of the user's private key can serve as this "decisional" trapdoor which supports an efficient detection mechanism.

For our new way of tracing, we found that it is easier for the GM to generate the user-specific tracing trapdoor instead, which is also stored in the archive. To make sure the signature of a member should be related to this chosen value, the GM should give a *signature on a block of messages*, i.e. on the trapdoor and a commitment of the user's private key. This may not be required for the typical group signature schemes or the traditional traceable signature schemes.

To enable a tracing agent TA to uniquely identify every signatures produced by a particular member, everyone is required to compute a deterministic tag

based on a seed given by the GM. A TA can then reproduce the tag to identify the signature. The tag is generated by a *pseudorandom function* taking the secret seed and the counter value as its input, which makes the signatures produced by the same user remain unlinkable to any one without the trapdoor.

Now we have a function that is only computable with the help of a secret seed, but what should be its input? We cannot afford an exponentially-large domain here since it will be time-consuming for a TA to re-generate all possible tags. We thus need to confine the domain. The verifier should ensure that this input value is an integer less than a limit N pre-selected by the GM. This can be done with the help of a zero-knowledge *range proof*.

The above idea has actually been employed by compact e-cash systems [4,8] to support "join once, spend many". In our case, we use the deterministic recovery nature to support efficient tracing. A weakness of this approach is that, N gives a trade-off between the number of unlinkable signature a member can produce and the efforts to trace the signatures.

2.2 Number-Theoretic Preliminaries

A mapping $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing if

- $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic multiplicative groups of prime order p.
- g, h are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.
- $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ is a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(h) = g$.
- Each group element has a unique binary representation.
- (Bilinear) $\forall x \in \mathbb{G}_1$, $y \in \mathbb{G}_2$ and a, $b \in \mathbb{Z}_p$, $\hat{e}(x^a, y^b) = \hat{e}(x, y)^{ab}$.
- (Non-degenerate) $\hat{e}(g, h) \neq 1$.

$\mathbb{G}_1$ and $\mathbb{G}_2$ can be the same group or different groups. We say that $(\mathbb{G}_1, \mathbb{G}_2)$ is a bilinear group pair if the group action in $\mathbb{G}_1$, $\mathbb{G}_2$, ψ and $\hat{e}$ are all efficiently computable. We name $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g, h)$ as bilinear map context $params_{BM}$.

Definition 1 (Decisional Diffie-Hellman (DDH)). *The DDH problem in* $\mathbb{G}$ *is, on input a quadruple* $(g, g^a, g^b, g^c) \in \mathbb{G}^4$*, output 1 if* $c = ab$ *and 0 otherwise.*

Definition 2 (eXternal Diffie-Hellman (XDH)). *The XDH problem in a bilinear group pair* $(\mathbb{G}_1, \mathbb{G}_2)$ *with trace map* ψ *is to solve the DDH problem in* $\mathbb{G}_1$*. If XDH is hard, there exists no efficiently computable isomorphism* $\psi' : \mathbb{G}_1 \to \mathbb{G}_2$*.*

Definition 3 (Decisional Linear (DLin)). *The DLin problem in* $\mathbb{G} = \langle g \rangle$ *is defined as follow: On input a sextuple* $(u, v, g, u^a, v^b, g^c) \in \mathbb{G}^6$*, decide if* $c = a+b$*.*

Definition 4 (q-Decisional Diffie-Hellman Inversion (q-DDHI)). *The q-DDHI problem in prime order group* $\mathbb{G} = \langle g \rangle$ *is defined as follow: On input a* $(q+2)$*-tuple* $(g, g^x, g^{x^2}, \ldots, g^{x^q}, g^c \in \mathbb{G}^{q+2})$*, decide if* $c = 1/x$*.*

Definition 5 (q-Strong Diffie-Hellman (q-SDH)). *The q-SDH problem in a bilinear group pair* $(\mathbb{G}_1, \mathbb{G}_2)$ *with an efficient (computable in polynomial time) trace map* $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ *is, on input a* $(q+2)$*-tuple* $(g, h, h^\gamma, h^{\gamma^2}, \ldots, h^{\gamma^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$ *where* $g = \psi(h)$*, output a pair* $(B, e) \in \mathbb{G}_1 \times \mathbb{Z}_p^*$ *such that* $B^{(\gamma+e)} = g$*.*

We say that an X assumption holds if no probabilistic polynomial time algorithm has non-negligible advantage (over random guessing if X is decisional) in solving problem X. The q-SDH assumption in $(\mathbb{G}_1, \mathbb{G}_2)$ with a trace map $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ is shown to be true [5] in the generic group model.

2.3 Cryptographic Building Blocks

Signature with Efficient Protocols. A signature scheme with efficient protocols refers to a signature scheme with two protocols for the following purposes.

1. The signer only needs a commitment of a block of messages $(m_1, \ldots, m_L)$ but not the messages themselves to give a signature on $(m_1, \ldots, m_L)$;
2. A signature holder can prove the knowledge of a signature on some block of messages without revealing the signature nor the block of messages.

Examples include BBS+ signature [2] (a variant of BBS signature in [6] as outlined in [9]), and P-signature [3,4]. The latter supports non-interactive zero-knowledge proofs in the common reference string model, and the construction in [4] supports $L \geq 1$.

In a P-signature, PSigSetup setups the global parameters used by all other algorithms to be described below. A signer uses PSigKG to generate a pair of signing / verification key. Any user can use an associated commitment scheme Com to make a commitment of the message(s) to be signed and run PObtain, which interacts with the algorithm PIssue executed by the signer. As a result, the user obtains a P-signature on the message(s). If the privacy of the message(s) is not a concern, the signer can simply use the PSign algorithm. The possession of a signature can be shown using the PProve algorithm, which can then be verified by anyone using the PVer algorithm. Details can be found in [4].

