Identity-based encryption: compress many public keys into short public parameters

Broadcast encryption: encrypt to many users with a short ciphertext (sublinear in number of users)

<u>Setup</u> (N): Takes as input the number of users N and outputs a master public key mpk and a set of decryption keys $sk_1, \dots, sk_N$

<u>Encrypt</u> (mpk, S, m): Takes a "broadcast set" $S \subseteq [N]$ and a message m and outputs a ciphertext ct

<u>Decrypt</u> (sk_i, S, ct): Takes a decryption key sk_i, the broadcast set $S \subseteq [N]$ and a ciphertext and outputs a message

<u>Correctness</u>:

$$\begin{array}{l} (mpk, sk_1, \dots, sk_N) \leftarrow \text{Setup}(N) \\ ct \leftarrow \text{Encrypt}(mpk, S, m) \end{array} \implies \begin{array}{l} \forall i \in S: \\ \quad \text{Decrypt}(sk_i, S, ct) = m \end{array}$$

<u>Security</u>:

<u>adversary</u> | <u>challenger</u>

$b \in \{0,1\}$

$(mpk, sk_1, \dots, sk_N) \leftarrow \text{Setup}$

$\xleftarrow{\quad mpk \quad}$

$\xrightarrow{\quad i \quad}$

$\xleftarrow{\quad sk_i \quad}$

$\xrightarrow{\quad S,\ m_0,\ m_1 \quad}$

$\xleftarrow{\quad \text{Encrypt}(mpk, S, m_b) \quad}$

$b' \in \{0,1\}$

<u>Requirement</u>: for all indices $i \in [N]$ queried by $\mathcal{A}$, it holds that $i \notin S$.

Scheme is (adaptively) secure if for all efficient adversaries $\mathcal{A}$:

$$\left| \Pr[b'=1 \mid b=0] - \Pr[b'=1 \mid b=1] \right| \leq \text{negl.}$$

<u>Succinctness</u>: $|ct| = o(N) \cdot \text{poly}(\lambda)$ where N is the number of users and λ is the security parameter

Without succinctness, can just concatenate N independent ciphertexts together

Boneh-Gentry-Waters (BGW) broadcast encryption scheme:

<u>Key idea</u>: use powers in the exponent

public parameters will contain group elements of the form

$$g \quad \underbrace{g^{\alpha} \quad g^{\alpha^2} \quad \cdots \quad g^{\alpha^N}}_{\substack{\text{"public keys" associated} \\ \text{with users } 1,\ldots,N}} \quad \text{---} \quad \underbrace{g^{\alpha^{N+2}} \quad g^{\alpha^{N+3}} \quad \cdots \quad g^{\alpha^{2N}}}_{\substack{\text{additional terms used for} \\ \text{decryption}}}$$

hole at $g^{\alpha^{N+1}}$ (used for security)

key idea: $g^{\alpha^{N+1}}$ not given out in base group $\leftarrow$ can encrypt with respect to this

to encrypt message $m \in \mathbb{G}_T$, sample $r \xleftarrow{R} \mathbb{Z}_p$ and compute

$$\left(g^r,\ e(g^{\alpha}, g^{\alpha^N})^r \cdot m \right)$$

$$\hookleftarrow \quad e(g^{\alpha}, g^{\alpha^N}) = e(g, g^{\alpha^{N+1}})$$

$g^{\alpha^{N+1}}$ <u>cannot</u> be given out: otherwise can compute $e(g^r, g^{\alpha^{N+1}})$

<u>problem</u>: how to decrypt?

if $i \in S \rightarrow$ should be able to compute $e(g, g^{\alpha^{N+1}})^r$

if $i \notin S \rightarrow$ should not be able to compute this

<u>approach</u>: let $g_i = g^{\alpha^i}$

observe that
$$e(g_i, g_j) = e(g^{\alpha^i}, g^{\alpha^j}) = e(g, g_{i+j})$$

in the ciphertext, include $\prod_{j \in S} g^r_{N+1-j}$

in this case,

$$e\left(g_i, \prod_{j \in S} g^r_{N+1-j}\right) = \prod_{j \in S} e(g_i, g_{N+1-j})^r$$

$$= e(g_i, g_{N+1-i})^r \prod_{j \in S \setminus \{i\}} e(g_i, g_{N+1-j})^r$$

$$= \underbrace{e(g, g_{N+1})^r}_{\text{quantity of interes!}} \prod_{j \in S \setminus \{i\}} \underbrace{e(g, g_{N+1-j+i})^r}_{= e(g^r, \underline{g_{N+1-j+i}})}$$

when $i, j \in [N]$,

$$2 \leq N+1-j+i \leq 2N$$

if $i \neq j$: $N+1-j+i \neq N+1$

this means $g_{N+1-j+i}$ is contained in the public parameters if $i \notin S$, then

$$e\left(g_i, \prod_{j \in S} g_{N+1-j}^r\right) \text{ does } \underline{\text{not}} \text{ contain } e(g, g_{N+1})^r$$

