Cryptanalysis of Aggregate Γ Signature with Sub-Exponential Complexity

Kaoru Takemure[1,2], Yusuke Sakai[2], Bagus Santoso[1],
Goichiro Hanaoka[2], and Kazuo Ohta[1,2]

[1] The University of Electro-Communications, Japan
[2] National Institute of Advanced Industrial Science and Technology (AIST), Japan

Abstract. We present a sub-exponential forger by using a k-sum algorithm against the aggregate Γ signature, which was proposed at AsiaCCS2019 by Zhao. Our forger is a universal forger under a key-only attack and effective in the knowledge of secret key model.

Keywords: k-sum algorithm · aggregate signature · universal forgery

1 Introduction

Aggregate signatures (AS) are a cryptographic primitive which allows combining individual signatures on different messages into a compact one. Boneh et al. introduced the notion of AS and proposed a pairing-based AS scheme which achieves a constant size signature [3]. In general, pairing-based AS schemes require pairing computation to verify a signature, and the security of them is based on the computational assumption in groups with bilinear maps which is stronger than the discrete logarithm assumption in elliptic curve (EC) groups. Deploying pairing-based AS schemes in existing applications, e.g., blockchain, is expensive because it requires not only replacing the algorithms of a signature scheme with the ones of the pairing-based scheme but also replacing an EC with pairing friendly ones. AS from general groups is attractive in terms of the computational complexity and the cost of deployment.

Zhao showed the subtlety in constructing secure AS scheme from general groups and proposed an AS scheme from general EC groups without bilinear maps by extending the Γ signature [8], which is called the aggregate Γ signature [9]. He also proved that it is secure in the plain public-key (PK) model [1] based on the new assumption, named the non-malleable discrete logarithm (NMDL) assumption.

In this paper, by using a k-sum algorithm [7], we propose a universal forger under a key-only attack [5] in the knowledge of secret key (KOSK) model [2, 6]. This forger runs in sub-exponential time due to the k-tree algorithm [7]. Although our proposed attack is not fatal theoretically since it is a sub-exponential time attack, it affects the practical performance of Zhao's scheme. More concretely, for most of schemes based on general groups the bit-length of the order of the underlying group of schemes is twice as long as the security parameter

due to the ρ method. In contrast, the aggregate Γ signature scheme requires the bit-length of the order of an underlying group to be approximately $\log n$ times as long as the security parameter where n is the number of signers.

2 Preliminaries

The following is notations and some definitions which are used in this paper.

2.1 Notation

For a prime integer q, we denote the ring of integers modulo q by Z_q and the multiplicative group of Z_q by Z_q^*. Let G be an additive cyclic group of order q and let P be a generator of G. For a set A, we write $a \xleftarrow{\$} A$ to mean that a is chosen uniformly at random from A.

2.2 k-sum Problem

We recall the definition of k-sum problem.

Definition 1 (k-sum problem). *The k-sum problem in group $(Z_q; +)$ for an arbitrary q provides k lists $L_1, \ldots, L_k$ of equal sizes, each list containing s_L elements sampled uniformly and independently from Z_q, and requires to find $x_1 \in L_1, \ldots, x_k \in L_k$ s.t. $\sum_{i=1}^{k} x_i \equiv 0 \pmod q$.*

In [7], Wagner proposed the k-tree algorithm which can solve the k-sum problem for $s_L = 2^{\log q/(1+\log k)}$ in time at most $O(k2^{\log q/(1+\log k)})$ with non-negligible probability.

2.3 Aggregate Signature

In this section, we show definitions of aggregate signatures and a security model of it.

Definition 2. *An aggregate signature scheme consists of the following six algorithms. Let n be the number of signers.*

Setup$(1^\lambda) \to pp$**.** *The public parameter generation algorithm takes as input a security parameter 1^λ, then it outputs a public parameter pp.*

KeyGen$(pp) \to (pk, sk)$**.** *The key generation algorithm takes as input a public parameter pp, then it outputs a public key pk and a secret key sk.*

Sign$(pp, pk, sk, m) \to \sigma$**.** *The signing algorithm takes as input a public parameter pp, a public key pk, a secret key sk, and a message m, then it outputs a individual signature σ.*

Verify$(pp, pk, m, \sigma) \to \{0, 1\}$ *The verification algorithm takes as input a public parameter pp, a public key pk, a message m, and a signature σ, then it outputs 0 (REJECT) or 1 (ACCEPT).*