For a BBS+ signature on (m_1, m_2), the global parameters contain (g, g_1, g_2, h). Using a signing key μ, the signer picks a random e and gives the signature as $\varsigma = (g g_1^{m_1} g_2^{m_2})^{1/(\mu+e)}$, which can be verified under the verification key $Z = h^{\mu}$ by checking if $\hat{e}(\varsigma, Z h^e) = \hat{e}(g g_1^{m_1} g_2^{m_2}, h)$. A computational zero-knowledge proof of signature (for a single message block) has been given in [6]. A perfect zero-knowledge proof (since the signature is not encrypted) for multiple message blocks has been given in [2]. These can be made non-interactive by using Fiat-Shamir heuristics in the random oracle model.

Pseudorandom Function, Weakly-Secure Signature and Strong One-Time Signature. We employed a variant of a pseudorandom function (PRF) due to Dodis and Yampolskiy [11], defined as $PRF_{g,s}(x) : x \mapsto g^{\frac{1}{s+x}}$ where $\mathbb{G}_p = \langle g \rangle$ is a cyclic group of prime order p, $s \in_R \mathbb{Z}_p$ is the secret seed and $x \in \mathbb{Z}_p$ is the input. We use it in our constructions for the tag-generation. Its pseudorandomness relies on the q-DDHI assumption, see [8,4] for details.

This PRF function appeared in a short signature scheme proposed by Boneh and Boyen [5]. The secret key is the seed s and the input x encodes the message. The signature is the PRF value. Verification of signature is possible if we use a bilinear group pair instead of $\mathbb{G}_p$. (On the other hand, the PRF is pseudorandom

only if the DDH is hard in $\mathbb{G}_p$.) We will use this signature in the range proof and the user signing part of our CRS-based construction. It is unforgeable under a non-adaptive chosen-message attack under the q-SDH assumption. Since it is deterministic, it is also strongly-unforgeable.

For the "anonymity against CCA attack", we use a strong one-time signature in our CRS-based construction, which informally means that the adversary can ask for the signature on a chosen message, but can neither create a different signature on that message nor forge a signature on a different message.

Non-Interactive Proofs for Bilinear Groups. Groth and Sahai [15] proposed an efficient non-interactive zero-knowledge (NIZK) or non-interactive witness-indistinguishable (NIWI) proof system for statements of the form

$$\prod_{q=1}^{Q} \hat{e}(a_q \prod_{m=1}^{M} x_m^{\alpha_{q,m}}, b_q \prod_{n=1}^{N} y_n^{\beta_{q,n}}) = t$$

where $t \in \mathbb{G}_T, \{a_q\} \subset \mathbb{G}_1, \{b_q\} \subset \mathbb{G}_2$, $\{\alpha_{q,m}\}, \{\beta_{q,n}\}, \subset \mathbb{Z}_p$, $\{x_m\} \subset \mathbb{G}_1, \{y_n\} \subset \mathbb{G}_2$ when given $\{C_m\}$ – commitments of $\{x_m\}$, $\{D_m\}$ – commitments of $\{y_n\}$, and a CRS $param_{BM}$. This is also referred to as pairing product equation.

The proof system can be instantiated by the subgroup decision assumption in composite order groups, the DLin assumption or the XDH assumption. However, the associated commitment scheme based on the first assumption is only binding over one of the prime order subgroups, which gives different PRF values for two identically distributed commitments of the same value. Hence, the e-cash scheme in [4] and our CRS-based construction employ either one of the latter two assumptions. It gives a NIZK protocol for Dodis-Yampolskiy PRF, a NIZK PoK for a tag-based encryption [17] (to be described), and a NIWI PoK for Boneh-Boyen signature in [14] It can be seen that the "structures" in all these primitives conformed to the pairing product equations.

Range Proof. Proving a secret value is within a public range can be done in this way – the verifier gives signatures on each value in the range, the prover then makes a commitment of the secret value and proves the knowledge of a signature that signs on the committed value. This idea appeared in the k-times anonymous authentication system in [20], and is used in [2,4] and our CRS-based construction. Camenisch, Chaabouni and shelat [7] gave a generalization of this approach. By writing the secret value in base-D and commit to these D-ary digits, this yields a proof of size $O(k/(\log k - \log\log k))$ instead of $O(k)$, for proving the secret lies in $[0, 2^{k+1} - 1]$.

Linear Encryption and Tag-Based Encryption. Linear encryption proposed in [6] is a natural extension of ElGamal encryption based on the decision linear assumption, which is secure even in groups where DDH problem is easy. The encryption key is $(u, v, g_0) \in \mathbb{G}_1^3$ where $u^a = v^b = g_0$, and the decryption key is (a, b). An encryption of a message $M \in \mathbb{G}_1$ is $(T_1, T_2, T_3) = (u^\alpha, v^\beta, Mg_0^{\alpha+\beta})$,

where $\alpha, \beta \in_R \mathbb{Z}_p^*$; which can be decrypted by $T_3/(T_1^a \cdot T_2^b)$. The scheme is secure against chosen-plaintext attacks (CPA) under the DLin assumption.

Kiltz [17] extended this linear encryption to a tag-based encryption which is secure against selective-tag weak chosen-ciphertext attacks (CCA), under the same assumption. The encryption key is $(u, v, g_0, U, V) \in \mathbb{G}_1^5$ where $u^a = v^b = g_0$, and the decryption key is (a, b). To encrypt a message $M \in \mathbb{G}_1$ under a tag (or a label) $t \in \mathbb{Z}_p^*$, picks $\alpha, \beta \in_R \mathbb{Z}_p^*$ and returns $(T_1, T_2, T_3, T_4, T_5) = (u^\alpha, v^\beta, Mg_0^{\alpha+\beta}, (g_0^t U)^\alpha, (g_0^t V)^\beta)$, which can be decrypted by $T_3/(T_1^a \cdot T_2^b)$ if $\hat{e}(u, T_4) = \hat{e}(T_1, g_0^t U)$ and $\hat{e}(v, T_5) = \hat{e}(T_2, g_0^t V)$ hold. The latter check can also be done without pairing if the discrete logarithm of U, V with respect to u, v respectively are kept. We will call the tag used in tag-based encryption as "label" to avoid any confusion with the tracing tag in the traceable signature.