<u>problem</u>: knowledge of g_i is sufficient to decrypt (when $i \in S$), but g_i is <u>public</u>!

<u>approach</u>: set secret key for user i as g_i^γ and publish $v = g^\gamma$ in the public parameters

ciphertext now contains

$$\left[v \cdot \prod_{j \in S} g_{N+1-j} \right]^r$$

now, when we compute

$$e\left(g_i, v^r \cdot \prod_{j \in S} g_{N+1-j}^r\right)$$

$$= e(g_i, v^r)\, e\left(g_i, \prod_{j \in S} g_{N+1-j}^r\right)$$

$$= \underbrace{e(g_i, v^r)}_{\text{new term introduced during decryption}} \underbrace{e(g, g_{N+1})^r \prod_{j \in S \setminus \{i\}} e(g^r, g_{N+1-j+i})}_{\text{same as before}}$$

$$e(g_i, v^r) = e(g_i, g^{\gamma r}) = e(\underbrace{g_i^\gamma}_{\text{secret key}}, \underbrace{g^r}_{\text{part of ciphertext}})$$

Setup (N): Sample $\alpha \xleftarrow{R} \mathbb{Z}_p$ and $\gamma \xleftarrow{R} \mathbb{Z}_p$

Let $g_i = g^{\alpha^i}$ and $v = g^{\gamma}$

$mpk = (g, g_1, g_2, \ldots, g_N, g_{N+2}, g_{N+3}, \ldots, g_{2N}, v)$

$sk_i = (g_i, g_i^{\gamma})$

Encrypt (mpk, S, m): $r \xleftarrow{R} \mathbb{Z}_p$

$e(g, g_{N+1})^r$

$$ct = \left(g^r, \left[v \cdot \prod_{i \in S} g_{N+1-i} \right]^r, \underbrace{e(g, g_N)^r}_{\text{KEM}} \cdot m \right)$$

Decrypt (sk_i, S, ct):

$sk_i = (g_i, d_i) \qquad ct = (ct_1, ct_2, ct_3)$

$$z \leftarrow \frac{e(g_i, ct_2)}{e\left(d_i \prod_{j \in S \setminus \{i\}} g_{N+1-j+i}, ct_1\right)}$$

output ct_3 / z

Correctness:

$$e(g_i, ct_2) = e\Big(g_i, v \cdot \prod_{j \in S} g_{N+1-j}\Big)^r$$

since $i \in S$

$$= e(g_i, g_{N+1-i})^r \, e\Big(g_i, v \cdot \prod_{j \in S \setminus \{i\}} g_{N-1-j}\Big)^r$$

$$= e(g, g_{N+1})^r \cdot e\Big(g_i, v \cdot \prod_{j \in S \setminus \{i\}} g_{N-1-j}\Big)^r$$

$$= e(g, g_{N+1})^r \, e(g_i, v)^r \prod_{j \in S \setminus \{i\}} e(g_i, g_{N-1-j})^r$$

$$e\Big(d_i \cdot \prod_{j \in S \setminus \{i\}} g_{N+1-j+i}, g^r\Big) = e(d_i, g)^r \prod_{j \in S \setminus \{i\}} e(g_{N+1-j+i}, g^r)$$

$$= e(g_i^\gamma, g)^r \prod_{j \in S \setminus \{i\}} e(g_i, g_{N+1-j})^r$$

$$= e(g_i, v)^r \prod_{j \in S \setminus \{i\}} e(g_i, g_{N+1-j})^r \qquad [v = g^\gamma]$$

Thus, $$\frac{e(g_i, ct_2)}{e\Big(d_i \prod_{\substack{j \in S \\ j \neq i}} g_{N+1-j+i}, ct_1\Big)} = e(g, g_{N+1})^r$$ Correctness holds!

