# Resisting Traitors in Linkable Democratic Group Signatures

Maged Hamada Ibrahim*
Department of Electronics, Communications and Computers, Faculty of Engineering, Helwan University
1, Sherif St., Helwan, Cairo, Egypt (Email: mhii72@hotmail.com)

*(Received Dec. 15, 2007; revised Mar. 3, 2008; and accepted Apr. 9, 2008)*

## Abstract

Linkable democratic group signatures (LDGS) [29] allow every member of a group to trace the identity of any other member who issued a signature while non-members (with the help of unique pseudonyms) are only able to link the signatures issued by the same signer without being able to trace the signer's identity. LDGS avoid centralized management authorities (group managers) and grant each group member the power to trace and identify the signer. Although LDGS add nice properties to group signatures, allowing each member of the group to trace the signer's identity requires a full trust in each group member not to trace or disclose the identity of the signer without a legal reason (e.g. a dispute). Such a requirement represents a major obstacle in practice. The existence of at least one saboteur member inside the group totally violates the anonymity attribute which is the main merit of group signatures. Such a traitor may reveal the identity of the signers to non-members without being detected. In this paper we introduce a simple, yet efficient traitors resistant LDGS (TR-LDGS) as a security improvement to the LDGS scheme of [29] to resist traitors in the sense that, the power to trace and disclose the identity of the signer must not be in the hands of each member. Instead, the power to trace and identify a signer will be distributed among the members of the group such that a fraction (majority) of the members may join together to trace and reveal the signer's identity while no minority coalitions are able to perform this task or to disturb the correct and legal progress of this task.

*Keywords: Democratic group signatures, group signatures, linkability, threshold cryptography, zero-knowledge proofs*

## 1 Introduction

*Group signatures* (GS) originally introduced in [10] allow members belonging to a group to sign messages on behalf of the group such that, the signature verifier (whether a group member or a non-member) is able to check that the signature is a valid group signature but cannot trace the identity of the signer. In case of a dispute, the trusted authority (group manager) can trace the identity of the signer.

*Democratic group signatures* (DGS) [24] eliminates the role of a group manager by (i) allowing the group members themselves to initialize and setup the group, (ii) controlling it over dynamic changes in a collective manner and (iii) distributing traceability rights to each group individual. In this case, every group member has the individual right to trace and disclose the identity of the signer and hence, anonymity is provided against non-members. The model in [24] requires unlinkability of signatures, i.e., the signature verifier cannot distinguish signatures issued by the same group member without this member being traced and disclosed.

DGS schemes differ from *threshold signature* schemes (e.g. [14, 15, 21, 30, 31]) in the sense that, in DGS each group member is granted the right to generate a signature on a given message individually; a non-member verifier recognizes the signature as anonymously generated by the group. On the other hand, in threshold signatures, the signature on a given message is generated by the majority of the group (exceeding a certain threshold), yet, no coalition of minority (less than or equals the threshold) can generate the signature.

*Linkable democratic group signatures* (LDGS) [29] realize linkability of signatures issued by the group members in a way that preserves the anonymity of the signer. More precisely, a non-member verifier is able to distinguish signatures issued by the same signer for future reference without being able to trace the identity of this signer. To achieve this property, LDGS actually employs the idea of pseudonym systems introduced in [23]. In this scenario, each group member (in addition to his unique identity) will be assigned a unique pseudonym. Given a certain group member, all signed messages generated by this particular member will carry his unique pseudonym. A non-member verifier is able to link signed messages of the same signer via his pseudonym, yet, the verifier gains no infor-

*Part of the work in this paper was accomplished while the author was visiting the department of computer science, ITE-UCONN, University of Connecticut, USA.

mation about the signer's identity from this pseudonym. On the other hand, each group member knows the secret tracing trapdoor parameter, by which, he is able to extract the identity of the signer from the signer's unique pseudonym.

*Ring signatures* introduced in [34] and further studied in [39] and [4] do not require any group manager to form a group. For signature generation, every user builds a set of public keys that includes his public key and the public keys of other users. A generated signature does not reveal the public key of the signer, but a set of public keys of all possible signers. Therefore, ring signatures cannot be used for a direct communication between a verifier and a signer. Additionally, ring signatures provide unconditional anonymity, i.e., no party can reveal the signer identity.

The problem we want to escalate is that, although the LDGS introduced in [29] adds nice properties to group signatures, it suffers from a serious weakness: The power to trace and disclose the identity of the signer is in the hands of each member of the group. Such individual traceability violates the anonymity attribute of group signatures, since, in practice, it maybe possible to guarantee the honesty of one authority (group manager) as in GS but it is difficult and almost impossible to guarantee the full honesty of all the group members as in LDGS. Malicious behavior of at least one of the group members violates anonymity and consequently, destroys the main merit of group signatures. Moreover, in LDGS, such traitors are undetectable.

LDGS consists mainly of four protocols/algorithms: protocol `setup`, algorithm `sign`, algorithm `verify` and algorithm `trace`. The `setup` protocol takes as input a security parameter $l$, and a number of members $n$. For each member $M_i$, the public output is an identity $id_i$ from the set of identities $ID$ and a pseudonym $ps_i$ from the set of pseudonyms $PS$ while the private output is the secret signing key $sk_i$ from the set of secret keys $SK$ and the secret tracing trapdoor parameter $k$ known to each member. The protocol requires the existence of a PKI that allows the group members to authenticate their messages during setup via their certified public keys. Yet, there is no third party actively involved in the protocol. The `sign` algorithm takes as an input a secret key $sk_i$ and a message $m$ and outputs a signature $\sigma$ on $m$. The `verify` algorithm takes as an input a signature $\sigma$, a message $m$ and the set of pseudonyms $PS$ and outputs a pseudonym $ps_i$ if it accepts the signature or $\perp$ if the signature is rejected. Algorithm-`trace` takes as input a signature $\sigma$, a message $m$, the secret tracing trapdoor parameter $k$, and the set of pseudonyms $PS$ and outputs either an identity $id_i$ or $\perp$ in case of failure.

During the setup algorithm, in the computation of the secret tracing trapdoor parameter $k$, the LDGS employs an authenticated Diffie-Hellman based group key agreement protocol as so-called the contributory group key agreement (CGKA) protocols e.g. [19, 27, 28]. In a CGKA protocol, each member $i$ of the group contributes his public key $y_i$ for the interactive computation of the secret common key $k$.