3 Framework

3.1 Syntax

Our new model of traceable signature is based on the original framework in [16]. A traceable signature involves three kinds of entities, namely, the group manager (GM), the users (U_i) and the tracing agents (TA). It consists of nine polynomial time algorithms or protocols. The following enumerates the syntax.

- Setup. On input a security parameter 1^λ for $\lambda \in \mathbb{N}$, a trusted party executes this algorithm to output the system parameters $params$. For simplicity of the framework, we assume that Setup also outputs the group public/private key (gpk, gsk), and the opening agent public/private key pair (opk, osk). For brevity, all algorithms below take $(params, gpk, opk)$ implicitly as inputs.
- Join. A (prospective) user U_i joins the group and obtains a member public/private key pair (pk_i, sk_i) as a result of the interaction with the GM via this protocol. The GM also adds U_i's identification and part of the transcript of the protocol to the membership archive $\mathcal{DB}$, which is kept private.
- Sign. Given a message m and a member private key sk_i, user U_i uses this algorithm to give a signature σ on m on behalf of the group gpk.
- Verify. Given a signature σ and a message m, anyone can use this algorithm to verify if σ is a valid signature on m signed by a member of the group gpk.
- Reveal. On input of the member archive $\mathcal{DB}$ and a user's identification U_i, the GM outputs a trapdoor s_i for tracing the signatures produced by U_i.
- Trace. Anyone can use Trace with the trapdoor s_i generated by Reveal to output a set of tags which can uniquely identify each of the signatures of U_i.
- Open. Given a valid signature σ, the GM uses the opening secret key osk to output some information ς_i which enables the retrieval of the user's identification information U_i in the membership archive $\mathcal{DB}$.
- Claim. Given the member private key sk_i and a valid signature σ, user U_i can give an evidence z that proves the original authorship of σ.
- ClaimVer. Given a message m, a valid signature σ and an evidence z produced by Claim, anyone can verify whether σ is originated from user U_i holding sk_i.

3.2 Requirements

Definition 6. *A traceable signature scheme (of security parameter λ) is correct if the four conditions below are satisfied (with overwhelming probability in λ).*

- *Sign-Correctness. For all messages m, and all sk_i obtained from the* Join *protocol,* $\mathsf{Verify}(\mathsf{Sign}(m, sk_i), m) = \top$.
- *Open-Correctness. For all messages m, all sk_i obtained from the* Join *protocol of user U_i, and all membership archives $\mathcal{DB}$ which contain the information for user U_i,* $\mathsf{Open}(\mathsf{Sign}(m, sk_i), osk, \mathcal{DB}) = \mathrm{U}_i$.
- *Trace-Correctness. For all messages m, all sk_i obtained from the* Join *protocol of user U_i, and all membership archives $\mathcal{DB}$ which contain the information for user U_i,* $\mathsf{Sign}(m, sk_i) \subseteq \mathsf{Trace}(\mathsf{Reveal}(\mathcal{DB}, \mathrm{U}_i))$, *where σ is in the set $\mathcal{S}$ when a specific component s of σ is in the set $\mathcal{S}$.*
- *Claim-Correctness. For all messages m and all sk_i obtained from the* Join *protocol,* $\mathsf{ClaimVer}(m, \sigma, \mathsf{Claim}(sk_i, \sigma)) = \top$, *where* $\sigma = \mathsf{Sign}(m, sk_i)$.

We briefly recall the security concerns. Formal definition can be found in [16].

- *Identification Security.* Any subset of colluded users and tracing agents cannot output a valid signature which cannot be opened to anyone in this collusion group or cannot be traced (by the trapdoors produced by an honest execution of Reveal algorithm) to one of them.
- *Anonymity.* No collusion of users and tracing agents can distinguish between the signatures of two honest group members. (Note that the tracing agents are not given the user-specific trapdoor of these two honest members.)
- *Non-Frameability.* There are two different ways an honest user may be framed. A conspiration of the GM and any subset of colluded users may construct a signature that opens or trace to an innocent user outside this group, or may claim a signature that was generated by an honest user as their own.

Due to the new traceability feature we introduce, our schemes can only be secure against a weaker variant of anonymity attack described below.

- *N-Anonymity.* Same as Anonymity Attack, but the adversary can only see at most N (determined in Setup) signatures from each of the honest members.

Remarks. Since Open is considered as an internal operation (which is different from Trace), the adversary in the first two attacks are not allowed to query an "open" oracle. Nevertheless, our CRS-based construction achieves "CCA" anonymity, i.e. the adversary has an "open" oracle in breaking anonymity, under the natural constraint that it cannot be queried with the challenge signature.

Given the deterministic nature of the tracing tag, the GM (or a tracing agent who "colludes" with the GM) may launch a misidentification or framing attack by giving a signature with a "legitimate" tracing tag [1]. However, a user can dispute if the self-claiming component is deterministically determined by the tracing tag and part of the membership private key which is only known to the

user. Specifically, the existence of two valid signatures with exactly the same tracing tag but different self-claiming components means that the GM "reused" the same seed in issuing "different" membership credentials.

4 Constructions

We first give our construction in the common reference string model. This can be seen as a concrete realization of the design we gave in Section 2. Then we will present a more efficient construction in the random oracle model.

4.1 Construction in the Common Reference String Model

This somewhat generic and moderately efficient construction is mostly based on the building blocks we presented in Section 2.3, except we have instantiated the signature in the range proof by Boneh-Boyen signature and the PRF by Dodis-Yampolskiy PRF. It is largely based on the compact e-cash scheme in [4]. We added a tag-based encryption [17] of the user's identity and the user self-claiming component, but removed the double-spending detection.

Setup. This algorithm setups all the building blocks. Namely, it runs $\mathsf{PSigSetup}$ and returns the P-signature parameters $params$, $\mathsf{PSigSetup}$ needs to run the setup of the Groth-Sahai proof system to get its parameters $params_{GS}$, which in turn contain the bilinear map context $params_{BM} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g, h)$. Let H be a collision-free hash function which maps to $\mathbb{Z}_p^*$. All these should be determined by the CRS, or executed by a trusted initializer.

For credential issuing, runs $\mathsf{PSigSetup}$ to generate a key pair (gpk, gsk).

