T-79.515 Cryptology: Special Topics / Pairing-based Cryptography: Aggregate Signatures from Bilinear Pairings

Tatiana Issaeva

Draft, March 16, 2003

1 Introduction

At the moment there are several real-world applications, which require signatures on many different messages produced by many different users. In such applications, the number of signatures is linear to the number of users. If number of users is large, such construct become not very practical and it is desirable to compress all signatures of the chain into a single signature. Such compresion can be achieved by aggregate signature scheme, proposed in [1].

Idea behind an aggregate signature scheme is that given signatures on n different messages from n different users, we can aggregate all signatures into one short signature, which will convince verifier that n users has signed n different messages. Aggregate signature scheme from [1] presented in this survey is build from bilinear map $e : G_1 \times G_1 \to G_T$ between two groups G_1 and G_T, such that a variant of Computational Deffie-Hellman (CDH) problem is hard in G_1. This aggregate signature scheme is also based on the short signature scheme from bilinear pairings described in [2]. This short signature scheme has already been presented previously during this seminar in [3].

The aggregate signature scheme is secure against existential forgery in the aggregate chosen-key model. In this model adversary is given a single public key and can choose all the rest needed public keys except the challenge public key. The adversary also has an access to signing oracle on the challenge key. The security is based on hardness of variant of Computational Deffie-Hellman (CDH) problem and secutiry proor reduces CDH to existential forgery of aggregate signature.

This survey presents aggregate signature scheme from [1] and is organized in a following way. First, all required definitions and notations for bilinear maps are given in Section 2. Then, in Section 3, aggragete signature scheme from bilinear pairings is described. Section 4 provides security definition for aggregate signature scheme and Section 5 contains a security proof. In Section 6 some possible real-world applications for aggregate signature scheme is briefly considered.

2 Required definitions and notation.

In this section we will intoduce all the definitions and notation which will be used later for definition and security proof of aggregate signature scheme.

Let G_1, G_2 and G_T be multiplicative cyclic groups of prime order p and $e : G_1 \times G_2 \to G_T$ be a computable bilinear map. Let g_1 be a generator of G_1 and g_2 be a generator of G_2. For the security proof of the system we will also need a computable isomorphism ψ from G_1 to G_2, such that $\psi(g_1) = g_2$.

Since we consider case $G_1 \neq G_2$, we will define a variant of computational Diffie-Hellman and decisional Diffie-Hellman.

The **computational co-Diffie-Hellman (co-CDH)** problem is given $g_1, g_1^a \in G_1$ and $g_2, g_2^b \in G_2$ to compute $g_2^{ab} \in G_2$.

The **decisional co-Diffie-Hellman (co-DDH)** problem is given $g_1, g_1^a \in G_1$ and $g_2, g_2^b, g_2^c \in G_2$ to decide if $ab = c$.

Definition 2.1. Two groups G_1 and G_2 are decision groups for co-Diffie-Hellman if the group actions on G_1 and G_2 and the map ψ from G_1 to G_2 can be computed in one time unit and co-DDH on G_1 and G_2 can be solved in one time unit.

Definition 2.2. The advantage of an algorithm $\mathcal{A}$ in solving the co-CDH problem in groups G_1 and G_2 is

$\mathrm{adv}_{\mathcal{A}} = \mathsf{Pr}\left[\mathcal{A}(g_1, g_1^a, g_2, g_2^b) = g_2^{ab} : a, b \leftarrow \mathbb{Z}_p\right]$.

The probability is taken over the choice of a and b and $\mathcal{A}$'s coin tosses.

An algorithm $\mathcal{A}$ is said to (t, ϵ)-break co-CDH on G_1 and G_2 if $\mathcal{A}$ runs in time at most t and $\mathrm{adv}_{\mathcal{A}} \geq \epsilon$.

Definition 2.3. Two groups G_1 and G_2 are bilinear groups if the group actions on G_1 and G_2 and the map ψ from G_1 to G_2 can be computed in one time unit and bilinear map $e : G_1 \times G_2 \to G_T$ exists and computable in one time unit.

Definition 2.4. Two groups G_1 and G_2 are (t, ϵ)-bilinear groups for co-Diffie-Hellman if they are bilinear groups and no algorithm (t, ϵ)-breaks co-CDH on G_1 and G_2.

3 Bilinear aggregate signature scheme

In this section we take a look at the definition of aggregate signature scheme from bilinear map $e : G_1 \times G_2 \to G_T$. This scheme uses a full-domain hash function $h : \{0,1\}^* \to G_2$. In secuty proof h is viewed as random oracle.