In our improvement to the LDGS, we perform modifications to the `setup` and `trace` protocols to allow the group to withstand possible traitors members. Our idea is to remove the CGKA protocol from the `setup` protocol and employ efficient threshold cryptographic protocols. In this case, the secret tracing trapdoor parameter $k$ will be shared among the members on a threshold basis. The members are able to jointly share a secret parameter which allows the tracing to be performed by the majority of the members. As a result, minority traitors will not be able to perform the tracing and are prevented from disclosing the identity of any signer without a legal reason. Also, the system must be robust against possible malicious behavior of the traitors during the `setup` and `trace` protocols.

## 2 Related Work

Since the introduction of group signatures in [10], several studies and improvements appeared, for example, in [6, 7, 8, 20, 37, 40]. However, these efforts assume the existence of a trusted party called a group manager. The group manager alone has the capabilities to disclose the identity of any signer within the group and to relate (link) two or more signatures as being produced by the same signer.

Democratic group signatures (DGS) introduced in [24] drift from the standard model of group signatures by eliminating the group manager and grant the tracing rights to each member in the group. Group members initialize the group and control it over dynamic changes in a collective manner. Every group member has the individual right to trace issued signatures to their signers. The signature anonymity is provided only against non-members assuming that all members within the group are honest in the sense that no member (a traitor) reveals the identities of other members to outsiders.

The work model of [24] requires unlinkability of signatures, that is, it is not possible to relate signatures produced by the same signer. The lose of linkability is a direct consequence of removing the group manager.

The recent work in [29] noticed the lose of linkability in DGS and introduced a solution to this problem via the idea of pseudonyms introduced in [23]. The linkable democratic group signature (LDGS) in [29] allows a non-member verifier to link signatures produced by the same signer within the group via his pseudonym and use this pseudonym for future reference to this signer without being able to link a pseudonym to any particular identity in the group. One more advantage worth noting for the LDGS of [29] is that the produced signature is efficiently shorter than that produced by the DGS of [24]. Yet, a serious weakness in the LDGS scheme is that, every member in the group has the individual capability to trace and identify a signer and to relate any pseudonym to its corresponding identity. At least one traitor within the group is able to completely destroy the anonymity property which is the main merit of group signatures.

## 3 Motivations and Contributions

In this section we present our motivations behind the work introduced in this paper and our contributions.

### 3.1 Motivations

The work in this paper is motivated by the observation that, although the linkable democratic group signature (LDGS) introduced in [29] is more efficient and adds new properties to democratic group signatures via the idea of pseudonyms, it suffers from a serious security weakness; the tracing trapdoor parameter is known to each member in the group and hence, all members must be honest not to reveal the identity-pseudonym relation to non-members. We consider such an assumption to be an obstacle in practice. The existence of at least one traitor within the group totally compromise the anonymity property which is the main merit of group signatures. Such a traitor may disclose all the identity-pseudonym relations of all the group members to non-members. Moreover, such a sabotage action is undetectable.

### 3.2 Contributions

The contributions of this paper is to improve the security of the LDGS introduced in [29] to resist traitors members within the group. The tracing trapdoor parameter must not be known to individuals, yet, it could be shared on a threshold basis such that, the identity-pseudonym link of any member is kept secret from the members unless there exists a legal reason (e.g. a dispute). In case of a dispute, the majority of the members join together to recover the identity of the signer (related to a given pseudonym). Our traitor resistant LDGS (TR-LDGS) must be robust against possible malicious behavior of the traitors. Our work assumes that there exists minority traitors (at most 1/4 of the members). However, this bound can be improved to 1/3 and further to 1/2 but with high computation and communication complexities.

## 4 Existing LDGS: An Overview

In this section we give a detailed description of the LDGS of [29]. The scheme consists of protocol `Setup`, algorithms `Sign`, `Verify` and `Trace`. There are $n$ members in the group, $M_1, \cdots, M_n$, each member has his own certified public key to allow authenticated and confidential communications among the group members. The system public parameters are two large primes $p$ and $q$ where $q|(p-1)$ and a generator $g \in Z_p^*$.

**Protocol.**
`Setup`: The members initialize the group as follows:

- Each member $M_i$ picks $x_i \in_R Z_q^*$ as a private parameter and computes $y_i = g^{x_i} \bmod p$, as his public identity,
- The members run a CGKA using their published identities, at the end, each member knows the secret tracing trapdoor parameter $k$,
- Each member is able to compute the blinded tracing trapdoor parameter $bk = g^k \bmod p$ and the pseudonym of any member $M_i$ as $ps_i = (bk, y_i^*)$ where $y_i^* = y_i^k \bmod p$.

Finally, the group members publish the set of identities, $ID = \{y_1, \cdots, y_n\}$ and the set of pseudonyms, $PS = \{ps_1, \cdots, ps_n\}$ as the group public parameters. Each member $M_i$ sets his secret key as $sk_i = (x_i, k, ps_i)$.

**Algorithm-`Sign`:** For a member $M_i$ to sign a message $m$ using his secret key $sk_i$, he:

- Picks $r \in_R Z_p^*$, hashes $m$ as $H = \mathcal{H}(m, r)$ and computes $z = H^{x_i} \bmod p$,
- Prepares a NIZK proof of equality of two discrete logarithms, $\Pi_{LogEq}$ which proves that, $\log_H z = \log_{bk} y_i^*$,
- Sends the signature $\sigma$ on $m$ to the verifier where, $\sigma = (r, z, ps_i, \Pi_{LogEq})$.

**Algorithm-`Verify`:** On the reception of a signature $\sigma$ on $m$, the verifier:

- Parses $\sigma$ as $(r, z, ps_i, \Pi_{LogEq})$ and $ps_i$ as $(bk, y_i^*)$.
- Checks whether $ps_i \in PS$, if not, then return $\perp$ and abort,
- Computes $H = \mathcal{H}(m, r)$,
- Performs a NIZK proof verification, $V_{LogEq}(H,\ bk,\ z,\ y_i^*,\ \Pi_{LogEq})$. If the verification is successful, then accept $\sigma$ as a signature on $m$, else, return $\perp$ and abort.

**Algorithm-`Trace`:** Given a pseudonym $ps_i = (bk, y_i^*)$, any member is able to trace the identity of $M_i$ by simply computing $(y_i^*)^{1/k} = y_i$.

The LDGS was proven secure in the random oracle model and proven anonymous assuming full honesty of the members which is not a practical assumption. The reader can easily notice that, any member is able to completely violate anonymity by simply revealing $k$. Once anonymity is violated, the group structure becomes totally useless.

