Security Flaws in a Pairing-based Group Signature Scheme

Zhengjun Cao*   Sherman S.M. Chow†

*Key Lab of Mathematics Mechanization, Academy of Mathematics and Systems Science,

Chinese Academy of Sciences, Beijing, P.R. China. 100080 zjcamss@hotmail.com

†Department of Computer Science, The University of Hong Kong

Pokfulam, Hong Kong smchow@cs.hku.hk

Abstract Deng and Zhao recently proposed a group signature scheme in [1]. We find that the scheme cannot satisfy all of the requirements of a secure group signature.

Keywords group signature scheme, bilinear pairings, Gap-Diffie-Hellman group, unforgeability, anonymity, linkability, exculpability, traceability, coalition-resistance

1 Introduction

Group signatures, introduced by Chaum and Heyst[2], allow individual members to make signatures on behalf of the group. A secure group signature scheme must satisfy the following properties[3]:

- 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 produced by the same group member is computationally hard for everyone but the group manager.
- Exculpability: Neither a group member nor the group manager can sign on behalf of other group member, but it precludes the group manager from creating fraudulent signers and then producing group signatures.
- Traceability: The group manager is always able to open a valid signature and identify the actual signer.

- 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.

Recently, Deng and Zhao proposed a new group signature scheme from Gap Diffie-Hellman groups, groups where computational Diffie-Hellman problem is intractable but decisional Diffie-Hellman problem is easy. Unfortunately, we find that the scheme is insecure. It is universally forgeable, linkable, untraceable, not anonymous, not exculpable and not coalition-resistant.

2 Deng-Zhao Group Signature Scheme

For the sake of completeness we review the group signature scheme proposed by Deng and Zhao here. Their group signature scheme consists of five algorithms, namely, `Setup`, `Join`, `Sign`, `Verify` and `Open`.

2.1 Setup

`Setup` generates the public parameter of the group signature scheme, the group manager's private key and the group's public key.

Let E/F_{k^n} be a supersingular elliptic curve whose order has large prime factor q. Let $P \in E/F_{k^n}$ be a point of order q. The subgroup $< P >$ generated by P is defined as $\mathbb{G}_1$. According to the isomorphism ϕ on the curve, define a bilinear map: $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, where $\mathbb{G}_2$ is a subgroup of $F^*_{k^{\alpha n}}$. Such a bilinear pairing is the key primitive for solving decisional Diffie-Hellman problem of $\mathbb{G}_1$. To map a string to a point on curve, we define $H_1 : \{0,1\}^* \to \mathbb{G}_1$ as a cryptographic hash function using the algorithm `MapToGroup`[4]. We also define $H_2 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{Z}^*_q$ be another cryptographic hash function. The group manager chooses a random number $s \in \mathbb{Z}^*_q$ and computes $P_{pub} = sP$. The public parameter is $(P, P_{pub}, H_1(\cdot), H_2(\cdot,\cdot), \hat{e}(\cdot,\cdot), \alpha, q, n)$. The master secret key and the group's public key is $SK = s$ and $PK = (P, P_{pub})$ respectively.

2.2 Join

`Join` is an interactive protocol between the group manager and a user, which is executed when a user wants to join the group. The input of the protocol contains the group manager's private key and the output of the protocol contains the public-private key pair for the user.

Suppose now that a user u_i wants to join the group. First, u_i randomly chooses $x_i \in \mathbb{Z}^*_q$. Then he/she computes $R_i = x_i P$. To obtain his/her membership certificate, each user must perform the following protocol with the group manager:

1. The user u_i sends R_i to the group manager.
2. The group manager regards R_i as some ID information and computes $D_i = sR_i$.

Then D_i is communicated secretly to the user u_i as group member secret key. R_i is u_i's public key.

2.3 Sign

`Sign` is a probabilistic algorithm which takes the private key of any arbitrary user who joined the group to produce a anonymous group signature on behalf of the group on a certain message. If user u_i wants to sign a message m on behalf of the group, he/she proceeds as follows:

1. Pick a random $r \in \mathbb{Z}_q^*$, compute $U = rR_i,\ h = H_2(m, U),\ V = (r + h)D_i$. Then the first part of the signature is $\sigma_1 = (U, V)$.
2. Compute $P_m = H_1(m),\ S_m = x_i P_m$, where σ_2 is the x-coordinate of S_m.

The final signature of user u_i is $(\sigma_1, \sigma_2, R_i)$, R_i can be treated as a traceability tag.

