FORCING OUT A CONFESSION
Threshold Discernible Ring Signatures

Swarun Kumar, Shivank Agrawal
Indian Institute of Technology, Madras, India

Ramarathnam Venkatesan, Satya Lokam
Microsoft Research, Bangalore, India

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Keywords: Ring signatures, Step out, Threshold discernible, Verifiable secret sharing.

Abstract: Ring signature schemes (Rivest et al., 2001) enable a signer to sign a message and remain hidden within an arbitrary group A of n people, called a ring. The signer may choose this ring arbitrarily without any setup procedure or the consent of anyone in A. Among several variations of the notion, *step out ring signatures* introduced in (Klonowski et al., 2008) address the issue of a ring member proving that she is not the original signer of a message, in case of dispute. First we show that the scheme in (Klonowski et al., 2008) has several flaws and design a correct scheme and prove formally the security of the same. Then we use the basic constructs of our scheme to design a protocol for a new problem, which we refer to as *threshold discernible ring signatures*. In threshold discernible ring signatures, a group B of t members can co-operate to identify the original signer of a ring signature that involved a group A of n alleged signers, where $B \subset A$ and $n > t$. This is the first time that this problem is considered in the literature and we formally prove the security of our novel scheme in the random oracle model.

1 INTRODUCTION

Ring signatures, introduced in (Rivest et al., 2001), allow a signer to sign a message on behalf of an arbitrary group A of n people, called the ring. The signer may hide behind the arbitrarily chosen ring A, without any setup procedure or the consent of the other members of A. Such signatures have been expanded to various applications: deniable ring authentication (Naor, 2002; Susilo and Mu, 2004), linkable ring signature schemes that allow one to link signatures signed by the same person, short versions of linkable ring signature (Tsang and Wei, 2005; Au et al., 2006). Furthermore, identity based ring signature schemes, which allow ring construction across different identity-based master domains (Cheng et al., 2004; Awasthi and Lal, 2005; Savola, 2006) and confessible threshold ring signature (Chen et al., 2006), where the actual signer can prove that she has created the signature, have also been proposed in literature.

Even though the original intent was to keep the real signer anonymous, in the event of a dispute, a member of the ring A may want to prove that she was not the actual signer of a particular message. A new variant called *step out* ring signature was introduced in (Klonowski et al., 2008); here the real signer can prove that she created the signature, while any one else in the ring can prove that she is not the original signer. Their proposal was an intermediate solution between the classical ring and group signatures, and can be used for instance in e-auction schemes, and this is the only scheme present in the literature. However, our attack presented here shows that their scheme allows a third party, who is not a member of the ring, to forge a signature on behalf of the ring. In another scenario, we break the anonymity of the signer of a ring signature. Hence, till date, there is no correct and provably secure scheme available for step

Kumar S., Agrawal S., Venkatesan R., Lokam S. and Pandu Rangan C. (2010).
FORCING OUT A CONFESSION - Threshold Discernible Ring Signatures.
In *Proceedings of the International Conference on Security and Cryptography*, pages 379-388
DOI: 10.5220/0002984203790388

out ring signatures.

Exposing the identity of the original signer of a ring signature may arise in several other contexts as well. Suppose a petitioner wishes to send a complaint regarding certain government officials on behalf of several people, say the residents of her locality. The signer wishes to remain anonymous in order to prevent harassment from the concerned officials. However, any resident who disagrees with the complaint must have the right to prove that she is not the petitioner. At the same time, a sufficiently large threshold of the residents should be able to discover the identity of the petitioner, in case the complaint was malicious.

Consider a joint bank account scenario, where n people share a single account. Any account holder among these n people is authorized to sign and transact with the bank. The bank will only know that someone among these n people has signed, but will not know the exact identity of the signer. Hence the situation cannot afford a centralized manager. Now, in case of fraud by any one of the n members, any threshold of t people among the n members can cooperate and identify the fraudulent person.

Our Contributions. We perform cryptanalysis on the step out ring signature scheme (Klonowski et al., 2008) and identify defects in unforgeability and anonymity. We additionally provide appropriate modifications in order to present a provably secure step out ring signature scheme under the random oracle model.

We introduce the concept of threshold discernible ring signatures, where a threshold of t signers are together capable of finding the identity of the original signer. This may be applied, for example, to situations where a message has been maliciously signed on behalf of a ring of signers and a majority (or a threshold t) of the ring members decide to unmask the original signer of the message. We shall use the basic constructs of our modified step out ring signature scheme to produce a threshold discernible ring signature scheme.

2 PRELIMINARIES

We shall consider rings with n members, denoted by $\mathcal{U}_1, \cdots, \mathcal{U}_n$. Let p, q be large primes ($p, q >> n$), $q|p-1$, and $G =< g >$ be an order q cyclic subgroup of $\mathbb{Z}_p^*$. For the sake of simplicity we shall skip "mod p" if it follows from the context. We assume that user $\mathcal{U}_i$ holds a private key x_i; the corresponding public key is $y_i = g^{x_i}$. The key y_i is publicly available. $\mathcal{H}$ denotes a secure hash function $\{0,1\}^* \rightarrow \{0,1\}^k$, where k is a fixed constant. We assume that the following assumptions are fulfilled in G:

Definition 1 - Decisional Diffie-Hellman Assumption. Let G be a cyclic group generated by g of order q. Let $\mathcal{A}^{DDH}$ be an algorithm that has to distinguish $c_0 = (g, g^a, g^b, g^{ab})$ from $c_1 = (g, g^a, g^b, g^c)$ for randomly chosen $a, b, c \in Z_q$. Let $\mathbf{Adv}_{\mathcal{A}}^{ddh} = |Pr[\mathcal{A}(c_1) = 1] - Pr[\mathcal{A}(c_0) = 1]|$ be called the advantage of $\mathcal{A}$ in breaking the DDH problem.

The DDH assumption holds for G, if advantage $\mathbf{Adv}_{\mathcal{A}}^{ddh}$ is negligible for each probabilistic polynomial-time algorithm $\mathcal{A}$, i.e. $\mathbf{Adv}_{\mathcal{A}}^{ddh} < \varepsilon_{ddh}$ where ε_{ddh} is negligible.

Definition 2 - Discrete Logarithm (DL) Assumption. Let G be a cyclic group generated by g of order q. Let $\mathcal{A}$ be an algorithm such that on input g^a, where $a \in Z_q$, $\mathcal{A}$ should output a. Let $\mathbf{Succ}_{\mathcal{A}}^{dl} = Pr[\mathcal{A}(g^a) = a]$ be called the success of $\mathcal{A}$ in breaking the DL problem.