For opening, setups a key pair (opk, osk) by tag-based encryption's $\mathsf{TEncSetup}$.

For tracing, manages a list of triple $(\mathrm{U}_i, pk_i, s_i)$, which is initially empty. For the range proof system, the GM picks a number N which is polynomial in λ, runs $\mathsf{PSigSetup}(params)$ again to generate another pair of signing key (pk_r, sk_r), generates and publishes the signatures $\Sigma_i = \mathsf{PSign}(i, sk_r), \forall i \in [1, 2, \cdots, N]$.

Join. User U_i obtains a credential from the GM through the interactions below.

1. User U_i randomly selects $x_i \in_R \mathbb{Z}_p^*$, computes a public key $pk_i = g^{x_i}$, and a commitment $comm_{sk} = \mathsf{Com}(x_i, open_{x_i})$. U_i sends $comm_{sk}$ to the GM, proves in zero-knowledge the knowledge of $open_{x_i}$, and that $comm_{sk}$ corresponds to the secret key used for computing pk_i.
2. The GM verifies the proof, randomly selects $s_i \in_R \mathbb{Z}_p^*$, computes $comm_{seed} = \mathsf{Com}(s_i, open_{s_i})$ and sends $(s_i, open_{s_i})$ to the user. The tracing trapdoor for this user will be s_i and the GM should ensure it is unique.
3. The user and the GM run the algorithms $\mathsf{PObtain}(gpk, (x_i, s_i), (open_{x_i}, open_{s_i}))$ $\leftrightarrow$ $\mathsf{PIssue}(gsk, (comm_{sk}, comm_{s_i}))$ respectively. The user obtains a P-signature ς_i on (x_i, s_i), and stores (ς_i, x_i, s_i) as the member private key.
4. The GM adds the entry $(\mathrm{U}_i, pk_i, s_i)$ to the membership archive.

Sign(m). User U_i manages a counter n_i on the number of signatures produced.

- U_i generates a one-time signature key pair (pk_o, sk_o).
- U_i signs on pk_o by $\sigma = g^{\frac{1}{x_i + H(pk_o)}}$.
- U_i computes the tracing tag $S = g^{\frac{1}{s_i + n_i}}$ and the self-claiming tag $R = S^{x_i}$.
- U_i encrypts pk_i by computing $\mathfrak{C} = \mathsf{TEnc}_{opk}(pk_i, H(pk_o))$, where $H(pk_o)$ serves as the label of the encryption.
- U_i proves in non-interactive zero-knowledge manner the relations (1) - (6):
  1. U_i is in possession of a P-signature ς_i from the GM on (x_i, s_i).
  2. U_i generated a commitment C_{sig} of $\sigma = g^{\frac{1}{x_i + H(pk_o)}}$, a signature on pk_o.
  3. $\mathfrak{C}$ is a tag-based encryption of pk_i with the label $H(pk_o)$.
  4. S is $PRF_{g,s_i}(n_i)$, that is, $S = g^{\frac{1}{s_i + n_i}}$.
  5. $R = S^{x_i}$.
  6. $0 \leq n_i < N$, i.e. U_i is in possession of a P-signature ς_i under pk_r on n_i.
- U_i uses sk_o to give a signature σ_{ots} on m concatenated with the above proofs.

All these proofs need to be done non-interactively by Groth-Sahai proof system or non-interactive P-signature (which utilizes the former). Specifically, U_i

1. runs PProve on ς_i and gpk to obtain commitments and proof $(C_{pk}, C_{seed}, \pi_1) \leftarrow \mathsf{PProve}(params, gpk, \varsigma_i, (x_i, s_i))$ for secret key x_i and seed s_i respectively.
2. runs PProve on Σ_{n_i} and pk_r to obtain commitment and proof for counter n_i, i.e. $(C_{ctr}, \pi_2) \leftarrow \mathsf{PProve}(params, pk_r, \Sigma_{n_i}, (n_i))$.
3. uses the Groth-Sahai proof system to construct proofs showing that the values $(R, S, \mathfrak{C}, C_{pk}, C_{seed}, C_{ctr}, C_{sig})$ are indeed well formed. This involves the proofs $\pi_S, \pi_R, \pi_O, \pi_C$ of the following languages:
   - $\mathcal{L}_S = \{C_s, C_n, y | \exists n, s, open_n, open_s$ such that $C_s = \mathsf{Com}(s, open_s) \wedge C_n = \mathsf{Com}(n, open_n) \wedge y = PRF_{g,s}(n)\}$
   - $\mathcal{L}_R = \{C_x, Y, y | \exists x, open_x$ such that $C_x = \mathsf{Com}(s, open_x) \wedge Y = y^x\}$
   - $\mathcal{L}_O = \{C_x, C_\sigma, pk_o | \exists x, open_x, open_\sigma$ such that $C_x = \mathsf{Com}(x, open_x) \wedge C_\sigma = \mathsf{Com}(\sigma, open_\sigma) \wedge \sigma = g^{\frac{1}{x + H(pk_o)}}\}$
   - $\mathcal{L}_C = \{C_x, \mathfrak{C} | \exists x, open_x$ such that $C_x = \mathsf{Com}(x, open_x) \wedge pk = g^x \wedge \mathfrak{C} = \mathsf{TEnc}_{opk}(pk)\}$

The signature is $(R, S, T, \mathfrak{C}, pk_o, \sigma_{ots}, C_{pk}, C_{seed}, C_{ctr}, C_{sig}, \pi_1, \pi_2, \pi_S, \pi_R, \pi_O, \pi_C)$. $\mathcal{L}_R$ is relatively simple. $\mathcal{L}_S$ can be found in [4, Section 4.2]. $\mathcal{L}_O, \mathcal{L}_C$ are somewhat simplified variants of the proofs in [14, Section 7].