## 5 Basic Tools

In this section we describe the basic tools that will be used to build our TR-LDGS. These tools are partitioned into two categories: threshold cryptography tools and proofs of knowledge tools. The reader must be familiar with these tools in order to follow the description of our TR-LDGS protocol.

### 5.1 Threshold Cryptography Tools

In this subsection we describe the threshold cryptographic tools used in building our TR-LDGS.

#### 5.1.1 Secret Sharing over a Prime Field

Let $s \in Z_q$ be a secret held by the dealer where $Z_q$ is a prime field. In order to share this secret among a set $\mathcal{P}= \{P_1, \cdots, P_n\}$ of $n > t$ players [33], the dealer defines a polynomial $f(x) = \sum_{j=0}^{t} a_j x^j \bmod q$, he sets $a_0 = s$ and each other coefficient $a_{j \neq 0} \in_R Z_q$. $\forall i = 1, \cdots, n$, the dealer secretly delivers $f(i)$ to player $P_i$. In the reconstruction phase, each player $P_i$ broadcasts $f(i)$, the players are able to compute $s$ from any $t + 1$ shares using Lagrange interpolation formula, $s = f(0) = \Sigma_{i \in \mathcal{B}} \lambda_i f(i) \bmod q$ where $\mathcal{B} \subset \mathcal{P}$, $|\mathcal{B}|= t + 1$ and $\lambda_i$ is lagrange coefficient for player $P_i$.

#### 5.1.2 Verifiable Secret Sharing

Verifiable secret sharing (VSS) extends polynomial secret sharing in a way that allows the recipients of the shares to verify that their shares are consistent (i.e., that any subset of $t+1$ shares determines the same unique secret). Assuming $n > 2t$, the protocol can tolerate malicious behaviors (e.g., illegal collaboration, sending wrong values, deleting values, etc.) of at most $t$ players. We distinguish two different contributions of VSS; the conditionally secure VSS due to Feldman [13] and the unconditionally secure VSS due to Pedersen [26]. To achieve best security [22], both of them will be used in our TR-LDGS protocol.

**Feldman's VSS.** Two large primes $p$ and $q$ are chosen such that $q|p - 1$. The primes $p$ and $q$ and an element $g \in Z_p^*$ of order $q$ are published as the system public parameters. The dealer shares the secret $s$ among the players on a $t$-degree polynomial $f(x) = \sum_{j=0}^{t} a_j x^j \bmod q$, the dealer also broadcasts the $t + 1$ commitments $c_j = g^{a_j} \bmod p$ $\forall j = 0, \cdots, t$. These commitments allow each player $P_i$ to verify the consistency of his share $f(i)$ by checking that, $g^{f(i)} = \prod_{j=0}^{t} c_j^{i^j} \bmod p$. If this check fails for any share $f(i)$, $P_i$ broadcasts a complaint. If more than $t$ players broadcasted a complaint, then at least one of them is honest and consequently the dealer is disqualified. Otherwise the dealer reveals the share $f(i)$ for each complaining player $P_i$, if the share is correct, $P_i$ is disqualified, otherwise, if the share does not satisfy the commitments or if the dealer does not respond, the dealer is disqualified. During reconstruction of the secret, any player can check the validity of the share broadcasted by any other player via the published commitments to filter out bad shares and safely perform the interpolation. When it comes to the distributed generation of a secret key $k$ and the joint computation of $g^k \bmod p$, Feldman's VSS alone is not secure due to the attacks described in [22].

**Pedersen's VSS.** The idea is to use double exponentiation which allows randomization. The public parameters are $p, q, g$ and $h$ where $p, q$ and $g$ are as in Feldman's VSS and $h$ is another element in $Z_p^*$ subject to the condition that $\log_g h$ is unknown and assumed hard to compute. In addition to the polynomial $f(x) = \sum_{j=0}^{t} a_j x^j \bmod q$ which holds the secret $s$ as the free term, the dealer sets up a randomizing $t$-degree polynomial $r(x) = \sum_{j=0}^{t} b_j x^j \bmod q$. He secretly delivers $(f(i), r(i))$ to player $P_i$ $\forall i = 1, \cdots, n$. The dealer also publishes the commitments $c_j = g^{a_j} h^{b_j} \bmod p$ $\forall j = 0, \cdots, t$. Each player $P_i$ verifies the consistency of his share $f(i)$ by checking that, $g^{f(i)} h^{r(i)} = \prod_{j=0}^{t} c_j^{i^j} \bmod p$. If this check fails for any share $f(i)$, $P_i$ broadcasts a complaint. If more than $t$ players broadcast a complaint, then at least one of them is honest and consequently the dealer is disqualified. Otherwise the dealer reveals the pair $(f(i), r(i))$ for each complaining player $P_i$, if the pair is correct, $P_i$ is disqualified, otherwise, if the pair does not satisfy the commitments or if the dealer does not respond, the dealer is disqualified. During reconstruction of the secret, any player can check the validity of the share broadcasted by any other player via the published commitments to filter out bad shares and safely perform the interpolation.

#### 5.1.3 Joint Secret Sharing

Joint secret sharing allows the players to jointly share some secret among themselves without the help of the dealer.

**Joint random secret sharing (JRSS).** JRSS [18] allows a set of $n$ players to jointly share a random secret without the help of the dealer. Each player $P_i \in \mathcal{P}$ picks a random integer $k_i \in Z_q$ and plays the role of the dealer to share $k_i$ among the players over a $t$-degree polynomial $f_i(x) = k_i + \sum_{j=1}^{t} a_j x^j \bmod q$. Each player $P_i \in \mathcal{P}$ simply sums the shares he receives from the other players to compute a share $f(i) = \sum_{j=1}^{n} f_j(i)$ which is a point on a $t$-degree polynomial $f(x)$ with its free term equals a random secret $k = \sum_{i=1}^{n} k_i \bmod q$.

**Joint random verifiable secret sharing (JRVSS).** To withstand malicious behavior of at most $t < n/2$ players during the JRSS, JRVSS combines the JRSS with Feldman's VSS for computational security or Pedersen's VSS for unconditional security. Simply, each player $P_i \in \mathcal{P}$ picks a random secret integer $k_i \in Z_q$ and plays the role of the dealer in the VSS protocol to share this secret among the other players. Complaints are solved as in the VSS protocol. Finally, each player sums what he has to compute his share on a $t$-degree polynomial, $f(x)$ with its free term $f(0) = \sum_{i=1}^{n} k_i \bmod q$.

