CS 6501 Week 6: Pairing-Based Cryptography

Application 2: Short signatures [Boneh, Lynn, Shacham, 2001]

Existing signature candidates: [128-bit security level]

RSA signatures: 2048 bits $\tilde{O}(\lambda^3)$ bits long
ECDSA signatures: 512 bits 4λ bits long
Schnorr signatures: 384 bits 3λ bits long
BLS signatures: 256 bits 2λ bits long [shortest practical / implemented signature]

$\mathrm{KeyGen}(1^\lambda) \to (vk, sk)$: $s \xleftarrow{R} \mathbb{Z}_p$ sk: s
vk: (g, g^s)

$\mathrm{Sign}(sk, m) \to \sigma$: $\sigma \leftarrow H(m)^s$ where $H: \mathcal{M} \to \mathbb{G}$ is a hash function (modeled as a random oracle)

$\mathrm{Verify}(vk, m, \sigma)$: check $e(\sigma, g) \stackrel{?}{=} e(H(m), g^s)$

Correctness: $e(\sigma, g) = e(H(m)^s, g) = e(H(m), g)^s = e(H(m), g^s)$ by bilinearity

Security: From CDH in $\mathbb{G}$ in the random oracle model:

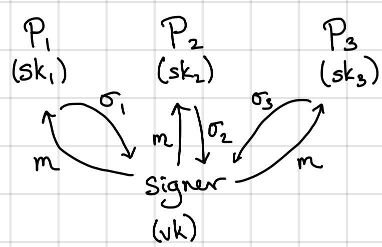
CDH assumption: given $g, g^a, g^b \in \mathbb{G}$, compute $g^{ab} \in \mathbb{G}$

Proof Sketch: Very similar to security proof for FDH:

- Given CDH challenge (g, g^a, g^b), reduction sets verification key to $vk = (g, g^a)$
- Assume without loss of generality that adversary queries random oracle before each signing query
- Choose one of the RO queries and program response to g^b [correct forgery is then g^{ab}]
- Remainder of signing queries can be simulated since reduction chooses the exponents (so can compute $H(m)^a$)

Properties:
- Signature is a single group element: ~256 bits (using point compression) [asymtotically: 2λ bits]
- Signature scheme naturally supports threshold signing, aggregation (i.e., compressing multiple signatures into one)

Threshold BLS signatures: Protect secret key by splitting it into many independent "shares" and giving shares to different parties



Goals:
1. Given $\sigma_1, \sigma_2, \sigma_3$, should be able to obtain signature σ on m (with respect to vk)
2. Given a subset of the key-shares $\{sk_1, sk_2, sk_3\}$, should not be able to sign (with respect to vk)

Recall signing in BLS: output $\sigma \leftarrow H(m)^s$ where $a \in \mathbb{Z}_p$ is secret key

To thresholdize BLS, choose $s_1, \ldots, s_n \xleftarrow{R} \mathbb{Z}_p$ such that $s_1 + s_2 + \cdots + s_n = s \in \mathbb{Z}_p$

↳ Each party's individual signing key is s_i, and signs using standard BLS

↳ Given $\sigma_1 = H(m)^{s_1}, \ldots, \sigma_n = H(m)^{s_n}$, we can compute

$$\sigma = \prod_{i \in [n]} \sigma_i = \prod_{i \in [n]} H(m)^{s_i} = H(m)^{\sum_{i \in [n]} s_i} = H(m)^s$$

↳ Security: Each party is implementing a BLS scheme (so partial signatures σ_i are unforgeable)

Puzzle: This is an "n-out-of-n" threshold signature scheme (i.e., need n out of n signatures to reconstruct).
Can we build a "t-out-of-n" threshold signature scheme (where any subset of t signatures suffice to reconstruct)?
↳ Will revisit when we discuss Shamir secret sharing.