Verify. To verify a signature, returns true if all of the following checks succeed:

1. $\mathsf{PVer}(params, gpk, \pi_1, (C_{pk}, C_{seed})) = \top$.
2. $\mathsf{PVer}(params, pk_r, \pi_2, C_{ctr}) = \top$.
3. σ_{ots} is a valid signature on $(m || \pi_1 || \pi_2 || \pi_S || \pi_R || \pi_O || \pi_C)$ under pk_o.
4. π_S is a valid proof on (C_{seed}, C_{ctr}, S).
5. π_R is a valid proof on (C_{pk}, R, S).
6. π_O is a valid proof on (C_{pk}, C_{sig}, pk_o).
7. π_C is a valid proof on $(C_{pk}, \mathfrak{C})$.

Open. Given a valid signature $(\cdots, \mathfrak{C}, pk_o, \cdots)$, anyone (the GM, or an opening agent) who holds the decryption key osk recovers $pk' = \mathsf{TDec}(\mathfrak{C}, H(pk_o))$. From the membership archive $\{(\mathrm{U}_i, pk', s_i)\}$, the GM outputs the corresponding U_i.

Reveal. From the membership archive, the GM retrieves s_i of the i^{th} user.

Trace. Given s_i, the TA computes $\{S_j = g^{\frac{1}{s_i+j}}\}_{0 \leq j < N}$. If a given signature has the S component inside this set, the TA concludes that user i is its originator.

Claim. U_i who gave the signature $\sigma = (R, S, \cdots)$ generates a non-interactive proof of knowledge π_R of the value x_i such that $R = S^{x_i}$ as a proof of authorship.

ClaimVer. Verify the proof π_R given by Claim.

4.2 Construction in the Random Oracle Model

Our second scheme assumes random oracle and employs CPA linear encryption instead of weak CCA tag-based encryption for better efficiency. The design is similar to the traditional-style traceable signature in [10] which is extended from [6]. However, we have moved the component for tracing component (which also helps in self-claiming) from $\mathbb{G}_T$ to $\mathbb{G}_p$. Since $\mathbb{G}_T$ is usually a subgroup of $\mathbb{Z}_{q^\alpha}$, it is vulnerable to sub-exponential discrete logarithm attacks and needs very large representation. For example, for 128-bit security, $|\mathbb{G}_T| \geq 3072$ bits. This can also been as a variant of [2], with a verifiable encryption and the self-claiming component added and double-spending detection removed.

Our scheme relies on the DLin assumption in $\mathbb{G}_1$, the q-SDH assumption in $(\mathbb{G}_1, \mathbb{G}_2)$, and the q-DDHI assumption in $\mathbb{G}_p$. We describe the scheme in $(\mathbb{G}_1, \mathbb{G}_2)$ but these two groups can be the same (since we instantiate the PRF in another DDH-hard group $\mathbb{G}_p$). Our scheme does not rely on the XDH assumption, this gives more flexibility in the choices of the underlying elliptic curve. If we are willing to make the XDH assumption, the signature can be made shorter since the linear encryption can be replaced by ElGamal encryption [6].

Setup. Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair with computable isomorphism ψ as discussed such that $\mathbb{G}_1 = \langle g \rangle$, $\mathbb{G}_2 = \langle h \rangle$ and $|g| = |h| = p$ for some prime p of λ bits. Let $\mathbb{G}_p = \langle f \rangle$ be a cyclic group of order p where the DDH assumption holds. Let $\mathfrak{g}_1$, g_1, $\mathfrak{g}_2$, g_2 be random elements in $\mathbb{G}_1$, which are for the zero-knowledge PoK protocols. Let H be a collision-free hash function which maps to $\mathbb{Z}_p^*$.

For credential issuing, the GM randomly selects $\mu \in_R \mathbb{Z}_p^*$ and computes $Z = h^\mu$ as the public key gpk of the group. The GM keeps $gsk = \mu$ in secret.

For opening, the GM setups an encryption key pair by picking $g_0 \in_R \mathbb{G}_1$, $osk = (a, b) \in_R (\mathbb{Z}_p^*)^2$. The public key is $opk = (u, v, g_0)$ where $u^a = v^b = g_0$.

For tracing, the GM manages a list of triple $(\mathrm{U}_i, s_i, \varsigma_i)$, which is initially empty; and picks a number N which is polynomial in λ. The GM also setup a range proof system, e.g. [7] to be discussed in Appendix A.

Join. User U_i obtains a credential from the GM through the interaction below.

1. U_i selects $x_i \in_R \mathbb{Z}_p^*$, sends $y_i = g_2^{x_i}$ to the GM with the proof $\mathsf{SPK}_0\{(x_i) : y_i = g_2^{x_i}\}$. This can be done non-interactively by Schnorr signature.
2. The GM verifies the proof, picks $s_i, e_i \in_R \mathbb{Z}_p^*$, computes $\varsigma_i = (g g_1^{s_i} y_i)^{\frac{1}{\mu + e_i}}$ and sends (ς_i, e_i, s_i) to the user. The GM also stores the triple $(\mathrm{U}_i, s_i, \varsigma_i)$. The tracing trapdoor for U_i will be s_i and the GM should ensure it is unique.
3. U_i checks if $\hat{e}(\varsigma_i, Z h^{e_i}) = \hat{e}(g g_1^{s_i} g_2^{x_i}, h)$. The member public key and secret key are (y_i, e_i) and (x_i, ς_i, s_i) respectively.

Sign(m). User U_i manages a counter n_i on the number of signatures produced.