Agg$(pp, \{(pk_i, m_i, \sigma_i)\}_{i=1}^n) \to \sigma_a$**.** *The aggregation algorithm takes as input a public parameter pp, and a set of all signers' public keys, messages, and signatures* $\{(pk_i, m_i, \sigma_i)\}_{i=1}^n$*, then it outputs an aggregate signature* σ_a*.*

AggVer$(pp, \{(pk_i, m_i)\}_{i=1}^n, \sigma_a) \to \{0, 1\}$**.** *The aggregate signature verification algorithm takes as input a public parameter pp, a set of all signers' public keys and messages* $\{(pk_i, m_i)\}_{i=1}^n$*, and an aggregate signature* σ_a*, then it outputs 0 (REJECT) or 1 (ACCEPT).*

For any set of messages $\{m_i\}_{i=1}^n$*, if a public parameter pp, all signers' public keys* $\{pk_i\}_{i=1}^n$*, and an aggregate signature* σ_a *are generated honestly by the above algorithms, then we require that* $\Pr[\mathbf{AggVer}(pp, \{(pk_i, m_i)\}_{i=1}^n, \sigma_a) = 1] = 1$.

Security Game. For aggregate signatures, we define *universal unforgeability under key-only attacks* in the knowledge of secret key (KOSK) model. In this security model, a forger who corrupts an aggregator and signers except one honest signer is given an honest signer's public key and a message and is required to generate a forgery on the given message by making only hash queries. When outputting a forgery, it must output cosigners' secret keys corresponding to cosigners' public keys which are chosen arbitrarily.

If, for all m^*, a forger $\mathcal{F}$ wins the following game with non-negligible probability, then we say that $\mathcal{F}$ is a universal forger under a key-only attack in the KOSK model.

Setup$(1^\lambda, m^*)$**.** The challenger chooses a public parameter $pp \xleftarrow{\$} \mathbf{Setup}(1^\lambda)$, a honest signer's key pair $(pk, sk) \xleftarrow{\$} \mathbf{KeyGen}(pp)$. It runs a forger $\mathcal{F}$ on input pp, pk and a message m^*.

Output. $\mathcal{F}$ outputs n key pairs $\{(pk_i, sk_i, m_i^*)\}_{i=1}^n$ and a forgery σ_a^* where the following holds.

- $(pk_1, m_1^*), \ldots, (pk_n, m_n^*)$ are mutually distinct.
- $(pk, m^*) \in \{(pk_i, m_i^*)\}_{i=1}^n$.
- sk_l is $\perp$ where l is s.t. $pk_l = pk$.
- sk_i is a correct secrete key corresponding to pk_i for $i \in [1, n] \backslash \{l\}$.

If $\mathbf{AggVer}(pp, \{(pk_i, m_i^*)\}_{i=1}^n, \sigma_a^*) = 1$ is true, then $\mathcal{F}$ wins.

3 Aggregate Γ Signature Scheme

In [9], the aggregate Γ signature scheme is proposed by Zhao. This scheme consists of the following six algorithms.

Setup$(1^\lambda) \to (G, q, P, H_0, H_1)$**.** It chooses (G, q, P), hash functions $H_0 : G \to Z_q$ and $H_1 : G \times M \to Z_q$ where M is the set of messages, then it outputs $pp = (G, q, P, H_0, H_1)$.

KeyGen$(pp) \to (X, x)$**.** It computes $x \xleftarrow{\$} Z_q^*$ and $X \leftarrow xP$, then it outputs a public key X and a secret key x.

Sign$(pp, X, x, m) \to \sigma$**.** It computes $r \xleftarrow{\$} Z_q^*$, $A \leftarrow rP$, $d \leftarrow H_0(A)$, and $e \leftarrow H_1(X, m)$. It computes $z \leftarrow rd - ex \mod q$, then it outputs $\sigma = (z, d)$ as a signature.

Verify$(pp, X, m, \sigma) \to \{0, 1\}$ It computes $e \leftarrow H_1(X, m)$ and $A \leftarrow zd^{-1}P + ed^{-1}X$. If $H_0(A) \neq d$ holds, then it outputs 0. Otherwise it outputs 1.

Agg$(pp, \{(X_i, m_i, \sigma_i)\}_{i=1}^n) \to (\hat{T}, \hat{A}, z)$**.** It initializes $\hat{T} = \emptyset$, $\hat{A} = \emptyset$, and $z = 0$. For $i = 1$ to n, if **Verify**$(pp, X_i, m_i, \sigma_i) = 1 \wedge (X_i, m_i) \notin \hat{T} \wedge A_i \notin \hat{A}$ holds, it sets $\hat{T} \leftarrow \hat{T} \cup \{(X_i, m_i)\}$ and $\hat{A} \leftarrow \hat{A} \cup \{A_i\}$ and computes $z \leftarrow z + z_i \mod q$. Finally, it outputs $(\hat{T}, \hat{A}, z)$.

