k-Times Full Traceable Ring Signature
Xavier Bultel, Pascal Lafourcade

To cite this version:
Xavier Bultel, Pascal Lafourcade. k-Times Full Traceable Ring Signature. 11th International Conference on Availability, Reliability and Security (ARES 2016), Aug 2016, Salzburg, France. pp.39–48, 10.1109/ARES.2016.37. hal-01691940

HAL Id: hal-01691940
https://hal.science/hal-01691940
Submitted on 24 Jan 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
$k$-times Full Traceable Ring Signature∗

Xavier Bultel
Université Clermont Auvergne
LIMOS, BP 10448, 63000 Clermont-Ferrand, France
Email: xavier.bultel@udamail.fr

Pascal Lafourcade
Université Clermont Auvergne
LIMOS, BP 10448, 63000 Clermont-Ferrand, France
Email: pascal.lafourcade@udamail.fr

Abstract—Ring and group signatures allow their members to anonymously sign documents in the name of the group. In ring signatures, members manage the group themselves in an ad-hoc manner while in group signatures, a manager is required. Moreover, $k$-times traceable group and ring signatures [1] allow anyone to publicly trace two signatures from a same user if he exceeds the a priori authorized number of signatures. In [2], Canard et al. give a 1-time traceable ring signature where each member can only generate one anonymous signature. Hence, it is possible to trace any two signatures from the same user. Some other works generalize it to the $k$-times case, but the traceability only concerns two signatures. In this paper, we define the notion of $k$-times full traceable ring signature ($k$-FTRS) such that all signatures produced by the same user are traceable if and only if he produces more than $k$ signatures. We construct a $k$-FTRS called Ktrace. We extend existing formal security models of $k$-times linkable signatures to prove the security of Ktrace in the random oracle model. Our primitive $k$-FTRS can be used to construct a $k$-times veto scheme or a proxy e-voting scheme that prevents denial-of-service caused by cheating users.

I. INTRODUCTION

Organization of elections is an old concern of Humanity. Therefore many kinds of elections have been invented. With the development of Internet and the progresses in cryptography, several e-voting schemes have been developed, see [3] for a survey. In all these systems the question of proxy voting exists independently of the form of the ballots and the counting phase. The problem is how a voting system can allow someone who cannot participate to the election to delegate his vote to someone else. In some elections, with the agreement of the voter, another voter can vote twice or more. Each voter can have different numbers of proxy-votes from 0 up to a limit fixed by the election rules. In traditional paper elections, using paper ballots and a transparent box, a cheatner who voted with an extra false proxy vote, is a posteriori detected during the tally phase, hence the election is canceled. An electronic voting system should offer a proxy mechanism that is better than in a traditional system. Once a fraud is detected then the cheater should be identified in order to blame him. Moreover, anyone should be able to find all the ballots that he has introduced in the system in order to remove them from the final count. Such a mechanism would prevent having to reorganize the election.

Another close problem is the elaboration of the program committee of a conference, that is often done by the members of the steering committee. Usually each member of the steering committee anonymously proposes a list of names for the program committee. Some members might want to discard some names for personal reasons, but they do not want other members of the steering committee to learn their choices. Our aim is to construct a system that allows each member to anonymously express a maximum of $k$ vetos. If one person gives more than $k$ vetos then we want to be able to detect it and to learn all vetos from the cheater in order to remove them. Using existing signature schemes such as [1], it is possible to design such a veto system and, in case someone uses more than his number of vetos, the cheater is identified. Unfortunately the cheater’s vetos remain anonymous, hence it is not possible to discard them. While with our ring signature scheme, all cheater’s vetos can be discarded without restarting the elaboration of the program committee.

For these two applications, we design a ring signature that offers the following security properties:

- A new signature cannot be forged without knowing a secret signature key.
- Each group signer $i$ can have his own number of authorized signatures $k_i$ which is determined when the public key is set up.
- If a signer $i$ signs a number of messages lower or equal to his authorized number of signatures $k_i$, it is not possible to determine which member of the group has signed the message. Moreover, no one can determine if two signatures have been generated with the same secret signature key (anonymity).
- If a signer $i$ signs more than his authorized number of signatures $k_i$ then anybody can link all his signatures (linkability) and also determine the identity of the cheater (traceability).
- Linkability and traceability properties exist only if the signatures are generated for the same event: for instance the first and the second round of an election are two different events.

In this paper, we propose a $k$-times full traceable ring signature that has all these features.

a) Contributions: We define a cryptographic primitive called $k$-times full traceable ring signature. This primitive allows everyone to publicly trace all signatures of a group member who has produced more than his personal threshold number of signatures $k$ during a specified event. Existing results in the literature only allow to link or open two of these signatures produced by the same user are traceable if and only if he produces more than $k$ signatures. Some other works generalize it to the $k$-times case, but the traceability only concerns two signatures. In this paper, we define the notion of $k$-times full traceable ring signature ($k$-FTRS) such that all signatures produced by the same user are traceable if and only if he produces more than $k$ signatures. We construct a $k$-FTRS called Ktrace. We extend existing formal security models of $k$-times linkable signatures to prove the security of Ktrace in the random oracle model. Our primitive $k$-FTRS can be used to construct a $k$-times veto scheme or a proxy e-voting scheme that prevents denial-of-service caused by cheating users.

∗ This research was conducted with the support of the “Digital Trust” Chair from the University of Auvergne Foundation.
signatures. We give a formal definition of our primitive and its security properties. More precisely, we have full anonymity of the signer if he does not over pass his threshold value. Moreover the ring signature verification key is only composed of the public key of each member, so it is easy to construct by an ad-hoc group of users with different thresholds for each user. Finally, the produced signatures are linear in the group size times the maximum of thresholds. We also give proofs of this scheme in the random oracle model (ROM).

Our \( k \)-times full traceable ring signature primitive can be used to design an anonymous veto mechanism for the steering committee where vetos are anonymous signatures on program committe names that are not desired. This mechanism can detect members that use more vetos than expected and remove cheater’s vetos while preserving the anonymity of other users. It can also be used to organize elections with multiple proxy votes such that when a user signs more ballots that he should, everybody can open all his signatures.

b) Related Works: Group and ring signatures are two well known cryptographic primitives that first appeared respectively in [4] and [5]. Both primitives allow users to anonymously sign a digital document within a group. In group signatures, a special authority manages the group using a manager secret key, while in ring signatures, members manage the group themselves in an ad-hoc manner. In [6], the authors present a scheme where the size of a signature is constant, i.e. it does not depend to the group size. Some group/ring signature schemes deal with linkability. Linkable ring signatures are first defined in [7]. In this paper, the authors present a ring signature that allows users to publicly link two signatures produced by the same user within the group. The security model of this primitive are formalized in [8]. Although the signature size grows linearly with the group size in most schemes, there exists constant size signatures such as [9] and [10]. The main applications of linkable signature are e-voting and e-cash, but this property is also used in several other applications, such as the direct anonymous attestation scheme described in [11].

In [12], Canard et al. present list signatures that add a property to linkable group signature: the identity of the signer of two linked signatures can be publicly computed. Few years later, Canard et al. give a list signature construction for ad-hoc group based on ring signature [2]. In this scheme, the signature size grows linearly with the group size. In [13], the authors give a constant size identity-based ring signature scheme with similar properties (linkability and traceability).