The DL assumption holds in G, if for each probabilistic polynomial-time algorithm $\mathcal{A}$, success $\mathbf{Succ}_{\mathcal{A}}^{dl}$ is negligible, i.e. $\mathbf{Succ}_{\mathcal{A}}^{dl} < \varepsilon_{dl}$ where ε_{dl} is negligible.

2.1 SKDL Proof of Knowledge

The SKDL proof of knowledge is a signature of knowledge of discrete logarithms defined in (Camenisch, 1997). It is based on the Schnorr signature scheme (Schnorr, 1991). This signature proves the knowledge of $x : y = g^x$ in the context of a message m. We explain the construction and verification below.

SKDL Construction. The construction SKDL(g, y, m) is described below. It is executed by the prover who possesses $x : y = g^x$. Note that g is a generator of the group G.

1. Pick $r \leftarrow_R \mathbb{Z}_q^*$.
2. Calculate $c = \mathcal{H}(g||y||g^r||m)$.
3. Calculate
$$s = r - cx \tag{1}$$

The procedure returns the values (c, s).

SKDL Verification. The verification procedure $\mathcal{V}_{SKDL}(g, y, m)$ is executed by the verifier and checks if:
$$c \stackrel{?}{=} \mathcal{H}(g||y||g^s y^c||m)$$
This proves that the prover is aware of discrete logarithm $x = log_g(y)$ without actually revealing x.

2.2 SEQDL Proof of Knowledge

The step-out ring signature scheme in (Klonowski et al., 2008) is based on a *signature of knowledge of equality of discrete logarithms* (SEQDL). Let $g, \hat{g}, \hat{y}_w \in G$ and tuples $(y_1, \cdots, y_n), (w_1, \cdots, w_n) \in G^n$. SEQDL allows a prover to prove in zero-knowledge that $log_{\hat{g}}\hat{y}_w = log_g(y_j w_j)$ for some index j, with j not revealed to the verifier.

Preliminaries. Recall that G is an order q cyclic subgroup of $\mathbb{Z}_p^*$ with g as its generator. Let $\mathcal{U}_j$ be a prover who has the following information:

- $Y = (y_1, \cdots, y_n) \in G^n$
- For a specific index j, $\mathcal{U}_j$ knows $x_j : y_j = g^{x_j}$. Note that $\mathcal{U}_j$ is not aware of the discrete logarithms of $y_i \in Y : i \neq j$.
- $W = (w_1, \cdots, w_n) \in G^n$ and $(r_1, \cdots, r_n) \in \mathbb{Z}_q^n$, where $w_i = g^{r_i}$ for all $i = 1, \cdots, n$. Note that unless $\mathcal{U}_j$ is the signer, she is not aware of the discrete logarithms of $w_i \in W : i \neq j$.
- $\hat{g} \in G$, which is randomly chosen by the signer and $\hat{y} = \hat{g}^{x_j + r_j}$.

Using these values, $\mathcal{U}_j$ wishes to convince the verifier that the discrete logarithms $log_{\hat{g}}\hat{y}_w$ and $log_g(y_j w_j)$ are equal, with the index j not revealed to the verifier. $\mathcal{U}_j$ achieves this by executing the SEQDL construction algorithm and passing the outputs to the SEQDL verification algorithm. The details are given below:

SEQDL Construction. The SEQDL construction algorithm, run by the $\mathcal{U}_j$, is SEQDL$(\hat{g}, g, x_j, r_j, \hat{y}_w, Y, W, m)$. Typically, the vector W is chosen by the signer. However, W may be created and used in different ways provide three different mechanisms for stepping out, as discussed in section 4. The construction of SEQDL is as follows:

1. Pick random elements $r \in \mathbb{Z}_q$ and $c_i, s_i \in \mathbb{Z}_q$, for $i \in \{1, \cdots, n\} \backslash \{j\}$.
2. For all $i \in \{1, \cdots, n\} \backslash \{j\}$, user U_j computes:
$$t_i \leftarrow \hat{g}^{s_i} \hat{y}_w^{c_i}, u_i \leftarrow g^{s_i}(y_i w_i)^{c_i}, t_j \leftarrow \hat{g}^r, u_j \leftarrow g^r \quad (2)$$
3. We denote $\overline{Y} = y_1 || \cdots || y_n$, $\overline{W} = w_1 || \cdots || w_n$
4. Compute:
$$c_j \leftarrow \mathcal{H}(\hat{g} || g || \hat{y}_w || \overline{Y} || \overline{W} || t_1 || u_1 || \cdots || t_n || u_n || m) - \sum_{i<n, i \neq j} c_i \quad (3)$$
$$s_j \leftarrow r - (x_j + r_j) c_j \bmod q \quad (4)$$

The algorithm finally returns (C, S) where $C = (c_1, \cdots, c_n)$, $S = (s_1, \cdots, s_n)$.

SEQDL Verification. Given a signature SEQDL$(\hat{g}, g, x_j, r_j, \hat{y}_w, Y, W, m) = (C, S)$, with parameters $\hat{g}$, g, $\hat{y}_w$, Y, W, and a message m, the verification algorithm $\mathcal{V}_{\text{SEQDL}}(\hat{g}, g, \hat{y}_w, Y, W, C, S, m)$, run by the verifier, checks if:

$$\sum_{i=1}^{n} c_i \stackrel{?}{=} \mathcal{H}(\hat{g} || g || \hat{y}_w || \overline{Y} || \overline{W} || \hat{g}^{s_1} \hat{y}_w^{c_1} || g^{s_1}(y_1 w_1)^{c_1} || \cdots || \hat{g}^{s_n} \hat{y}_w^{c_n} || g^{s_n}(y_n w_n)^{c_n} || m) \quad (5)$$

The verifier returns 1 if the above condition succeeds, 0 otherwise. When verification returns 1, the verifier is convinced of the equality of discrete logarithms $log_{\hat{g}}\hat{y}_w$ and $log_g(y_j w_j)$ with the index $j \in \{1, \cdots, n\}$ unknown to the verifier.

3 STEP OUT RING SIGNATURES (SRS)

3.1 Scheme Outline

Let us assume that $\mathcal{U}_j$ is the real signer and $\mathcal{U}_1, \cdots, \mathcal{U}_n$ are all ring members. Let the private and public key of user $\mathcal{U}_i$ be x_i and $y_i = g^{x_i}$ respectively. For Step-out Ring Signatures (SRS) we have the following procedures:

Signing Procedure. $\mathcal{S}_{SRS}(g, \hat{g}, x_j, Y, m)$ is a randomized algorithm that takes generator g and a random element $\hat{g} \in < g >$, $\hat{g} \neq 1$, the secret key x_j, the set of public keys $y_1, \cdots, y_n \subset < g >$ and a message m. It returns a signature σ.

