International Journal of Computer (IJC)

ISSN 2307-4523 (Print & Online)

© Global Society of Scientific Research and Researchers

http://ijcjournal.org/

# Performance and Security of Group Signature in Wireless Networks

Joshua J. Tom[a]*, Prof Boniface. K. Alese[b], Dr. Aderonke F. Thompson[c], Dr. Nlerum P. Anebo[d]

[a,b,c]*Department of Computer Science, Federal University of Technology, Akure, Nigeria*

[d]*Department of Computer Science, Federal University, Otuoke, Nigeria*

[a]*Email: tomjoshua.tom@gmail.com*

[b]*Email: bkalese@futa.edu.ng*

[c]*Email: afthompson@futa.edu.ng*

[d]*Email: nlerumpa@fuotuoke.edu.ng*

## Abstract

A Group signature protocol is a cryptographic scheme that decouples a user identity and location from verification procedure during authentication. In a group signature scheme, a user is allowed to generate signatures on behalf of other group members but identity and location information of the signer is not known by a verifier. This ensures privacy, authentication and unlinkability of users. Although group signature is expensive to implement, its existential anonymity, non-repudiation and untraceablility properties make it attractive especially for resources-constrained devices in wireless network. A general group signature scheme usually contains six basic phases: setup (or key generation), join, message signing (or signature generation), signature verification, open and user revocation. In this paper, an evaluation of the performance of group signature based on three of the phases mentioned above is considered and its security in wireless networks examined. The key generation, signing and verification algorithms are implemented in Java 8. A proof of security of group signature by implication is also presented.

***Keywords:*** Wireless network; authentication; security; anonymity; untraceability; group signature.

## 1. Introduction

A Group Signature (GS) scheme is a method for allowing a member of a group to anonymously sign a message on behalf of the group. The concept was first introduced by David Chaum and Eugene van Heyst [1] in 1991.

------------------------------------------------------------------------

* Corresponding author.

Group signatures can be used in many privacy-preserving services and authentication schemes. Group signatures can be understood as a subset of attribute authentication systems which contain only one attribute representing a membership in a group. A user who is a member of a group can sign a message on behalf of the group and send the message anonymously to a verifier. According to [1], a secure group signature scheme should satisfy two basic requirements, anonymity and traceability. Anonymity stipulates that the identity of the signer should remain unknown to anyone verifying the signature including other group elements. On the other hand, traceability requires that there should be an entity, called the group manager, capable of revoking the anonymity of signer whenever necessary. Elliptic Curve Digital Signature Algorithm (ECDSA) is a variant of Digital Signature Algorithm (DSA) used extensively in a vast area of applications and across many different fields to verify the authenticity of messages and confirm that they have not been altered in transmission. ECDSA is a version of DSA using elliptic curves. ECDSA has been proven to be more effective than using DSA as it provides the same security with a smaller key size [2]. The two schemes (GS and ECDSA) are public key algorithms. Public key cryptography provides two keys for authentication. These keys are public key and private key. In terms of digital signatures (including ECDSA), the private key is used for creating signatures and the public key is copied and handed out to validate signatures. For GS, the group members each have differentiated private keys used for signing and a common group public key for verifying the signatures made available to all verifiers. Applications that benefit from group signatures are: vehicle safety communications (VSC) [12] system to preserve the privacy of its users, anonymous attestation (e.g. DAA-Direct Anonymous Attestation [13]), bidding [6], electronic cash [14], anonymous fingerprinting [15] etc.

### *1.1. Properties of a Group Signature Scheme*

Group signature schemes usually provide the following properties:

- Unforgeability - only an unrevoked group member can create a valid signature on behalf of the group.
- Anonymity - a verifier is not able to determine the identity of a signer.
- Complete anonymity - if an attacker obtains a valid signature and knows *gpk* and all keys of group members' *gsk*[i], he is not able to determine the identity of a signer.
- Traceability - all members can be tracked by the group manager or the revocation manager by member's signed message.
- Untraceability - any member cannot be tracked by a verifier and/or other group members by his/her signed messages.
- Unlinkability - a verifier and other members are not able to link two signatures which have been signed by one member of the group.
- Coalition-resistance - it is impossible to create a valid signature by a subgroup of users.
- Exculpability - even group manager is not able to create the valid signature of a group member.
- Correctness - every correct signature of a group member has to be always accepted during verification.
- Revocation - a revoked member is not able to create valid signatures on behalf of the group.
- Differentiation of group members - all members of a group must have a different gsk[i].
- Immediate-revocation - if a group member is revoked, his capability of creating the group signatures is

immediately disabled.

### *1.2. Security Assumptions of Group Signature*

One of the most critical requirements in cryptographic research is identifying (strongest and/or weakest) assumptions required for the construction of secure primitives. This helps to close the gap between which primitive is sufficient and what is necessary to build a given cryptographic function such as a group signature to determine the exact conditions that must be met for them to exist.