Anonymous authentication is closely related to group/ring signatures. For instance, \( k \)-times anonymous authentications [14], [15], [16] allow a group member to anonymously authenticate himself \( k \) times, but the \((k+1)\)th authentication allows the verifier to trace his identity. In [1] the authors adapt \( k \)-times anonymous authentications to linkable group signatures. This signature primitive allows everyone to trace a user that has produced more than \( k \) signatures by linking two of theses signatures. This primitive can be viewed as a generalization of list signature schemes. However, the generalization is incomplete since only two signatures among \( k+1 \) are linked, and the anonymity of the signer is publicly revealed only for these two signatures but his other signatures remain anonymous. Moreover, the number \( k \) is the same for each group member. To the best of our knowledge, there exists no ring signature that ensures full anonymity for members that produce less than \( k \) signatures, and public full linkability/traceability on all signatures of a member after the \((k+1)\)th signature (for a personal bound \( k \)). Solving this problem is the goal of our \( k \)-times full traceable ring signature primitive.

c) Outline: In the next section, we recall some cryptographic notions. In Section III, we present our security models. In Section IV, we give our scheme and discuss about its security. In Section V, we give applications of our scheme. Finally, we conclude in Section VI.

II. CRYPTOGRAPHIC TOOLS

We present some cryptographic assumptions and some results on zero knowledge proofs. First, we recall the decisional Diffie-Hellman assumption and his bilinear variant.

Definition 1 (Decisional Diffie-Hellman [17]). Let \( G \) be a multiplicative group of prime order \( p \) and \( g \in G \) be a generator. The decisional Diffie-Hellman problem (DDH) is to decide whether \( z = ab \) given \( (g^a, g^b, g^z) \) for unknown \( a, b \in \mathbb{Z}_p^* \). The decisional Diffie-Hellman hypothesis states that there exists no polynomial time algorithm that solves DDH with non-negligible advantage.

Definition 2 (Bilinear Decisional Diffie-Hellman [18]). Let \( G_1, G_2 \) and \( G_t \) be three groups of prime order \( p \) and \( e : G_1 \times G_2 \to G_t \) be a type 3 non-degenerate bilinear pairing. Let \( g_1 \in G_1 \) and \( g_2 \in G_2 \) be two generators. The Bilinear Decisional Diffie-Hellman problem (BDDH) is to decide whether \( z = abc \) given \( (g_1^a, g_2^b, g_t^c, e(g_1, g_2)^z) \) for unknown \( a, b, c \in \mathbb{Z}_p^* \). The Bilinear Decisional Diffie-Hellman problem states that there exists no polynomial time algorithm that solves BDDH with non-negligible advantage.

In this work, we use the following variant of BDDH.

Definition 3 (2-Bilinear Decisional Diffie-Hellman). Let \( G_1, G_2 \) and \( G_t \) be three groups of prime order \( p \) and \( e : G_1 \times G_2 \to G_t \) be a type 3 non-degenerate bilinear pairing. Let \( g_1 \in G_1 \) and \( g_2 \in G_2 \) be two generators. The Bilinear Decisional Diffie-Hellman problem (2BDDH) is to decide whether \( z = ab \) given \( (g_1^a, g_2^b, g_t^c, e(g_1, g_2)^z) \) for unknown \( a, b, c, d \in \mathbb{Z}_p^* \). The 2-Bilinear Decisional Diffie-Hellman problem states that there exists no polynomial time algorithm that solves 2BDDH with non-negligible advantage.

This assumption is very close to BDDH. Actually, an instance of 2BDDH contains \( (g_1^a, g_2^b, e(g_1, g_2)^z) \) that is an instance of the BDDH problem and two additional elements \( (g_2^d, e(g_1, g_2)^{z \cdot abc}) \). We do not know any reduction from 2BDDH to BDDH (or any other standard assumption such that the decisional Diffie-Hellman assumption in \( G_t \) [17] or the pairing inversion problem [19]). However, this problem seems to be difficult to solve: values \( (g_2^d, e(g_1, g_2)^{z \cdot abc}) \) are hard
to exploit since \( e(g_1, g_2)^{abc} \) is indistinguishable to a random group element under BDDH, and it is hard to extract \( g_1^b \) from \((g_1^a, g_1^b, e(g_1, g_2)^{abc})\) under the pairing inversion problem.

The hardness of this assumption can be argued by the following property: 2BDDH is hard under BDDH when \( e \) is a type 1 or type 2 pairing. Indeed, if \( \mathcal{G}_1 = \mathbb{G}_2 = \mathbb{G} \) and \( g_1 = g_2 = g \), it is possible to transform an instance \((g^a, g^b, g^d, e(g, g)^3)\) of BDDH into an instance of 2BDDH. Picking \( c \in \mathbb{Z}_p^* \), one can compute \( g^c \) and \( e(g^a, g^b)^c = e(g, g)^{abc} \) and build the instance \((g^a, g^b, g^c, e(g, g)^{abc}, e(g, g)^3)\) which is an instance of 2BDDH. Then, knowing an algorithm that solves the 2BDDH problem, we can deduce an algorithm that solves the BDDH problem in a similar running time. It is possible to construct a similar reduction using type 2 pairing. Unfortunately, the same reduction does not work for type 3 but the problem seems to be at least as hard as in the type 1 or type 2 cases.

**Definition 4** (Zero-knowledge proofs [20]). A proof of knowledge is a two-party protocol between two polynomial time algorithms \( P \) (the prover) and \( V \) (the verifier). It allows the prover \( P \) to convince the verifier \( V \) that he knows a solution \( s \) to the instance \( I \) of a problem \( P \). Such a protocol is said zero-knowledge proof of knowledge (ZKP) if it satisfies the following properties:

- **Completeness**: If \( P \) knows \( s \), then he is able to convince \( V \) (i.e. \( V \) outputs "accept.")
- **Soundness**: If \( P \) does not know \( s \), then he is not able to convince \( V \) (i.e. \( V \) outputs "reject") except with negligible probability.
- **Zero-knowledge**: \( V \) learns nothing about \( s \) except \( I \), i.e. there exists a probabilistic polynomial time algorithm \( S_i \) (called the simulator) such that outputs of the real protocol and outputs of \( S_i(I, \text{input}_V) \) follow the same probability distribution, where \( \text{input}_V \) denotes the input used by \( V \) for the real protocol.

Honest-verifier ZKP (HZKP) is a weaker notion of ZKP which is restricted to case where the verifier is honest, i.e. \( V \) correctly runs the protocol.

If we only have one transaction from the prover to the verifier, we say that the ZKP is non-interactive (NIZKP). In the literature, sigma protocols are ZKP with three exchanges between the prover and the verifier: a commitment, a challenge, and a response (by example the Schnorr protocol [21]). If the challenge is chosen on a large set, it is possible to transform a sigma protocol into a NIZKP using the Fiat-Shamir heuristic [22] replacing the challenge by the digest of a hash function on the commitment.

Finally, our scheme uses the generic transformation of ZKP designed by [23]. The authors propose a generic transformation from the ZKP of the solution to some problem instance to a ZKP of the solution to one problem instance out of \( n \) problem instances (without revealing this problem instance). This transformation holds with any sigma protocol. The computational and space cost of the resulting ZKP is \( n \) times the cost of the primary ZKP. It is possible to use the Fiat-Shamir transformation on such a ZKP to obtain an equivalent NIZKP.

### III. Model and Security

We first formally define \( k \)-times full traceable ring signature (\( k \)-FTRS) scheme, next we define the security models.

Let \( k \) be the maximum of authorized anonymous signatures then a \( k \)-FTRS is a ring signature scheme that has three additional functionalities depending on the parameter \( k \):

- (i) a link algorithm allows users to link two signatures produced by the same member who has produced more than \( k \) signatures;
- (ii) a match algorithm extracts the identity of a member \( u \) and a tracer, denoted by \( \omega_{(E, u)} \), from two linked signatures for the same event \( E \);
- (iii) a trace algorithm allows users to decide whether a signature has been produced by the user \( u \) for the same event \( E \), using the corresponding tracer \( \omega_{(E, u)} \).