Verification Procedure. $\mathcal{V}_{SRS}(\sigma, m)$ is a deterministic algorithm that takes a message m, and a signature σ for m. It returns a bit: 1 or 0 to indicate whether σ is valid, i.e. someone having a public key in a set Y indicated by σ has signed m.

Confession Procedure. Let σ be a step-out ring signature on m produced by member $\mathcal{U}_j$ of the ring. In the confession procedure, $\mathcal{U}_j$ proves that she is indeed the original signer of m and produced σ. Towards this, $\mathcal{U}_j$ produces a confession record σ', which is yet another signature by $\mathcal{U}_j$ on m. The verifier runs $\mathcal{C}_{SRS}(\sigma, \sigma', y_j, m)$, a deterministic algorithm which takes as input σ, σ', m and the public key y_j of user $\mathcal{U}_j$, and returns either a bit 1 to confirm that U_j has created σ or a bit 0 otherwise.

Step-out Procedure. Let σ be a step-out ring signature on m produced by member $\mathcal{U}_j$ of the ring. During step-out, a ring member $\mathcal{U}_i, i \neq j$ proves that she not the original signer of m. Here, $\mathcal{U}_i$ produces

step-out records σ'', σ''', which are SRS signatures for the message $\tilde{m}$ ="I have not signed m". The verifier runs $\mathcal{D}_{SRS}(\sigma, m, \sigma'', \sigma''', y_i, \tilde{m})$, a deterministic algorithm which takes as input σ, σ'', σ''', m and the public key y_i of user $\mathcal{U}_i$, and returns either a bit 1 to confirm that U_i has not created σ or a bit 0 otherwise.

3.2 Step Out Ring Signatures

We recall the signing and verification procedures of the step out ring signature scheme in (Klonowski et al., 2008).

3.2.1 Signing Algorithm

The signing algorithm is run by user $\mathcal{U}_j$ with private key x_j to produce a ring signature corresponding to n users with public keys $Y = (y_1, \cdots, y_n)$. Note that parameter $\hat{g} \in G$ is randomly chosen by the signer.

Algorithm $\mathcal{S}_{SRS}(g, \hat{g}, x_j, Y, m)$
repeat
$r_1, \cdots, r_n \leftarrow_R \mathbb{Z}_q^*$
$w_i \leftarrow g^{r_i}$ for each $i = 1, \cdots, n$
until $(y_i w_i \neq y_j w_j$ for each $i \neq j)$
$\hat{w} \leftarrow \hat{g}^{r_j}, \hat{y} \leftarrow \hat{g}^{x_j}, \hat{y}_w \leftarrow \hat{y}\hat{w}$
$(C, S) \leftarrow \text{SEQDL}(\hat{g}, g, x_j, r_j, \hat{y}_w, Y, W, m)$
$Y \leftarrow y_1, \cdots, y_n$
$W \leftarrow w_1, \cdots, w_n$
$\sigma \leftarrow (g, \hat{g}, \hat{y}, \hat{w}, Y, W, C, S)$
return (m, σ)

3.2.2 Verification Algorithm

This algorithm is run by a verifier using only public information. Algorithm $\mathcal{V}_{\text{SEQDL}}$ verifies the SEQDL proof of knowledge output by the signer.

Algorithm $\mathcal{V}_{SRS}(\sigma, m)$
$\hat{y}_w \leftarrow \hat{y}\hat{w}$
$d \leftarrow \mathcal{V}_{\text{SEQDL}}(\hat{g}, g, \hat{y}_w, Y, W, C, S, m)$
if $d = 1$
then return 1
else return 0

3.2.3 Scenarios for r_i

Three different ways of using parameter r_i, targeting three different applications, are suggested in (*Klonowskiet al.*, 2008):

1. The numbers r_i are created by the signer at random. They are kept secret unless the signer enables a member of a ring to step out.
2. The numbers r_i are given together with the signature. In this case the ring participants can immediately step out.
3. $\mathcal{U}_i$ generates r_i herself and publishes w_i. Moreover, each w_i can be a kind of time stamp - a signature generated with w_i has to be created no earlier than at the time of creating w_i.

4 CRYPTANALYSIS OF SCHEME

We have found weaknesses in the paper in the case of scenario (1) and scenario (2) above. We explain these below:

4.1 Forgery in Scenario 1

Under scenario 1, we show that it is easy for anyone, even without the knowledge of any of the ring members' secret keys, to produce $\hat{y}_w, w_j$ for some j such that $log_{\hat{g}}\hat{y}_w = \log_g(y_j w_j)$. We explain an algorithm $\mathcal{F}_{SRS}$ which forges a signature of (Klonowski et al., 2008) in this manner below:

4.1.1 Forger Algorithm

Algorithm $\mathcal{F}_{SRS}(g, \hat{g}, Y, m)$
repeat
$r_i \leftarrow_R \mathbb{Z}_q^*$ for each $i \in \{1, \cdots, n\} \backslash \{j\}$
$w_i \leftarrow g^{r_i}$ for each $i \in \{1, \cdots, n\} \backslash \{j\}$
$\alpha \leftarrow_R \mathbb{Z}_q^*$
$w_j \leftarrow g^{\alpha}/y_j$
until $(y_i w_i \neq y_k w_k$ for each $i \neq k)$
$\beta \leftarrow_R \mathbb{Z}_q^*$
$\hat{w} \leftarrow \hat{g}^{\beta}, \hat{y} \leftarrow \hat{g}^{\alpha-\beta}, \hat{y}_w \leftarrow \hat{y}\hat{w}$
$(C, S) \leftarrow \text{SEQDL}(\hat{g}, g, \alpha - \beta, \beta, \hat{y}_w, Y, W, m)$
$Y \leftarrow y_1, \cdots, y_n$
$W \leftarrow w_1, \cdots, w_n$
$\sigma \leftarrow (g, \hat{g}, \hat{y}, \hat{w}, Y, W, C, S)$
return (m, σ)

4.1.2 Validity

We will show that the signature produced by $\mathcal{F}_{SRS}$ verifies successfully. Note that the verification algorithm $\mathcal{V}_{SRS}(\sigma, m)$ will in turn call $\mathcal{V}_{\text{SEQDL}}$ $(\hat{g}, g, \hat{y}_w = \hat{g}^{\alpha}, Y, W, C, S, m)$. By construction in equation (2) and (3), verification equation (5) holds provided: $g^{s_j}(y_j w_j)^{c_j} = g^{s_j + c_j \alpha} = g^r$ and $\hat{g}^{s_j}\hat{y}_w^{c_j} = \hat{g}^{s_j + c_j \alpha} = \hat{g}^r$. However, these hold, since by construction in (4), $r = s_j + c_j \alpha$. Hence the forged signature is considered valid.

