Journal of Mobile, Embedded and Distributed Systems, vol. IV, no. 3, 2012
ISSN 2067 – 4074

About Group Digital Signatures

Adriana-Cristina ENACHE
Military Technical Academy
Bucharest,
ROMANIA
adryanaenache@gmail.com

Abstract: Group signatures try to combine **security** (no framing, no cheating) and **privacy**(anonymity, unlinkability).A **group digital signature** is a digital signature with enhanced privacy features that allows members of a given group to anonymously sign messages on behalf of the group, producing a group signature. However, in the case of dispute the identity of the signature's originator can be revealed by a designated entity (group manager). The present paper describes the main concepts about group signatures, along with a brief state of the art and shows a personal cryptographic library implemented in Java that includes two group signatures.

Key-words: group digital signature, privacy, identification, ACJT signature, CG signature.

1. Introduction

Privacy is acknowledged and protected as a fundamental and inviolable right by many national and international laws. Even if it is a concept hard to clearly define: it is often used as a base term with many dimensions such as identity, location and personal data privacy, and meanings that are very context-dependent. Furthermore, many systems that we use every day were not designed at all for privacy protection, and most of the times, security functionality are added as an afterthought.

Most frequently, **identification** is a disclose-all-or-nothing procedure. During our everyday life we are requested to identify ourselves, either digitally or not, and most of the times we must reveal a great deal of additional details, even if the only relevant information required is, for instance, the membership of a particular group. Imagine a big company where employees are asked to identify every time they move around the company site, to check if they have authorized to move to a new area (e.g. a lab or a department) or to get access to different equipments inside the company (e.g. printers etc.).

Group signatures allow a member of some group to sign anonymously on the behalf of the group. Thus, a party receiving a signature cam be sure that its originator is a member of the group, but receives no other additional information. However, in exceptional cases such as when the anonymity is misused and a legal dispute arises, a designated revocation manager can reveal the unambiguous identity of the originator of the signature.

These type of signatures provide both privacy and security, without hindering the other. Group signatures are used in **applications** where the recipient only needs to know that the signature came from a group, and revealing which specific group member formed the signature does not offer any advantage. Applications that benefit from group signatures are: vehicle safety communications (VSC) [8] system to preserve the privacy of its users, anonymous attestation (e.g. DAA-Direct Anonymous Attestation[7]), bidding [12], electronic cash [13], anonymous fingerprinting [14] etc.

Ateniese et al.[3] proposed the first practical group signature based on strong RSA assumption. Later it is improved and proved in a formal model[4]. The drawback of this signature is that it does not allow member revocation. The Camenisch and Groth[19] group signature is based on the ACJT group signature, but in addition

it offers the possibility of member revocation in the VLR[1] model.
This paper describes the main concepts of group signatures and a personal implementation of the ACJT and CG group signatures in Java that supports integration with the JCA (Java Cryptography Architecture).

2. Group Digital Signatures

2.1. Overview

Group digital signatures scheme deal with a group whose users are called *members* (or players) and most of the time a *group manager* (or group leader) who is the designated authority with the ability to "open" a group signature in case of a dispute and reveal the identity of the actual signer, a member of the group.
A group signature scheme is composed of the following steps:(1)The group manager, GM, along with some third trusted party, chooses the security parameters as well as a group secret key and a group public key. (2)Any group member candidate is required to choose his member secret key, and run an interactive protocol with GM to join in the group, during which GM generates a signature on the member secret key blindly, i.e., not knowing the secret key value, the signature is also called member certificate. (3)Any group member can generate group signatures using his group signing key which includes member secret key and member certificate.
A common paradigm of constructing group signatures[3] is as follows: the group manager adopts an ordinary signature scheme to generate membership certificate for group members, i.e., sign on some secret key known only to members. The group signature is in fact a **non-interactive zero-knowledge** proof of knowledge of member certificate and member secret key, transformed in *Fiat-Shamir's heuristic method* [5] from interactive proofs.