The considered aggregate signature scheme consists of five algorithms: keyGen, sign, veriry, aggregate and aggregateVerify, which are described below.

Key generation. Pick a random $x_i \leftarrow \mathbb{Z}_p$ and compute $v_i \leftarrow g_1^{x_i}$ for some user u_i. Then for user u_i the secret key is $x_i \in \mathbb{Z}_p$ and the public key is $v_i \in G_1$.

Signing. For user u_i, given a secret key x_i and message $M_i \in \{0,1\}^*$, compute $h_i \leftarrow h(M_i)$, where $h_i \in G_2$. Then compute $\sigma_i = h_i^{x_i}$. The signature for user u_i is $\sigma_i \in G_2$.

Verification. Given user's u_i pubic key v_i , a message M_i and signature σ_i, compute $h_i \leftarrow h(M_i)$ and accept if $e(g_1, \sigma_i) = e(v_i, h_i)$ holds.

Aggregation. Aggregating subset of users is $U = \left\{u_i, i = \overline{1,k},\right\} \subseteq \mathbb{U}$. Each user $u_i \in U$ provides signature $\sigma_i \in G_2$ on a message $M_i \in \{0,1\}^*$ of his choice, where $M_i \neq M_j$ for $i \neq j$, $i, j = \overline{1,k}$.

Compute aggregate signature as $\sigma = \prod_{i=1}^{k} \sigma_i$, $\sigma \in G_2$.

Aggregate Verification. Given aggregate signature $\sigma \in G_2$ for aggregating subset of users U and original messages $M_i \in \{0,1\}^*$ and public keys $v_i \in G_1$ for all users $u_i \in U$:

Step 1: Check $M_i \neq M_j$ for $i \neq j$, $i, j = \overline{1,k}$ to ensure that all messages are distinct. Reject otherwise.

Step 2: Compute $h_i \leftarrow h(M_i)$ for $i = \overline{1,k}$.

Step 3: Accept if $e(g_1, \sigma) = \prod_{i=1}^{k} e(v_i, h_i)$.

Latter can be proven by using properties of bilinear pairings:

$e(g_1, \sigma) = e(g_1, \prod_i \sigma_i) = e(g_1, \prod_i h_i^{x_i}) = \prod_i e(g_1, h_i)^{x_i} = \prod_i e(g_1^{x_i}, h_i) = \prod_i e(v_i, h_i)$.

It is also worth to note that aggregate verification time is linear to the number of users and aggregation can be done incrementaly.

In next section we will look at the security definition of this scheme.

4 Security defintion

The aggregate signature scheme is secure against existential forgery in the aggregate chosen-key model. In this model adversary attempts to forge an aggregate signature on messages of his choice by some set of users. An adversary is also given a single public key pk_1 and can addaptively request signatures with pk_1 on messages of his choice. At some point adversary will output additional public keys $pk_2, \ldots, pk_k$, messages $M_1, \ldots, M_k$ and an aggregate signature σ by k users. An adversary succeeds if the aggregate signature σ is a valid signature. Since adversary is able to generate public keys we need additional security requirement that an aggregate signature is only valid only if all messages are distinct. This is taken care of on aggregate verification step 1.

Definition 4.1. An aggregate forger $\mathcal{A}$ $(t, q_H, q_S, N, \epsilon)$-breaks an N-user aggregate signature scheme in the aggregate chosen-key model if: $\mathcal{A}$ runs in time at most t; $\mathcal{A}$ makes at most q_H queries to the hash function and at most q_S queries to the signing oracle; $\mathrm{adv}_{\mathcal{A}}$ is at least ϵ; and the forged aggregate signature is by at most N users. An aggregate signature scheme is $(t, q_H, q_S, N, \epsilon)$-secure against existential forgery in the aggregate chosen-key model if no forger $(t, q_H, q_S, N, \epsilon)$-breaks it.

Theorem 3.1. *Let G_1 and G_2 are (t,ϵ)-bilinear groups for co-Diffie-Hellman of order p, with respective generators g_1 and g_2 and with computable isomorphism ψ from G_1 to G_2 and a bilinear map $e : G_1 \times G_2 \to G_T$. Then the bilinear aggregate signature scheme on G_1 and G_2 is $(t, q_H, q_S, N, \epsilon)$-secure against existential forgery in the aggregate chosen-key model, where $t \le t' - 2c_{\mathcal{C}} \log p (q_H + q_S + N + 2)$, $\epsilon \ge 2e(q_S + N - 1)\epsilon'$, $c_{\mathcal{C}}$ is a small constant and e is a base of natural logarithm.*

5 Secutiry Proof

Proof of theorem 3.2. Asume that there exists an algorithm $\mathcal{A}$ that can forge aggregate signature for the bilinear groups for co-Diffie-Hellman G_1 and G_2. Let us then build an algorithm $\mathcal{C}$ that will use $\mathcal{A}$ to break co-CDH for these groups. We also define B_ζ to be probability distribution over $\{0,1\}$, where 1 is drawn with probability ζ and 0 with probability $1 - \zeta$.