4.1.3 Salient Features

The above algorithm clearly does not use private information x_j to forge a ring signature. If this were performed by the k^{th} ring member, she can step out using the value r_k. An adversary can also allow every ring member other than the j^{th} one to step out by releasing the values r_i for each $i \in \{1,\cdots,n\}\backslash\{j\}$. In fact, it can be shown that the forged sign is indistinguishable from a signature by the j^{th} ring member in polynomial time. In the next section we will demonstrate how to fix this break. We will provide a corrected scheme and unforgeability proof in the following sections.

4.2 Break of Anonymity in Scenario 2

The anonymity of the signer can be broken in the second scenario using the parameter $\hat{w}$. Since the parameters r_i are released together with the signature, a distinguisher simply tests if $\hat{g}^{r_i} \stackrel{?}{=} \hat{w}$ for each $i = 1,\cdots,n$. According to the protocol, this will only hold for the signer j, thus revealing the identity of the signer.

5 MODIFIED STEP OUT RING SIGNATURES

We will explain in this section how we can modify the step out ring signature scheme to restore unforgeability and anonymity.

5.0.1 Providing Unforgeability

The signer generates a random value r_j, but uses only the value $x_j + r_j$ in equation (4) of the SEQDL protocol for generating the components, (c,s). However, there is no proof of knowledge of r_j (or the other r_i's) insisted by the verification algorithm. Hence a forger can generate the value '$x_j + r_j$' in an arbitrary manner without even knowing or proving that she knows x_j and r_j individually. This is exactly what we did in our forgery algorithm by *reverse engineering* the $(x_j + r_j)$ values. In fact, in our forging algorithm the values α and β are chosen in such a way that when $\alpha - \beta$ and β are used as parameters for SEQDL, the algorithm produces the same value that SEQDL would have produced with x_j and r_j. Hence, to fix the above problem, we add SKDL's for w_i's and verify them during verification.

5.0.2 Providing Anonymity

The anonymity can be broken if the parameter $\hat{w}$ is known and the signature σ output by the signer contains $\hat{w}$ explicitly as a part of it. Notice that σ contains both $\hat{w}$ and $\hat{y}$ but the verification algorithm needs only the product $\hat{w}\hat{y}$. Hence, it is sufficient to provide only the product value $\hat{w}\hat{y}$ as a component of σ instead of providing $\hat{w}$ and $\hat{y}$ as separate components. As one can not compute $\hat{w}$ from the product $\hat{w}\hat{y}$, this modification prevents one from breaking the anonymity. In fact, we formally prove the same.

5.1 Modified SRS Scheme

The modified SRS scheme overcomes the flaws of the step out ring signature scheme in (Klonowski et al., 2008). This uses the SKDL which is a zero knowledge proof of discrete logarithm.

5.1.1 Modified Signing Algorithm

The algorithm is run by user $\mathcal{U}_j$ with private key x_j to produce a ring signature corresponding to n users with public keys $Y = (y_1,\cdots,y_n)$. Note that parameter $\hat{g} \in G$ is randomly chosen by the signer.

Algorithm $\mathcal{S}_{MSRS}(g,\hat{g},x_j,y_1,\cdots,y_n,m)$
  repeat
    $r_1,\cdots,r_n \leftarrow_R \mathbb{Z}_q^*$
    $w_i \leftarrow g^{r_i}$ for each $i = 1,\cdots,n$
  until $(y_i w_i \neq y_j w_j$ for each $i \neq j)$
  $\hat{y}_w \leftarrow \hat{g}^{x_j + r_j}$
  $(c_1,\cdots,c_n,s_1,\cdots,s_n) \leftarrow$ SEQDL$(\hat{g},g,x_j,r_j,\hat{y}_w$
    $,y_1,\cdots,y_n,w_1,\cdots,w_n,m)$
  $Y \leftarrow y_1,\cdots,y_n$
  $W \leftarrow w_1,\cdots,w_n$
  $\sigma = (g,\hat{g},\hat{y}_w,Y,W,c_1,\cdots,c_n,s_1,\cdots,s_n,$
    $\{\text{SKDL}(g,w_i,m), i = 1,\cdots,n\})$
  return(m,σ)

5.1.2 Modified Verification Algorithm

This algorithm is run by a verifier using only public information. Algorithm $\mathcal{V}_{\text{SKDL}}$ is used to verify the SKDL proofs of knowledge output by the signer.

Algorithm $\mathcal{V}_{MSRS}(\sigma,m)$
  if$(\mathcal{V}_{SKDL}(g,w_i,m) = 0,$
    for any $i = 1,\cdots,n)$
    then return 0
  $d \leftarrow \mathcal{V}_{\text{SEQDL}}(\hat{g},g,\hat{y}_w,y_1,\cdots,y_n,w_1,\cdots,w_n,$
    $c_1,\cdots,c_n,s_1,\cdots,s_n,m)$
  if $d = 1$
    then return 1
    else return 0

5.1.3 Modified Confession Algorithm

We denote $Y' = (y_1', \cdots, y_n')$, $Y'' = (y_1'', \cdots, y_n'')$, $Y''' = (y_1''', \cdots, y_n''')$. The confession record $\sigma' = (g, \hat{g}, \hat{y}, \hat{w}, Y', W, \text{SEQDL}(\hat{g}, g, x_i, r_i, \hat{y}.\hat{w}, Y', W, m))$ is a new signature with the same parameters $g, \hat{g}, W$ as in σ and some new set of potential signers $Y' : Y \cap Y' = \{y_j\}$, where y_j stands at the same position in both sequences.

The confession algorithm verifies whether a member of the ring $\mathcal{U}_j$ has generated the ring signature σ by obtaining σ' from her as shown below. Note that the verifier verifies σ' using $\mathcal{V}_{SRS}$ because the SKDL's corresponding to W have already been verified in the verification of σ.