2.4 Verify

`Verify` is a deterministic algorithm which takes the public key of the group and the message as the input to check whether the signature is a valid one. Given a signature $(\sigma_1, \sigma_2, R_i)$ and a message m, verification can be divided into two parts:

1. The verifier makes sure that the signature is generated by a group member by checking that $\hat{e}(P_{pub}, \phi(U + hR_i)) = \hat{e}(P, \phi(V))$, using σ_1.
2. The verifier checks that the signature is definitely generated by u_i rather than other members of the group. He/she does the following :
   (a) Find a point $S \in E/F_{k^n}$ of order q whose x-coordinate is σ_2 and whose y-coordinate is some $y \in F_{k^n}$. If no such point exists, reject the signature as invalid.
   (b) Set $c = \hat{e}(P, \phi(S))$ and $d = \hat{e}(R_i, \phi(H_1(m)))$
   (c) If either $c = d$ or $c^{-1} = d$, accept the signature. Otherwise, reject it.

2.5 Open

`Open` is the algorithm executed by the group manager when the anonymity of the group signature should be revoked. The group manager knows the identity of the member for each u_i. As a result, it is easy for the group manager, given a message m and a valid group signature, to determine the identity of the signer corresponding to the public key R_i.

3 Analysis

3.1 Unforgeability

We show the scheme is universally forgeable by describing a way that everyone can generate a private key by his/her own without the execution of `Join` protocol with the group manager.

Autonomous Join Procedure:

1. Randomly choose $\hat{x} \in \mathbb{Z}_q^*$.
2. Compute the public key by $\hat{R} = \hat{x}P$.
3. Compute the corresponding private key by $\hat{D} = \hat{x}P_{pub}$.

It is easy to see $\hat{D}$ produced is a valid private key since $\hat{D} = \hat{x}P_{pub} = \hat{x}sP = s\hat{x}P = s\hat{R}$. With the help of this autonomous join procedure, any body outside the group can sign on behalf of the group.

A simple fix to this flaw is to require the user to choose an identifier $ID \in \{0,1\}^*$ and send ID instead of R_i to group manager in `Join` phase. R_i will be set to $H_1(ID)$. Then, every signature should include ID as the traceability tag instead of R_i. However, after such fix, x_i such that $R_i = x_iP$ cannot be easily deduced. To remedy this problem, we can employ another pair of key x_i' and $R_i' = x_i'P$, while x_i' is randomly generated by the user and R_i is set to $H_1(ID||R')$ in order for this new pair of key to be "certified" by the group manager.

3.2 Anonymity

In the `Join` protocol, the user first generates R_i and requests for the corresponding private key, and the private key is subsequently sent to the user in a secure way. Any eavesdropper in this process can associate the R_i to the identity of the user. Since the user's public key R_i is to be included in every signature he/she produced, the anonymity of the group signature is lost. A simple fix is to employ an anonymous and encrypted key issuing protocol.[5]

3.3 Unlinkability

It is obvious that the scheme is linkable because a user u_i must use his/her key R_i to make a valid group signature $(\sigma_1, \sigma_2, R_i)$. To overcome the flaw, the authors proposed a modification that each user uses different public-private key pair for each signature.[1] In a sense, the scheme is not practical.

3.4 Exculpability

We first show how the group manager can generate a valid group signature, then we describe how this attack affect the exculpability of the scheme.

A key observation on the `Sign` phase in the original scheme is that user u_i has to use his/her key triple (x_i, R_i, D_i), where x_i is a random number picked by user u_i in $\mathbb{Z}_q^*$, $R_i = x_i P$ and $D_i = sR_i$.

Intuitively, we know that the group manager has absolute superiority in `Join` phase in the scheme because he/she can generate an arbitrary $R_0 = x_0 P$ ($x_0 \in_R \mathbb{Z}_q^*$) solely, then he/she can compute $D_0 = sR_0$ using his/her master key s. Therefore, he/she obtains a valid key triple (x_0, R_0, D_0) for signing like any group member.

Group Manager's Attack:

If group manager wants to sign a message m on behalf of the group, he/she performs the following steps:

1. Pick a random $r \in \mathbb{Z}_q^*$, compute $U = rR_0,\ h = H_2(m, U),\ V = (r + h)D_0$. Then the first part of the signature is $\sigma_1 = (U, V)$.
2. Compute $P_m = H_1(m),\ S_m = x_0 P_m$, where σ_2 is the x-coordinate of S_m.