- U_i computes the tracing tag $S = f^{\frac{1}{s_i + n_i}}$ and the self-claiming tag $R = S^{x_i}$.
- U_i encrypts ς_i in $(T_1 = u^\alpha, T_2 = v^\beta, T_3 = \varsigma_i g_0^{\alpha+\beta})$ where $\alpha, \beta \in_R \mathbb{Z}_p^*$.
- U_i proves in non-interactive zero-knowledge manner such that (T_1, T_2, T_3) is a linear encryption of ς_i, where (ς_i, e_i) is a BBS + signature from the GM on (s_i, x_i), $S = f^{\frac{1}{s_i + n_i}}$, $R = S^{x_i}$ and $0 \le n_i < N$. This can be abstracted as

$$\mathsf{SPK}_1\Big\{ \big(\varsigma_i, e_i, s_i, x_i, n_i, \alpha, \beta\big) : \begin{array}{rcll} \hat{e}(\varsigma_i, Z h^{e_i}) & = & \hat{e}(g g_1^{s_i} g_2^{x_i}, h) & \wedge \\ (T_1, T_2, T_3) & = & (u^\alpha, v^\beta, \varsigma_i g_0^{\alpha+\beta}) & \wedge \\ (S, R) & = & (f^{\frac{1}{s_i+n_i}}, f^{\frac{x_i}{s_i+n_i}}) & \wedge \\ 0 \le & n_i & < N & \end{array} \Big\}(m)$$

To conduct SPK_1, U_i computes $\mathfrak{A}_1 = \mathfrak{g}_1^{e_i} \mathfrak{g}_2^{\rho_1}$, $\mathfrak{A}_2 = \mathfrak{g}_1^{x_i} \mathfrak{g}_2^{\rho_2}$, $\mathfrak{A}_3 = \mathfrak{g}_1^{n_i} \mathfrak{g}_2^{\rho_3}$ for $\rho_1, \rho_2, \rho_3 \in_R \mathbb{Z}_p^*$. Next, U_i computes the following two SPK's.

$$\mathsf{SPK}_{1A}\Big\{ \big(e_i, s_i, x_i, n_i, \alpha, \beta, \rho_1, \rho_2, \rho_3, \gamma_1, \gamma_2\big) : \begin{array}{rcl} \mathfrak{A}_1 = \mathfrak{g}_1^{e_i} \mathfrak{g}_2^{\rho_1} \wedge \mathfrak{A}_2 = \mathfrak{g}_1^{x_i} \mathfrak{g}_2^{\rho_2} \wedge \mathfrak{A}_3 = \mathfrak{g}_1^{n_i} \mathfrak{g}_2^{\rho_3} & & \wedge \\ T_1 = u^\alpha \quad \wedge & T_2 = v^\beta & \wedge \\ T_1^{e_i} = u^{\gamma_1} \quad \wedge & T_2^{e_i} = v^{\gamma_2} & \wedge \\ f = S^{s_i} S^{n_i} \quad \wedge & R = S^{x_i} & \wedge \\ \hat{e}(T_3, Z h^{e_i}) = & \hat{e}(g g_1^{s_i} g_2^{x_i}, h) \hat{e}(g_0, Z^{\alpha+\beta} h^{\gamma_1+\gamma_2}) & \end{array} \Big\}(m)$$

$$\mathsf{SPK}_{1B}\Big\{ (n_i, \rho_3) : \mathfrak{A}_3 = \mathfrak{g}_1^{n_i} \mathfrak{g}_2^{\rho_3} \wedge 0 \le n_i < N \Big\}(m)$$

We show how to instantiate SPK_{1A} below and SPK_{1B} in Appendix A.

(Commitment.) U_i picks r_{e_i}, r_{s_i}, r_{x_i}, r_{n_i}, r_α, r_β, r_{ρ_1}, r_{ρ_2}, r_{ρ_3}, r_{γ_1}, $r_{\gamma_2} \in_R \mathbb{Z}_p^*$ and computes
$\mathfrak{T}_1 = \mathfrak{g}_1^{r_{e_i}} \mathfrak{g}_2^{r_{\rho_1}}$, $\mathfrak{T}_2 = \mathfrak{g}_1^{r_{x_i}} \mathfrak{g}_2^{r_{\rho_2}}$, $\mathfrak{T}_3 = \mathfrak{g}_1^{r_{n_i}} \mathfrak{g}_2^{r_{\rho_3}}$, $\mathfrak{T}_4 = u^{r_\alpha}$, $\mathfrak{T}_5 = v^{r_\beta}$ in $\mathbb{G}_1$,
$\mathfrak{T}_6 = \hat{e}(T_3, h)^{r_{e_i}} \hat{e}(g_1, h)^{-r_{s_i}} \hat{e}(g_2, h)^{-r_{x_i}} \hat{e}(g_0, Z)^{-r_\alpha - r_\beta} \hat{e}(g_0, h)^{-r_{\gamma_1} - r_{\gamma_2}}$ in $\mathbb{G}_T$,
$\mathfrak{T}_7 = T_1^{r_{e_i}} u^{-r_{\delta_3}}$, $\mathfrak{T}_8 = T_2^{r_{e_i}} v^{-r_{\delta_4}}$ in $\mathbb{G}_1$, $\mathfrak{T}_9 = S^{r_{s_i}} S^{r_{n_i}}$, $\mathfrak{T}_{10} = S^{r_{x_i}}$ in $\mathbb{G}_p$.
(Challenge and Response) Let $\mathfrak{T} = (\mathfrak{T}_1 || \ldots || \mathfrak{T}_{10})$. For a challenge $c = H(m||R||S||T_1||T_2||T_3||\mathfrak{T})$, U_i computes, in $\mathbb{Z}_p$, $z_{e_i} = r_{e_i} - c e_i$, $z_{s_i} = r_{s_i} - c s_i$, $z_{x_i} = r_{x_i} - c x_i$, $z_{n_i} = r_{n_i} - c n_i$, $z_{\rho_1} = r_{\rho_1} - c\rho_1$, $z_{\rho_2} = r_{\rho_2} - c\rho_2$, $z_{\rho_3} = r_{\rho_3} - c\rho_3$, $z_{\gamma_1} = r_{\gamma_1} - c\alpha e_i$, $z_{\gamma_2} = r_{\gamma_2} - c\beta e_i$, U_i sets $\mathfrak{z} = (z_{e_i}, z_{s_i}, z_{x_i}, z_{n_i}, z_{\rho_1}, z_{\rho_2}, z_{\rho_3}, z_{\gamma_1}, z_{\gamma_2})$.

Verify. To verify a signature $(S, T_1, T_2, T_3, \pi_{SPK})$, where π_{SPK} denotes the commitments (e.g. $\mathfrak{A}_1$, $\mathfrak{A}_2$, $\mathfrak{A}_3$, $\mathfrak{T}$) and the responses (e.g. $\mathfrak{z}$) generated by the above PoK protocol, this algorithm executes the verification algorithm of SPK_1.