Algorithm $\mathcal{C}_{MSRS}(\sigma, \sigma', y_j, m)$
  if(the same $g, \hat{g}, \hat{y}, \hat{w}, W$ were used in σ and σ') then
    $d_1 \leftarrow \mathcal{V}_{MSRS}(\sigma, m), d_2 \leftarrow \mathcal{V}_{SRS}(\sigma', m)$
    if($d_1 = d_2 = 1$ and $\{y_j\} = Y \cap Y'$ and y_j stands on position j in Y') then
      return 1 else return 0
  else return 0

5.1.4 Modified Step-out Algorithm

We define the step-out records σ'', σ''' below:

- $\sigma'' = (g, \hat{g}, \hat{y}'', \hat{w}'', Y'', W, \text{SEQDL}(\hat{g}, g, x_i, r_i, \hat{y}''.\hat{w}'', Y'', W, \tilde{m}))$ - a SRS signature with the same parameters $g, \hat{g}, W$ as in σ and $\hat{y}'' = \hat{g}^{x_i}$, $\hat{w}'' = \hat{g}^{r_i}$, some new set of potential signers Y'', for the control message $\tilde{m}$ = "I have not signed m".
- $\sigma''' = (g, \hat{g}, \hat{y}'', \hat{w}'', Y''', W, \text{SEQDL}(\hat{g}, g, x_i, r_i, \hat{y}''.\hat{w}'', Y''', W, \tilde{m}))$ - a SRS signature for the same control message $\tilde{m}$ with the same $g, \hat{g}, \hat{w}'', W$ and Y''' such that $Y'' \cap Y''' = \{y_i\}$ and y_i stands on the same position in Y'' and Y'''. Moreover, $y_{i_1}''w_{i_1} \neq y_{i_2}'''w_{i_2}$ for $i_1 \neq i_2$

The step-out algorithm verifies whether a member of the ring $\mathcal{U}_i$ has not generated the ring signature σ by obtaining (σ'', σ''') from her as shown below. Note that the verifier verifies σ'' and σ''' using $\mathcal{V}_{SRS}$ because the SKDL's corresponding to W have already been verified in the verification of σ.

Algorithm $\mathcal{D}_{MSRS}(\sigma, m, \sigma'', \sigma''', y_i, \tilde{m})$
  if(the same $g, \hat{g}, W$ were used in $\sigma, \sigma'', \sigma'''$ and the same $\hat{y}'', \hat{w}''$ were used in σ'', σ''') then
    $d_1 \leftarrow \mathcal{V}_{MSRS}(\sigma, m), d_2 \leftarrow \mathcal{V}_{SRS}(\sigma'', \tilde{m}),$
    $d_3 \leftarrow \mathcal{V}_{SRS}(\sigma''', \tilde{m})$
    if($d_1 = d_2 = d_3 = 1$ and $\{y_i\} = Y'' \cap Y'''$, and y_i stands at the same position in Y'' and Y''', and $\hat{y}\hat{w} \neq \hat{y}''\hat{w}''$) then
      return 1 else return 0
  else return 0

6 ANALYSIS

6.1 Unforgeability

Informally, forking lemma(Pointcheval, 2005) for adaptive chosen message attacks states that if an algorithm $\mathcal{A}$ can with non-negligible probability ε, produce a valid signature $(m, \sigma_1, h, \sigma_2)$ without knowing the secret key, then, a replay of the attacker $\mathcal{A}$ may output two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m, \sigma_1, h', \sigma_2')$ such that $h \neq h'$, within a bounded time and non-negligible probability. Forking lemma is applicable for modified step-out ring signatures. This can be proved similar to (Klonowski et al., 2008), the difference being the computation of SKDL's by the simulators. We state the lemma below.

Lemma 1. Modified SRS signatures can be simulated by a simulator, with oracle access to $\mathcal{H}$, under DDH assumption without knowing the corresponding secret signing key and with distribution probability indistinguishable from SRS signatures produced by a legitimate signer.

Now, we shall construct an adversary that can solve the DL problem by finding $x_i = log_g y_i$ for some i. Note that the y_i's are supplied to the forger as input. Hence a DL solver attempting to find $log_g X$ can do so by setting $y_t = X$ for some t. With success probability $1/n$, this is the index of the signer whose signature the forger generates.

6.1.1 Construction of DL Solver

We now apply forking lemma in the chosen message attack scenario (section 2.1). The signature σ is written as (σ_1, h, σ_2) where:

$\sigma_1 =$ $(\hat{g}, \hat{y}_w, W, u_1, \cdots, u_n, t_1, \cdots, t_n)$, where u_i, t_i are constructed like in (2)

$h =$ $(H(\hat{g}||g||\hat{y}_w||\overline{Y}||\overline{W}||u_1||t_1||...||u_n||t_n||m), \{H(g||w_i||g^{\tilde{s}_i}w_i^{\tilde{c}_i}||m), i = 1, \cdots, n\}$

$\sigma_2 =$ $(C, S, \tilde{c}_1, \tilde{s}_1, \cdots, \tilde{c}_n, \tilde{s}_n)$

After acquiring two valid signatures (σ_1, h, σ_2) and $(\sigma_1, h', \sigma_2')$, such that $h \neq h'$ and $\sigma_2 \neq \sigma_2'$, the DL solver can compute the $x_i = log_g(y_i)$ corresponding to the signer whose signature the forger generated.

The solver first computes $\alpha_i = x_i + r_i = (s_i' - s_i)/(c_i - c_i')$ for all $i = 1, \cdots, n$ where $c_i \neq c_i'$, which holds due to equation 4 in SEQDL construction. It then computes $r_i' = (\tilde{s}_i' - \tilde{s}_i)/(\tilde{c}_i - \tilde{c}_i')$ for all $i = 1, \cdots, n$ where $\tilde{c}_i \neq \tilde{c}_i'$, which is evident from equation (1) in SKDL construction. Finally, it computes $x_i' = \alpha_i - r_i'$ for all obtained values of α_i and r_i'. Clearly, if the forger produced a signature by the user with public key y_j, then solver has obtained

$x'_j : g^{x'_j} = y_j$.

Hence the solver has the solution to the DL problem $x' = log_g X$ provided $j = t$. The probability that this happens is $1/n$. Since we assume that the DL assumption holds, the above algorithm must have negligible probability of success, therefore the forger has negligible success probability too.