2.2. Procedures, security and efficiency

A group signature scheme[2] *is a digital signature scheme with enhanced privacy features and is comprised of the following five* ***procedures****[3]:*
SETUP: *A probabilistic algorithm which on input of a security parameter outputs the initial group public key* Y *(including all system parameters) and the secret key* S for the group manager.
JOIN**:** *A protocol between the group manager and a user that results in the user becoming a new group member. The user's output is a membership certificate and a membership secret*.
SIGN**:** *A probabilistic algorithm that on input a group public key, a membership certificate, a membership secret, and a message m outputs the group signature of the message.*
VERIFY: *An algorithm for establishing the validity of an alleged group signature of a message with respect to a group public key.*
OPEN: *A procedure that, given a message, a valid group signature on it, a group public key and a group manager's secret key, determines the identity of the signer.*
Some group signature schemes also support a ***REVOKE*** procedure that allows the member to be revoked from the group.
A **secure** group signature scheme must satisfy the following ***properties***[3]:
Correctness: Signatures produced by a group member using SIGN must be accepted by VERIFY.
Unforgeability: Only group members are able to sign messages on behalf of the group.
Anonymity: Given a valid signature of some message, identifying the actual signer is computationally hard for everyone but the group manager.
Unlinkability: Deciding whether two different valid signatures were computed by the same group member is computationally hard.
Exculpability: Neither a group member nor the group manager can sign on behalf of other group members.

[1] Verifier Local Revocation

ISSN 2067 – 4074

Traceability: The group manager is always able to open a valid signature and identify the actual signer. Therefore, any colluding subset of group members cannot generate a valid signature that the group manager cannot link to one of the colluding group members.
Coalition-resistance: A colluding subset of group members (even if comprised of the entire group) cannot generate a valid signature that the group manager cannot link to one of the colluding group members.
The ***efficiency*** of a group signature scheme is measured on a set of indicators: the size of group public key, the size of the group signature, the efficiency of Sign, Verify, Setup, Join and Open, or the efficiency of Revoke, when present [3].

2.3. State of the art

Group signatures [1][2] are a relatively new advanced cryptographic tools. They were introduced by Chaum and van Heist [6] who proposed four static schemes, in which the addition of new members implies reforming the group with a new group public key and new membership certificates.
It is interesting to notice that a concept dual to group signatures is that of *identity escrow*[15] schemes. They can be seen as group-member identification schemes with revocable anonymity. Actually, any group signature scheme can be turned into an identity escrow scheme and vice versa.
The [9] group signature scheme is a major breakthrough as there is no need to change the group public key every time a new member is added to the group. This scheme paved the way for *dynamic group signatures* of possibly large number of members. Several schemes [10][11] based on the previous assumptions from [9] have been proposed, but some of them have been proven insecure or are inefficient.
The first practical, coalition-resistant and provable secure scheme is from Ateniese et al. [3], that we will refer to as ACJT from now on. This scheme has stood the test of time and for many years has been regarded as the state of the art.
However, the member revocation problem, without greatly increasing the computational costs, remained opened. Many experts have proposed solutions such as: using member revocation lists, considering some periods of time etc., but most have been proven inefficient and implied a cumbersome implementation. One of the first viable solution was given by Camenisch and Lysyanskaya [20], that introduce a dynamical accumulation scheme. The accumulator is a mathematical tool which accumulates small prime numbers and allows insertion and deletion of the records. By using this tool for the member revocation the complexity of the ACJT scheme remains the same and it only adds a constant factor, which is less than two. Therefore a member can prove his member status by showing that his certificate is part of the accumulator.
In order to make the revocation process more efficient, the VLR model is introduced in many schemes. This model implies that verifiers must locally adjust their parameters, for them to be able to recognize a corrupt member, and the signers are not obliged to change their signature procedure. One of the first group signatures that enclose this model is Camenisch and Groth [19]. The CG signature is based on the ACJT[3], CL [20] and it offers an efficient member revocation procedure.
The subsequent chapters will illustrate the ACJT and the CG algorithms as well as a personal implementation in Java.

3. The ACJT algorithm

This section provides an overview of the ACJT scheme[3]. The scheme is proven secure and coalition resistant under the strong RSA[2] and DDH[3] assumptions and it offers the set of properties seen in 2.2.. Furthermore, it is scalable and dynamic, provably secure, but it does not support member revocation.