Each group member has his own parameter \( k_u \) that is used to generate his pair of signing/verification key. Thus, to publicly detect a cheater who has produced more than \( k_u \) signatures in a set of \( r \) signatures, it suffices to use the link algorithm on all pairs of signatures. The match algorithm allows users to identify the cheater and returns a tracer \( \omega_{(E, u)} \). Using this tracer and the trace algorithm, everyone can detect all other signatures produced by the identified cheater. The number of calls to the link algorithm is quadratic in \( r \) and the number of calls to the trace algorithm is linear in \( r \).

As it is often required in linkable signatures, all signatures are computed for a particular event. Thus, link, match and trace algorithms can be only used for signatures coming from the same event. More formally, the signature algorithm requires a bit string \( E \), called an event, that corresponds to the identification of a given event (for example the concatenation of the date, title and the location of the election).

**Definition 5** \( (k \)-times full traceable ring signature (\( k \)-FTRS)).

A \( k \)-FTRS is defined by the following algorithms:

- **Init**\(^{(i)}\): This algorithm outputs an \( \text{init} \) value from security parameter \( t \).
- **Gen**\((\text{init}, k)\): This algorithm outputs a signing key pair \((ssk, svk)\) from \( \text{init} \) and a threshold value \( k \) denoting the maximum number of anonymous signatures authorized for the key \( ssk \).
- **Sig**\((\text{ssk}, m, L, j)\): This algorithm outputs a signature \( \sigma \) on the message \( m \) using the event \( E \), the signing key \( ssk \), the set of public keys of all members of the group \( L \) and the witness \( j \in \{1, \ldots, k\} \), where \( k \) is the value used to generate \( ssk \).
- **Ver**\((E, \sigma, m)\): This algorithm checks that \( \sigma \) is a valid signature of \( m \) for the event \( E \) and the set \( L \).
- **Link**\((E, L_1, L_2, \sigma_1, \sigma_2, m_1, m_2)\): This algorithm checks whether the two signatures \( \sigma_1 \) and \( \sigma_2 \), for the messages \( m_1 \) and \( m_2 \) and the sets of public keys \( L_1 \) and \( L_2 \), come from the same signing key \( ssk \) for the event \( E \). In this
case, it returns 1, else it returns 0. This algorithm only links two signatures of the same member out of k + 1, for k the value used to generate sskk.

\text{Match}_{E}(L_1, L_2, \sigma_1, \sigma_2, m_1, m_2): This algorithm outputs \( \perp \) if \( \text{Link}_{E}(L_1, L_2, \sigma_1, \sigma_2, m_1, m_2) \neq 1 \). Else, let svk, be the key of the signer who has produced \( \sigma_1 \) and \( \sigma_2 \). If \( \text{svk}_u \in L_1 \cup L_2 \) then this algorithm outputs \( \text{svk}_u \) and a tracer \( \omega_{(E, u)} \), else \( \perp \).

\text{Trace}_{E}(L, \sigma, m, \omega_{(E, u)}): This algorithm checks that the signature \( \sigma \) of \( m \) for the set of public keys L has been produced in the event E by the user u using the tracer \( \omega_{(E, u)} \). In this case it returns 1, else 0.

The following definition presents the correctness of such a scheme. Loosely speaking, a k-FTRS is correct when the algorithms \text{Ver}, \text{Link}, \text{Match} and \text{Trace} correctly work using signatures coming from the algorithm \text{Sig}.

\textbf{Definition 6 (Correctness).} We say that a k-FTRS is correct if for any key pair \( (\text{ssk}_i, \text{svk}_i) \) generated by \text{Gen}(init, k) and any signatures \( \sigma = \text{Sig}_E(\text{ssk}_i, m, L) \), \( \sigma_1 = \text{Sig}_E(\text{ssk}_i, m_1, L_1, j) \) and \( \sigma_2 = \text{Sig}_E(\text{ssk}_j, m_2, L_2, j) \) (where \( i, j \in \{1, \ldots, k\} \)), the following conditions hold:

1) \( \text{Ver}_E(L_i, \sigma, m, \omega) \) outputs 1.
2) \( \text{Link}_{E}(L_1, L_2, \sigma_1, \sigma_2, m_1, m_2) \) outputs 1.
3) If the algorithm \text{Match}_{E}(L_1, L_2, \sigma_1, \sigma_2, m_1, m_2) outputs \( (\text{svk}, \omega) \) then \( \text{svk} = \text{svk}_i \).
4) If the algorithm \text{Match}_{E}(L_1, L_2, \sigma_1, \sigma_2, m_1, m_2) outputs \( (\text{svk}, \omega) \) then \( \text{Trace}_{E}(L, \sigma, m, \omega) = 1 \).

The first point checks the verification algorithm works correctly for any signature generated by the algorithm \text{Sig}.

The second one shows that the algorithm \text{Link} outputs 1 if the two given signatures use the same witness j and the same signature key \text{ssk}_i.

The third point ensuresthat the algorithm \text{Match} outputs the verification key associated to two given linked signatures.

The last point verifies that the tracer and the verification signature key \text{svk}_i outputed by the algorithm \text{Match} allow the algorithm \text{Trace} to trace all the signatures produced by \text{ssk}_i.

We formalize the following properties of k-FTRS:

\textbf{Unforgeability:} It is computationally infeasible to forge a valid signature without the secret key of a group member.

\textbf{Traceability:} More then k signatures coming from the same user in the same event are always traceable.

\textbf{Anonymity:} It is computationally infeasible to determine the identity of an honest user from less than \( (k + 1) \) of his signatures for each event.

Our security models are based on [8] that formalizes the security of linkable ring signatures.

\textbf{Unforgeability:} A k-FTRS is unforgeable when there exists no polynomial adversary able to create a new valid signature for the group. The adversary has access to a signature oracle that computes signatures for given messages and events using the secret key of chosen user. To win the adversary must generate a valid signature that does not come from the oracle.

In what follow, we denote by \text{out}_\mathcal{O} the set of all the values outputed by the oracle \mathcal{O} during an experiment. Unforgeability is defined as follows.

\textbf{Definition 7 (EUF-CMA security).} Let P be a k-FTRS of security parameter t and let \mathcal{A} be a polynomial time adversary. We define the \((n, k)\)-existential unforgeability against chosen message attack experiment for \mathcal{A} as follows:

\text{Exp}_{n,k}(\mathcal{P}, \mathcal{A})(t):

\begin{align*}
\text{init} \leftarrow \text{Init}(1^t) \\
\forall i \in \{0, \ldots, n\}, (\text{ssk}_i, \text{svk}_i) \leftarrow \text{Gen}(\text{init}, k) \\
U \leftarrow \{\text{svk}_i\}_{0 \leq i \leq n} \\
(L_i, \sigma_i, m_i, E_i) \leftarrow \mathcal{A}_{\text{SO}_1}(i, t, U) \\
\text{if Ver}_E(L_i, \sigma_i, m_i) = 1 \text{ and } (L_i, \sigma_i, m_i, E_i) \notin \text{out}_{\text{SO}_1}, \text{ then output 1 else output 0.}
\end{align*}

Where \text{SO}_1(\cdot) is a signing oracle that takes \( (\text{svk}_k, L, E, m, j) \) as input. If \text{svk}_i \notin U then it returns \( \perp \), else it computes \( \sigma = \text{Sig}_E(\text{ssk}_i, m, L, j) \) and returns \( (L, \sigma, m, E) \).

The advantage of the adversary \mathcal{A} against \((n, k)\)-EUF-CMA is \( \text{Adv}_{\mathcal{P}, \mathcal{A}}^{(n, k)\text{-euf-cma}}(t) = \text{Pr}[\text{Exp}_{n,k}(\mathcal{P}, \mathcal{A})(t) = 1] \).

