CS355

Lecture #7

Pairing-based Cryptography

Last time: Elliptic curves.

Big idea: elliptic curves let us build prime-order groups that are faster & have more compact representations than $\mathbb{Z}_p^*$, for equivalent security.

$$E: y^2 = x^3 + Ax + B$$

$\Rightarrow E(\mathbb{F}_p)$ is the set of solutions (x,y) to E, for $x, y \in \mathbb{F}_p$.

$\Rightarrow \boxplus$ is the group operation; g^a is

$$\underbrace{g \boxplus g \boxplus \cdots \boxplus g}_{a \text{ times}}$$

$\leftarrow$ computed in $O(\log a)$ time via "square & multiply"

Today: Pairings & applications

1. What is a pairing?
2. Attacking discrete log [MOV'93]
3. Tripartite DH [Joux'00]
4. Signatures [BLS'01]
   ⇒ and: hashing to elliptic curves!
5. Identity-based encryption [BF'01]

⇒ Pairings have many more applications, too! e.g., SNARKs — Lecture #11.

$\boxed{1}$ What is a pairing?

Definition: Let $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups of prime order q.

A symmetric pairing is a mapping

$$e: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$$

with 3 properties:

$\Rightarrow$ Bilinearity: $\forall a,b \in \mathbb{Z}_q,\ g \in \mathbb{G}$

$$e(g^a, g^b) = e(g,g)^{ab}$$

$\Rightarrow$ Non-degeneracy: if g generates $\mathbb{G}$, $e(g,g)$ generates $\mathbb{G}_T$.

$\Rightarrow$ Efficiency: e is efficiently computable.

Why these properties?

$\Rightarrow$ Need e to be non-degenerate because $e(g,g) \mapsto \mathbb{1} \in \mathbb{G}_T$ is bilinear

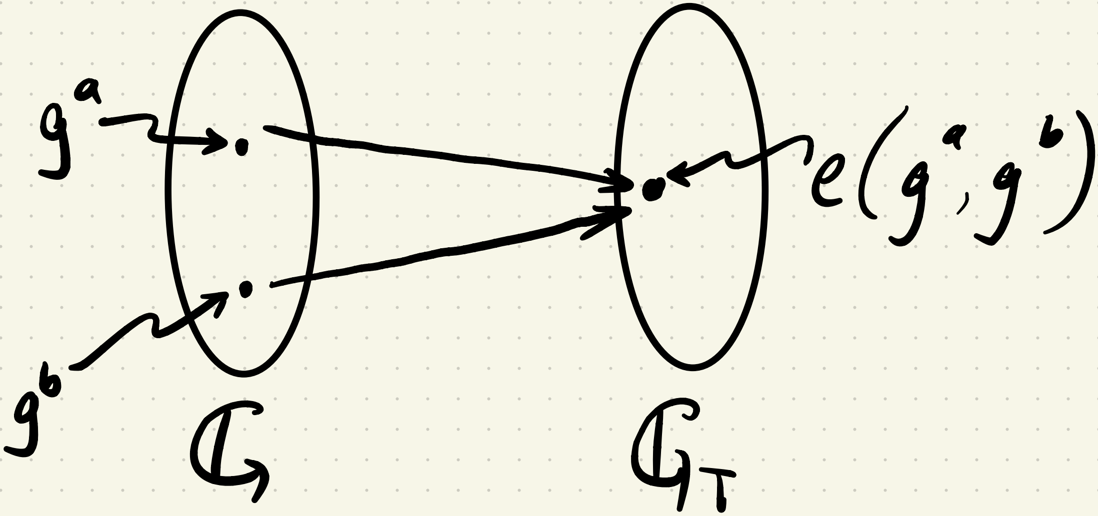
the identity element in $\mathbb{G}_T$.

$\Rightarrow$ Need e to be efficiently computable because

$$e(g^a, g^b) \mapsto g^{ab} \in \mathbb{G}_T$$

is bilinear & non-degenerate but is hard to compute (we hope — this is the computational D-H problem!)

The pairing e:



Question: Given a pairing

$$e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$$

$r \xleftarrow{\$} \mathbb{Z}_q$

how can we efficiently solve DDH in $\mathbb{G}$? Recall: want to distinguish (g^a, g^b, g^{ab}) from (g^a, g^b, g^r).

$\Rightarrow$ $\mathbb{G}$ is sometimes called a gap-DH group: CDH is hard, DDH is easy.