[2] Rivest Shamir Adleman
[3] Decisional Diffie–Hellman

The scheme admits five **procedures [3]**:
SETUP: it admits as input the security parameters and outputs the group public key (GPK) and the group secret key (GSK). The GPK, is then made publicly available even to non-members of the group;
JOIN: is a protocol between the group manager (GM) and a user outside the group that results in the user becoming a new group member, mi. The member's output is a membership certificate with a membership secret key (MSK);
SIGN: is a probabilistic algorithm that on input GPK, a membership certificate, a membership secret and a message Msg outputs a group signature Signature of Msg;
VERIFY: is a deterministic algorithm for establishing the validity of an alleged Signature of Msg with respect to a GPK;
OPEN: is an algorithm that, given a Msg, a valid Signature on it, a GPK and a GSK, determines the identity mi of the signer.
To continue the model of the group signature is described below. First the security parameters that must be chosen, than the procedures, including the implicated actors in the process.

3.1. Security Parameters

The security parameters for the ACJT scheme are given in the table below:

Table 1: ACJT Security Parameters

Security parameter	**Parameter description**	**Bits [16]**
lp	RSA modulus factor	512
ε	Tightness of the zero-knowledgeness	1.1
λ1 **λ2** **γ1** **γ2**	Interval ranges for proving discrete logarithm knowledge	838 600 1102 800
k **H**	Length of the digest Collision resistant hash function	160

The security parameters must be chosen, such that the following relationships are satisfied:
$\lambda 1 > (\lambda 2 + k) + 2$, $\lambda 2 > 4lp$, $\gamma 1 > \varepsilon(\gamma 2 + k) + 2$, $\gamma 2 > \gamma 1 + 2$ (1)

The security parameters λ1, λ2, γ1 and γ2 are used for defining the intervals

$\Lambda = [2^{\lambda 1} - 2^{\lambda 2} ; 2^{\lambda 1} + 2^{\lambda 2}]$ and $\Gamma = [2^{\gamma 1} - 2^{\gamma 2}; 2^{\gamma 1} + 2^{\gamma 2}]$ (2)

on which the zero-knowledge protocol is based on.

3.2. The model

This section describes the actual operation of the ACJT group signature scheme. As noted in [3] it can observed that there are two different ordinary signatures: the first is for issuing certificates of membership, the second is for the actual group signing/verifying. The second scheme is based on a proof of knowledge of a membership certificate through the Fiat-Shamir heuristic. The efficiency of the signature scheme is therefore strictly linked to the efficiency of the membership proof scheme.

(A) SETUP

1. Select random secret lp-bit primes p' and q' such that p = 2p' + 1 and q =2q' + 1 are prime. Set the modulus n = pq.
2. Choose random elements a, a0, g, h ϵ_R QR(n) (of order p'q').
3. Choose a random secret element x ϵ_R $Z^*_{p'q'}$ and set y = g^x mod n.
4. The group public key is: Y = (n; a; a0; y; g; h).
5. The corresponding secret key (known only to GM) is: S = (p' ; q' ; x).

(B) JOIN

1. User Mi generates a secret exponent x'i ϵ_R [0; n^2], a random integer r' ϵ_R [0; 2^{2lp}] and sends C1 = $g^{x'i}h^{r'}$ mod n to GM and proves him knowledge of the representation of C1 w.r.t. bases g and h.
2. GM checks that C1 ϵ QR(n). If this is the case, GM selects αi and

ISSN 2067 – 4074

βi ϵ_R [0; $2^{\lambda 2}$] at random and sends (αi, βi) to Mi.

3. User Mi computes xi = $2^{\lambda 1}$ + (αi x'i + βi mod $2^{\lambda 2}$) and sends GM the value C2 = a^{xi} mod n. The user also proves to GM:

(a) that the discrete log of C2 w.r.t. base a lies in Λ

(b) knowledge of integers u, v, and w such that

i. u lies in [$-2^{\lambda 2}$,$2^{\lambda 2}$]

ii. u equals the discrete log of C2/$a^{2\wedge\lambda 1}$ w.r.t. base

iii. $C1^{\alpha i}g^{\beta i}$ equals $g^u(g^{2\wedge\lambda 2})^v h^w$.