Security: N- bilinear Diffie-Hellman exponent (N-BDHE) assumption

sample $\alpha \xleftarrow{R} \mathbb{Z}_p$, $\gamma \xleftarrow{R} \mathbb{Z}_p$, and $r \xleftarrow{R} \mathbb{Z}_p$:

distinguish $\quad (g^\gamma, g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^N}, g^{\alpha^{N+2}}, \ldots, g^{\alpha^{2N}}, e(g,g)^{\alpha^{N+1}\gamma})$

from $\quad (g^\gamma, g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^N}, g^{\alpha^{N+2}}, \ldots, g^{\alpha^{2N}}, e(g,g)^r)$

similar to BDHE except exponents are α^{N+1} and γ, and adversary is given additional powers $g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^N}, g^{\alpha^{N+2}}, \ldots, g^{\alpha^{2N}}$

assumption is often called a "q-type assumption" (size of assumption depends on a scheme parameter)

Note: if $p-1$ has a factor $t \leq N$, then there is an algorithm for computing α from $(g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^N})$ in time $\tilde{O}(\sqrt{p/N} + \sqrt{N})$.

Generic discrete log algorithm runs in time $\tilde{O}(\sqrt{p})$, so hardness of assumption degrades with size of assumption.

Proving security: Will consider weaker notion of security where adversary declares the challenge set S before seeing the security parameters (selective security)

We do not know how to prove the BGW scheme is adaptively secure (i.e., in a model where adversary can choose the set S after seeing public parameters).

Proof of selective security:

1. Algorithm B receives N-DHBE challenge

$$\begin{array}{cccccccc} (h, & g, & g_1, \ldots, & g_N, & g_{N+2}, \ldots, & g_{2N}, & T) \\ \| & \| & & \| & \| & \| & \| \\ g^r & g^\alpha & & g^{\alpha^N} & g^{\alpha^{N+2}} & g^{\alpha^{2N}} & e(g,g)^{\alpha^{N+1} r} \text{ or } e(g,g)^t \end{array}$$

2. Algorithm B runs A. Algorithm A chooses a set $S \subseteq [N]$. Algorithm B now needs to construct public key and decryption keys for indices $i \notin S$.

In the ciphertext, we will want to use $h = g^r$ as the first component of the challenge ciphertext. Then $e(g,g)^{\alpha^{N+1} r}$ is the <u>real</u> blinding factor while $e(g,g)^t$ is a <u>random</u> blinding factor.

If $h = g^r$ is the first element, we need a way to construct

$$\left[v \cdot \prod_{j \in S} g_{N+1-j} \right]^r$$

but we do <u>not</u> know r (and cannot compute g^r_{N+1-j}).

<u>Idea</u>: Sample $s \xleftarrow{R} \mathbb{Z}_p$.

Set $v = \left[g^s / \prod_{j \in S} g_{N+1-j} \right]$.

Now $v \cdot \prod_{j \in S} g_{N+1-j} = g^s$.

Then $\left(v \cdot \prod_{j \in S} g_{N+1-j} \right) = g^{rs} = h^s$

which we can compute ourselves!

$$mpk = (g, g_1, g_2, \ldots g_N, g_{N+2}, \ldots, g_{2N}, v)$$

Next, we need to construct keys for $i \notin S$.

But we don't know α^i...

Real scheme: $d_i = g_i^\gamma = g^{\alpha^i \gamma} = v^{\alpha^i}$ $\quad [v = g^\gamma]$

For our value of v, we can write

$$d_i = v^{\alpha^i} = \frac{g^{s\alpha^i}}{\prod_{j\in S} g_{N+1-j}^{\alpha^i}} = \frac{g_i^s}{\prod_{j\in S} g_{N+1-j+i}}$$

We can compute this as long as $i \notin S$!

Observe: reduction can only construct key for $i \notin S$. Important since otherwise, reduction does not need A and can decrypt itself.

3. Algorithm B gives mpk and d_i for $i \notin S$ to A. Algorithm A outputs a message m to be encrypted.

[We again consider a stronger pseudorandomness game where the adversary tries to distinguish ciphertext from a random string.]

To construct challenge ciphertext:

$$ct = (h, h^s, m \cdot T)$$

Observe: $h = g^r$, so second component should be

$$\left(v \cdot \prod_{j\in S} g_{N+1-j}\right)^r = (g^s)^r = h^s$$

if $T = e(g,g)^{\alpha^{N+1} r}$, then this is a real ciphertext

if $T = e(g,g)^t$, then this is random group element