AggVerify$((\hat{T}, \hat{A}, z)) \to \{0, 1\}$**.** If the elements in $\hat{T}$ are not mutually distinct, the elements in $\hat{A}$ are not mutually distinct, or $|\hat{T}| \neq |\hat{A}|$ holds, then outputs 0. It sets $n' \leftarrow |\hat{T}|$, and for $j = 1$ to n', it computes $d_j \leftarrow H_0(A_j)$ and $e_j \leftarrow H_1(X_j, m_j)$. If $\sum_{j=1}^{n'} d_j A_j = zP + \sum_{j=1}^{n'} e_j X_j$ holds, it outputs 1, Otherwise it outputs 0.

Zhao presented the ephemeral rouge-key attack against an intuitive AS scheme built from the Schnorr signature which combines only the response components of the Σ protocol [4] and showed that the above AS scheme can prevent this attack. Also the security of this scheme is proved based on the non-malleable discrete logarithm (NMDL) assumption. We review the definition of this assumption.

Definition 3 (non-malleable discrete logarithm (NMDL) assumption). *Let $H_1, \ldots, H_K : \{0,1\}^* \to Z_q^*$ be cryptographic hash functions, which may not be distinct. On input (G, P, q, X) where $X = xP$ for $x \leftarrow Z_q^*$ a PPT algorithm $\mathcal{A}$ (called an NMDL solver) succeeds in solving the NMDL problem, if it outputs $(\{b_i, Y_i, m_i\}_{i=1}^K, z)$ satisfying:*

- *$z \in Z_q$, and for any i, $1 \leq i \leq K$, $Y_i \in G$, $m_i \in \{0,1\}^*$ that can be the empty string, and $b_i \in \{0, 1\}$.*
- *For any $1 \leq i, j \leq K$, it holds that $(Y_i, m_i) \neq (Y_j, m_j)$. It might be the case that $Y_i = Y_j$ or $m_i = m_j$.*
- *$X \in \{Y_i\}_1^K$, and $zP = \sum_{i=1}^K (-1)^{b_i} e_i Y_i$ where $e_i = H_i(Y_i, m_i)$.*

The NMDL assumption means that there are no PPT algorithm which succeeds in solving the NMDL problems with non-negligible probability in $\log q$.

For more detail of this assumption, see Section 5.1 of [9].

4 Sub-Exponential Universal Forgery under a Key-Only Attack against Aggregate Γ Signature in the KOSK Model

Here we present a sub-exponential universal forger under a key-only attack against the aggregate Γ signature in the KOSK model. The cause of this cryptanalysis is that there is an algorithm that can solve the NMDL problem in sub-exponential time by using a k-sum algorithm.

The input and the goal of a forger against aggregate Γ signature in the security game in Section 2.3 are as follows:

Input: A challenge key X_1 and a target messages m_1^*.
Goal: To output a forgery $(z^*, \{A_i\}_{i=1}^n)$ and a set of cosigners' keys and messages $\{(X_i, x_i, m_i^*)\}_{i=2}^n$ s.t. the following holds:

$$\sum_{i=1}^{n} d_i A_i = z^* P + \sum_{i=1}^{n} e_i X_i \tag{1}$$

where $X_i = x_i P$ for $i \in [2, n]$, $d_i = H_0(A_i)$, and $e_i = H_1(X_i, m_i^*)$ for $i \in [1, n]$.

Below, we show the procedure of our proposed forger $\mathcal{F}$.

Main Procedure

1. Choose arbitrary cosigners' secret keys $\{x_i\}_{i=2}^n \in (Z_q^*)^{(n-1)}$ and assign the public keys as follows:

$$X_2 \leftarrow x_2 P, \ldots, X_n \leftarrow x_n P. \tag{2}$$

2. Launch a n-sum attack via $n \cdot s_L$ times hash computations to obtain $\{(d_i, r_i, A_i)\}_{i=1}^n$ s.t. the following holds:

$$\sum_{i=1}^{n} d_i \equiv e_1 \pmod{q} \tag{3}$$

where $A_i = r_i P + X_1$, $d_i = H_0(A_i)$ for $i \in [1, n]$ and $e_1 = H_1(X_1, m_1^*)$.
3. Choose any messages $\{m_i^*\}_{i=2}^n$ and assign the followings:

$$e_2 \leftarrow H_1(X_2, m_2^*), \ldots, e_n \leftarrow H_1(X_n, m_n^*), \tag{4}$$

$$z^* \leftarrow -\sum_{i=2}^{n} x_i e_i + \sum_{i=1}^{n} r_i d_i. \tag{5}$$

4. Output $(z^*, \{A_i\}_{i=1}^n)$ and $\{(X_i, x_i, m_i^*)\}_{i=2}^n$.

In Step 2 of the above, $\mathcal{F}$ executes the n-sum algorithm according to the following.