At startup, $\mathcal{C}$ is given a challenge $\left(g_1, g_1^a, g_2^b\right)$, which it uses to construct a public key $v_1 \leftarrow g_1^a$. Then $\mathcal{C}$ gives a public key v_1 to $\mathcal{A}$ and let it run. During its running $\mathcal{A}$ will issue at most q_H queries to random oracle for message hashes and to signing oracle for message signatures by the private key corresponding to v_1. These oracles are under control of $\mathcal{C}$ and $\mathcal{A}$ will always asks the hash of message before it will ask its signature. To be able to answer $\mathcal{A}$'s queries, $\mathcal{C}$ will maintain some information as explained below. For every message M arriving from $\mathcal{A}$ at query step i $\mathcal{C}$ will:

- store message M in $M^{(i)}$;
- pick a random number $r^{(i)} \leftarrow \mathbb{Z}_p$;
- generate a random bit $s^{(i)} \leftarrow B_\zeta$;
- if $s^{(i)} = 1$ set $h^{(i)} \leftarrow g_2^b \psi(g_1)^{r^{(i)}}$and $\sigma^{(i)} \leftarrow *$a placeholder;
- if $s^{(i)} = 0$ set $h^{(i)} \leftarrow \psi(g_1)^{r^{(i)}}$ and $\sigma^{(i)} \leftarrow \psi(g_1^a)^{r^{(i)}}$; in this case note also that $\left(g_1, v_1, h^{(i)}, \sigma^{(i)}\right)$ is a valid co-Diffie-hellman tuple and hence $\sigma^{(i)}$ is a signature on any message whose hash is $h^{(i)}$.

After that, if *$\mathcal{A}$'s* query was for a hash, $\mathcal{C}$ will respond with $h^{(i)}$. If $\mathcal{A}$ asked for signature $\mathcal{C}$ will respond with $\sigma^{(i)}$ if $s^{(i)} = 0$. Otherwise $\mathcal{C}$ declares failure and halts.

Finaly $\mathcal{A}$ either halts with failure or returns a value $k \le N$, $k-1$ public keys $v_2, \ldots, v_k$, k messages $M_1, \ldots, M_k$ and forged signature σ. The messages must be all distinct and must not have asked a signature on M_1. Also $\mathcal{A}$ should have asked for the hash of every message in the aggregate.

Let μ be the mapping from aggregate indexing to indexing of $\mathcal{C}$. Then $M_i = M^{(\mu(i))}$ for all $i = \overline{1,k}$. For each $i = \overline{1,k}$ $\mathcal{C}$ sets $s_i \leftarrow s^{(\mu(i))}$, $r_i \leftarrow r^{(\mu(i))}$ and $h_i \leftarrow h^{(\mu(i))}$.

Now $\mathcal{C}$ continue only if $s_1 = 1$ and $s_i = 0$ for $i = \overline{2,k}$. From that we have that $h_1 = g_2^b \psi(g_1)^{r_1}$ and $h_i = \psi(g_1)^{r_i}$, $i = \overline{2,k}$.

The aggregate signature σ must satisfy aggregate verification $e(g_1, \sigma) = \prod_{i=1}^{k} e(v_i, h_i)$. Since for each $i = \overline{1,k}$ $\mathcal{C}$ sets $\sigma_i \leftarrow \psi(v_i)^{r_i}$ then

$e(g_1, \sigma_i) = e(g_1, \psi(v_i)^{r_i}) = e(g_1, \psi(v_i))^{r_i} = e(v_i, \psi(g_1))^{r_i} = e(v_i, \psi(g_1)^{r_i}) = e(v_i, h_i)$.