Several implications and separations are known in literature for primitives such as standard signatures (DSA, ECDSA, RSA, etc.) and public-key encryption, very little is known for group signatures despite the intuition that the latter appears to be a stronger primitive than standard signatures [4].

Apart from the original work of [1], many other schemes have been proposed in the literature (e.g., [5, 6, 7]). Each of these has its own set of security properties and requirements. Recently, a formal model of security for group signatures was put forward [8], which integrated the many sets of security requirements into two basic categories, called full-anonymity and full-traceability.

These two basic properties were shown to imply in the case of static groups all of the existing security properties of previous scheme. Formal definitions for dynamic groups were also provided in subsequent works [9, 10]. The significance of such formal definitions includes concrete and simpler proofs of security (only two properties need be satisfied) and better understanding a group signature scheme being secure with its implications. Another benefit is that precise relations between group signatures and other cryptographic primitives can be drawn. The implications proven in [4] are only possible in the presence of such formal models of security. In this paper, we show that the group signature cryptographic primitive is secure by implied security of the constituent primitives used to build the group signature scheme.

## 2. Protocol Specification

The group manager computes initialization parameters. The parameters include

(i) H selects a random number $\gamma \in \mathbb{Z}$ and computes a group manager private key (gmpk) computed as:

$$gmsk = \gamma \quad (1)$$

(ii) The group manager public key (gmpk) is given as,

$$gmpk = (g_1, g_2, h_1, h_2, \dots h_T, w) \quad (2)$$

where given $Fq$ a finite field with an elliptic curve $E$, $G_1$ a multiplicative cyclic group of prime order p and $G_2$ a multiplicative group of exponent p, with some power of p as its order, $g_1$ is a generator of $G_1$ and $g_2$ is an order-$p$ element of $G_2$. The elements $g_1$ and $g_2$ will be selected at random as part of system setup. $h_1, h_2, \dots h_T$

represent randomly selected $h_j \in G$ for each interval j and $w$ is given as $w = g_2{}^{\gamma}$.

(iii) H uses the above computed parameters to generate a vector of N user's secret keys *usk* and a vector of N x T revocation tokens (*urt*) for each registered user with current time intervals to ensure unlinkability as follows:

$usk$ = ($usk$[1], $usk$[2], …, usk[$N$]) (3)

$urt$ = ($urt$[1][1],…, $urt$[1][T],…, $urt$[$N$][$T$]) (4)

(iv) The user secret key for $N$ users is given as:

$usk[i] = (A_i, x_i)$ (5)

Where $Ai = g_1{}^{1/(\gamma+x_i)}$ for all i $\in [1, N]$ and $x_i \in \mathbb{Z}$ is selected randomly.

(v) Next, randomly selected $h_j \in [1, T]$ is used to compute the revocation token at time interval $j$ of $user_i$ with secret key $(A_i, x_i)$ as:

$B_{ij} = h_j^{x_i}$ for all i $\in [1, N]$ and j $\in \mathbb{N}$ (6)

(vi) H computes an alias for each registered user intending to roam using secret splitting mechanism from:

$alias_U = (w||ID_H) \oplus ID_U \oplus ID_H$ (7)

Where $ID_H$ is the group manager identity, $ID_U$ is the identity of the user, U and $\oplus$ is an Exclusive-Or operator. In the authentication phase a group signature with revocation support is illustrated as follows:

(i) User selects a random number, $N_U$ and computes $N_U G$ (we assume $G_1 = G_2 = G$, from bilinear map) and sends $(ID_H, N_U G)$ to $V$.
(ii) Verifier, $V$ selects a random number $N_V$, computes a session key used between $V$ and U as follows

$k_{VU} = N_V(N_U G)$ (8)

and sends $N_V$ to U.

(iii) U computes a group signature $pk_{V_1}$ by running group signature $\sigma$ by executing the signing algorithm, G.Sig(gmpk, *gsk*[i], j, alias, $N_{V_1}G$) and use it to sign a message sent to *V*.

It then computes a temporary alias by encrypting $alias_U$ given to it by group manager using its session key $k_{UV}$. Then it sends (alias, $\sigma_u$) to $V$. Otherwise, if the signature $\sigma_{V_1} = 0$, connection rejected.

(iv) $V$ verifies the signature from U with the group manager public key, *gmpk* by running the verification algorithm, G.Ver(gmpk, usk[i], j, alias, $N_V$G, $\sigma_u$).

$\sigma_U = \begin{cases} 1, & \sigma\ valid, \text{allow}\ connection \\ 0, & otherwise\ \text{disallow} \end{cases}$ $V$ allows connection if $\sigma_U$ =1 and disallow otherwise.