**Joint zero secret sharing (JZSS).** JZSS is a special case of the JRSS in which the random secret shared by each player is a zero. At the end of the JZSS, each player holds a share $f(i)$ on a $t$-degree polynomial $f(x)$ with its free term $f(0) = 0$.

**Joint zero verifiable secret sharing (JZVSS).** As in the JRVSS, to with stand malicious behavior of at most $t < n/2$ players during the JZSS, JZVSS combines the JZSS with Feldman's VSS for computational security or Pedersen's VSS for unconditional security. Simply, each player $P_i \in \mathcal{P}$ plays the role of the dealer in the VSS protocol to share a zero among the other players. Complaints are solved as in the JRVSS protocol. Finally, each player sums what he has to compute his share on a $t$-degree polynomial, $f(x)$ with its free term $f(0) = 0$. Notice that, from the published commitments, each player can verify that the shared secret is really a zero. This is true since the commitment to a zero will be $g^0 = 1$.

#### 5.1.4 The Multiplication Protocol

Given two secrets $a$ and $b$ shared over $t$-degree polynomials $A(x)$ and $B(x)$ respectively, the `multiplication` protocol [38] computes $\xi = ab \bmod q$ in a robust way with no information revealed about $a$ or $b$. Each player $P_i$ locally computes $C(i) = A(i)B(i) \bmod q$. In this case, each $C(i)$ is a share on a $2t$-degree polynomial $C(x) = A(x)B(x) \bmod q$ with $C(0) = \xi$. However, publishing and interpolating the shares $C(1), \cdots, C(n)$ reveals information about $A(x)$ and $B(x)$ [38], consequently, randomizing the shares of $C(x)$ is necessary. To randomize the shares of $C(x)$ without changing $C(0)$ the players run a `JZVSS` to share a zero over a $2t$-degree polynomial $R(x)$ with $R(0) = 0$. Finally, each player $P(i)$ computes and publishes $D(i) = C(i) + R(i)$. The result $\xi$ could be reached by interpolating the $2t$-degree polynomial $D(x)$ with the help of the Berlekamp-Welch decoding scheme [5] to filter out corrupted shares. Since we are interpolating a polynomial of degree $deg = 2t$ and we have a maximum of $t$ malicious players (i.e. at most $t$ possible $faults$), using the Berlekamp-Welch bound, the number of shares needed in order to correctly interpolate the polynomial is at least $deg + 2faults + 1 = 4t + 1$. Hence, we need $n > 4t$.

#### 5.1.5 The Reciprocal Protocol

In the tracing protocol of our TR-LDGS we are faced with the following problem. Given a secret $k$ which is shared among the players, generate a sharing of the reciprocal of $k$ modulo $q$ with no information revealed about $k$. Each player $P_i$ holds a share $f(i)$ which is a point on a $t$-degree polynomial $f(x)$ with $f(0) = k$. To compute shares of $k^{-1}$, we need $n > 4t$, the $n$ players run the reciprocal protocol [1] as follows:

- The players run the `JRVSS`, at the end each player holds a share $v(i)$ of a random secret $v$ over a polynomial of degree $t$.
- The players run the `multiplication` protocol and reconstruct the value $\xi = kv \bmod q$.
- Finally each player $P_i$ computes his share of the reciprocal as $\xi^{-1}v(i) \bmod q$, which is a share over a $t$-degree polynomial with its free term equals $k^{-1} \bmod q$.

## 5.2 Proofs of Knowledge Tools

In this subsection we describe the proof of knowledge tools used in building our TR-LDGS.

#### 5.2.1 Proof of Equality of Two Discrete Logarithms

We review the protocol of [11, 35] that is believed to be a zero knowledge proof of equality of two discrete logarithms. In this protocol, the public parameters are two large primes $p$ and $q$ such that $q|p-1$, two elements $\alpha, \beta \in Z_p^*$ and the two quantities $G_1, G_2 \in Z_p^*$. The prover $(\mathcal{P})$ proves to a verifier $(\mathcal{V})$ that he knows $x \in Z_q^*$ such that $G_1 = \alpha^x \bmod p$ and $G_2 = \beta^x \bmod p$. The protocol is as follows:

- $\mathcal{P} \rightarrow \mathcal{V}$:
  Choose $r \in_R Z_q^*$ and send $(A = \alpha^r \bmod p,\ B = \beta^r \bmod p)$.
- $\mathcal{V} \rightarrow \mathcal{P}$:
  Choose $c \in_R Z_q^*$ and send $c$.
- $\mathcal{P} \rightarrow \mathcal{V}$:
  Send $y = r + cx \bmod q$.
- $\mathcal{V}$:
  Check that $\alpha^y = AG_1^c \bmod p$ and $\beta^y = BG_2^c \bmod p$.

The above protocol can be made non-interactive (we denote it, $\Pi_{LogEq} \leftarrow P_{LogEq}(\alpha, \beta, G_1, G_2, x)$ ) using a sufficiently strong hash function $\mathcal{H}()$ [3] and setting $c = \mathcal{H}(A, B)$. The protocol $\Pi_{LogEq}$ becomes as follows:

- $\mathcal{P} \rightarrow \mathcal{V}$:
  Choose $r \in_R Z_q^*$ and send $(A = \alpha^r \bmod p,\ B = \beta^r \bmod p,\ c = \mathcal{H}(A, B)$ and $y = r + cx \bmod q)$.
- $\mathcal{V}$: Check that $\alpha^y = AG_1^c \bmod p$ and $\beta^y = BG_2^c \bmod p$.

#### 5.2.2 Proof of Existence of Discrete Log Equality

Let $y_i = \alpha^{x_i} \bmod p$ for $i = 1, .., n$ and let $z = \beta^{x_i} \bmod p$ for some $i \in \{1, \cdots, n\}$. A prover $\mathcal{P}$ demonstrates to a verifier $\mathcal{V}$ that he knows one of the logarithms of $y_i$ ($i \in \{1, \cdots, n\}$) to the base $\alpha$ and that $\log_\alpha y_i = \log_\beta z \bmod q$ without revealing which $i$. Let the relation holds for $i = 1$ (i.e. $x_1 = \log_\alpha y_1 = \log_\beta z \bmod q$). The protocol is as follows [32]:

- $\mathcal{P} \rightarrow \mathcal{V}$:
  Choose $k_i \in_R Z_q^*$ for $i = 1, \cdots, n$, $c_j \in_R Z_q^*$ for $j = 2, \cdots, n$ and compute:
  - $r_1 = \alpha^{k_1} \bmod p$, $r_i = \alpha^{k_i} y_i^{-c_i} \bmod p$ for $i = 2, \cdots, n$.
  - $t_1 = \beta^{k_1} \bmod p$, $t_i = \beta^{k_i} z^{-c_i} \bmod p$ for $i = 2, \cdots, n$.