In particular, SPK_{1A} can be verified by first computing

$$\mathfrak{T}'_1 = \mathfrak{A}_1^c \mathfrak{g}_1^{z_{e_i}} \mathfrak{g}_2^{z_{\rho_1}} \qquad \mathfrak{T}'_2 = \mathfrak{A}_2^c \mathfrak{g}_1^{z_{x_i}} \mathfrak{g}_2^{z_{\rho_2}} \qquad \mathfrak{T}'_3 = \mathfrak{A}_3^c \mathfrak{g}_1^{z_{n_i}} \mathfrak{g}_2^{z_{\rho_3}}$$
$$\mathfrak{T}'_4 = T_1^c \mathfrak{T}_4 \qquad \mathfrak{T}'_5 = T_2^c \mathfrak{T}_5$$
$$\mathfrak{T}'_7 = T_1^{z_{e_i}} u^{-z_{\gamma_1}} \qquad \mathfrak{T}'_8 = T_2^{z_{e_i}} v^{-z_{\gamma_2}}$$
$$\mathfrak{T}'_9 = f^c S^{z_{s_i}} S^{z_{n_i}} \qquad \mathfrak{T}'_{10} = R^c S^{z_{x_i}}$$

$$\mathfrak{T}'_6 = \hat{e}(T_3, h^{z_{e_i}}/Z^c)\hat{e}(g^c g_1^{-z_{s_i}} g_2^{-z_{x_i}}, h)\hat{e}(g_0, Z^{-(z_\alpha + z_\beta)} h^{-(z_{\gamma_1} + z_{\gamma_2})})$$

and checking if $c \stackrel{?}{=} H(m||R||S||T_1||T_2||T_3||\mathfrak{T}'_1|| \ldots ||\mathfrak{T}'_{10})$. If the equation holds, (R, S) and (T_1, T_2, T_3) are well-formed, thus outputs $\top$, $\perp$ otherwise.

Open. Given a valid signature $(m, S, T_1, T_2, T_3, \pi_{SPK})$, anyone who holds the decryption key (a, b) computes $\varsigma' = T_3/(T_1^a \cdot T_2^b)$. From the membership archive, the GM outputs U_i of the entry with the ς_i component matches with ς'.

Reveal. From the membership archive, the GM retrieves s_i of the i^{th} user.

Trace. Given s_i, the TA computes $\{S_j = f^{\frac{1}{s_i + j}}\}_{0 \leq j < N}$. If a given signature has the S component inside this set, the TA concludes that user i is its originator[1].

Claim. U_i who gave the signature $\sigma = (R, S, \cdots)$ can provide a proof of authorship π by generating a proof of knowledge π of the value x_i such that $R = S^{x_i}$.

ClaimVer. Verify the proof π given by Claim.

4.3 Security Analysis

Since both of our constructions are based on the same design, below we give an overall picture of the security analysis. The security of both schemes is hinged upon DLin, q-DDHI and q-SDH assumptions. Our RO-based construction relies on the DDH assumption, while our CRS-based one relies on the XDH assumption and any others required for the security of the multi-block P-signature [4]. Details for the RO-based construction can be found in [2,6,10] and those for the CRS-based construction follow from [4,14].

– *Identification Security.* Misidentification is an attack by a subset of colluded users and TA's. To model the former requires the signing oracle of the underlying signature. For our CRS-based construction, the underlying signing protocol (the credential issuing protocol in our case) of the P-signature has

[1] The TA may recover f^{x_i} from (R, S), but it does not help in forging a signature. Nevertheless, the GM can simply send $\{S_j = f^{\frac{1}{s_i + j}}\}_{0 \leq j < N}$ to the TA to avoid this.

guaranteed that computing these signatures in such an interactive manner reveals nothing else about the secret key (by the fact that the protocol can be simulated by blackbox access of the signing oracle). For the TA's, they are just *given* the seeds (but they cannot choose it), which can be easily simulated. In fact, the seeds are just part of the messages to be signed by the underlying signature scheme, and their secrecy is not relevant here.

There are two attacks goals in misidentification attack. The first one is to output a valid signature which cannot be opened to anyone outside this collusion group. With the soundness of the ZK proof for the encryption, this translates to giving an encryption of a credential which the GM never issues. If this happens, the simulator $\mathcal{S}$ can extract this credential and returns it as the forgery of the signature scheme used for credential issuing.

The second attack goal is to produce a tag that cannot be computed by the Trace algorithm. There are two possibilities. The adversary $\mathcal{A}$ used an entirely new seed that is never "certified" by the GM, or $\mathcal{A}$ used a seed from the GM but produced something that cannot be produced by the Reveal algorithm. For the first case, $\mathcal{S}$ can decrypt the ciphertext and obtain a forgery of the underlying signature scheme. The second case will break either the soundness of the ZK proof for the PRF or that of the range proof.

– *N-Anonymity.* The compromised parties controlled by an adversary $\mathcal{A}$ is the same as those in misidentification attack. N-anonymity goes by a series of transformation such that a signature produced by an honest member is eventually transformed to one produced by another honest member, and argue that each transformation is computationally indistinguishable to $\mathcal{A}$.

  Firstly, we change the parameter for the ZK proofs to a simulated one such that the commitments are perfect hiding and the ZK proofs involved in a signature will be "faked" by a simulation instead of giving a real proof. The adversary cannot notice this change by the security of the ZK proofs. Now the proofs are faked, we can change the elements that can differentiate one honest user from another in the signatures. We replace the tracing tag S with random element, then replace it with the tracing tag of another user. As long as these honest users produced less than N signatures, the indistinguishability of these changes are guaranteed by the pseudorandomness of the PRF. We then change the self-claiming tag R, by the DDH assumption (in $\mathbb{G}_p$ or in $\mathbb{G}_1$ – XDH). Finally, we encrypt a random element instead of the signature, by the indistinguishability of the encryption.