$\boxed{2}$ Attacking discrete log

[Menezes, Okamoto, Vanstone '93]

For **any** elliptic curve E, for **some** $\alpha \in \mathbb{Z}$ there is a pairing

$$e : E(\mathbb{F}_p) \times \mathbb{G}^* \to \mathbb{F}_{p^\alpha}$$

In general, $\mathbb{G}^* \neq E(\mathbb{F}_p)$. This is an asymmetric pairing.

Idea: Use pairing to "convert" EC dlog to $\mathbb{F}_{p^\alpha}$ dlog, which is easy if α is small:

$\Rightarrow$ dlog over $E(\mathbb{F}_p)$: $O(\sqrt{p})$

$\Rightarrow$ dlog over $\mathbb{F}_{p^\alpha}$: $2^{\tilde{O}(\sqrt[3]{\alpha \log p})}$

$\rightarrow$ To avoid MOV, most curves have $\alpha > 2^{100}$.

In more detail:

Assume g^* generates $\mathbb{G}^*$.

Then, to solve dlog:

(1) $\gamma \leftarrow e(g, g^*)$

$\rightarrow$ γ generates $\mathbb{G}_T$ (Non-degeneracy)

(2) $\delta \leftarrow e(g^a, g^*)$

$\rightarrow$ $\delta \triangleq \gamma^a$ (Bilinearity)

(3) Compute $\log_\gamma \delta$ in $\mathbb{G}_T$

$\Rightarrow$ this is a !

$\boxed{3}$ Tripartite Diffie-Hellman key exchange [Joux '00]

3 parties

Recall 2-party DH:

Alice: $a \xleftarrow{\$} \mathbb{Z}_q$

Bob: $b \xleftarrow{\$} \mathbb{Z}_q$

$$\xrightarrow{A \triangleq g^a}$$

$$\xleftarrow{B \triangleq g^b}$$

Alice: $B^a = g^{ab}$

Bob: $A^b = g^{ab}$

Both: $k \leftarrow KDF(g^{ab})$

Security comes from DDH:

$$(g^a, g^b, g^{ab}) \approx_c (g^a, g^b, g^r) \qquad r \xleftarrow{\$} \mathbb{Z}_q$$

→ So: k is indistinguishable from random.

3-party DH:

Alice

$a \xleftarrow{\$} \mathbb{Z}_q$

$A \leftarrow g^a$

Bob

$b \xleftarrow{\$} \mathbb{Z}_q$

$B \leftarrow g^b$

Card

$c \xleftarrow{\$} \mathbb{Z}_q$

$C \leftarrow g^c$

$$\begin{tikzcd} & \text{Alice} \arrow[dl, shift right, "A"'] \arrow[dr, shift left, "A"] & \\ \text{Bob} \arrow[ur, shift right, "B"'] \arrow[rr, shift right, "B"'] & & \text{Card} \arrow[ul, shift left, "C"] \arrow[ll, shift right, "C"'] \end{tikzcd}$$

Alice : $e(B,C)^a = e(g,g)^{abc}$

Bob : $e(A,C)^b = e(g,g)^{abc}$

Card : $e(A,B)^c = e(g,g)^{abc}$

Security of 3°DH:

Bilinear DDH assumption:

$$\left(g, g^a, g^b, g^c, e(g,g)^{abc}\right) \approx_c \left(g, g^a, g^b, g^c, e(g,g)^r\right) \qquad r \xleftarrow{\$} \mathbb{Z}_q$$

$\Rightarrow$ Pairings give us one multiplication "in the exponent"

$\Rightarrow$ But *not* 2!

Open problem: N-party key exchange for $N > 3$.

Also: 3°DH from other assumptions

$\boxed{4}$ Signatures from pairings

[Boneh, Lynn, Shacham '01]

<u>Why</u> another signature scheme?

$\Rightarrow$ signature is 1 element of $\mathbb{G}$

↳ in practice, <u>not</u> smaller than other EC-based signatures @ 128-bit security

$\Rightarrow$ signatures can be aggregated

↳ turn many signatures on same message into one "multi-signature"

$\Rightarrow$ Blockchains: many people sign each block.