We define the advantage on \((n, k)\)-EUF-CMA experiment by \( \text{Adv}_{\mathcal{P}, \mathcal{A}}^{(n, k)\text{-euf-cma}}(t) = \max_{\mathcal{A} \in \text{POLY}(t)} \{\text{Adv}_{\mathcal{P}, \mathcal{A}}^{(n, k)\text{-euf-cma}}(t)\} \).

We say that a k-FTRS scheme P is \((n, k)\)-EUF-CMA secure when the advantage \text{Adv}_{\mathcal{P}, \mathcal{A}}^{(n, k)\text{-euf-cma}}(t) is negligible.

\textbf{Traceability:} A k-FTRS is traceable when there exists no polynomial adversary who is able to generate at least k valid signatures for the same event knowing only one secret key from the group such that the match and trace algorithms fail to trace the corresponding public key on each signatures. To help him, the adversary has access to a signature oracle that computes signatures for given messages and given events using secret keys of chosen users. This property is formally given in the following definition.

\textbf{Definition 8 (Traceability).} Let P be a k-FTRS of security parameter t and let \mathcal{A} = (A_1, A_2) be a pair of algorithm in \text{POLY}(t). We define the \((n, k)\)-traceability experiment for adversary \mathcal{A} as follows:

\text{Exp}_{n,k}(\mathcal{P}, \mathcal{A})(t):

\begin{align*}
\text{init} \leftarrow \text{Init}(1^t) \\
\forall i \in \{0, \ldots, n\}, (\text{ssk}_i, \text{svk}_i) \leftarrow \text{Gen}(\text{init}, k) \\
U \leftarrow \{\text{svk}_i\}_{0 \leq i \leq n} \\
\pi \leftarrow A_1^{\text{SO}_1}(i, t, U) \\
(z, \{(L_i, \sigma_i, m_i)\}_{1 \leq i \leq z}, E) \leftarrow A_2^{\text{SO}_2}(i, \text{ssk}_i, U) \\
\text{if } (z > k) \\
\text{and } (\forall i \in \{1, \ldots, z\}, \text{Ver}_E(L_i, \sigma_i, m_i) = 1 \text{ and } (L_i, \sigma_i, m_i, E) \notin \text{out}_{\text{SO}_1})
\end{align*}

and
\begin{align*}
&(\forall a, b, \text{Link}_{E}(L_a, L_b, \sigma_a, \sigma_b, m_a, m_b) \neq 1) \text{ or } (\forall a, b, i, \text{Match}_{E}(L_a, L_b, \sigma_a, \sigma_b, m_a, m_b) = (\text{svk}_a, \omega_a) \Rightarrow (\text{svk} \neq \text{svk}_a \text{ or Trace}_{E}(L_i, \sigma_i, m_i, \omega_i) \neq 1)) \\
\text{then output 1 else output 0.}
\end{align*}

Where \text{SO}_1(\cdot) is defined as in Definition 7. The advantage of the adversary \mathcal{A} against \((n, k)\)-traceability is defined by \( \text{Adv}_{\mathcal{P}, \mathcal{A}}^{(n, k)\text{-trace}}(t) = \text{Pr}[\text{Exp}_{n,k}(\mathcal{P}, \mathcal{A})(t) = 1] \).

We define the advantage on \((n, k)\)-traceability experiment by
\textbf{Anonymity:} A \(k\)-FTRS is anonymous when there exists no polynomial adversary able to distinguish the signer of a given message between two given honest group members. The adversary chooses two honest user’s public keys, a group of users, a message and an event, the he sends it to the challenger. The challenger signs the message using one of the two corresponding secret keys. Then the adversary must guess who the signature is. To help him, the adversary has access to a signature oracle that computes signatures for given messages and given events using the secret key of chosen users. However, the oracle does not produce more that \(k-1\) signatures for the two chosen public keys and the corresponding event used to produce the challenge since signatures would be traceable in this case. This notion is formally introduced in the following definition.

**Definition 9 (Anonymity).** Let \(P\) be a \(k\)-FTRS of security parameter \(t\) and let \(A = \{A_1, A_2\}\) be polynomial time adversary. We define the \((n, k)\)-anonymity experiment for adversary \(A\) against \(P\) as follows:

\[
\text{Exp}_{P, A}^{\text{anony}}(t) = \text{Init}(t) = \{0, 1\}
\]

\[
\text{Output} = \text{A} \cdot \text{Sig}_{E}(\text{ssk}, m, j).
\]

The advantage of the adversary \(A\) in the \((n, k)\)-anonymity experiment is

\[
\text{Adv}_{P, A}^{\text{anony}}(t) = \text{Pr}(\text{Exp}_{P, A}^{\text{anony}}(t) = 1) - \frac{1}{2}.
\]

We define the advantage on \((n, k)\)-anonymity experiment by

\[
\text{Adv}_{P}^{\text{anony}}(t) = \max_{A \in \text{poly}(t)} \left\{ \text{Adv}_{P, A}^{\text{anony}}(t) \right\}.
\]

We say that a \(k\)-FTRS scheme \(P\) is \((n, k)\)-anonymous when the advantage \(\text{Adv}_{P}^{\text{anony}}(t)\) is negligible.

**B. An Helpful ZKP**

In order to have all the necessary tools to construct our scheme, we derive the following ZKP, denoted \(I\).

Let \(G_1, G_2, G_t\) be three groups of prime order \(p\), \(g_1\) and \(g_2\) be two respective generators of \(G_1\) and \(G_2\), \(e: G_1 \times G_2 \rightarrow G_t\) be a non-degenerate bilinear pairing and \(n\) be an integer. Let \(A, B, C, W\) be seven elements of \(G_1\), \(T_i\) be an element of \(G_2\), \(T_i\) be an element of \(G_t\) and \(v\) be two elements of \(Z_p^*\). Finally, for all \(i \in \{1, \cdots, n\}\), let \((h_i, l_i)\) be some couples of \(G_t^2\). Using

\[
Q = (g_1, g_2, A, B, C, W, u, v, T_1, T_2, T_3, T_4, T_5)
\]

\[
S = \{(Q, h_i, l_i)\}_{i \in \{1, \cdots, n\}}
\]

build \(S\), a NIZKP of knowledge of \((x, y, z) \in (Z_p)^3\) such that

\[
T_1 = A^x, T_2 = B^x \cdot g_1^u, T_3 = C^x \cdot W^v, T_4 = g_2^z, T_5 = e(W^v, T_4), h = g_1^l \text{ and } l = g_2^l \text{ for one } (Q, h, l) \in S.
\]

We first describe in Figure 1 the interactive case \(I_1\) where \(n = 1\), hence there is only one couple \((h, l)\). It is based on the classical methodology of ZKP of discrete logarithm knowledge [21] and equality of two discrete logarithms [24].

This proof is by construction a sigma-protocol.

**Lemma 10.** The ZKP \(I_1\) is complete, sound, and honest-verifier zero-knowledge.

See Appendix A, for the proof of Lemma 10. As \(I_1\) is honest-verifier zero knowledge and a sigma protocol, we can use the generic transformation of [23] to obtain the interactive version of our proof for any \(n \geq 0\). Finally, using this transformation and the Fiat-Shamir heuristic on \(I_1\), we build the non-interactive proof \(I_2\) in the random oracle model.
Theorem 11. The NIZKP II is complete, sound, and zero-knowledge in the random oracle model.

It is a direct implication of [23] and Lemma 10.

C. Our Construction: Ktrace

Our construction, given in Scheme 1, requires three groups \( G_1, G_2 \) and \( G_3 \) of prime order \( p \) and a bilinear pairing \( e \). Let \( g_1 \) and \( g_2 \) be two respective generators of \( G_1 \) and \( G_2 \). Each user generates an \( ElGamal \) signing key \( \{(\bar{x}_i,j, g_1^{x_i,j})\}_{1 \leq j \leq k} \) and an extra key pair \( (x_i, g_1^x) \) used to identify him with the algorithms match and trace.