## 3. Group Signature Algorithms

### *3.1. Key Generation Algorithm*

This algorithm takes as input integers $N, T \in \mathbb{N}$ indicating the number of subscribers (users) and the number of time intervals, respectively.

```
VLR-GS.KenGen (N, T)
select randomly a generator $g_2 \in G_2$
set $g_1 = \psi(g_2)$
select randomly $\gamma \in \mathbb{Z}_p$
set $w = g_2^{\gamma}$
set $hsk = \gamma$
for i = 0 to N – 1 // generate an SDH tuple ($A_i$ , $x_i$ )
select randomly $x_i \in \mathbb{Z}$ // $x_i + \gamma$ must be nonzero
set $A_i = g_1^{1/(\gamma+x_i)}$
set $alias_i = (w||ID_H) \oplus ID_i \oplus ID_H)$
for j = 0 to T – 1 // generate time intervals for user$_i$
select $h_j \in G$ // to ensure backward unlinkability
set $B_{ij} = h_j^{x_i}$
set $musk[i] = (A_i, x_i)$
set $murt[i][j] = B_{ij}$
end for
end for
```

```
set hspk = ($g_1$, $g_2$, $h_1$, …, $h_T$, $w$)
end
```

### *3.2. Signing Algorithm*

The inputs to this signing algorithm are a home server public key, *hspk* = ($g_1$, $g_2$, $h_1$, …, $h_T$, $w$), the current time interval $j$, the mobile user secret key, *musk*[$i$] = ($A_i$, $x_i$) and a signed message $M \in \{0, 1\}^*$ assumed to include all time intervals $j$ in order to bind the signature to the interval.

```
VLR-GS.Sig (hspk, j, usk[i], M)
select randomly a generator $g_2 \in G_2$
set $g_1 = \psi(g_2)$
compute SPK $V$
select random number $\alpha, \beta, \delta \in \mathbb{Z}_p$
set $\varepsilon = x_i\alpha,\ \zeta = x_i\beta,\ \eta = x_i\delta$
compute $T_1 = A_i g_2^{\alpha}$ and $T_2 = g_1^{\alpha} g_2^{\beta},\ T_3 = e(g_1^{x_i}, h_j)^{\delta}$ and $T_4 = g_1^{\delta}$
select $r_\alpha, r_\beta, r_\delta, r_{x_i}, r_\varepsilon, r_\zeta, r_\eta \in \mathbb{Z}_p$ //blinding factors to compute SPK
compute $R_1 = g_1^{r_\alpha} g_2^{r_\beta},\ R_2 = T_2^{r_{x_i}} (\frac{1}{g_1})^{r_\varepsilon} (\frac{1}{g_2})^{r_\zeta}$,
compute $R_3 = \left(\frac{1}{e(T_1, g_1)}\right)^{r_{x_i}} e(g_2, w)^{r_\alpha} e(g_1, g_2)^{r_\varepsilon}$
compute $R_4 = e(g_1, h_j)^{r_\eta},\ R_5 = g_1^{r_\delta}$
compute $R_6 = T_4^{r_{x_i}} (\frac{1}{g_1})^{r_\eta}$
compute $c = H(gpk, j, M, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4, R_5, R_6)$
compute $s_\alpha = r_\alpha + c\alpha,\ s_\beta = r_\beta + c\beta,\ s_\delta = r_\delta + c\delta,\ s_{x_i} = r_{x_i} + cx_i$
compute $s_\varepsilon = r_\varepsilon + c\varepsilon,\ s_\zeta = r_\zeta + c\zeta,\ s_\eta = r_\eta + c\eta$
compute $\sigma = (T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_\delta\ s_{x_i}, s_\varepsilon, s_\zeta, s_\eta)$
```

```
output $\sigma$ // as the signature
end
```

***3.3. Verification Algorithm***

The input to this verification algorithm include a home server public key *gpk* = ($\boldsymbol{g_1}$, $\boldsymbol{g_2}$, $\boldsymbol{h_1}$, …, $\boldsymbol{h_T}$, *w*), the current time interval *j*, a revocation list $\boldsymbol{RL_j}$ that consists of *murt*[*i*][*j*] for all revoked *i* at the time interval *j* and a signature $\boldsymbol{\sigma}$ = ( $\boldsymbol{T_1}$, $\boldsymbol{T_2}$, $\boldsymbol{T_3}$, $\boldsymbol{T_4}$, *c*, $\boldsymbol{s_\alpha}$ , $\boldsymbol{s_\beta}$, $\boldsymbol{s_\delta}$ $\boldsymbol{s_{x_i}}$, $\boldsymbol{s_\varepsilon}$ , $\boldsymbol{s_\zeta}$, $\boldsymbol{s_\eta}$).

```
VLR-GS.Ver(gpk, RLj , $\sigma$ , M).
select randomly a generator $g_2 \in$ G2
set $g_1 = \psi(g_2)$
check SPK $V$
set $R_1' = g_1^{s_\alpha} g_2^{s_\beta} (1/\mathrm{T}_2)^c$    // Recomputed $R_1$, $R_2$, $R_3$
set $R_2' = T_2^{s_{x_i}} (1/\mathrm{g}_1)^{s_\varepsilon} (1/\mathrm{g}_2)^{s_\zeta}$
set $R_3' = \left(\frac{1}{e(T_1,g_1)}\right)^{s_{x_i}} e(g_2,w)^{s_\alpha} e(g_1,g_2)^{s_\varepsilon} \left(\left(\frac{1}{e(T_1,g_1)}\right) e(g_1,g_1)\right)^c$
set $R_4' = e(g_1,h_j)^{s_\eta} (1/\mathrm{T}_3)^c$
set $R_5' = g_1^{s_\delta} (1/\mathrm{T}_4)^c$
set $R_6' = T_4^{s_{x_i}} (1/\mathrm{g}_1)^{s_\eta}$
if $c = H(gpk, j, M, T_1, T_2, T_3, T_4, R_1', R_2', R_3', R_4', R_5', R_6')$// compare with c in $\sigma$
Output "signature valid" else "signature invalid"
for j = 1 to T //revocation check in RLj
if $T_3 \neq e(T_4, B_{ij})$
output "user not revoked"
allow access
```

```
else
output “user revoked”
deny access
endif
end for
end
```