n-sum Attack Procedure

1. Choose $\{r_{i,j}\}_{i=1,j=1}^{n,s_L} \in (Z_q^*)^{n \times s_L}$ and computes $\{A_{i,j}\}_{i=1,j=1}^{n,s_L}$ where

$$A_{i,j} = r_{i,j} P + X_1. \tag{6}$$

2. Compute $d_{i,j} \leftarrow H_0(A_{i,j})$ for $i \in [1, n], j \in [1, s_L]$.

3. Make lists as follows:

$$L_1 \leftarrow \{d_{1,j} - e_1\}_{j=1}^{s_L},$$
$$\text{and } L_i \leftarrow \{d_{i,j}\}_{j=1}^{s_L} \text{ for } i \in [2, n].$$

4. Run the n-sum algorithm on input the n lists $\{L_i\}_{i=1}^{n}$ to obtain $\{d_{i,j_i}\}_{i=1}^{n}$ s.t. Eq. (3) holds.
5. Output $\{(d_{i,j_i}, r_{i,j_i}, A_{i,j_i})\}_{i=1}^{n}$.

Correctness. Now we confirm the correctness of the above attack procedure. For an output of $\mathcal{F}$, $(z^*, \{A_i\}_{i=1}^{n})$ and $\{(X_i, x_i, m_i^*)\}_{i=2}^{k}$, we have the following equations hold:

$$\begin{aligned}
& z^*P + \sum_{i=1}^{n} e_i X_i \\
&= (-\sum_{i=2}^{n} x_i e_i + \sum_{i=1}^{n} r_i d_i)P + e_1 X_1 + \sum_{i=2}^{n} e_i x_i P \quad (\text{from, Eq.(5)}) \\
&= \sum_{i=1}^{n} r_i d_i P + e_1 X_1 \\
&= \sum_{i=1}^{n} r_i d_i P + \left(\sum_{i=1}^{n} d_i\right) X_1 \quad (\text{from Eq.(3)}) \\
&= \sum_{i=1}^{n} d_i (r_i P + X_1) \\
&= \sum_{i=1}^{n} d_i A_i \quad (\text{from Eq.(6)}).
\end{aligned}$$

Computational Complexity. By using Wagner's k-tree algorithm, Step 4 of $(n-1)$**-sum Attack Procedure** takes at most $O\left(n2^{\log q/(1+\log n)}\right)$ time. In addition, in **Main Procedure**, there are $n-1$ exponentiations and n computations of the hash function in Step 1 and 3, respectively. Also, in n**-sum Attack Procedure**, there are respectively $n \times s_L$ exponentiations and $n \times s_L$ computations of the hash function in Step 1 and 2 where s_L is $2^{\log q/(1+\log n)}$.

A value of $2^{\log q/(1+\log n)}$ is minimized when $n = 2^{\sqrt{\log q}-1}$. In particular, assuming that the bit-length of q is 256-bits, the running time of the above forger is minimized to $O(2^{31})$ when n is approximately 2^{15}. In this parameter (i.e., $n = 2^{15}$), the number of cosigners should be fixed to $2^{15} - 1$ and cannot be chosen flexibly. Instead, if we want to reduce the number of cosigners, we can mount the above attack with smaller n at the cost of much time and space complexity.

References

1. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: CCS 2006. pp. 390–399 (2006)
2. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: PKC 2003. pp. 31–46 (2003)
3. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: EUROCRYPT 2003. pp. 416–432 (2003)
4. Cramer, R.: Modular design of secure, yet practical cryptographic protocols. Ph.D. thesis, University of Amsterdam (1996)
5. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1988)
6. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: EUROCRYPT 2006. pp. 465–485 (2006)
7. Wagner, D.A.: A generalized birthday problem. In: CRYPTO 2002. pp. 288–303 (2002)
8. Yao, A.C., Zhao, Y.: Online/offline signatures for low-power devices. IEEE Trans. Information Forensics and Security **8**(2), 283–294 (2013)
9. Zhao, Y.: Practical aggregate signature from general elliptic curves, and applications to blockchain. In: AsiaCCS 2019. pp. 529–538 (2019)