To sign a message, a user picks \( r \in Z_p^* \) and chooses one of these ElGamal secret keys and computes \( T_1 = A^{x_{i,j}} \) and \( T_2 = B^{x_{i,j}} \cdot (g_1^x)^{x_i} \) as in the scheme [2] except that \( u = H_1(E, m, 0, g_2^z) \). Thus, using these two values, two signatures can be linked when the same secret key is used, and it is possible to deduce the public key \( g_1^x \) from these two signatures. Then the signer computes additional values \( T_3, T_4, T_5 \) and \( T_6 \) as follows: he sets \( C = H_0(E, 2), W = H_0(E, 3) \) and \( v = H_1(E, m, 1, g_2^z) \) and sets \( T_3 = C^{x_{i,j}} \cdot W^{x_i}, T_4 = g_2^z \) and \( T_5 = e(W, T_4)^{x_i} \). As in list signatures, the signer computes in \( T_6 \) a non-interactive proof that all other parts of the signatures \( T_1, T_2, T_3, T_4 \) and \( T_5 \) are "correctly formed" according to one of the verification keys of the group using the proof II given in Section IV-B. The complete signature is the tuple \( (T_1, T_2, T_3, T_4, T_5, T_6) \) and is verified by checking the validity of the proof \( T_6 \).

Note that using two linked signatures with respective third terms \( T_{1,3} = C^{x_{i,j}} \cdot W^{v_1 \cdot x_i} \) and \( T_{2,3} = C^{x_{i,j}} \cdot W^{v_2 \cdot x_i} \), it is possible to compute the tracer \( \omega(E,i) = (T_{1,3}/T_{2,3})^{(v_1 - v_2)} = W^{v_1} \) in addition to the public key \( g_1^x \) in the match algorithm, for an event \( E \). Finally, to trace a signature \( T \) generated by the owner of the public key \( g_1^x \) using the tracer \( \omega_i \), anybody can check that \( e(\omega_i, T_4) = e(W^{x_i}, T_4) = e(W, T_4)^{x_i} \). Since the proof \( T_6 \) assures that terms \( T_3, T_4 \) and \( T_5 \) are well formed, this equation always holds when the signer is the owner of the public key \( g_1^x \).

1We omit the event \( E \) in the notation of the tracer. When it is clear from the context, we simply write \( \omega_i \).

---

### Scheme 1 (Ktrace scheme)

Ktrace is a \( k \)-FTRS with the following algorithms:

**Init(1)i:** This algorithm generates three groups \( G_1, G_2 \) and \( G_3 \) of prime order \( p \), two respective generators \( g_1 \) and \( g_2 \) of \( G_1 \) and \( G_2 \), a non-degenerate bilinear pairing \( e \) of \( G_1 \times G_2 \rightarrow G_3 \) of type 3 and two hash functions \( H_0 : \{0, 1\} \rightarrow G_1, H_1 : \{0, 1\} \rightarrow Z_p^* \). The algorithm outputs \( \text{init} = (p, G_1, G_2, G_3, g_1, g_2, e, H_0, H_1) \).

**Gen(init, k_i):** This algorithm randomly picks \( x_i \) in \( Z_p^* \). It then picks \( k_i \) other values \( x_{i,j} \) in \( Z_p^* \) for \( j \in \{1, \ldots, k_i\} \).

It sets \( \text{svk}_i = (g_1^{x_i}, \{g_1^{x_{i,j}}\}_{1 \leq j \leq k_i}) \) and \( \text{ssk}_i = (x_i, \{x_{i,j}\}_{1 \leq j \leq k_i}) \) and returns \( \text{(svk}_i, \text{ssk}_i) \).

**Sig(2, ss_k, m, j):** Using \( \text{ssk}_i = (x_i, \{x_{i,j}\}_{1 \leq j \leq k_i}) \) and the event \( E \), this algorithm picks \( r \in Z_p^* \) and computes the following values:

\[
\begin{align*}
A & = H_0(E, 0) \\
C & = H_0(E, 2) \\
v & = H_1(E, m, 0, g_2^z) \\
T_1 & = A^{x_{i,j}} \\
T_2 & = B^{x_{i,j}} \cdot g_1^{x_i} \\
T_3 & = C^{x_{i,j}} \cdot W^{x_i} \\
T_4 & = g_2^z \\
T_5 & = e(W, T_4)^{x_i}
\end{align*}
\]

Let \( I \) be the set of user indexes such that:

\[
L = \{(g_1^{x_i}, \{g_1^{x_{i,j}}\}_{1 \leq j \leq k_i})\}_{i \in I}
\]

We set:

\[
Q = (g_1, g_2, A, B, C, W, u, v, T_1, T_2, T_3, T_4, T_5)
\]

The algorithm computes the following set:

\[
S = \{ (Q, g_1^{x_i}, \{g_1^{x_{i,j}}\}) \}_{(i,j) \in I \times \{1, \ldots, k_i\}}
\]

Finally, it generates a non-interactive proof of knowledge \( T_6 \) using \( \Pi \) (see section IV-B) of the solution \( (x_i, x_{i,j}, r) \) of one instance \( I = (Q, g_1^{x_i}, \{g_1^{x_{i,j}}\}) \) out of the instance set \( S \) without revealing neither \( (x_i, x_{i,j}, r) \) nor \( I \). It outputs the signature \( \sigma = (T_1, T_2, T_3, T_4, T_5, T_6) \).
\begin{align*}
\text{Ver}_E(L, \sigma, m) & : \text{ We set } \sigma = (T_1, T_2, T_3, T_4, T_5, T_6). \text{ Let } I \text{ be a set of user indexes such that } L = \{(g_1^{x_i}, \{g_1^{x_{i,j}}\}_{1 \leq j \leq k})\}_{i \in I}. \text{ This algorithm computes:}
A &= H_0(E, 0) \quad B = H_0(E, 1) \\
C &= H_0(E, 2) \quad W = H_0(E, 3) \\
u &= H_1(E, m, 0, T_4) \quad v = H_1(E, m, 1, T_4) \\
\text{It also computes:}
Q &= (g_1, g_2, A, B, C, W, u, v, T_1, T_2, T_3, T_4, T_5) \\
S &= \{(Q, g_1^{x_i}, g_1^{x_{i,j}})\}_{(i,j) \in I \times \{1, \ldots, k\}} \\
\text{Finally, it returns } 1 \text{ if the proof } T_6 \text{ is valid for } \Pi (\text{see section IV-B}) \text{ according to the set of instances } S.
\end{align*}

\begin{align*}
\text{Link}_E(L_1, L_2, \sigma_1, \sigma_2, m_1, m_2) & : \text{ We set}
\sigma_1 &= (T_{1,1}, T_{1,2}, T_{1,3}, T_{1,4}, T_{1,5}, T_{1,6}) \\
\sigma_2 &= (T_{2,1}, T_{2,2}, T_{2,3}, T_{2,4}, T_{2,5}, T_{2,6}) \\
\text{This algorithm returns } 1 \text{ if and only if}
\text{Ver}_E(L_1, \sigma_1, m_1) = \text{Ver}_E(L_2, \sigma_2, m_2) = 1 \text{ and } T_{1,1} = T_{2,1}.
\end{align*}