Aggregating BLS signatures: BLS signatures support a property called aggregation:
given message-signature pairs $(m_1, \sigma_1), \ldots, (m_t, \sigma_t)$ under vk,
can compress into a single BLS signature σ that authenticates $(m_1, \ldots, m_t)$
Suppose we have $(m_1, \sigma_1), \ldots, (m_t, \sigma_t)$ where each $\sigma_i = H(m_i)^s$.
Observe that:

$$\prod_{i \in [n]} \sigma_i = \prod_{i \in [n]} H(m_i)^s = \left[\prod_{i \in [n]} H(m_i)\right]^s$$

Then, define the aggregate signature $\sigma = \prod_{i \in [n]} \sigma_i$. To verify σ on $(m_1, \ldots, m_t)$, compute

$$e(g, \sigma) \stackrel{?}{=} e\left(g^s, \prod_{i \in [n]} H(m_i)\right)$$

$$\begin{array}{cc} \| & \| \\ e\left(g, \left[\prod_{i \in [n]} H(m_i)\right]^s\right) & e\left(g^s, \prod_{i \in [n]} H(m_i)\right) \end{array}$$

Very useful property when we have many signatures and want to compress them (e.g., certificate chains, Bitcoin transactions, etc.)

Open Question: Can we obtain even shorter signatures?
Lower bound: for λ bits of security, need at least λ bits
Feasibility result: Using indistinguishability obfuscation, we can do this, but no other constructions known...
Source of difficulty: Need to consider exponential-time adversaries (security against 2^λ-time adversaries)
↳ generic discrete log algorithm is reason for 2λ size in BLS

Application 3: Identity-based encryption

Beyond public-key encryption: pairing-based cryptography enabled for the first time new forms of advanced cryptographic primitives beyond traditional public-key encryption and digital signatures

Going beyond public-key encryption: with traditional PKE, sender needs to know public key of recipient in order to encrypt
Question: Can the public key be an arbitrary string (e.g., email address, username, etc.)?

Identity-based encryption [Shamir, 1984]: encrypt with respect to identities
↳ major open problem resolved by Boneh-Franklin in 2001 using pairings (and also concurrently by Cocks in 2001)

Schema: $\text{Setup}(1^\lambda) \to (\underbrace{mpk}_{\text{global public parameters}}, \underbrace{msk}_{\text{master secret key}})$

$\text{Encrypt}(mpk, id, m) \to ct_m$ [encrypts message m with respect to identity id]

$\text{KeyGen}(msk, id) \to sk_{id}$ [generates a secret decryption key for the identity id]

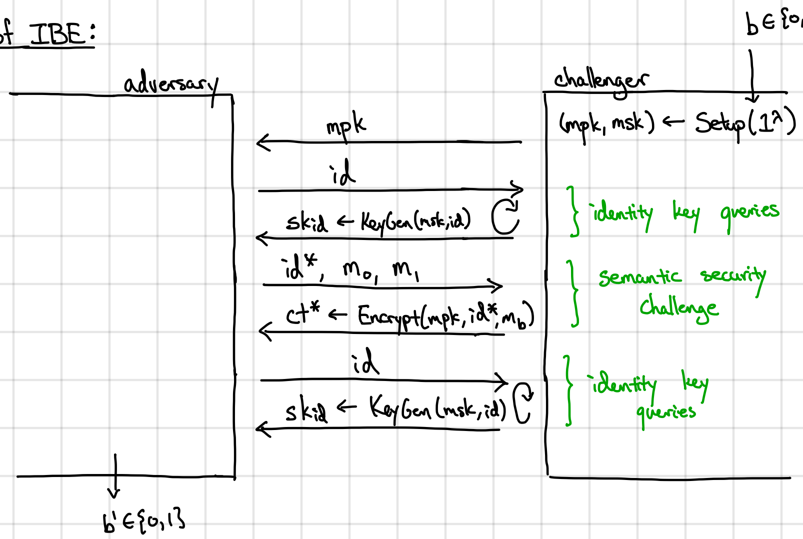
$\text{Decrypt}(sk_{id}, ct_m) \to m / \perp$ [decryption should output m if ct_m is encryption to id and $\perp$ otherwise]