6.2 Anonymity

The anonymity argument in (Klonowski et al., 2008) can be readily extended to the proof of anonymity of the modified scheme. As the r_i's are chosen randomly, the SKDLs reveal no additional information about the signer. Also, the proof of anonymity in (Klonowski et al., 2008) assumes that the only distinguishing property of two signature tuples of the form $\sigma = (m, g, \hat{g}, \hat{w}, y_1, y_2, w_1, w_2, c_1, c_2, s_1, s_2)$ by two different signers 1 and 2, is that in the former, $log_g(y_1 w_1) = log_{\hat{g}}(\hat{y}\hat{w})$ and in the latter, $log_g(y_2 w_2) = log_{\hat{g}}(\hat{y}\hat{w})$. However, the fact that the adversary, in scenario 2, may use $r_i = log_{\hat{g}}(\hat{w})$, when r_i is released along with the signature was not considered. This can be rectified when the product $\hat{y}_w = \hat{y}\hat{w}$ is released with the signature instead of the individual values $\hat{y}$, $\hat{w}$.

6.3 Security of Confession and Step Out

We will prove the following lemmas in order to show the security of confession and step-out protocols in our modified step-out ring signature scheme.

Lemma 2. A confession has a positive outcome only if performed by the original signer of a modified step-out ring signature according to protocol.

Proof. Since $\mathcal{V}_{\text{SEQDL}}(\hat{g}, g, \hat{y}_w, Y, W, C, S, m) = 1$, there exists α such that $g^\alpha \in \{y_1 w_1, \cdots, y_n w_n\}$ and $\hat{g}^\alpha = \hat{y}\hat{w}$. Moreover, if σ' is constructed appropriately and $\mathcal{V}_{\text{MSRS}}(\sigma', m) = 1$, then $g^\alpha \in \{y'_1 w_1, \cdots, y'_n w_n\}$ as well. So $g^\alpha \in \{y_1 w_1, \cdots, y_n w_n\} \cap \{y'_1 w_1, \cdots, y_n w'_n\}$. Since $\{y1, ..., yn\} \cap \{y'_1, ..., y'_n\} = \{y_j\}$, and $y_{i_1} w_{i_1} \neq y'_{i_2} w_{i_2}$ for $i_1 \neq i_2$, we know that $g^\alpha = y_j w_j$, so in this case user $\mathcal{U}_j$ was a creator of σ and $C_{MSRS}(\sigma, \sigma', y_j, m) = 1$.

Lemma 3. A step-out has a positive outcome only if performed by a ring-member of a modified step-out ring signature, other than the original signer, according to protocol.

Proof. It is easy to see that the see that σ''' is a confession that a message $\tilde{m}$ has been signed as σ'' by the user $\mathcal{U}_i : y_i = Y'' \cap Y'''$. Clearly, this user is a member of the ring. We will show that the outcome of the step-out procedure performed by this user is positive. Let us assume that $\mathcal{D}_{\text{MSRS}}(\sigma, m, \sigma, \sigma, y_i, \tilde{m}) = 0$. This happens if $\hat{y}\hat{w} = \hat{y}''\hat{w}''$. As in the proof of Lemma 1, we can see that the signatures σ'' and σ''' guarantee that there exists α' such that $g^{\alpha'} = y_i w_i$ and $\hat{g}^{\alpha'} = \hat{y}''\hat{w}''$. So $\alpha = log_g(y_i w_i) = log_{\hat{g}}(\hat{y}''\hat{w}'') = log_{\hat{g}}(\hat{y}\hat{w}) = log_g(y_j w_j)$, where $\mathcal{U}_j$ is the signer of σ. We have got that $y_i w_i = y_j w_j$, but this contradicts the assumption about generating secrets r_i and computing w_i during the signing procedure, provided $i \neq j$.
Let us consider the case when an actual signer attempts to step-out. When performing the step-out procedure and generating signatures σ' and σ'', the user $\mathcal{U}_j$ has to generate $y''w'' = g^{x_j + r_j}$. However, this product is the same as in σ, so this would lead to a failure of the test of the step-out procedure.

7 THRESHOLD DISCERNIBLE RING SIGNATURES

Threshold discernible ring signatures are ring signatures where a threshold of t signers are together capable of finding the identity of the original signer. This may be applied for example to situations where a message has been maliciously signed on behalf of a ring of signers and a majority (or a threshold t) of the signers decide to unmask the original signer of the message.

We extend the modified step out ring signature scheme from section 6.3 to allow threshold discernibility. The signing algorithm additionally outputs a set of verifiably encrypted shares of the secret $l = log_g(\hat{g})$. This can be done using verifiable sharing of discrete logarithms (Stadler, 1996) and verifiable encryption of discrete logarithms (Stadler, 1996; Camenisch and Shoup, 2003). Once l is gathered by any set of t ring members, the original signer is easily found by inspecting for which index i of the ring members, the equation $(y_i w_i)^l = \hat{y}_w$ holds. This is the index of the original signer.

7.1 Preliminaries

We assume the same settings and complexity assumptions as the SRS signature scheme as in section 2. The algorithm uses a verifiable encryption scheme (Stadler, 1996; Camenisch and Shoup, 2003; Camenisch and Damgard, 2000). The notations used for this scheme are explained below. We also explain Shamir's secret sharing scheme (Rivest et al., 2001) which is used in the verifiable secret sharing of discrete logarithms (Stadler, 1996).

7.1.1 Verifiable Encryption

We denote verifiable encryption of a discrete logarithm $\alpha = log_g(\beta)$ under public key PK as $VE_{PK}(\alpha : \beta = g^{\alpha})$. This denotes the cipher-text created by the *Encrypt* algorithm. The encryption scheme has three algorithms namely:

1. $Encrypt(\alpha : \beta = g^{\alpha})$: Takes a message α, a public key PK and outputs cipher text $VE_{PK}(\alpha : \beta = g^{\alpha})$ where $g, \beta = g^{\alpha}$ are publicly known.
2. $Decrypt(VE_{PK}(\alpha : \beta = g^{\alpha}))$: Takes a cipher-text $VE_{PK}(\alpha : \beta = g^{\alpha})$ and obtains the original message α. This requires the secret key SK.
3. $Verify(VE_{PK}(\alpha : \beta = g^{\alpha}))$: Takes the cipher-text $VE_{PK}(\alpha : \beta = g^{\alpha})$ and verifies the zero knowledge proof that the cipher text indeed encrypts α such that $\beta = g^{\alpha}$.

7.1.2 Shamir's Secret Sharing Scheme

A (t, n) secret sharing scheme is a scheme where a secret d is shared among n users where only a coalition of size at least t can recover the secret. Such a scheme was proposed by Shamir (Rivest et al., 2001) and is explained below. A user $\mathcal{U}_i$ has a well known public parameter $\alpha_{u_i} \in \mathbb{Z}_q$.

Preliminaries. Let q be a large prime ($q >> n$), and $d \in \mathbb{Z}_q$ be the secret to be shared. There are $n \geq t$ users in total.