(The statements (i–iii) prove that the user's membership secret xi = loga C2 is correctly computed from C1, αi and βi).

4. GM checks that C2 ϵ QR(n). If this is the case and all the above proofs were correct, GM selects a random prime ei ϵ_R Γ and computes Ai := $(C2\ a0)^{1/ei}$ mod n. GM sends Mi the new membership certificate [Ai; ei]. (Note that Ai = $(a^{xi}\ a0)^{1/ei}$ mod n.)

5. User Mi verifies that a^{xi} a0 ≡ Ai^{ei} (mod n).

(C) SIGN

1. Generate a random value w ϵ_R $\{0, 1\}^{2lp}$ and compute: T1 = Ai y^w mod n, T2 = g^w mod n, T3 = g^{ei} h^w mod n .
2. Randomly choose

r1 ϵ_R ±$\{0, 1\}^{\varepsilon(\gamma 2 +k)}$, r2 ϵ_R ±$\{0, 1\}^{\varepsilon(\lambda 2 +k)}$,

r3 ∈R ±$\{0, 1\}^{\varepsilon(\gamma 1 +2\ p +k+1)}$, and r4 ∈R ±$\{0, 1\}^{\varepsilon(2lp +k)}$ and compute:

(a) d1 = $T1^{r1}$ /(a^{r2} y^{r3}) mod n, d2 = $T2^{r1}$ /g^{r3} mod n, d3 = g^{r4} mod n, and d4 = g^{r1} h^{r4} mod n;

(b) c = H(g || h || y || a0 || a || T1 || T2 || T3 || d1 || d2 || d3 || d4 || m);

(c) s1 = r1 − c(ei − $2^{\gamma 1}$), s2 = r2 − c(xi − $2^{\lambda 1}$), s3 = r3 − c ei w, and s4 = r4 − c w (all in Z).

3. Output (c, s1, s2, s3,s4, T1 , T2 , T3).

(D) VERIFY - Any verifier can check the validity of the signature on message m

(E) OPEN

Only the group manager (GM) can open a signature and reveal the identity of the signer by executing the following:

1. Check the signature's validity via the VERIFY procedure.
2. Recover Ai (and thus the identity of Pi) as Ai = $T1/T2^x$ mod n

The group signature can be regarded as a ***signature of knowledge*** of a value ***xi ∈ Λ*** such that a^{xi} a0 is the value that is ElGamal-encrypted in (T1 , T2) under y and of an ***ei -th root*** of that encrypted value, where ei is the first part of the representation of T3 w.r.t. g and h and that ei lies in Γ .

4. The CG algorithm

This signature is based on the ACJT [3] and the CL [20] schemes. It is proven as secure, practical, scalable, dynamical, it also supports member revocation and in addition the authors prove that it is more efficient that the ACJT signature [3]. In [19] the authors show three variants: one for static groups and two for dynamic groups. Two level of revocation are given for the dynamic versions: revoke and full revoke.

This algorithm admits the same procedures as the ACJT, but it may also contain the revoke and full-revoke procedures. The last procedure, full-revoke, implies a low security level, but an advantage is that any member can prove that a specific signature belongs to him, this is called the claming property.

4.1. The model

The signature model is based on two groups: QR(n) where n is a strong RSA module and a group of order Q in Z_p^* where Q|(P-1).

The values proposed by the authors for the security parameters of this signature are shown in the table below.

Table 2: CG Security Parameters [19]

Security parameter	**Description**	**Length (bits) [19]**
ln	Length of the RSA modulus	1024
lP	Order of the group	1024

lc	Order of the group , where Q|(P-1)	230
lE	The length of the prime number for the member's certificate	450
le	A large number such that each member can have a different number	30
ls	Tightness of the zero-knowledgeness	30
lc	Hash length	160
H	Hash function collision resistant	SHA-1

A member can prove that he is part of the group by using the CL signature [20]. For the complete model of the signature we refer the reader to the original paper [19].

5. Group Signature development

The goal of this project is to create a **standardized framework** for group signatures that will easily support the integration of different group signature algorithms. The implementation should be application independent, algorithm independent, extensible and should preferably be cross-platform. The solution we found is Java which satisfies all of the criteria previously listed. The implementation follows the JCA (Java Cryptography Architecture) standard.