↳ challenge of IBE is to compress exponential number of (public/secret) key-pairs into a single set of short public parameters

Correctness: for all messages m and identities id, if we generate $(mpk, msk) \leftarrow \text{Setup}(1^\lambda)$ and $sk_{id} \leftarrow \text{KeyGen}(msk, id)$,

$$\Pr[\text{Decrypt}(sk_{id}, \text{Encrypt}(mpk, id, m)) = m] = 1$$

Security of IBE:



$$\text{IBEAdv}[A] = |\Pr[b'=1 \mid b=0] - \Pr[b'=1 \mid b=1]|$$

↳ Require that A does not query for a decryption key for its target identity id^* (otherwise can trivially break security)

Boneh-Franklin IBE Scheme:

$\text{Setup}(1^\lambda) \to (mpk, msk)$: $s \xleftarrow{R} \mathbb{Z}_p$

mpk: $h = g^s$ msk: s

$\text{Encrypt}(mpk, id, m) \to ct_m$: $r \xleftarrow{R} \mathbb{Z}_p$

$ct_m = (g^r, m \cdot e(h^r, H(id)))$

How to decrypt?

$$e(h^r, H(id)) = e(g^{rs}, H(id)) = e(\underbrace{g^r}_{\text{included in ciphertext}}, \underbrace{H(id)^s}_{\text{secret key for identity id}})$$

$\text{KeyGen}(msk, id) \to sk_{id}$: $H(id)^s$

Compare with ElGamal:

$\text{Setup}(1^\lambda) \to (pk, sk)$: $s \xleftarrow{R} \mathbb{Z}_p$

pk: $h = g^s$ sk: s

$\text{Encrypt}(pk, m) \to ct_m$: $r \xleftarrow{R} \mathbb{Z}_p$

$ct_m = (g^r, m \cdot h^r)$

Key idea in pairing-based cryptography: exploit bilinearity: two ways to compute each quantity ↳ using public parameters ↳ using secret parameters

BLS signatures:

verification relation: $e(H(m)^s, g) = e(H(m), g^s)$

exponent can be "moved"

computed using the secret signing key

part of the public verification parameters

Boneh-Franklin IBE:

decryption relation: $e(g^r, H(id)^s) = e((g^s)^r, H(id))$

secret key

public parameters

Security of Boneh-Franklin IBE: Will rely on the bilinear DDH (BDDH) assumption (and modeling H as a random oracle)

$$(g, g^a, g^b, g^c, e(g,g)^{abc}) \overset{c}{\approx} (g, g^a, g^b, g^c, e(g,g)^r) \text{ where } a,b,c,r \xleftarrow{R} \mathbb{Z}_p$$

Proof idea. Given BDDH challenge (g, g^a, g^b, g^c, T):

this is the algorithm/adversary we construct in the reduction

- Set $mpk = h = g^a$ (so a is the corresponding secret key, unknown to the simulator)
- Assume (without loss of generality) that adversary queries RO on each identity before making the corresponding key query or challenge query
- Guess which RO query corresponds to challenge identity id^*
  - On RO query $id \neq id^*$: choose random $x \xleftarrow{R} \mathbb{Z}_p$ and reply with g^x
  - On RO query $id = id^*$: reply with g^b (from the challenge)

  In both cases, the response is uniformly random
- On a key query for identity $id \neq id^*$: reply with $(g^a)^x$ where x is the exponent chosen for $H(id)$
  - ↳ Observe that by construction, $sk_{id} = g^{ax} = (g^x)^a = H(m)^a$, so these keys are correctly simulated
- For the challenge ciphertext, reply with $(g^c, m \cdot T)$ where g^c, T are from the challenge
  - ↳ Observe that if $T = e(g,g)^{abc}$, then in particular

$$T = e(g,g)^{abc} = e(g^{ac}, g^b) = e((g^a)^c, g^b) = e(h^c, H(id^*)),$$

exactly as required in the real scheme.

Therefore, under the BDDH assumption, the challenge ciphertext is independent from two random group elements (independent of the message), and so security holds.