The final signature of group manager is $(\sigma_1, \sigma_2, R_0)$.

The correctness of the forged group signature is obvious.

Now we analyze the scheme's exculpability. The `Open` procedure uses only the secret knowledge held by the group manager and is not publicly verifiable. From the group manager's forgery attack, any user can be framed to be the "actual signer".

It is true that the user can provide a knowledge proof of sR_i and try to claim that the secret key sR_0 is not the real private key. Unfortunately, this knowledge proof cannot be used as a proof of treachery of the group manager. Note that for the scheme to achieve anonymity, the user's key R_i (and also x_i) should not be related to the identity of the user, i.e. they should be some "random-looking" bit strings. The knowledge proof of sR_i where $sR_i \neq sR_0$ does not imply sR_i is associated to the user, it is possible that sR_i is a key from conspiracy.

A simple fix is to employ a trusted authority (not the group manager) to certify the relationship between the user and the public key.

3.5 Traceability

With the help of the autonomous join procedure in Section 3.1, any body outside the group can sign on behalf of the group without fearing the tracing of the group manager since the `Open` procedure of the original scheme requires the group manager's knowledge of the real identity of the person invoking the `Join` protocol.

Even without the help of the autonomous join procedure, any real member of the group (i.e. those who have executed the `Join` protocol with the group manager) can generate a untraceable but valid group signature on behalf of the group.

Untraceable Signing Procedure:

1. Pick a random $r \in \mathbb{Z}_q^*$ and a random untraceable factor $y \in \mathbb{Z}_q^*$, compute $R_i' = yR_i$. Then compute $U = rR_i',\ h = H_2(m, U),\ V = (r+h)yD_i$. Then the first part of the signature is $\sigma_1 = (U, V)$.
2. Compute $P_m = H_1(m),\ S_m = x_i y P_m$, where σ_2 is the x-coordinate of S_m.

The final signature of user u_i is $(\sigma_1, \sigma_2, R_i')$.

It is easy to see that the signature is valid. However, since the traceability tag R_i of the signature is spoiled by the random untraceable factor, the signature produced is thus information theoretically untraceable, even the group manager cannot help.

We note that the simple fix in Section 3.1 can be used to prevent this attack.

3.6 Coalition-Resistance

Our attack can be carried out by any group of colluding signers of arbitrary size. For the simplicity we show the case for 2 colluding signers, u_i and u_j.

Untraceable Coalition Signing Procedure:

1. Pick a random $r \in \mathbb{Z}_q^*$, compute $U = r(R_i + R_j),\ h = H_2(m, U),\ V = (r+h)(D_i + D_j)$. Then the first part of the signature is $\sigma_1 = (U, V)$.

2. Compute $P_m = H_1(m),\ S_m = (x_i + x_j)P_m$, where σ_2 is the x-coordinate of S_m.

The final signature of user is $(\sigma_1, \sigma_2, (R_i + R_j))$.

It is easy to see that the signature is valid and untraceable. Without knowing the group size of the coalition, the group manager needs to try $O(2^z)$ combinations of R_is (where z is the size of the group) to trace the identities of the colluding signers. If untraceable factor is used, the group manager cannot trace at all even exhausted all of the $O(2^z)$ possibilities. Again, we note that the simple fix in Section 3.1 can be used to prevent this attack.

4 Conclusion

In the paper, we analyze Deng-Zhao group signature scheme and show its insecurity from different aspects. We hold that it is not easy to design a perfect group signature scheme.

References

[1] Donghua Deng, Yiming Zhao. An Efficient Group Signature from Gap Diffie-Hellman Groups. ChinaCrypt 2004, pp. 186-194 (in English), 2004.

[2] D.Chaum, F.Heyst. Group Signatures. EUROCRYPT 1991, LNCS Vol. 547, pp.257-265, Donald W. Davies ed. Springer, 1992.

[3] M. Bellare, D. Micciancio, B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction based on General Assumptions. EUROCRYPT 2003. LNCS Vol. 2656, pp.614-629, Eli Biham ed. Springer, 2003.

[4] Boneh D, Lynn B, Shacham H. Short Signatures from the Weil Pairing. ASIACRYPT 2001, LNCS Vol. 2248, Colin Boyd ed. Springer, 2001.

[5] Sherman S.M. Chow, Secure Identity-Based Key Issuing without Secure Channel. Manuscript, 2004.