## 4. ECDSA Scheme Algorithms

### *4.1. ECDSA Key Pair Generation Algorithm*

```
ECDSA.KeyGen
Input $G$, n // base point on $E\backslash\mathbb{F}_q$, $n$ is order of G
Randomly select $d \in \mathbb{Z}_p$ in the interval [1, $n$ -1]
compute Q = d$G$
set public key = Q
set private key = d
output public key, private key
end
```

### *4.2. ECDSA Signature Generation Algorithm*

```
ECDSA.SignGen
Input: $d$, $G$, $n$, hash function H, and message m
Output: Signature (r; s)
compute random integer $k$, within $1 \leq k \leq n-1$
compute $kG = (x_1, y_1)$
convert $x_1$ to an integer $x_1'$
```

```
compute $r$ = $x_1'$ mod $n$
If $r$ = 0 then go to step 1
compute $H$(m) and convert this bit string to an integer $e$
compute s = $k^{-1}$(e + dr) mod n.
If s = 0 go to step 1
output signature = ($r$, s) end
```

***4.3. ECDSA Signature Verification Algorithm***

```
ECDSA.Verify
Input: ($r; s$), $m, n, e, G, Q$, and hash function $H$
Output: Accept or reject signature (r; s)
verify $r$ and $s$ are integers in the interval [1, $n$ - 1]
compute $H$($m$) and convert bit string to an integer $e$
compute $w$ = $s^{-1}$ mod n
compute $u_1$ = $ew$ mod $n$ and $u_2$ = $rw$ mod n
compute X = $u_1$G + $u_2$Q
If X = $O$(the point at infinity), reject the signature.
Otherwise, convert $x_1$of X to an integer $x_1'$
compute $v$ = $x_1'$ mod n
accept the signature if and only if $v = r$
end.
```

**5. Implementation and Results**

To implement the group signature framework developed in this paper, we analyze the group signature model because the model differs from conventional digital signature models including ECDSA, ElGammel, RSA signature models. This difference is due to the number of parties participating in the protocol. Unlike the schemes stated above, group signature involves more members (group manager, group members and verifiers). Each member of a group can sign on behalf of a group with its identity unknown by any verifier. This requires each member in the group to have a different private key but use same public key. Hence there are three distinct processes in the scheme i.e. signing, group management, and signature verification. The group management function is comprised of key generation, revocation and open. So in this paper, each of the functionalities is handled by a specialized java class as shown in Table 2. Even though key generation and issuance, revocation, and opening are carried out by one entity in protocol run, we treat these operations as being as being the responsibility of three different servers. We implemented the group scheme in this paper using Java as a programming language and Eclipse as IDE and JUnit (a unit testing framework for the Java programming language) for testing. The tools used for the development of this project are shown in table 1:

**Table 1:** Tools used for implementation

| Tool | Version |
|---|---|
| Eclipse | Neon 2 |
| Java JDK | 8 |
| JUnit | 4 |

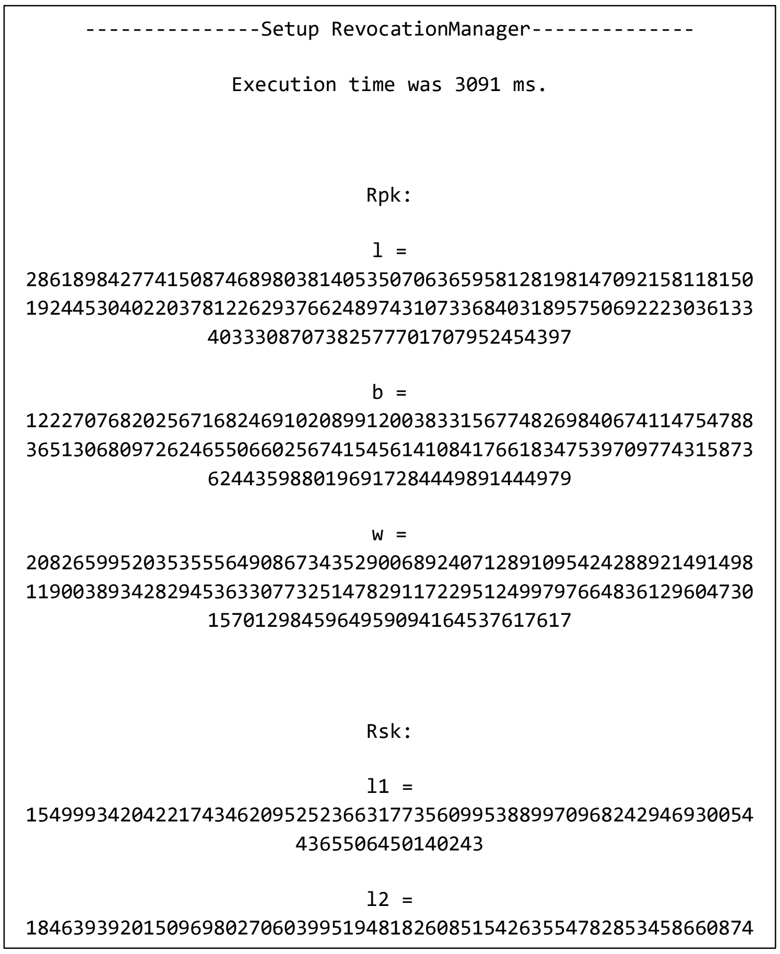
**Table 2:** Java classes for the implementation

| Package | Classes in Package |
|---|---|
| groupsignature.client | User.java<br>Verifier.java |
| groupsignature.interface | CommentsGUI.java<br>Main.java |
| groupsignature.elliptic | ECParameters.java<br>ECPoint.java<br>EllipticCurve.java<br>InsecureCurveException.java<br>NoCommonMotherException.java<br>NotOnMotherException.java<br>secp112r1.java<br>secp160r1.java<br>secp256r1.java |
| groupsignature.signature | RevocationCertificate.java<br>Signature.java |
| groupsignature.server | IssuingManager.java<br>OpeningManager.java<br>RevocationManager.java |

The hardware platform for testing is made up of the following configuration: a laptop with Pentium(R) Dual Core CPU T4400 @ 2.20Ghz, 6GB RAM, L2 cache size 4MB with 64 bits Windows 8 operating system.

## 6. Group Signature Performance

### *6.1. Implementation Results*

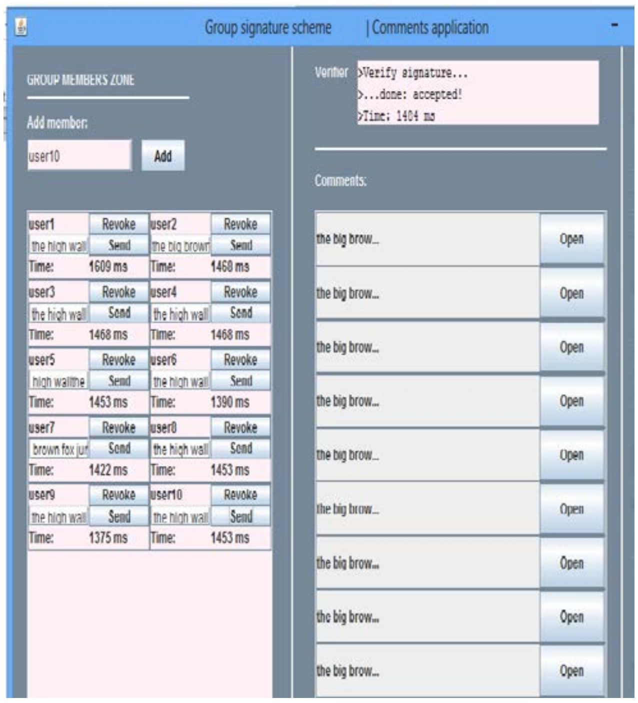


```
---------------Setup RevocationManager--------------

Execution time was 3091 ms.



Rpk:

l =
28618984277415087468980381405350706365958128198147092158118150
19244530402203781226293766248974310733684031895750692223036133
40333087073825777017079524543 97

b =
12227076820256716824691020899120038331567748269840674114754788
36513068097262465506602567415456141084176618347539709774315873
62443598801969172844498914449 79

w =
20826599520353555649086734352900689240712891095424288921491498
11900389342829453633077325147829117229512499797664836129604730
15701298459649590941645376176 17



Rsk:

l1 =
15499934204221743462095252366317735609953889970968242946930054
4365506450140243

l2 =
18463939201509698027060399519481826085154263554782853458660874
```

**Figure 1:** output screen

The main interface is shown in figure 2.