5.1. Java Cryptography Architecture

The JCA[17] takes care of the cryptography part of the Java platform by offering a set of APIs for services such as digital signatures, message digest, certificates and certificate validation, encryption, key generation and management, and secure random number generation, to name a few. Furthermore, some architecture principles that guided the design of the JCA are interesting for our development: **implementation independence** and **interoperability** and **algorithm extensibility**.
Implementation independence of the JCA is achieved using a "provider"-based architecture. The main concepts related to JCA are: Cryptographic service, the Engine class, the Service Provider Interface (SPI) which is an abstract class, the Cryptographic Service Provider (CSP) that contains the concrete implementations of one or more cryptographic services and the Provider class that publishes all the services implemented in the CSP.
For a better understanding of the concepts the figure below is more intuitive:

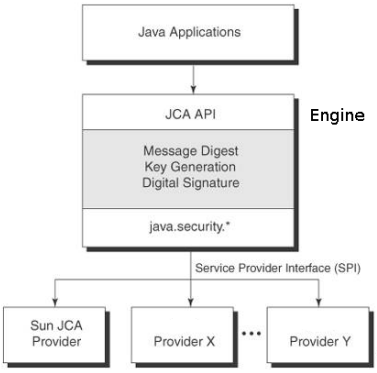

Figure 1: JCA Architecture

5.2. Implementation

In order to implement the group signature framework, a general **analysis** of the group signature model is needed. We must notice that unlike the classical signature model, group signatures imply more parties (members and the group manager), each of them having their own keys and certificates, and the group manager is the only one who can execute some specific procedures. So in conclusion the group signature implies two distinct processes: the message signing and the group management. Thus, two engine classes

ISSN 2067 – 4074

were needed for each process: **GSignature** (for signature generation and validation) and **GGroupManager** (for the group management). For each engine class a corresponding SPI class has been implemented.

For the actual implementation of the ACJT and CG algorithms into the CSP, we have created some **data structures** (classes and interfaces) as follows:

1. Data structures related to **keys,** that mimic the case of corresponding interfaces of the standard JCA framework: *GKey, GPrivateKey, GPublicKey,GKeyPair, GParameterSpec.*
2. Data structures for **certificates** of the group , *GGroupCertificate* that has a X.509 like structure: version, serial number, group signature algorithm, issuer, validity, subject, timestamp, GPublicKey. Group members will not have certificates because their content is mainly secret.
3. The structure for the **member**'s data *GMember* that contains the subject's name and an array object that will store some additional information about a specific member.
4. The **revocation certificate** and the **member revocation list**, for the CG signature(GRevocationCertificate, GrevocationList, …).
5. A class for the **security parameters** that stores their values (CParameterSpec, …).
6. For each engine class a CSP class is created, and it contains the actual implementation of the algorithm's procedures (ACJTGroupManager, ACJTSignature, …)

The **Provider classes** , *ACJTProvider and CGProvider*, contains the mapping between the algorithm names and implementation as a list of string mappings. It registers three services GSignature, GGroupManager and a KeyPaiGenerator.

The package organization is given in the table below:

Table 3: Library Organization

Group Signature Library	
P a c k a g e	**Classes**
mta.libSign.gs	GGroupManager GGroupManagerSpi GSignature GSignatureSpi GMath
mta.libSign.gs.certs	GGoupCertificate GmemeberCertificate GRevocationCertificate
mta.libSign.gs.interfaces	GKey GKeyPair GPrivateKey GPublicKey GMember GparameterSpec GRevocationList
ACJT	
mta.libSign.gs.acjt	ACJTGroupManager ACJTSignature ACJTSignatureCSP ACJTProvider ACJTGroupKeyPairGenerator ACJTGroupPrivateKey ACJTGroupPublicKey ACJTMemberSecretKey ACJTParameterSpec ACJTMember
mta.libSign.gs.acjt.certs	ACJTGroupCertificate
mta.libSign.gs.acjt.interfaces	ACJTKey ACJTPrivateKey ACJTPublicKey
CG (Camenisch and Groth)	
mta.libSign.gs.cg	CGGroupManager CGSignature CGSignatureCSP CGProvider CGGroupKeyPairGenerator CGGroupPrivateKey CGGroupPublicKey CGMemberSecretKey CGParameterSpec CGMember CGRevocationList
mta.libSign.gs.cg.certs	CGGroupCertificate CGRevocationCertificate
mta.libSign.gs.cg.interfaces	CGKey CGPrivateKey CGPublicKey