  Send the values $(r_1, \cdots, r_n, t_1, \cdots, t_n)$.
- $\mathcal{V} \rightarrow \mathcal{P}$:
  Choose and send $c \in_R Z_q^*$.
- $\mathcal{P} \rightarrow \mathcal{V}$:
  Calculate $c_1 = c - \sum_{i=2}^{n} \bmod q$, $s = x_1 c_1 + k_1 \bmod q$ and set $s_i = k_i$ for $i = 2, \cdots, n$. Send $(c_1, \cdots, c_n, s_1, \cdots, s_n)$.
- $\mathcal{V}$:
  Check that $c = \sum_{i=1}^{n} c_i$ and that $\forall i = 1, \cdots, n$: $\alpha^{s_i} = y_i^{c_i} r_i \bmod p$ and $\beta^{s_i} = z^{c_i} t_i \bmod p$.

The above interactive proof can be transformed into a non-interactive proof that we will denote it by:

$$\Pi_{\exists LogEq} \leftarrow P_{\exists LogEq}(\alpha, \beta, y_1, \cdots, y_n, z)$$

using a strong hash function $\mathcal{H}()$ [3]. This can be done by setting $c = \mathcal{H}(y_1, \cdots, y_n, \alpha, z, \beta, \alpha^{s_1} y_1^{-c_1}, \cdots, \alpha^{s_n} y_n^{-c_n}, \beta^{s_1} z^{-c_1}, \cdots, \beta^{s_n} z^{-c_n})$.

## 6 Assumptions and Model

We assume the existence of $n$ group members, $M_1, \cdots, M_n$, where $n > 4t$ and $t \geq 1$ is a particular threshold. There exists at most $t$ possible traitors (malicious members). We assume a static adversary model, however, security against adaptive adversary could be achieved by plugging a suitable non-committing encryption scheme (see [2, 25]). Also security against coercive adversaries could be realized by plugging a suitable deniable public key encryption scheme (see [9, 12, 16, 17]).

Each group member has his own certified public key provided to him by a certain certification authority (CA), yet, this CA does not participate actively in the group. Certified public keys allow the members to realize authenticated and private channels among themselves. We also assume that the non-member verifiers are semi-honest in the sense that, they follow the execution steps word for word but they are willing to learn any information leaked during verification. Finally, the bound on the number of members could be reduced to $n > 3t$ but with an increase in computation and communication complexities [21].

## 7 Our Traitors Resistant LDGS

We are ready to describe the details of our TR-LDGS which consists of: Protocol `setup`, algorithm `sign`, algorithm `verify` and protocol `trace`. We assume that there exists $n > 4t$ group members with at most $t \geq 1$ possible traitors. Also we assume that each member has his own certified public key to realize authenticated and confidential communications among the members. The initial public parameters of the system are: two large primes $p, q$ such that $q | p - 1$ and two elements $g, h \in Z_p^*$ such that $\log_g h$ is unknown and assumed hard to compute.

### 7.1 The Protocol

Our TR-LDGS is as follows:

**Protocol.**
`Setup`: The members initialize the group as follows:

- Each member $M_i$ picks a private key $x_i \in_R Z_q$, computes and publishes the corresponding authenticated public key $y_i = g^{x_i} \bmod p$. At the end, we have the set of identities $ID = \{y_1, \cdots, y_n\}$.
- The members run a `JRVSS` with Pedersen's VSS as the VSS in place. At the end, each member $M_i$ holds a share $K(i)$ of a random secret $k \in_R Z_q^*$ over a $t$-degree polynomial $K(x)$ with $K(0) = k$.
- The members that are not disqualified in the JRVSS in the previous step broadcast the commitments to their shared polynomial based on Feldman's VSS. More precisely, if $K_i(x) = k_i + \sum_{j=1}^{t} a_j$ is the polynomial of member $M_i$, $M_i$ broadcasts $g^{k_i}$ and $g^{a_j} \bmod p$ $\forall j = 1, \cdots, t$.
- For any member $M_i$ who receives at least one valid complaint in the previous step, the other members join to reconstruct his polynomial $K_i(x)$ and values $g^{k_i}$ and $g^{a_j} \bmod p \ \forall j = 1, \cdots, t$ in the clear.
- Finally, the remaining good members join to safely compute $g^k = \prod_{i=1}^{n} g^{k_i} \bmod p$.

  At this point, the members share the tracing trapdoor parameter $k$ over a $t$-degree polynomial and have jointly computed the blinded tracing trapdoor, $bk = g^k \bmod p$. As a preparation for the tracing protocol, the members need to compute shares of $k^{-1} \bmod q$, so they proceed:
- The members run the `reciprocal` protocol, at the end, each member holds a share $D(i)$ on a $t$-degree polynomial, $D(x)$ with its free term $D(0) = k^{-1} \bmod q$.

- Each member broadcasts Feldman's VSS commitments (i.e. to the base $g$) of all his chosen random polynomials during the reciprocal protocol. These commitments allow the members to validate the quantities $G_i = g^{D(i)} \bmod p \ \forall i$.

Next, the members compute their pseudonyms. *We must emphasize that, in the LDGS of [29], any member can compute the pseudonym of any other member, since the tracing trapdoor parameter $k$ is known to each member, the pseudonym of member $M_i$ is computed as $ps_i = (bk, y_i^k)$. In our case, the pseudonym of member $M_i$ cannot (and must not) be computed by any member other than $M_i$ himself so that no member within the group can relate any pseudonym to a particular identity unless the tracing protocol is initialized by the majority of the members or the signer himself declares his identity.* So we proceed,

- Each member $M_i$ computes $y_i^* = (bk)^{x_i} \bmod p$ and parse his pseudonym as $ps_i = (bk, y_i^*)$.

This finalizes the setup protocol, each member $M_i$ sets his secret information $sk_i$ as $sk_i = (x_i, D(i), ps_i)$. The public parameters are the set of identities, $ID = \{y_1, \cdots, y_n\}$, the set of commitments, $\mathcal{G} = \{G_1, \cdots, G_n\}$ where $G_i = g^{D(i)} \bmod p$ and the blinded tracing trapdoor parameter, $bk$.