\begin{align*}
\text{Match}_E(L_1, L_2, \sigma_1, \sigma_2, m_1, m_2) & : \text{ We set}
\sigma_1 &= (T_{1,1}, T_{1,2}, T_{1,3}, T_{1,4}, T_{1,5}, T_{1,6}) \\
\sigma_2 &= (T_{2,1}, T_{2,2}, T_{2,3}, T_{2,4}, T_{2,5}, T_{2,6}) \\
\text{This algorithm computes the following hashed values:}
\omega_1 &= H_1(E, m_1, 0, T_{1,4}) \quad \omega_2 = H_1(E, m_1, 1, T_{2,4}) \\
\omega_3 &= H_1(E, m_2, 0, T_{2,4}) \quad \omega_4 = H_1(E, m_2, 1, T_{2,4}) \\
\text{If } \text{Link}_E(L_1, L_2, \sigma_1, \sigma_2, m_1, m_2) = 0 \text{ this algorithm outputs } \perp, \text{ else it computes:}
\text{id} &= \left( \frac{T_{1,2}}{T_{2,2}} \right)^{\frac{1}{\lambda_1 + \lambda_2}} ; \quad \omega_{(E,i)} = \left( \frac{T_{1,3}}{T_{2,3}} \right)^{\frac{1}{\lambda_1 + \lambda_2}}.
\end{align*}

\begin{align*}
\text{svk}_i &= (g_1^{x_i}, \{g_1^{x_{i,j}}\}_{1 \leq j \leq k}) \text{ be the element of } L_1 \cup L_2 \text{ such that } g_1^{x_i} = \text{id}. \text{ If such an element does not exist then this algorithm outputs } \perp, \text{ else this algorithm outputs (svk}_i, \omega_{(E,i)})
\text{Trace}_E(L, \sigma, m, \omega_{(E,i)}) : \text{ This algorithm set } \sigma = (T_1, T_2, T_3, T_4, T_5, T_6). \text{ If } \text{Ver}_E(L, \sigma, m) = 1 \text{ and } e(\omega_{(E,i)}, T_4) = T_5 \text{ then it returns } 1, \text{ else } 0.
\end{align*}

\section{D. Correctness}

\textbf{Theorem 12.} \textit{Ktrace is correct.}

\textbf{Proof.} \textit{Ver:} The set of instances \textit{S} is similarly computed in both algorithms Sig and Ver. The part \textit{T}_6 of a signature is a proof \textit{P} on the set of instances \textit{S}. Since the verification algorithm consists to check this proof, and since \textit{P} is complete, then algorithm Ver is correct, i.e. \textit{Ver}_E(L, \sigma, \textit{ssk}_k, m, L, j, m) = 1.

\textit{Link:} We show that when the signature \textit{ssk} = (\textit{ssk}_k, m, L, j) and \textit{ssk} = (\textit{ssk}_k, m, L, j), then \textit{Link}_E(L_1, L_2, \sigma, m_1, m_2) = 1. Using the same event \textit{E}, the same secret key \textit{ssk} = (x_i, \{x_{i,j}\}_{1 \leq j \leq k}) and the same witness \textit{j}, the first part of the signature is \textit{T}_1 = \textit{A}^x_j where \textit{A} = \textit{H}_0(E, 0). Let \textit{T}_1,1 and \textit{T}_2,1 be the first part of the two signatures (\textit{ssk}, \textit{ssk}), clearly \textit{T}_1,1 = \textit{T}_2,1 = \textit{A}^x_j.

\textit{Match:} We show that given two signatures \textit{ssk} = \textit{Sig}_E(\textit{ssk}_k, m_1, L_1, j) and \textit{ssk} = \textit{Sig}_E(\textit{ssk}_k, m_2, L_2, j) then \textit{Match}_E(L_1, L_2, \sigma_1, \sigma_2, m_1, m_2) outputs the public key of the signer \textit{ssk}_k:
\begin{align*}
id &= \left( \frac{T_{1,2}}{T_{2,2}} \right)^{\frac{1}{\lambda_1 + \lambda_2}} = \left( \frac{B^{x_{1,j}} \cdot g_1^{y_1 \cdot x_{i,j}}}{B^{x_{1,j}} \cdot g_1^{y_2 \cdot x_{i,j}}} \right)^{\frac{1}{\lambda_1 + \lambda_2}} \\
&= g_1^{x_i \cdot \frac{y_1 - y_2}{\lambda_1 + \lambda_2}} = g_1^{x_i}
\end{align*}

Then the algorithm outputs \textit{svk}_k = (g_1^{x_i}, \{g_1^{x_{i,j}}\}_{1 \leq j \leq k}) which is the public key corresponding to \textit{ssk}_k.

\textit{Trace:} Given two signatures \textit{ssk} = \textit{Sig}_E(\textit{ssk}_k, m_1, L_1, j) and \textit{ssk} = \textit{Sig}_E(\textit{ssk}_k, m_2, L_2, j) we show that \textit{Match}_E(L_1, L_2, \sigma_1, \sigma_2, m_1, m_2) outputs a tracer \omega_{(E,i)} such that:
\begin{align*}
\omega_{(E,i)} &= \left( \frac{T_{1,3}}{T_{2,3}} \right)^{\frac{1}{\lambda_1 + \lambda_2}} = \left( \frac{C^{x_{i,j}} \cdot W^{y_1 \cdot x_{i,j}}}{C^{x_{i,j}} \cdot W^{y_2 \cdot x_{i,j}}} \right)^{\frac{1}{\lambda_1 + \lambda_2}} \\
&= W^{x_i \cdot \frac{y_1 - y_2}{\lambda_1 + \lambda_2}} = W^{x_i}
\end{align*}

Then, using any \textit{m}, \textit{E}, \textit{L} and \textit{l}, we show that \textit{Trace}_E(L, \textit{Sig}_E(\textit{ssk}_k, m, L, l), \omega_{(E,i)}) = 1. Indeed, using \textit{Sig}_E(\textit{ssk}_k, m, L, l) = (T_1, T_2, T_3, T_4, T_5, T_6) such that \textit{T}_4 = g_2^j and \textit{T}_5 = e(W, T_4)^{x_i}, we have:
\begin{align*}
e(\omega_{(E,i)}, T_4) &= e(W^{x_i}, T_4) = e(W, T_4)^{x_i} = T_5
\end{align*}
\qed

\section{E. Security}

According to the security model introduced in Section III, we have the following theorem.

\textbf{Theorem 13.} \textit{Ktrace is (n,k)-EUF-CMA secure, (n,k)-traceable and (n,k)-anonymous under the decisional Diffie-Hellman assumption in } \mathbb{G}_1 \textit{ (Def. 1) and the 2-bilinear decisional Diffie-Hellman assumption (Def. 3) in the random oracle model for any polynomial bounded } k \textit{ and } n.

Full proof of this theorem will appear in the full version of this paper. In the following, we just give some proof intuitions for each properties.

\textit{(n,k)-EUF-CMA security:} In order to build a valid signature of \textit{m} using \textit{E}, the adversary must to compute a valid NIZKP in \textit{T}_6. This proof convinces the verifier that the signer has used a valid secret key \textit{ssk} according to the set of public keys \textit{L}. As a consequence, since the NIZKP is sound, the signer cannot generate a valid signature except with negligible probability.

\textit{(n,k)-traceability:} In a similar way, as \textit{P} is sound, a signature that success the verification check is correctly constructed. Since the adversary is not able to forge a signature using the secret key of an honest user under the soundness of \textit{P}, the traceability of \textit{Ktrace} is a direct implication of the correctness of the algorithms link, match and trace.
Let $T_1 = A^{x_{1, k}}$, $T_2 = B^{x_{2, k}}$, $T_3 = C^{x_{3, k}}$, $W^{x_{1, k}}$ be the three first parts of the challenge. To discover the user’s identity using these three values, the adversary must deduce $g_1^{x_{1, k}}$ or $g_1^{\frac{x_{1, k}}{x_{3, k}}}$ from $T_1$, $T_2$, and $T_3$, knowing the public hash values $A$, $B$, and $C$. However, since the key $x_{1, k}$ is used only once per event, $A^{x_{1, k}}$, $B^{x_{2, k}}$, and $C^{x_{3, k}}$ are computationally indistinguishable to three random elements according to the Diffie-Hellman assumption in $G_1$.