The implementation was done in Java as a programming language and Eclipse as IDE and JUnit for testing. The tools used for the development of this project are shown in the following table:

Table 4: Implementation Tools

Tool	Version
Eclipse	INDIGO Service Release 2
JDK	6
JUnit	4

The tests were done using a notebook with Intel(R) Core(TM)2 Duo CPU T7500 @ 2.20Ghz, 2GB RAM, L2 cache size 4MB with an Ubuntu 9.10 operating system. The following table shows the execution time (in milliseconds) of each operation. The SETUP algorithm was not included because its execution requires a lot of time, almost ten times the verify operation, because it implies operations such as prime generator, quadric residues etc. that computed in Java using the BigInteger class are costly.

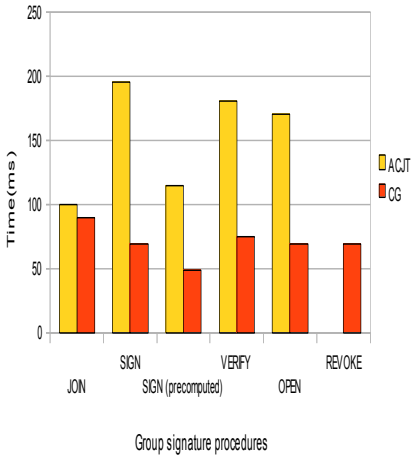

Figure 2: ACJT and CG execution time for a 512 bits message

As expected, the procedure for the CG scheme are much faster. The **JOIN** procedure includes four data exchanges between the member and the group manager for the ACJT, while for the CG only two data exchanges are needed. The small difference of time between them is dew to the more complex operations from the CG scheme.
For the **SIGN** procedure we have also implemented an optimization, by precomputing some values (such as g^r mod n sau F^{RR} mod). At the end the execution time is almost half of the non-optimized version. To conclude, the signing operation is three times faster, while **VERIFY** and **OPEN** are 2.5 faster for the CG.
We have also tested the signature for larger messages, by first applying hash functions (Skein-512, SHA-512). The time differences are determined by the computation of the hash, so we will not include them in this paper.
Another important aspect is to analyze the memory necessary for the implementation (especially when implementing the algorithm on devices with small memory, eg. Phones, PDAs etc.). Theoretically the ACJT signature is 6lp + k + ε(λ1 +λ2 + γ2 + 4lp + 4k) + 4 (~ 8700 bits) and the public group key is 12lp (6144 bits), while the CG signature is a little shorter 2ln+4lP+3lQ+5lc+3ls+le (~ 7758 bits) and the group public key is longer 6ln+4lP+lQ (10470 bits).
The tests showed that CG needs slightly more heap memory, as expected, because it uses more data structures that are more complex (for the revocation process). In the case of the non-heap memory (contructors, constants etc.) the two algorithms use approximately the same memory quantity.
Taking into account the efficiency indicators from section 2.2 we can deduce that the CG signature is more efficient than ACJT.

6. Applications that can use the group signature library

The mechanisms from our library can be used to create security services that can assure the privacy of an entity in many cryptographic protocols. In the following we will give some examples of two applications:

- **electronic payments** – in the classical scheme the financial information from the card is visible to the vendor, although he is not authorized. To improve this system we can use group signatures. For this, each bank has a group of clients and each client has his secret key (the member's secret key). The financial data can be signed with this secret key of the member such that

ISSN 2067 – 4074

the vendor can verify that the signature is valid, but he cannot know the customer's identity.

- **electronic passports** – in this case the terminal can send a group signature to the ICC from the electronic passport instead of the certificate chain. Also the passport owner is offered anonymity if the electronic passport has a member secret key (is part of the group). In this last case only one validation (the group signature) is needed instead of two validations like in the classical PKI model (the validation of the public key of the CA and the validation of the public certificate).