**Algorithm-`Sign`:** For a member $M_i$ to sign a message $m$:

- Picks an integer $r \in_R Z_q^*$, hashes $m$ and $r$ as $H = \mathcal{H}(m, r)$ and computes $z = H^{x_i} \bmod p$.
- Generates a NIZK proof, $\Pi_{LogEq} \leftarrow P_{LogEq}(H, bk, z, y_i^*, x_i)$, which proves that $\log_H(z) = \log_{bk}(y_i^*)$.
- Generates a NIZK proof, $\Pi_{\exists LogEq} \leftarrow P_{\exists LogEq}(g, bk, y_1, \cdots, y_n, y_i^*)$, which proves that there exists some index $i \in \{1, \cdots, n\}$ such that $\log_g y_i = \log_{bk} y_i^*$.
- Parses $\sigma_i$ as $(r, z, ps_i, \Pi_{LogEq}, \Pi_{\exists LogEq})$. $\sigma_i$ is $M_i$'s signature on $m$.

**Algorithm-`Verify`:** On the reception of $\sigma_i$ and $m$ by the verifier:

- The verifier Parses $\sigma_i$ as $(r, z, ps_i, \Pi_{LogEq}, \Pi_{\exists LogEq})$ and parses $ps_i$ as $(bk, y_i^*)$.
- The verifier performs the verification algorithm, $V_{LogEq}(H, bk, z, y_i^*, \Pi_{LogEq})$, if not successful then reject $m$ and abort. Else,
- The verifier performs the verification algorithm $V_{\exists LogEq}(g, bk, y_1, \cdots, y_n, y_i^*, \Pi_{\exists LogEq})$, if not successful then reject $m$ and abort. Else, accept $\sigma_i$ as a signature on $m$.

**Protocol `Trace`:** In case of – for example – a dispute, the members join to reveal the identity of the signer from the argued pseudonym $ps_j = (bk, y_j^*)$. Each member $M_i$:

- Broadcasts $Y_i = (y_j^*)^{D(i)} \bmod p$, where $D(i)$ is $M_i$'s share of $k^{-1} \bmod q$.
- Broadcasts $\Pi_{LogEq} \leftarrow P_{LogEq}(g, y_j^*, G_i, Y_i, D(i))$ to prove that $\log_g G_i = \log_{y_j^*} Y_i \bmod q$.

From any $t+1$ quantities, $Y_i$'s, that pass the proof $\Pi_{LogEq}$ successfully, any member can perform interpolation in the exponent to compute $(y_j^*)^{1/k} = y_j$. Interpolation in the exponent is as simple as computing:
$\prod_{i\in\mathcal{B}} (y_j^*)^{D(i)\lambda_i} = (y_j^*)^{\sum_{i\in\mathcal{B}} D(i)\lambda_i} = (y_j^*)^{k^{-1}} = y_j$, where $|\mathcal{B}| = t+1$ and $\lambda_i$ is lagrange coefficient of member $M_i$.

## 7.2 Members' Join and Leave

In our scheme, at least $t+1$ members are enough to run the trace protocol, consequently, $n-(t+1)$ members can leave the group without affecting the correct operations. Those members simply declare themselves as wanting to leave the group and their identities are removed from the group set of identities and their shares of the reciprocal are made public by the rest of the members. If more than $n-(t+1)$ members leave, the rest of the members must re-initialize the group.

For a member to join the group and be part of the trace protocol, the existing members simply initialize a verifiable secret sharing redistribution (e.g. [36]), to distribute shares of the reciprocal to the new member. In brief, given that each player $P_i$ holds a share $D(i)$ of $k^{-1}$, then $k^{-1} = \sum_{i\in\mathcal{B}} D(i)\lambda_i$, $|\mathcal{B}| = t+1$. For the new set of $n'$ players, each player $P_i$ in $\mathcal{B}$ re-share the quantity $D(i)\lambda_i$ among the other players over a $t$-degree polynomial. Finally each player sums the shares he receives from the other players to compute his new share of $k^{-1}$ for the set $n'$. Verifiability is achieved by injecting VSS schemes.

## 8 Security Analysis

In the core of the `setup` protocol, the generation of the tracing trapdoor parameter (which is a random, uniformly distributed value $k$) is distributed on a threshold basis and the value $bk = g^k$ is made public. The protocol is called $t$-secure, that is, in the presence of at most $t$ malicious members:

**Correctness:**

- All subsets of t+1 valid shares reconstruct to the same unique secret parameter $k$.
- Each member is able to compute the common public value $bk = g^k$.

- $k$ is uniformly distributed in $Z_q$ and hence, $bk$ is uniformly distributed in the subgroup generated by $g$.

**Secrecy:** No information on $k$ can be learned by the coalition of at most $t$ members except for what is implied by the value $bk = g^k$.

Performing JRVSS with Feldman's VSS alone is insecure, since, traitors can influence the distribution of the result of Feldman's VSS to a non-uniform distribution [22]. More precisely, the attack works as follows: Assume that two traitors – say $M_1$ and $M_2$ – want to bias the distribution towards values $bk$ whose last bit is 0. $M_1$ gives members, $M_3, \cdots, M_{t+2}$ shares which are inconsistent with his broadcast values, the rest of the members receive consistent shares. Thus, there will be $t$ complaints against $M_1$, yet $t$ complaints are not enough for disqualification. The traitors compute $\alpha = \sum_{i=1}^{n} g^{k_i}$ and $\beta = \sum_{i=2}^{n} g^{k_i}$. If $\alpha$ ends with 0 then $M_1$ will do nothing and continue the protocol as written. If $\alpha$ ends with 1 then force the disqualification of $M_1$, this is achieved by asking $M_2$ to also broadcast a complaint against $M_1$, which brings the number of complaints to $t + 1$. This action sets the public value $bk$ to $\beta$ which ends with 0 with probability 1/2. An illustrative example is as follows: Let $00, 01, 10, 11$ be the possible two MSBs of $bk$. Now, if $bk$ ends with zero $(00, 01)$ then let $\alpha$ be, but if $bk$ ends with one $(10, 11)$ then $M_2$ complains (let $\beta$ be)and hence the probability that $bk$ ends with one $(11)$ is 1/2 (notice that $(11)$ is the only situation that makes the attack fails), hence, the attack fails with probability 1/4 and so we have. Thus effectively the traitors have forced strings ending in 0 to appear with probability 3/4 rather than 1/2. One must notice that synchronous broadcast does not prevent such attack to take place. Hence, the third requirement for correctness and the secrecy requirement dramatically fail. In Pedersen's VSS, the view of the traitors is independent of the value of the secret $k$, and therefore the secrecy of $k$ is unconditional, this eliminates the possibility of the described attack. The security of the JRVSS can be proven via simulation [22].

