Aggregate signatures from bilinear pairings

Tatiana Issaeva

March 16, 2003

Introduction

- some applications require signatures on many different messages produced by many different users
- number of signatures is linear to the number of users
- it is desirable to compress all signatures of the chain into a single signature
- such compresion can be achieved by aggregate signature scheme

Definitions

- G_1, G_2 and G_T - multiplicative cyclic groups of prime order p

- $e : G_1 \times G_2 \to G_T$ - a computable bilinear map

- g_1 - a generator of G_1, g_2 - a generator of G_2

- ψ - a computable isomorphism from G_1 to G_2: $\psi(g_1) = g_2$

Diffie-Hellman problems

- **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 $g_2^{ab} \in G_2$

- **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 $ab = c$

Diffie-Hellman problems

- Advantage of $\mathcal{A}$ in solving co-CDH in G_1 and G_2:

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

- $\mathcal{A}$ is said to (t, ϵ)-break co-CDH on G_1 and G_2 if $\mathcal{A}$ runs in time atmost t and $\mathbf{adv}_{\mathcal{A}} \geq \epsilon$

Bilinear groups

G_1 and G_2 - *bilinear groups* if group actions and 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

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

Aggregate signature scheme

Let $h : \{0,1\}^* \to G_2$ is a full-domain hash function

Key generation. Pick a random $x_i \leftarrow \mathbb{Z}_p$ (secret key) and compute $v_i \leftarrow g_1^{x_i}$ (public key) for some user u_i.

Signing. For user u_i, given 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: $U = \left\{ u_i, i = \overline{1, k,} \right\} \subseteq \mathbb{U}$

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 $i \neq j$, $i, j = \overline{1, k}$

Aggregate signature: $\sigma = \prod_{i=1}^{k} \sigma_i$, $\sigma \in G_2$

Aggregate verification

Given $\sigma \in G_2$, $M_i \in \{0,1\}^*$ and $v_i \in G_1$ for all users $u_i \in U$:

- check $M_i \neq M_j$ for $i \neq j$, $i, j = \overline{1,k}$
- compute $h_i \leftarrow h(M_i)$ for $i = \overline{1,k}$
- accept if $e(g_1, \sigma) = \prod_{i=1}^{k} e(v_i, h_i)$

Why works?

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

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

Aggregate signature scheme: notes

- aggregate verification time is linear to the number of users

- aggregation can be done incrementaly.

Aggregate signature scheme: security

secure against existential forgery in the aggregate chosen-key model

adversary is given a single public key pk_1 and can addaptevely request signatures with pk_1 on messages of his choice to forge an aggregate signature

additional security requirement: all messages must be distinct

Aggregate signature scheme: security

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.

N -number of users

q_H - number of queries to the hash function

q_S - number of queries to the signing oracle

Security theorem

Let G_1 *and* G_2 *are* (t, ϵ)*-bilinear groups for co-Diffie-Hellman of order* p *and* $e : G_1 \times G_2 \to G_T$ *is a bilinear map. 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*

$t \leq t' - 2c_{\mathcal{C}} \log p(q_H + q_S + N + 2)$,

$\epsilon \geq 2e(q_S + N - 1)\epsilon'$,

$c_{\mathcal{C}}$ *is a small constant*

e *is a base of natural logarithm.*

Security proof

- *assume* $\mathcal{A}$ can forge aggregate signature

- build an algorithm $\mathcal{C}$ that use $\mathcal{A}$ to break co-CDH

- define B_ζ - probability distribution over {0,1}, where 1 is drawn with probability ζ and 0 with probability $1 - \zeta$.

Security proof (cont)

- $\mathcal{C}$ is given a challenge $\left(g_1, g_1^a, g_2^b\right)$
- $\mathcal{C}$ constructs a public key $v_1 \leftarrow g_1^a$
- $\mathcal{C}$ gives v_1 to $\mathcal{A}$ and let it run

$\mathcal{A}$ issues at most:

- q_H queries to random oracle for message hashes
- q_S queries to signing oracle for message signatures by the private key corresponding to v_1

Security proof (cont)

For every message M from $\mathcal{A}$ at query step i
$\mathcal{C}$:

- 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 $s^{(i)} = 1$: $h^{(i)} \leftarrow g_2^b \psi(g_1)^{r^{(i)}}$and $\sigma^{(i)} \leftarrow *$;
- if $s^{(i)} = 0$: $h^{(i)} \leftarrow \psi(g_1)^{r^{(i)}}$ and $\sigma^{(i)} \leftarrow \psi(g_1^a)^{r^{(i)}}$

Security proof (cont)

$\mathcal{A}$ returns:

- value $k \leq N$
- $k-1$ public keys $v_2, \ldots, v_k$
- k messages $M_1, \ldots, M_k$
- forged signature σ

μ - mapping: $M_i = M^{(\mu(i))}$ for $i = \overline{1,k}$.

For $i = \overline{1,k}$: $s_i \leftarrow s^{(\mu(i))}$, $r_i \leftarrow r^{(\mu(i))}$, $h_i \leftarrow h^{(\mu(i))}$

Security proof (cont)

$\mathcal{C}$ continue only if $s_1 = 1$ and $s_i = 0$ for $i = \overline{2,k}$

- $h_1 = g_2^b \psi(g_1)^{r_1}$, $h_i = \psi(g_1)^{r_i}$
- $e(g_1, \sigma) = \prod_{i=1}^{k} e(v_i, h_i)$
- σ_i is a valid signature for M_i:

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

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

Security proof (cont)

$\mathcal{C}$ constructs σ_1 : $\sigma_1 \leftarrow \sigma \left(\prod_{i=2}^{k} \sigma_i\right)^{-1}$

σ_1 is a valid signature:

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

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

Security proof (cont)

- $(g_1, v_1, h_1, \sigma_1) = \Big(g_1, v_1, g_2^b \psi(g_1)^{r_1}, \sigma_1\Big)$ - valid co-CDH tuple

- $\Big(g_1, g_1^a, g_2^b, \sigma_1/\psi(g_1)^{r_1}\Big)$ is also a valid co-CDH tuple

- $\sigma_1/\psi(g_1)^{r_1}$ - solution for computational co-CDH challenge

Security proof (cont)

$\mathcal{A}$ succeeds with probability at least:

- $\zeta\,(1-\zeta)^{q_S+N-1}\cdot\epsilon$
- it is maximized at $\zeta = 1/\,(q_S+N)$.
- running time of $\mathcal{C}$:

$t' \leq t - 2c_{\mathcal{C}} \log p(q_H + q_S + N + 2)$

$\square$

Applications

in Public Key Infrastructure (PKI):

- chain of n distinct certificates contains n different signatures usued by n distinct Certificate Authorities (CAs)

- each CA can incrementally aggregate its signature in the chain

- certificate chains could be shortened

Applications (cont)

in secure routing protocols.

Example: Secure Border Gateway Protocol (SBGP)

- each router on the path receives a packet with a list of n signatures

- each router 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

- size of the signature list is linear to the number of routers on the packet path

- aggregate signature can reduce size of a packet

Reference

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

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.