An examination of the results as can be found in the output of figure 1 showed that the different algorithms execution took variable amount of time. The following table shows the execution time (in milliseconds) of each operation.



user10

```
>Verify signature...
>...done: accepted!
>Time: 1404 ms
```

| | | | |
|---|---|---|---|
| user1 | the high wall | Time: | 1609 ms |
| user2 | the big brown | Time: | 1468 ms |
| user3 | the high wall | Time: | 1468 ms |
| user4 | the high wall | Time: | 1468 ms |
| user5 | high wallthe | Time: | 1453 ms |
| user6 | the high wall | Time: | 1390 ms |
| user7 | brown fox jur | Time: | 1422 ms |
| user8 | the high wall | Time: | 1453 ms |
| user9 | the high wall | Time: | 1375 ms |
| user10 | the high wall | Time: | 1453 ms |

the big brow...
the big brow...
the big brow...
the big brow...
the big brow...
the big brow...
the big brow...
the big brow...
the big brow...

**Figure 2:** main interface

**Table 3:** Operations and their execution times

| Operation | Execution Time |
|---|---|
| Key generation and issuance | 329 ms |
| Opening a signature to determine its signer | 172 ms |
| Revocation | 3091 ms |
| Join (1st User) | 1735 ms |
| (10th User) | 283 ms |
| Signing message to be sent | 1609 ms |
| verification | 1515 ms |

It was however observed that in the signing phase, the execution times remained almost constant with negligible variability. The same observations were made concerning verifications of the signatures. The size of messages had no effect on the signing and verification times. As can be noticed from table 3, the running time of the Join algorithm seems to go downward as more members join the group. This is depicted in the graph in figure 3.

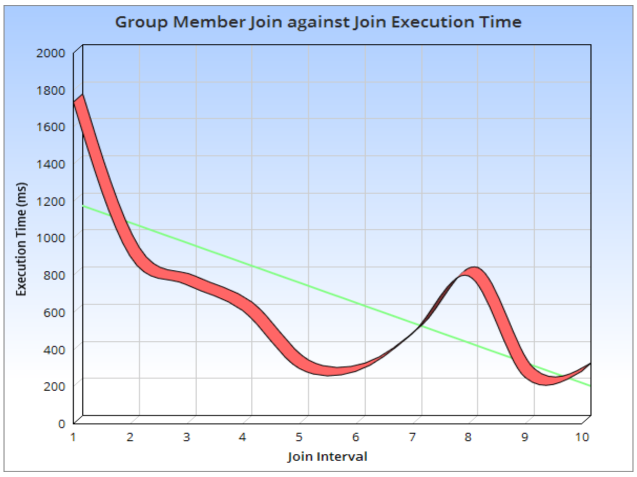


**Figure 3:** Join interval plot against join algorithm execution time.

### *6.2. Group Signature vs ECDSA*

We compared group signature with ECDSA signature scheme in general using the desirable security requirements of a digital signature scheme as discussed in section 1.1. This is shown in Table 4.

**Table 4:** Group signature vs ECDSA

| Property | GS | ECDSA |
|---|---|---|
| Unforgeability | yes | yes |
| Anonymity | yes | no |
| Untraceability | yes | no |
| Traceability | yes | yes |
| Unlinkability | yes | no |
| Coalition-resistance | yes | yes |
| Exculpability | yes | yes |
| Immediate-revocation | yes | - |
| Correctness | yes | yes |
| Revocation | yes | - |

## 7. Security Proof

Suppose we have an arbitrary group signature scheme GS = (GKg; GSig; GVf; Open) with number of signers only two, i.e. {user0, user1}, we describe a public encryption scheme where the public key for verification is *gpk* and the secret keys for signing by user0 and user1 are *gsk*[0] and *gsk*[1] respectively. These keys are related to the group manager secret key, *gmsk* used for revocation and opening, and the group manager public key *gmpk*. The group signature, $\sigma$ is the encryption of message thus:

$$M = b_0 b_1 \ldots b_n \text{ with } b_i \in \{0, 1\} \tag{9}$$

which is done bitwise. To decrypt the encryption, $\sigma$ of the bit stream *b*, we simply verify that $\sigma$ is a valid group signature, and if so use the group manager's secret key to recover the identity of the signer. This also applies to arbitrary length messages. We proof the security of group signature as implying the security of public key cryptography.

Let B be an attacker seeking to break the indistinguishability under chosen plaintext attack (*IND-CPA*) security of the encryption scheme A $\mathcal{E}[GS]$ used for encryption which results in $\sigma$. We construct an adversary A against the group signature scheme GS such that

$$\mathbf{Adv}_{AE;A}^{\mathbf{ind-cpa}}(\boldsymbol{k}) \leq \boldsymbol{p}\mathrm{A}(\boldsymbol{k}).\,\mathbf{Adv}_{\mathbf{GS};B}^{\mathbf{anon}}(\boldsymbol{k}, \mathbf{2}) \tag{10}$$

where pA (k) means that adversary A runs in polynomial time bound. A $\mathcal{E}$ is an IND-CPA secure encryption scheme assuming GS is fully-anonymous, the function on the right-hand side of the inequality is negligible. A runs the guess stage of algorithm B for encryption scheme A $\mathcal{E}$ and obtains two messages $m_0$ and $m_1$. These messages, together with the state information output by B is forwarded to the choose stage of A. In this stage, A selects at random a position *j* on which $m_0$ and $m_1$ are different, and creates a challenge ciphertext for B. The challenge ciphertext is an encryption (gpk; gsk) of a word which on its first (j – 1) positions coincides with $m_1$ and on its last n - j positions coincides with $m_0$, where $n = |m_0| = |m_1|$. The bit b on position j in the plaintext encrypted by the challenge ciphertext is precisely the identity of the player that generated the challenge signature $\sigma$ which A received from its environment.