Share. (d) The dealer chooses a random polynomial $f(x) = d + \sum_{i=1}^{t-1} a_i x^i$, of degree $t-1$ from $\mathbb{Z}_q[x]$ where the constant term is set to d. The dealer then distributes the secret shares $s_i = f(\alpha_{\mathcal{U}_i})$, to the i^{th} user, for each $i = 1 \cdots n$.

Reconstruct. $((\alpha_{v_1}, s_1), \cdots, (\alpha_{v_{|S|}}, s_{|S|}))$ This process is a simple polynomial interpolation to compute $f(0) = d$. Suppose a coalition $S, |S| \geq t, S = \{v_1, \cdots, v_{|S|}\}$ wants to reconstruct the secret. They can compute the secret polynomial f(x) and the secret by Lagranges polynomial interpolation:

$$f(0) = \sum_{i \in S} y_i \lambda_{i0} \quad \text{,where } \lambda_{ij}^{S} = \prod_{j' \in S \setminus \{i\}} \frac{j - j'}{i - j'}$$

The additional requirement to Shamir's secret sharing our scheme requires is that the shared secrets are encrypted and these encrypted portions must still be verifiable.

7.2 Scheme Description

Outline. Let us assume that $\mathcal{U}_j$ is the real signer and $\mathcal{U}_1, \cdots, \mathcal{U}_n$ are all ring members. Let the private and public key of user $\mathcal{U}_i$ be x_i and $(y_i = g^{x_i}, \alpha_i)$ respectively, where $\alpha_i \in \mathbb{Z}_q$. For Threshold Discernible Ring Signatures (TDS) we have the following procedures:

Signing Procedure. $\mathcal{S}_{TDS}$ $(g, x_j, y_1, \cdots, y_n, \alpha_1, \cdots, \alpha_n, t, m)$ is an algorithm that takes generator g, the secret key x_j, the set of public keys $\{y_1, \cdots, y_n\} \subset < g >$, threshold t and a message m. It returns a threshold discernible signature σ.

Verification Procedure. $\mathcal{V}_{TDS}(m, \sigma)$ is an algorithm that takes a message m, and a signature σ for m. It returns a bit: 1 or 0 to indicate whether σ is valid, i.e., someone having a public key in a set Y indicated by σ has signed m, and whether it is indeed threshold discernible by t of the members of the ring.

Threshold Distinguisher Procedure. $\mathcal{T}_{TDS}(m, \sigma)$ is an algorithm that takes a message m, and a signature σ for m, and returns i, the index of the original signer among the public key sequence Y in the signature σ. The algorithm requires inputs by at least t signers among the n members of the ring indicated by σ.

7.3 Signing Algorithm

The signing algorithm verifiably encrypts n shares of the secret $l = log_g(\hat{g})$, along with the MSRS signature. It performs the sharing by encrypting the values of $t-1$ degree polynomial function $f(x) = l + \sum_{j=1}^{t-1} f_j x^j$, at n points viz. at $x = \alpha_1, \cdots, \alpha_n$.

Algorithm $\mathcal{S}_{TDS}(g, x_j, y_1, \cdots, y_n, \alpha_1, \cdots, \alpha_n, t, m)$
$f_1, f_2, \cdots, f_{t-1} \leftarrow_R \mathbb{Z}_q^*$
$F_i \leftarrow g^{f_i}, i = 1, \cdots, t-1$
$l \leftarrow_R \mathbb{Z}_q^* \setminus \{1\}$
$\hat{g} \leftarrow g^l$
$s_i \leftarrow l + \sum_{j=1}^{t-1} f_j \alpha_i^j,\ i = 1, \cdots, n$
$V_i \leftarrow VE_{y_i}(s_i : g^{s_i} = \hat{g} \prod_{j=1}^{t-1} F_j^{\alpha_i^j}),\ i = 1, \cdots, n$
$\sigma_1 \leftarrow \mathcal{S}_{MSRS}(g, \hat{g}, x_j, y_1, \cdots, y_n, m)$
$\sigma \leftarrow (\sigma_1, \{V_i : i = 1, \cdots, n\}, \{F_i : i = 1, \cdots, t-1\})$
return (m, σ)

7.4 Verification Algorithm

The verification algorithm verifies the MSRS signature as well as the verifiably encrypted shares of the secret l. The verification algorithm must check

whether t is an acceptable value based on the required policy. For instance, one may require that $t = \lceil \frac{n}{2} \rceil$.

Algorithm $\mathcal{V}_{TDS}(m, \sigma)$
  if (Verify$(VE_{y_i}(s_i : g^{s_i} = \hat{g} \prod_{j=1}^{t-1} F_j^{\alpha_i^j})) = 0$
    for any $i = 1, \cdots, n$)
    return 0
  return $\mathcal{V}_{MSRS}(m, \sigma)$

7.5 Threshold Distinguisher Algorithm

The threshold distinguisher algorithm requires that at least t of the signers in the ring share their respective s_i's. It is required that each of these s_i's are such that $S_i = \hat{g} \prod_{j=1}^{t-1} F_j^{y_i^j} = g^{s_i}$. Now, using Lagrange's interpolation formula, the function f, hence the value $f(0) = l$, can be computed. Once l is computed, the verifier checks for which value of i, the equation, $(y_i w_i)^l = \hat{y}_w$ holds. This i is the index of the original signer.

Algorithm $\mathcal{T}_{TDS}(m, \sigma)$
  if $(\mathcal{V}_{TDS}(m, \sigma) = 0)$
    then return $\perp$
  Obtain $s_i = Decrypt(VE_{y_i}(s_i : g^{s_i} = \hat{g} \prod_{j=1}^{t-1} F_j^{\alpha_i^j}))$
    from t signers w.l.o.g. $i = 1, \cdots, t$.
  $l \leftarrow Reconstruct((\alpha_0, s_0), \cdots, (\alpha_t, s_t))$
  for $i = 1$ to n
    if $((y_i w_i)^l = \hat{y}_w)$
      then return i
  return $\perp$

7.6 Security

In this section we define the security models for threshold discernible ring signatures. Due to lack of space, we provide sketches of the security proofs. Detailed proofs will be provided in the full version of this paper.

A threshold discernible ring signature (TDS) scheme must follow the following conditions:

Unforgeability. Unforgeability in threshold discernible ring signatures requires that no entity other than a member of the ring must be able to produce a ring signature with non-negligible advantage in polynomial time.