As a consequence, these three values hide all the information about the signer public key contained in $T_1$, $T_2$, and $T_3$. On the other hand, let $T_4 = g_1^{x_1}$, $T_5 = e(W^{x_1}, g_2)$, and $T_6$ be the three last terms of the signatures. Clearly, $T_6$ leaks no information about the user identity since it is zero-knowledge. Moreover, it seems to be hard to deduce the signer identity from $T_3$ since it is hard to guess that $T_3 = e(W^{x_1}, g_2^k)$ knowing $W$, $g_1^{x_1}$ and $g_2^k$ under the BDDH assumption. Actually, we do not reduce the security of Ktrace to BDDH but to 2BDDH because asking the signing oracle, the adversary can learn the part $T_3 = e(W^{x_1}, g_2^k)$ of an other signature using the same key as the challenge. Thus, the adversary wins the experiment if he guesses whether $T_3 = e(W^{x_1}, g_2^k)$ or not knowing $W$, $g_1^{x_1}$, $g_2^k$, $g_2^{x_1}$ and $e(W^{x_1}, g_2^k)$. Finally, it is computationally infeasible to guess the identity of an honest voter from one of its signatures under the decisional Diffie-Hellman assumption in $G_1$ and the 2-bilinear decisional Diffie-Hellman assumption.

F. Performances

The performances of the scheme Ktrace depend of $n$ the number of users in the group and $k = \max_{0 \leq i \leq n}(k_i)$ the maximum in the set of all user threshold values $k_i$. The secret key of each user contains at most $(k + 1)$ elements of $\mathbb{Z}_p^*$, i.e. the size of the public key of each user is $(k + 1) \cdot s_p$ where $s_p = \log_2(p)$ is the bit size of the elements of $\mathbb{Z}_p^*$. Thus, the group public key size is $(k + 1) \cdot n \cdot s_1$ where $s_1$ is the bit size of an element of $G_1$.

In the signature algorithm, we use the NIZKP II which is constructed using the generic transformation [23] on $\Pi_1$. The size of this NIZKP is $6 \cdot s_1 + 2s + s_T + 3 \cdot s_p$ where $s_2$ is the bit size of an element of $G_2$ and $s_T$ is the bit size of an element of $G_T$. The size of a NIZKP built from $[23]$ is $x$ times the size of the original proof, where $x$ is the number of instances used. In our scheme, the cardinal of the set of instances $S = \{(Q, g_1^{x_{1, k}}, g_1^{x_{2, k}})\}_{(i,j) \in I \times \{1, \ldots, n\}}$ is at most $n$-k. Then the size of the NIZKP is bounded by $(6 \cdot s_1 + 2s + s_T + 3 \cdot s_p) \cdot n \cdot k$.

Finally, the size of a signature is $(T_1, T_2, T_3, T_4, T_5, T_6)$ is $(6 \cdot s_1 + s_2 + s_T + 3 \cdot s_p) \cdot n \cdot k$ which is in $O(n \cdot k)$. The complexity of the signing algorithm and the verification algorithm are in $O(n \cdot k)$.

The complexity of the three algorithms link, match and trace are constant. Moreover, the tracer outputted by the match algorithm is also in constant size $s_1$. According to the comparative table in Figure 2, we show that the signature size of our scheme loses at least a factor $k$ compared to other ring signature schemes, particularly comparing to the list signature scheme [2]. However, it is the only one which is $k$-times, ad-hoc and fully traceable together. Note that a 1-times scheme is always full traceable by construction.

Finally, note that in both applications given in the next section, $k$ is a priori much smaller than $n$. Moreover, our applications do not implicate that $k$ increases when $n$ does.

Thus, depending to the context, the signature size $O(n \cdot k)$ can be often considered to be relatively closed to $O(n)$.

V. Applications

A. $k$-times Proxy E-Voting

Linkability in group and ring signatures is a useful property to design cryptographic e-voting schemes. As a matter of fact, one can build a simple e-voting scheme from any linkable group signature scheme as follows. To vote for the candidate $c$, the voter signs $c$ in $\sigma$ using a linkable group signature scheme. He then publishes $(c, \sigma)$ on a public bulletin board.

The bulletin board is a publicly readable storage such that everybody can write data on the board but it is not possible to discard any data from it. To compute the result of the election, it is sufficient to remove all non-valid signatures according to the group public key and all linked signatures, and to count the number of ballots for each candidate. Such a scheme presents several properties:

- Ballots are anonymous thanks to the anonymity of the group signature.
- A voter cannot vote more than his number of authorized votes thanks to the linkability of the group signature.
- The result of the election can be publicly computed by anyone.
- Using linkable ring signature, this scheme does not require any manager or trust party.
- Using event oriented signatures, voters can use the same keys for several elections. In a multi-group context, using the group identity as event allows the users to use the same signing key for different groups.
- Using traceable signatures, the identity of a cheater is public, and it is possible to revoke the cheaters.

On the other hand, $k$-times schemes can be used to design such an e-voting scheme with proxy mechanism. In proxy voting, a signer who receives a proxy vote from another one must be able to vote one more time. Thus, any voter can vote $k$ times where $k$ is the number of proxy that he has received. Of course, if the voter votes more than $k$ times, he must be traced and revoked. In this case, we would like to discard all votes of the cheaters during the counting of votes. However, such a property cannot be achieved with a basic $k$-times signature scheme since it is not full traceable. To solve this problem, we propose the following $k$-times proxy e-voting scheme based on our $k$-FTRS scheme.

Setup: Each voter uses $Gen$ to generate secret/public keys depending on his number of proxy $k$. The voters construct the group public key $GPK$ which is the set of all voters public keys.

Initialization of an election: Users choose the event $E$ and the set of candidates $C$. 

Voting protocol: Each user chooses $k$ candidates $(c_1, \ldots, c_k) \in O^k$ and publishes on the public bulletin board the ballots $(c_j, \text{Sig}_G(\text{ssk}, c_j, \text{GPK}, j))$ for all $j \in \{1, \ldots, k\}$ using his secret key $\text{svk}$.

Counting of votes: Copy all ballots from the bulletin board. First, everyone removes each ballot which contains invalid signature using the algorithm $\text{Ver}$. Then anyone uses the $\text{Link}$ algorithm on each pair of ballots, and the $\text{Match}$ algorithm on each pair of linked ballots to obtain the cheaters identities and the corresponding tracers. For each tracer, we can use the $\text{Trace}$ algorithm on each ballot, and remove it if $\text{Trace}$ returns 1. Finally, everyone counts the number of ballots for each candidate and deduces the election winner.

Note that cheaters can also be revoked of the group for the future elections.

B. $k$-times Anonymous Veto

A $k$-times anonymous veto allows members of a group to anonymously express $k$ vetos. For example, during the organization of a conference, the members of the steering committee constitute the list of members of the technical committee. For some reason, a member of the steering committee might want to refuse the designation of some people on this list. However, members who exclude people might want to be anonymous. On the other hand, it is desirable that the number of people excluded per member is limited to a fixed value $k$. Members who exceed the limit must be revoked to the steering committee. In this case, it is necessary to discard all requests produced by the revoked members. Such a veto scheme can be easily designed using a $k$-FRS scheme:

Setup: Each group member uses $\text{Gen}$ to generate secret/public keys depending to his number of veto $k$. The members construct the group public key $\text{GPK}$ which is the set of all members public keys.

Initialization: Users choose the event $E$ and the set of items $C$.