So σ_i is a valid signature for M_i. Now $\mathcal{C}$ constructs value of $\sigma_1 : \sigma_1 \leftarrow \sigma \left(\prod_{i=2}^{k} \sigma_i\right)^{-1}$. Then

$$e(g_1, \sigma_1) = e(g_1, \sigma) \prod_{i=2}^{k} e(g_1, \sigma_i)^{-1} =$$
$$= e(g_1, \sigma) \prod_{i=2}^{k} e(v_i, h_i)^{-1} =$$
$$= \prod_{i=1}^{k} e(v_i, h_i) \prod_{i=2}^{k} e(v_i, h_i)^{-1} = e(v_1, h_1).$$

Thus σ_1 is a valid co-CDH signature by key v_1 on a message whose hash is h_1. Also $(g_1, v_1, h_1, \sigma_1) = \left(g_1, v_1, g_2^b \psi(g_1)^{r_1}, \sigma_1\right)$ is a valid co-CDH tuple. But then $\left(g_1, g_1^a, g_2^b \cdot \psi(g_1)^{r_1} / \psi(g_1)^{r_1}, \sigma_1 / \psi(g_1)^{r_1}\right) = \left(g_1, g_1^a, g_2^b, \sigma_1 / \psi(g_1)^{r_1}\right)$ is also a valid co-CDH tuple. Therefore $\sigma_1 / \psi(g_1)^{r_1}$ is the solution for the posed computational co-CDH challenge.

Now we analyze the probability of success for forging the signature.

Probability of $\mathcal{C}$ answering all q_S *$\mathcal{A}$'a* signature queries is at least $(1-\zeta)^{q_S}$. Also $s_1 = 1$ occurs with probability of ζ. For each $i > 1$, $\mathcal{A}$ either asked for a signature of M_i, in which case $s_i = 0$ with probability 1 or it didn't and $s_i = 0$ with probability $(1-\zeta)$. So the probability that all $s_i = 0$ for all $i = \overline{2,k}$ is $(1-\zeta)^{k-1} \ge (1-\zeta)^{N-1}$. Hence, if $\mathcal{A}$ succeeds with probability ϵ, $\mathcal{A}$ succeeds with probability at least $\zeta (1-\zeta)^{q_S + N - 1} \cdot \epsilon$, which is maximized at $\zeta = 1/(q_S + N)$.

The running time of $\mathcal{C}$ includes running time of $\mathcal{A}$. Also $\mathcal{C}$ needs to evaluate group exponentiation to generate solution from forged aggreagate signature. $\mathcal{C}$ may need to complete this operation by using at most $2 \log p$ group actions and thus within at most $2 \log p$ time units on G_2. $\mathcal{C}$ may need to perform such operations, so overall running time of $\mathcal{C}$ is $t' \leq t - 2c_{\mathcal{C}} \log p(q_H + q_S + N + 2)$, where $c_{\mathcal{C}}$ is a small constant to take into acount some $\mathcal{C}$*'s* administrative overhead.□

6 Applications for aggregate signatures

There are several real-world applications where aggregate signatures could be utilized to compress resulting set of signatures on messages from many users.

For example in Public Key Infrastructure (PKI) of depth n each user receive a chain of n distinct certificates. This chain contains n different signatures used by n distinct Certificate Authorities (CAs) on those certificates. Since aggregation can be performed incrementally, each CA can incrementally aggregate its signature in the chain. Thus the certificate chains could be shortened.

Presented aggregate signature scheme can also be used in secure routing protocols for reducing the size of protocol packet. For example in Secure Border Gateway Protocol (SBGP) [4] routing protocol each router on the path receives a packet with a list of n signatures, which represents some certain path of the length n in the network. Each router then signs its own segment in the path, adds it to the list and forwards the resuling list of $n+1$ signatures to the next router. The size of the signature list is linear to the number of routers on the packet path. The proposed aggregate signature scheme could be used to recuce the size of the signature list by each router incrementaly aggregating its own signatures into one aggregate signature. The requirement that all messages are different is also satisfied for SBGP since each signature is done for different segment of the path.

7 Conclusion

This survey has presented an aggregate signature scheme based on bilinear pairings. The concept of aggregate signatures can be utilized in a number of real-world applications and several such examples are presented.

References

[1] D.Boneh, B.Lynn, C.Gentry, H.Shacham "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps", *Cryptology ePrint Archive*, Report 2002/175.

[2] D.Boneh, B.Lynn, H.Shacham "Short Signatures from the Weil Pairing", in *Advances in Cryptology - Asiacrypt* 2001, Lecture Notes in Computer Science, Vol. 263, Springer-Verlag, pp. 186-194, 1986.

[3] J.Wallén, "Short Signatures from Bilinear Pairings", survey at T-29.515 Cryptology: Special Topisc / Pairing-based Cryptography, January 30, 2003

[4] S.Kent, C.Lynn, K.Seo. "Secure Border Gateway Protocol (Secure BGP)". IEEE J. Selected Areas in Com., 18(4):582-92, April 2000.