In the `sign` algorithm of the LDGS of [29], it was enough that a signer $M_i$ proves to the verifier that $\log_H(z) = \log_{bk}(y_i^*)$ via $P_{LogEq}(H, bk, z, y_i^*, x_i)$ since the set of pseudonyms $PS$ is already published by the group and already known to the verifier. The verifier simply verifies that $y_i^*$ is a valid pseudonym by searching this value in $PS$. In our TR-LDGS, the set $PS$ cannot be published or else the group members are able to trace each other (notice that, if a member publishes his pseudonym, he must sign it for authentication using his certified public key, which discloses his identity to the group members). Consequently, in our protocol, when the verifier receives a signed message from a signer for the first time, the signer must prove that the included pseudonym is valid (i.e. related to an identity in the set $ID = \{y_1, \cdots, y_n\}$), hence, the signer $M_i$ includes the proof $P_{\exists LogEq}(g, bk, y_1, \cdots, y_n, y_i^*)$ which proves to the verifier that there exists some index $i \in \{1, \cdots, n\}$ such that $\log_g y_i = \log_{bk} y_i^*$ with no information revealed about the exponent $x_i$ or the index $i$.

In the `verify` algorithm, since $\Pi_{LogEq}$ is ZK, a verifier that receives a signed message with a certain pseudonym $ps_i = (bk, (bk)^{x_i})$ is faced with the computational Diffie-Hellman problem (CDHP) to compute $x_i$ given $bk$ and $(bk)^{x_i}$. Given the set of identities $ID$, anonymity of the signer is preserved, since $\Pi_{\exists LogEq}$ is ZK. The verifier is faced with the decisional Diffie-Hellman problem (DDHP), that is, given, $g, g^k, g^{kx_i}$, distinguish $g^{x_i}$ from $g^{x_j}$ for a random $x_{j \neq i} \in Z_q^*$.

Finally, algorithm `trace` is $t$-resilient, that is at most $t$ members traitors cannot trace the identity of the signer. This statement follows from the properties of threshold structures [22, 38].

## 9 Evaluations and Comparisons

Since we add a new security service to the LDGS [29], it is predictable that the computations and communications complexities will increase accordingly. In the `setup` protocol, we replaced the CGKA which is used to produce a common key $k$, among the members with a verifiable secret sharing protocol to share $k$ and to resist possible malicious behaviors attempted by at most $t$ members. There is an increase in the computations and communications complexities due to the need to compute and publicize polynomial commitments chosen by each member. As a result of the `setup` protocol, the group public parameters increases over that in [29] by $n$ group elements (contained in the set of public commitments, $\mathcal{G}$). One must notice that the setup protocol is performed only once for initializing the group.

In the `sign` algorithm, the length of the signature (as a first signature from this group signer) increases due to the need to include $\Pi_{\exists LogEq}$ in the signature. However, this proof is performed by the signer only once per verifier to proof that the signer's pseudonym is a valid pseudonym related to some anonymous identity $y_i$, once the verifier successfully verifies the correctness of the pseudonym, he simply stores the pseudonym for future messages from this particular signer. Hence, $\Pi_{\exists LogEq}$ is included only in the first signed message to this verifier and removed from the future messages. The increase of complexity in the algorithm verify is also due to the need to perform the verification step, $V_{\exists LogEq}()$ for each new group signer.

In our protocol `trace` (an algorithm in the LDGS of [29]), which is not supposed to be frequently performed unless there is a legal reason (e.g. a dispute) agreed by the majority of the members, the complexities increase due to the need to perform the proof $\Pi_{LogEq}$ for each broadcasted quantity $Y_i$ and the need to perform interpolation in the exponent.

## 10 Conclusions

The linkable democratic group signature of [29] has the serious weakness that the power to trace the signer is granted to each member of the group and hence the assumption of the full honesty of each member is a must. At least one member traitor totaly violates group anonymity by simply revealing the tracing trapdoor parameter to a non-member. In this paper, we introduced a solution to this security flaw by distributing the power to trace and identify the signer among the group members on a threshold bases to resist traitors and prevent them from violating anonymity of the group. Due to the new security service we provide, the complexities increase accordingly, yet, our solution is efficient and practical.

## Acknowledgments

The work in this paper is supported, in part, by a grant from the Egyptian Culture and Education Bureau (ECEB) Washington-DC.

## References

[1] D. Beaver, and J. B. Ilan, "Non-cryptographic fault-tolerant computing in a constant number of rounds," *Proceedings of the ACM Symposium on Principles of Distributed Computation*, pp. 201-209, 1989.

[2] D. Beaver, and S. Haber, "Cryptographic protocols provably secure against dynamic adversaries," *Eurocypt '92*, pp. 307-323, 1993.

[3] M. Bellare, and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," *First ACM Conference on Computer and Communications Security*, pp. 62-73, 1993.

[4] A. Bender, J. Katz, and R. Morselli, "Ring signatures: Stronger definitions, and constructions without random oracles," *Theory of Cryptography Conference*, pp. 60-79, Springer-Verlag, 2006.

[5] E. Berlekamp, and L. Welch, *Error Correction for Algebraic Block Codes*, US Patent 4,633,470.

[6] J. Camenisch, and M. Stadler, "Efficient group signature schemes for large groups," *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, LNCS 1294, pp. 410-424, Springer-Verlag, 1997.

[7] J. Camenisch, and M. Michels. "A group signature scheme with improved efficiency," *Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security*, pp. 160-174, 1998.

[8] J. Camenisch, and J. Groth, "Group signatures: Better efficiency and new theoretical aspects," *SCN*, LNCS 3352, pp. 120-133, Springer-Verlag, 2004.

[9] R. Canetti, and R. Gennaro, "Incoercible multiparty computation," *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pp. 504-513, 1996.

[10] D. Chaum, and E. Heyst, "Group signatures," *Advances in Cryptology - Eurocrypt '91*, LNCS 547, pp. 257-265, Springer-Verlag, 1991.