Veto protocol: Each member chooses $l$ items $(c_1, \ldots, c_l) \in C^k$ such that $l \leq k$. To veto each item $c_j$, the member publish on the public bulletin board the witness $(c_j, \text{Sig}_E(\text{ssk}, c_j, \text{GPK}, j))$ for all $j \in \{1, \ldots, l\}$ using his secret key $\text{svk}$.

Cheater detection: Anybody can use the following procedure to detect a member who places more than $k$ vetos: we use $\text{Link}$ algorithm on each pair of witness, and $\text{Match}$ algorithm on each pair of linked witness to obtain the cheater’s identities and the corresponding tracers. Thanks to $\text{Match}$, it is possible to revoke the cheaters.

Finding invalid witness: All witnesses which contain an invalid signature are invalid. If the previous procedure detects cheaters, the $\text{Trace}$ algorithm is used on each witness for each tracer outputted by $\text{Match}$. If $\text{Trace}$ outputs 1 then the corresponding witness is invalid.

Limitation of other $k$-times schemes: Since other schemes in the literature are not full traceable, they present some limitation for building $k$-times anonymous veto schemes. Without full traceability, it is not possible to discard all witnesses produced by a cheater. However, the cheater is revoked, so he is no longer in the committee, and he no longer has the right to exclude people from the list. In this case, other members of the committee have no other choice but to restart the veto procedure since other witness of the cheater remain anonymous. Our scheme allows to solve this problem.

VI. Conclusion

In this paper, we give a more general definition of $k$-times linkable/traceable signatures and their security. Our construction allows members to chose a different threshold $k$ for each group member, and anybody can trace all signatures generated by the same member. Future work will investigate the design of such a scheme with smaller (or constant) size of signature. On the other hand, we aim at designing $k$-times full traceable schemes using the same methodology on other existing linkable/traceable ring/group signatures.

References


APPENDIX A

PROOF OF LEMMA 10

Lemma 10. Completeness: Knowing the solution \((x, y, z)\) of the instance \((T_1 = A^2, T_2 = B^x \cdot g_1^y, T_3 = C^z, W^v = T_4 = g_2^z, T_5 = e(W, T_4)^y = e(W, T_4)^x \cdot e(W, T_4)^{z'}, h = g_1^x, l = g_1^y)\), an honest prover is always able to compute \((\bar{x}, \bar{y}, \bar{z})\) from any commitments \(R_0 = g_1^x, R_1 = A^r, R_2 = B^s, R_3 = C^t, S_0 = g_1^z, S_1 = W^v, S_2 = e(W, T_4)^s, Q_0 = g_1^y\) following the protocol. Then the following equations hold:

\[
A^2 = R_1 \cdot T_1^{c_0}; \\
B^x = g_1^{y}; \\
C^x \cdot W^v = T_4 = g_2^z \cdot T_5^{c_0}; \\
\begin{align*}
A^2 \cdot B^x \cdot C^x \cdot W^v & = R_1 \cdot T_1^{c_0} \\
B^x \cdot C^x \cdot W^v & = g_1^{y} \\
B^x \cdot C^x \cdot W^v & = R_2 \cdot S_0^{c_0} \cdot T_2^{c_0} \\
B^x \cdot C^x \cdot W^v & = R_2 \cdot S_0^{c_0} \cdot T_2^{c_0} \\
B^x \cdot C^x \cdot W^v & = R_3 \cdot S_1^{c_0} \cdot T_3^{c_0} \\
B^x \cdot C^x \cdot W^v & = R_3 \cdot S_1^{c_0} \cdot T_3^{c_0} \\
g_2^z & = g_2^z \cdot T_5^{c_0} = Q_0 \cdot T_4^{c_0} \\
g_2^y & = g_2^y \cdot T_5^{c_0} = S_0 \cdot T_4^{c_0} \end{align*}
\]

Soundness: We consider a prover \(\bar{x}, \bar{y}, \bar{z}\) who is able to respond to two different challenges \(c_0 \in Z_p^*\) and \(c_1 \in Z_p^*\) from the same commitment \((R_0, R_1, R_2, R_3, S_0, S_1, S_2, Q_0)\). We then prove that this prover is able to compute a valid witness \((x, y, z)\). Let \((\bar{x}, \bar{y}, \bar{z})\) be the response to the challenge \(c_0\) (resp. \(c_1\)). By hypothesis:

\[
e(W, T_4)^y = e(W, T_4)^x \cdot e(W, T_4)^{z'} = S_2 \cdot T_5^{c_0} \\
g_2^z = g_2^z \cdot T_5^{c_0} = Q_0 \cdot T_4^{c_0}
\]

We prove that \((x, y, z) = (\frac{1}{c_1} \cdot (\bar{x} + \bar{y} \cdot \bar{z}) \cdot \bar{z}^2, \bar{y}^2, \frac{1}{c_1} \cdot (\bar{x} + \bar{y} \cdot \bar{z}) \cdot \bar{z}^2)\) is a valid witness. We then check that:

\[
A^2 = R_1 \cdot T_1^{c_0}; \\
B^x = g_1^{y}; \\
C^x \cdot W^v = T_4 = g_2^z \cdot T_5^{c_0}; \\
\begin{align*}
A^2 \cdot B^x \cdot C^x \cdot W^v & = R_1 \cdot T_1^{c_0} \\
B^x \cdot C^x \cdot W^v & = g_1^{y} \\
B^x \cdot C^x \cdot W^v & = R_2 \cdot S_0^{c_0} \cdot T_2^{c_0} \\
B^x \cdot C^x \cdot W^v & = R_2 \cdot S_0^{c_0} \cdot T_2^{c_0} \\
B^x \cdot C^x \cdot W^v & = R_3 \cdot S_1^{c_0} \cdot T_3^{c_0} \\
B^x \cdot C^x \cdot W^v & = R_3 \cdot S_1^{c_0} \cdot T_3^{c_0} \\
g_2^z & = Q_0 \cdot T_4^{c_0} \\
e(W, T_4)^y & = S_2 \cdot T_5^{c_0} \\
e(W, T_4)^y & = S_2 \cdot T_5^{c_0} \\
g_2^y & = S_0 \cdot T_4^{c_0} \end{align*}
\]

This concludes the proof.

Honest verifier zero knowledge: We show how to construct a simulator \(\bar{S}im\) that outputs a valid transcript for \(\Pi_1\) from the same distribution as a real \(\Pi_1\) transaction. \(\bar{S}im\) picks \(\bar{x} \in Z_p^*\), \(\bar{y} \in Z_p^*\), \(\bar{z} \in Z_p^*\), \(\epsilon \in Z_p^*\) and \(S_1 \in \mathbb{G}_1\). It then computes the following values:

\[
S_0 = g_1^y \cdot \epsilon; \\
S_2 = e(W, T_4)^y / T_5^{c_0}; \\
R_0 = g_2^z / \epsilon^2; \\
R_1 = A^r / T_1^{c_0}; \\
R_2 = B^s \cdot g_1^y / S_0^{c_0} \cdot T_2^{c_0}; \\
R_3 = C^t \cdot W^v / S_1^{c_0} \cdot T_3^{c_0}; \\
Q_0 = g_2^z / T_4^{c_0}
\]

Then the transcript \((\bar{R}_0, \bar{R}_1, \bar{R}_2, \bar{R}_3, \bar{S}_0, \bar{S}_1, \bar{S}_2, \bar{Q}_0, \epsilon, (\bar{x}, \bar{y}, \bar{z}))\) is valid for \(\Pi_1\). Moreover, since \(\bar{x}, \bar{y}, \bar{z}\) and \(\epsilon\) come from a uniform distribution, then \(\bar{S}im\) outputs valid transcripts from the same distribution as real \(\Pi_1\) transactions, it concludes the proof. □