For security proof of unforgeability we formalize the attacks of a forger $\mathcal{F}_{TDS}$ in the chosen-message scenario. We consider the following experiment of running a forger $\mathcal{F}_{TDS}$:

Experiment $\text{Exp}_{\mathcal{F}_{TDS}}$
  for k = 1 to q_{max}
    query for (m_k , σ_k), such that $\mathcal{V}_{TDS}(\sigma_k, m_k) = 1$
  let $(m, \sigma) \leftarrow \mathcal{F}_{SRS}(g, \hat{g}, y_1, ..., y_n, m, (m_1, \sigma_1), ...$
    $, (m_k, \sigma_k))$
  if $\mathcal{V}_{SRS}(\sigma, m) = 1$ return 1
  else return 0

Then we define the advantage $\text{Adv}_{\mathcal{F}_{TDS}}$ of the forger. $\mathcal{F}_{TDS}$ as the probability $Pr[\text{Exp}_{\mathcal{F}_{TDS}} = 1]$.

Theorem 1. Threshold discernible ring signatures are secure against forgery, i.e., $\text{Adv}_{\mathcal{F}_{TDS}}$ is negligibly small.

Proof Sketch: We assume the following security results on the verifiable secret sharing of discrete logarithms. This means that no polynomial time adversary can with non-negligible probability produce verifiably encrypted shares $\sigma_2 = \{V_i, i = 1, \cdots, n\}, \{F_i, i = 1, \cdots, t-1\}$ of secret l, without knowledge of the secret l. Additionally in verifiable secret sharing of discrete logarithms, no set of $t-1$ or fewer users can obtain the secret l from σ_2 in polynomial time with non-negligible probability. These results can be obtained from (Stadler, 1996).

This guarantees that the quantity σ_2 cannot be produced without the prior knowledge of l such that $\hat{g} = g^l$ by any adversary. Note that l is the only common value used in generation of σ_1 and σ_2. As σ_1 is an MSRS signature, the unforgeability of σ_1 follows from the unforgeability of σ_2. Hence the tuple (σ_1, σ_2) is unforgeable.

Threshold Anonymity. Threshold anonymity in threshold discernible ring signatures requires that no entity other than a group of at least t ring members must be able to identify the original signer of a ring signature with non-negligible advantage in polynomial time.

Theorem 2. Let $\mathcal{A}_{ATDS}$ be a probabilistic polynomial time algorithm that can distinguish between σ_x, σ_y produced by two different signers for an arbitrary message m by any group of $t-1$ signers among n signers. Let advantage of $\mathcal{A}_{ATDS}$ be defined as $Adv_{\mathcal{A}_{ATDS}} = Pr[A(\sigma_b) = b]$, where $b \in \{x, y\}$. We say that the scheme provides threshold anonymity, if for any efficient algorithm $\mathcal{A}_{ATDS}$ the value of $Adv_{\mathcal{A}_{ATDS}}$ is at most negligibly greater than $1/n$. The threshold discernible ring signature scheme discussed above has the threshold anonymity property.

Proof Sketch: From the anonymity of MSRS

no polynomial time algorithm can discover the original signer of σ using the component σ_1 alone. This ensures that no group of size below t can find the original signer of a signature σ_1. Additionally, in verifiable secret sharing of discrete logarithms, no set of $t-1$ or fewer users can obtain the secret l from σ_2 in polynomial time with non-negligible probability. Hence l cannot be obtained to find the original signer of σ_1, unless a group of at least t users cooperate. Hence, threshold anonymity holds for the signature (σ_1, σ_2) and theorem 2 holds.

8 CONCLUSIONS AND OPEN PROBLEMS

Step out ring signatures, introduced in (Klonowski et al., 2008), had security flaws. We identified those flaws present in the scheme and fixed them in order to make it secure. We have introduced the new concept of the Threshold discernible ring signature using the corrected version of the step out ring signature. Our scheme is proved secure under DDH assumption. The problem of finding a scheme which is secure in the standard model and formulating step out ring signatures using bilinear groups remain open.

REFERENCES

Au, M. H., Chow, S. S. M., Susilo, W., and Tsang, P. P. (2006). Short linkable ring signatures revisited. In *EuroPKI*, pages 101–115. Springer.

Awasthi, A. K. and Lal, S. (2005). Id-based ring signature and proxy ring signature schemes from bilinear pairings. *CoRR*.

Camenisch, J. (1997). Efficient and generalized group signatures. In *EUROCRYPT*, pages 465–479. Springer.

Camenisch, J. and Damgard, I. (2000). Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *ASIACRYPT*, pages 331–345. Springer.

Camenisch, J. and Shoup, V. (2003). Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, pages 126–144. Springer.

Chen, Y.-S., Lei, C.-L., Chiu, Y.-P., and Huang, C.-Y. (2006). Confessible threshold ring signatures. In *ICSNC '06: Proceedings of the International Conference on Systems and Networks Communication*, page 25. IEEE Computer Society.

Cheng, W., Lang, W., Yang, Z., Liu, G., and Tan, Y. (2004). An identity-based proxy ring signature scheme from bilinear pairings. In *ISCC '04: Proceedings of the Ninth International Symposium on Computers and Communications 2004 Volume 2 (ISCC"04)*, pages 424–429. IEEE Computer Society.

Klonowski, M., Krzywiecki, L., Kutylowski, M., and Lauks, A. (2008). Step-out ring signatures. In *MFCS '08: Proceedings of the 33rd international symposium on Mathematical Foundations of Computer Science*, pages 431–442. Springer-Verlag.

Klonowski, M., Krzywiecki, L., Kutyowski, M., and Lauks, A. (2009). Step-out group signatures. *Computing*, 85(1-2):137–151.

Naor, M. (2002). Deniable ring authentication. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 481–498. Springer-Verlag.

Pointcheval, D. (2005). Provable security for public key schemes. In *Contemporary Cryptology*, pages 133–190. Birkhuser Basel.

Rivest, R. L., Shamir, A., and Tauman, Y. (2001). How to leak a secret. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–565. Springer-Verlag.

Savola, R. (2006). A requirement centric framework for information security evaluation. In *IWSEC*, pages 48–59. Springer.

Schnorr, C.-P. (1991). Efficient signature generation by smart cards. *J. Cryptology*, pages 161–174.

Stadler, M. (1996). Publicly verifiable secret sharing. In *EUROCRYPT*, pages 190–199. Springer-Verlag.

Susilo, W. and Mu, Y. (2004). Deniable ring authentication revisited. In *ACNS*, pages 149–163. Springer.

Tsang, P. P. and Wei, V. K. (2005). Short linkable ring signatures for e-voting, e-cash and attestation. In *ISPEC*, pages 48–60. Springer.