– *Non-Frameability.* The group member gives a signature based on the secret key x_i. The simulator $\mathcal{S}$ does not know x_i, but it gives out many signatures by the simulators of the underlying signature scheme and the ZK proofs (and manipulating the random oracle in our RO-based scheme). Eventually, the adversary $\mathcal{A}$ produces a valid forgery. In our CRS-based construction, $\mathcal{S}$ extracts the knowledge of σ by the extractability of the underlying proof system, which is a solution of the q-SDH problem as long as $H(pk_o)$ does not match with those public keys of a one-time signature scheme pre-selected by the simulator in the simulation of the signing oracle. In our RO-based construction, $\mathcal{S}$ can use the standard rewinding technique to extract x_i.

5 Conclusion

We found that the original idea of tracing signatures is nice but may not fully solve the problems of the group signature as expected. We propose a new and efficient way of tracing by borrowing the idea from compact e-cash. Our notion gives an alternative when timely tracing is important and when the signatures are scattered around in an anonymous system.

References

1. Au, M.H.: Personal communication (2009)
2. Au, M.H., Susilo, W., Mu, Y.: Constant-Size Dynamic k-TAA. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer, Heidelberg (2006)
3. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and Noninteractive Anonymous Credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
4. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: Compact E-Cash and Simulatable VRFs Revisited. In: Boyen, X., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 114–131. Springer, Heidelberg (2009)
5. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. J. Cryptology 21(2), 149–177 (2008)
6. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin [12], pp. 41–55
7. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient Protocols for Set Membership and Range Proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (2008)
8. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact E-Cash. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
9. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin [12], pp. 56–72
10. Choi, S.G., Park, K., Yung, M.: Short Traceable Signatures Based on Bilinear Pairings. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S.-i. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 88–103. Springer, Heidelberg (2006)
11. Dodis, Y., Yampolskiy, A.: A Verifiable Random Function with Short Proofs and Keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
12. Franklin, M. (ed.): CRYPTO 2004. LNCS, vol. 3152. Springer, Heidelberg (2004)
13. Ge, H., Tate, S.R.: Traceable Signature: Better Efficiency and Beyond. In: Gavrilova, M., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganà, A., Mun, Y., Choo, H. (eds.) ICCSA 2006. LNCS, vol. 3982, pp. 327–337. Springer, Heidelberg (2006)
14. Groth, J.: Fully Anonymous Group Signatures Without Random Oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
15. Groth, J., Sahai, A.: Efficient Non-interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)

16. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable Signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Heidelberg (2004)
17. Kiltz, E.: Chosen-Ciphertext Security from Tag-Based Encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
18. Nguyen, L., Safavi-Naini, R.: Efficient and Provably Secure Trapdoor-Free Group Signature Schemes from Bilinear Pairings. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 372–386. Springer, Heidelberg (2004)
19. IEEE P1556 Working Group. VSC Project. Dedicated short range communications, DSRC (2003)
20. Teranishi, I., Sako, K.: k-Times Anonymous Authentication with a Constant Proving Cost. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 525–542. Springer, Heidelberg (2006)

A Signature-Based Range Proof

To implement a range proof system with D signatures [7], the GM setups a signing key $Z' = h^{\nu}$ and gives D signatures $\{\sigma_i = g^{1/\nu+i}\}$ for each $i \in \mathbb{Z}_D$.

To conduct SPK_{1B} in Section 4.2, i.e. to prove a secret value $t = n_i$ lies in $[0, N = D^{\ell})$, the prover U_i writes t in base u where $t = \sum_j (t_j D^j)$ to obtain ℓ elements $\{t_j\}$, then picks τ_j, computes $\mathfrak{V}_j = \sigma_{t_j}^{\tau_j}$ for $j \in \mathbb{Z}_\ell$. Finally, U_i computes:

$$\mathsf{SPK}_{1C}\Big\{ (\{t_j\}, \{\tau_j\}, \rho) : \mathfrak{A} = (\textstyle\prod_j \mathfrak{g}_1^{D_j})^{t_j} \mathfrak{g}_2^{\rho} \wedge (\wedge_j \mathfrak{V}_j = \sigma_j^{\tau_j}) \Big\}(m)$$

which can be instantiated by

(Commitment.) U_i picks $r_{t_1}, \ldots, r_{t_\ell}, r_{\tau_1}, \ldots, r_{\tau_\ell}, r_\rho \in_R \mathbb{Z}_p^*$ and computes $\{\mathfrak{U}_j = \hat{e}(\sigma_j, h)^{-r_{t_j}} \hat{e}(g,h)^{r_{\tau_j}}\}$ and $\mathfrak{U}^* = \prod_j (\mathfrak{g}_1^{D_j r_{t_j}}) \mathfrak{g}_2^{r_\rho}$.

(Challenge and Response) Let $\mathfrak{U} = (\mathfrak{U}_1 || \ldots || \mathfrak{U}_\ell || \mathfrak{U}^*)$. For a challenge $c = H(m||\mathfrak{U})$, U_i computes, in $\mathbb{Z}_p$, $z_{t_1} = r_{t_1} - ct_j, \ldots, z_{t_\ell} = r_{t_\ell} - ct_\ell$, $z_{\tau_1} = r_{\tau_1} - c\tau_j, \ldots, z_{\tau_\ell} = r_{\tau_\ell} - c\tau_\ell$, $z_\rho = r_\rho - c\rho$ and U_i sets $\mathfrak{z} = (z_{t_1}, \ldots, z_{t_\ell}, z_{\tau_1}, \ldots, z_{\tau_\ell}, z_\rho)$.

To verify, compute $\mathfrak{U}'_j = \hat{e}(\mathfrak{V}_j, Z'^c h^{-z_{t_j}}) \hat{e}(g,h)^{z_{\tau_j}}, \forall j \in [1, 2, \cdots, \ell]$, $\mathfrak{U}'' = \mathfrak{A}^c \prod_j (\mathfrak{g}_1^{D_j z_{t_j}}) \mathfrak{g}_2^{z_\rho}$ and check if $c \stackrel{?}{=} H(m||\mathfrak{U}')$ where $\mathfrak{U}' = (\mathfrak{U}'_1 || \ldots || \mathfrak{U}'_\ell || \mathfrak{U}'')$.