We must remark that group signatures can substitute the classical PKI system and thus we can have offline transactions, or online transactions in which case credentials are validated only once. Thus, by adding group signatures into different systems we can obtain privacy without hindering the security level. But this anonymity is still questionable as the identification of entities can be obtained by using the current systems (IPs, GPS etc.), so a full global change must be made.

7. Conclusion

The main contribution of this project is to show the feasibility of implementing a standardized framework for group signatures. This framework has some interesting properties: implementation and application independence as well as algorithm extensibility. Furthermore, its practicability has been proven by adding the ACJT and the CG group signature algorithms, the Skein hash algorithm which have been described in this paper, but other group signature algorithms can be enclosed.

The framework has been integrated with java's security architecture, thus showing that modern security can be easily added to old cryptographic frameworks.

Further research should be done to test this framework in a practical application, such as electronic cash etc., and other group signature algorithms should be included. Moreover, the framework should also be extended by adding other modern signature types (blind signature, short signatures etc.).

References

[1] A. Menezes, P. van Orschot, and S. Vanstone, "*Handbook of Applied Cryptography*", CRC Press, 1996.

[2] J. Camenisch. "Group signature schemes and payment systems based on the discrete logarithm problem". PhD thesis, vol. 2 of *ETH Series in Information Security an Cryptography*, Hartung-Gorre Verlag, Konstanz, 1998. ISBN 3-89649-286-1.

[3] Giuseppe Ateniese , Jan Camenisch , Marc Joye , and Gene Tsudik, "*A Practical and Provably Secure Coalition-Resistant Group Signature Scheme "*, Advances in Cryptology – CRYPTO 2000, vol. 1880 of Lecture Notes in Computer Science, pp. 255–270, Springer-Verlag, 2000.

[4] "Efficient constructions and anonymity from trapdoor-holders," in Cryptology ePrint Archive, Report 2004/076, 2004.

[5] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," in Advances in Cryptology - CRYPTO'86, LNCS 263, pp. 186-194, Springer, 1987.

[6] D. Chaum and E. van Heyst. "Group signatures". In D. W. Davies, editor, Proceedings of Eurocrypt 1991, volume 547of LNCS, pages 257-65. Springer-Verlag, Apr. 1991.

[7] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation, Oct. 2004.

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

[9] Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In Advances in Cryptology - CRYPTO '97, volume 1296 of Lecture Notes in Computer Science, pages 410-424. Springer, 1997.

[10] Jan Camenisch and Markus Michels. A Group Signature Scheme

with Improved Efficiency. In Advances in Cryptology – ASIACRYPT '98, volume 1514 of Lecture Notes in Computer Science, pages 160-174. Springer, 1998.

[11] Aggelos Kiayias and Moti Yung. Group Signatures: Provable Security, Ecient Constructions and Anonymity from Trapdoor-Holders. Cryptology ePrint Archive, Report 2004/076, 2004. http://eprint.iacr.org/2004/076.

[12] L. Chen and T. P. Pedersen. "New group signature schemes". In EUROCRYPT '94, vol. 950 of LNCS, pp. 171-181. Springer-Verlag, 1995.

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

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

[15] J. Kilian and E. Petrank. Identity escrow. In CRYPTO '98, vol. 1642 of LNCS, pp. 169-185, Berlin, 1998. Springer Verlag.

[16] Toru Nakanishi, Fumiaki Kubooka, Naoto Hamada, and Nobuo Funabiki. Group Signature Schemes with Membership Revocation for Large Groups. In Information Security and Privacy, 10th Australasian Conference, volume 3574 of Lecture Notes in Computer Science, pages 2443454. Springer, 2005.

[17] Java Cryptography Architecture http://docs.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html ;

[18] http://docs.oracle.com/javase/1.4.2/docs/guide/security/CryptoSpec.html

[19] Camenisch şi Groth**,** "Group Signatures: Better Efficiency and New Theoretical Aspects"

[20] Jan Camenisch and Anna Lysyanskaya. A Signature Scheme with Efficient Protocols. In Security in Communication Networks volume 2576 of Lecture Notes in Computer Science , pages 268-289. Springer, 2003.