Given messages $m_0$ and $m_1$ and $s_0, \ldots,$ sp denotes a sequence of $p = |\mathrm{diff}(m_0,m_1)|$ words such that $s_0 = m_0$, $s_p = m_1$, and any two consecutive words $s_{i-1}$ and $s_i$ differ exactly in one bit position. More precisely, let j be the element of rank i in $\mathrm{diff}(m_0,m_1)$. We can construct word $s_i$ from word $s_{i-1}$ by flipping the j-th bit of $s_{i-1}$, for i = 1, …, p. Now, let *i* be the rank of the value j selected by A during the **choose** stage of A. Therefore, adversary B receives as challenge either the encryption of $s_{i-1}$ or the encryption of $s_i$, depending on the key used to create challenge signature $\sigma$. With this in mind, notice that in the experiment $\mathbf{Exp}_{\mathbf{GS};A}^{\mathbf{anon-2}}(\boldsymbol{k}, \mathbf{2})$ (for $b \in \{0, 1\}$), adversary A successfully guesses the bit b whenever adversary B correctly identifies if the challenge ciphertext is the encryption of $s_{i-1}$ or that of $s_i$. To simplify notation, we will write B (Enc(pk, $s_i$)) for B (guess, St, Enc((*gpk*, *gsk*), $s_i$)). It follows from the above discussion that

$$\Pr[\mathrm{Exp}_{GS;A}^{\mathrm{ianon-0}}(k, 2) = 1] = \frac{1}{|\mathrm{diff}(m_0, m_1)|} \sum_{\boldsymbol{i=1}}^{|\mathrm{diff}(m_0,m_1)|} \Pr\left[B\left(\mathrm{Enc}(\mathrm{pk}, s_{i-1})\right) = 1\right]$$

and

$$\Pr[\mathrm{Exp}_{GS;A}^{\mathrm{ianon}-1}(k,2)=1]=\frac{1}{|\mathrm{diff}(m_0,m_1)|}\sum_{\boldsymbol{i=1}}^{|\mathrm{diff}(m_0,m_1)|}\Pr\big[B\big(\mathrm{Enc}(\mathrm{pk},s_i)\big)=1\big]$$

where the first factor represents the probability that the value j selected by A has rank $i$. Let p = |diff(m$_0$,m$_1$)|. We can now bind the advantage of A by:

$$\mathbf{Adv}_{\boldsymbol{AE;A}}^{\mathbf{ind-cpa}}(\boldsymbol{k},\mathbf{2})=\Pr[\mathrm{Exp}_{GS;A}^{\mathrm{ianon}-1}(k,2)=1]-\Pr[\mathrm{Exp}_{GS;A}^{\mathrm{ianon}-0}(k,2)=1]$$

$$=\frac{1}{\mathrm{p}}.\sum_{\mathrm{i}=1}^{\mathrm{p}}\Pr\big[B\big(\mathrm{Enc}(\mathrm{pk},s_i)\big)=1\big]-\frac{1}{\mathrm{p}}.\sum_{\mathrm{i}=1}^{\mathrm{p}}\Pr\big[B\big(\mathrm{Enc}(\mathrm{pk},s_{i-1})\big)=1\big]$$

$$=\frac{1}{\mathrm{p}}.\sum_{\mathrm{i}=1}^{\mathrm{p}}(\Pr\big[B\big(\mathrm{Enc}(\mathrm{pk},s_i)\big)=1\big]-\Pr\big[B\big(\mathrm{Enc}(\mathrm{pk},s_{i-1})\big)=1\big])$$

$$=\frac{1}{\mathrm{p}}.(\Pr\Big[B\Big(\mathrm{Enc}\big(\mathrm{pk},s_p\big)\Big)=1\Big]-\Pr\big[B\big(\mathrm{Enc}(\mathrm{pk},s_0)\big)=1\big])$$

$$=\frac{1}{\mathrm{p}}.(\Pr\big[B\big(\mathrm{Enc}(\mathrm{pk},m_1)\big)=1\big]-\Pr\big[B\big(\mathrm{Enc}(\mathrm{pk},m_0)\big)=1\big])$$

$$=\frac{1}{\mathrm{p}}.\mathrm{Adv}_{A\mathcal{E},B}^{\mathrm{ind-cpa}}(\boldsymbol{k})$$

$$=\frac{1}{|m_0|}.\mathrm{Adv}_{A\mathcal{E},B}^{\mathrm{ind-cpa}}(\boldsymbol{k})$$

We can also bind the length of $m_0$ by the total running of algorithm A, which is some polynomial $p\mathrm{A}(.)$ in the security parameter. As a result,

$$\mathrm{Adv}_{\mathrm{GS},B}^{\mathrm{anon}}(k,2)\ \geq\ \frac{1}{p\mathrm{A}(k)}\ .\mathrm{Adv}_{A\mathcal{E},B}^{\mathrm{ind-cpa}}(k)$$

This gives the result claimed in Equation 10 by rearranging the terms.