[11] E. D. Chaum, and T.P. Pedersen, "Wallet databases with observers," *Advances in Cryptology - Crypto '92*, pp. 89-105, 1992.

[12] C. Dwork, M. Naor, R. Canetti, and R. Ostrovsky," Deniable encryption," *Proceedings of the 17th Annual international Cryptology Conference on Advances in Cryptology*, pp. 90-104, Springer-Verlag, London, 1997.

[13] P. Feldman, " A practical scheme for non-interactive verifiable secret sharing," *Proceeding 28th Annual Symposium on the Foundations of Computer Science*, pp. 427-437, IEEE, 1987.

[14] M. H. Ibrahim, "Eliminating quadratic slowdown in two-prime rsa function sharing," *International Journal of Network Security (IJNS)*, vol. 7, no. 1, pp. 107-114, July 2008.

[15] M. H. Ibrahim, *Efficient Dealer-Less Threshold Sharing of Standard RSA, International Journal of Network Security (IJNS)*, to appear.

[16] M. H. Ibrahim, *A method for obtaining deniable public-key encryption, in the International Journal of Network Security (IJNS)*, to appear.

[17] M. H. Ibrahim, "Receiver-deniable public-key encryption," *The International Journal of Network Security (IJNS)*, to appear.

[18] I. Ingemarsson, and G. J. Simmons, "A protocol to set up shared secret schemes without the assistance of a mutualy trusted party," *Advances in Cryptology - Eurocrypt '91*, LNCS 473, pp. 266-282, I. B. Damgard, Editor, Springer-Verlag, 1990.

[19] J. Katz, and M. Yung, "Scalable protocols for authenticated group key exchange," *Advances in Cryptology - Crypto '03*, LNCS 2729, pp. 110-125, Springer-Verlag, 2003.

[20] A. Kiayias, and M. Yung, "Group Signatures With efficient Concurrent Join," *Advances in Cryptology - Eurocrypt '05*, LNCS 3494, pp. 198-214, Springer-Verlag, 2005.

[21] H. Krawczyk, R. Gennaro, S. Jarecki, and T. Rabin, "Robust threshold DSS signatures," *Information and Computation*, vol. 164, no. 1, pp. 54-84, 2001.

[22] H. Krawczyk, R. Gennaro, S. Jarecki, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *Journal of Cryptology* , vol. 20, no. 1, pp. 51-83, 2007.

[23] A. Lysyanskaya, A. Sahai, R. L. Rivest, and S. Wolf, "Pseudonym systems," *Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography* , pp. 184-199, 2000.

[24] M. Manulis, "Democratic group signatures - on an example of joint ventures," *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS 2006)*, pp. 365-365, Taiwan, China, 2006.

[25] M. Naor, O. Goldreich, R. Canetti, and U. Feige, "Adaptively secure multi-party computation," *28th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 639-648, 1996.

[26] T. Pedersen, "Non-interactive and information theoretic secure verifiable secret sharing," *Advances in Cryptology - Crypto '91*, LNCS 576, pp. 129-140, J. Feigenbaum, Editor, Springer-Verlag, 1992.

[27] A. Perrig, G. Tsudik, and Y. Kim, "Tree-based group key agreement," *ACM Transactions on Information and System Security*, vol. 7, no. 1, pp. 60-96, 2004.

[28] C. N. Rotaru, G. Tsudik, Y. Amir, and Y. Kim, " On the performance of group key agreement protocols," *ACM Transactions on Information and System Security*, vol. 7, no. 3, pp. 457-488, 2004.

[29] A. Sadeghi, J. Schwenk, and M. Manulis, "Linkable democratic group signatures," *Proceedings of the 2nd Information Security Practice and Rexperience Conference (ISPEC 2006)*, pp. 11-14, China, 2006.

[30] A. H. E. Sawy, I. A. Ali, I. I. Ibrahim, and M. H. Ibrahim, "Fast fully distributed and threshold rsa function sharing," *Proceedings of Information Systems: New Generation Conference (ISNG 2004)*, pp. 11-15, Las Vegas, Nevada, USA, 2004.

[31] A. H. E. Sawy, I. I. Ibrahim, and M. H. Ibrahim, "Fast three party shared generation of rsa keys without distributed primality test," *Proceedings of Information Systems: New Generation Conference (ISNG 2004)*, pp. 5-10, Las Vegas, Nevada, USA, 2004.

[32] B. Schoenmakers, *Efficient Proofs of OR*, Manuscript, 3 pages, 1993.

[33] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, pp. 612-613, 1979.

[34] A. Shamir, R. L. Rivest, and Y. Tauman, "How to leak a secret," *Asiacrypt 1, Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, LNCS 2248, pp. 552-565, Springer-Verlag, 2001.

[35] C. C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161-174, 1991.

[36] C. Wang, J. M. Wing, and T. M. Wong, *Verifiable Secret Redistribution for Threshold Sharing Schemes*, Technical Report CMU-CS-02-114-R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, Sep. 2002.

[37] B. Warinschi, D. Micciancio, and M. Bellare, "Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions," *Advances in Cryptology - Eurocrypt '03*, LNCS 2656, pp. 614-629, Springer-Verlag, 2003.

[38] A. Wigderson, M. B. Or, and S. Goldwasser, "Completeness theorems for noncryptographic fault-tolerant distributed computations," *Proceeding 20th Annual Symposium on the Theory of Computing, ACM*, pp. 1-10, 1988.

[39] D. S. Wong, J. K. Liu, and V. K. Wei, "Linkable spontaneous anonymous group signature for ad-hoc groups, " *Information Security and Privacy, 9th Australasian Conference, ACISP 2004*, LNCS 3108, pp. 325-335, Springer-Verlag, 2004.

[40] C. Zhang, H. Shi, and M. Bellare, "Foundations of group signatures: The case of dynamic groups," *CT-RSA*, LNCS 2656, pp. 136-153, Springer-Verlag, 2005.

**Maged Hamada Ibrahim** received his BSc in communications and computers engineering from Helwan University, Cairo; Egypt. Received his MSc and PhD in Cryptography and Network security systems from Helwan University in 2001 and 2005 respectively. Currently, working as an assistant professor and also joining several network security projects in Egypt. His main interest is Cryptography and network security. More specifically, working on the design of efficient and secure cryptographic algorithms, in particular, secure distributed multiparty computations. Other things that interest him are number theory and the investigation of mathematics for designing secure and efficient cryptographic schemes.