## 8. Recommendation

It is a matter of utmost importance that in designing a security protocol suitable for deployment in an unsecure environment such as the wireless networks, attention must be given to determining the exact conditions that must be met for a secure protocol design. Hence there should be a remarkable distinction between what is known to be sufficient to construct secure and efficient group signatures (considering device limitations) and what is known to be necessary. This can be achieved by closing the gab existing between which primitive is sufficient and what is necessary to build a given cryptographic function such as encryption or group signatures. In this scheme, it is additionally recommended that γ should be erased from group master’s storage after the key generation process and can be stored somewhere outside the group master *H*. This is possibly due to the fact that the private key is rarely used except in the case of a legal warrant for purposes of privacy revocation.

## 9. Conclusion

Group signature has been shown to be a secure primitive existentially providing anonymity, addressing traceability and unlinkability. Other properties of group signature found desirable in this paper include nonrepudiation, unforgeability, and exculpability. The main advantage of proving that the existence of secure group signature schemes implies public-key encryption schemes is that one can apply several of the results that are known for public-key encryption to the case of group signatures. However, group signature schemes have some limitation which is transferred to schemes using group signatures. In the event of exposure of a member's signing key probably due to a compromise of the underlying storage system or human errors, this danger may escalate as the group size increases.

## References

[1] Chaum, D. and van Heyst, E. Group signatures. In D. W. Davies, editor, Advances in Cryptology, EUROCRYPT 1991 (Lecture Notes in Computer Science 547), pages 257–265. Springer-Verlag, April 1991. Brighton, U.K.

[2] Khalique, A. Singh, K. Sood, S. ”Implementation of Elliptic Curve Digital Signature Algorithm”, May 2010.Web.http://www.ijcaonline.org/volume2/number2/pxc387876.pdf.

[3] Blumenthal, Matt. ”Encryption: Strengths and Weaknesses of Public key Cryptography”, Web. http://www.csc.villanova.edu/_tway/courses/csc3990/f2007/csrs2007/01-pp1-7-MattBlumenthal.pdf.

[4] Abdalla M., Warinschi B. (2004) On the Minimal Assumptions of Group Signature Schemes. In: Lopez J., Qing S., Okamoto E. (eds) Information and Communications Security. ICICS 2004. Lecture Notes in Computer Science, vol 3269. Springer, Berlin, Heidelberg

[5] Ateniese, G. Camenisch, J. Joye, M. and Tsudik, G. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, Advances in Cryptology - CRYPTO 2000, volume 1880 of Lecture Notes in Computer Science, pages 255-270, Santa Barbara, CA, USA, Aug. 20-24, 2000. Springer-Verlag,Berlin, Germany.

[6] Chen, L. and Pedersen, T. P. New group signature schemes. In A. D. Santis, editor, Advances in Cryptology, EUROCRYPT'94, volume 950 of Lecture Notes in Computer Science, pages 171-181, Perugia, Italy, May 9-12, 1994. Springer-Verlag, Berlin, Germany.

[7] Camenisch, J. Efficient and generalized group signatures. In W. Fumy, editor, Advances in Cryptology EUROCRYPT'97, volume 1233 of Lecture Notes in Computer Science, pages 465{479, Konstanz, Germany, May 11{15, 1997. Springer-Verlag, Berlin, Germany.

[8] Bellare, M. Micciancio, D. and Warinschi, B. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, Advances in Cryptology { EUROCRYPT 2003, volume 2656 of Lecture Notes in Computer Science, pages 614{629, Warsaw, Poland, May 4{8, 2003. Springer-Verlag, Berlin, Germany.

[9] Kiayias, A. and Yung, M. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. http://eprint.iacr.org/.

[10] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. Cryptology ePrint Archive, Report 2004/077, 2004. http://eprint.iacr.org/.

[12] IEEE P1556 Working Group, VSC Project. Dedicated short range communications (DSRC), 2003

[13] E. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation, Oct. 2004.

[14] A. Lysyanskaya and Z. Ramzan. “Group blind digital signatures: A scalable solution to electronic cash”. In Proc. Financial Cryptography, 1998.

[15] J. Camenisch. “Efficient anonymous fingerprinting with group signatures”. In ASIACRYPT 2000, vol. 1976 of LNCS, pp. 415{428. Springer Verlag, 2000.