BLS signature definitions:

Fix $\mathbb{G}, \mathbb{G}_T, g \in \mathbb{G}, e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$

(order q)

$H : \{0,1\}^* \to \mathbb{G}$

modeled as a random oracle

"hash to curve"

- Key Gen() $\to (pk, sk)$:

  $a \xleftarrow{\$} \mathbb{Z}_q$

  return $(g^a, a) \in \mathbb{G} \times \mathbb{Z}_q$

- Sign(sk, m) $\to \sigma$

  return $H(m)^{sk} \in \mathbb{G}$

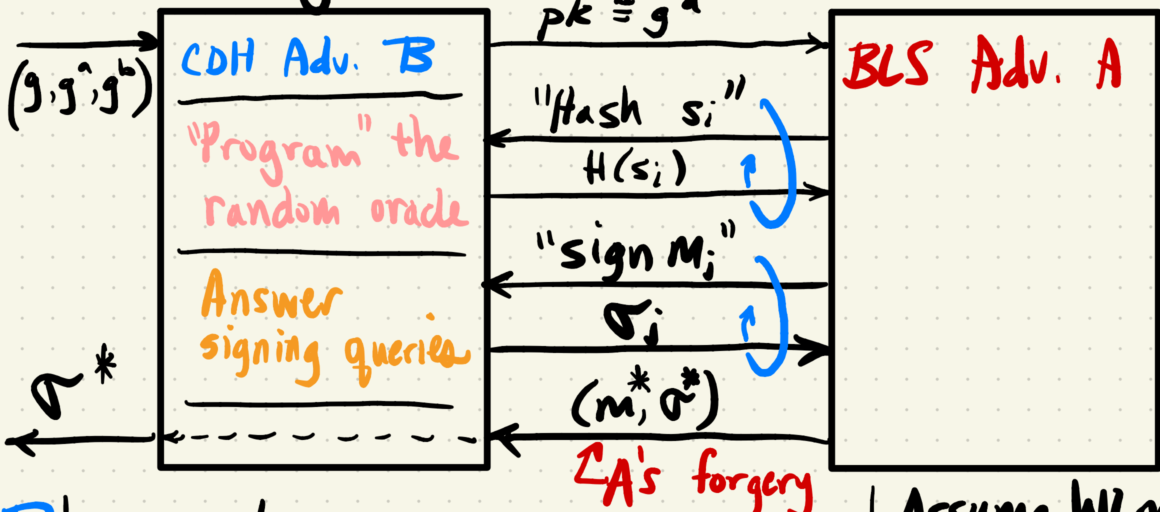
- Verify(pk, m, σ) $\to$ {True, False}

  return $e(pk, H(m)) \stackrel{?}{=} e(g, \sigma)$

Correctness:

$$
\begin{aligned}
e(pk, H(m)) &= e(g^{sk}, H(m)) && \text{def'n of } pk \\
&= e(g^{sk}, g^{\mu}) && \exists \mu : H(m) = g^{\mu} \\
&= e(g,g)^{sk \cdot \mu} && \text{Bilinearity} \\
&= e(g, g^{sk \cdot \mu}) && \text{Bilinearity} \\
&= e(g, H(m)^{sk}) && \text{def'n of } \mu \text{ (above)} \\
&= e(g, \sigma) && \text{def'n of sign}(\cdot)
\end{aligned}
$$

Security: CDH in $\mathbb{G}$ in ROM



$\mathcal{B}$'s procedure

1. send g^a to $\mathcal{A}$ as BLS pk
2. guess i^*, the RO query index on which $\mathcal{A}$ will query eventual forgery m^*
3. For $i \neq i^*$, let $b_i \xleftarrow{\$} \mathbb{Z}_q$ and set $H(s_i) = g^{b_i}$
   For $i = i^*$, send g^b
4. For $m_j = s_i$ for $i \neq i^*$ send $(g^a)^{b_i}$
   If $m_j = s_{i^*}$ abort
5. If $\mathcal{B}$ guessed i^* correctly,

Assume WLOG
- $\mathcal{A}$ queries RO on message m^*
- $\mathcal{A}$ does not repeat RO queries

CDH solution!

$$e(g^a, H(m^*)) = e(g^a, g^b) = e(g,g)^{ab} = e(g, g^{ab}) = e(g, \sigma^*)$$

$\mathcal{B}$ wins if it guesses i^* correctly

$\mathcal{A}$ makes poly(λ) queries $\Rightarrow$ CDH-ADV($\mathcal{B}$) $\geq \dfrac{\text{BLS-ADV}(\mathcal{A})}{\text{poly}(\lambda)}$

Aggregating BLS signatures

[BGLS'03, BDN'18]

https://eprint.iacr.org/2018/483

Idea: compress 2 (or more) signatures into one!

Let $H_q : \{0,1\}^* \to \mathbb{Z}_q$ be a random oracle.

- $\text{Aggregate}(pk_A, \sigma_A, pk_B, \sigma_B) \to \sigma_{AB}$:

  $t \leftarrow H_q(pk_A \| pk_B)$

  return $\sigma_A^t \cdot \sigma_B$

- $\text{Agg-Verify}(pk_A, pk_B, m, \sigma_{AB}) \to \{\text{True}, \text{False}\}$:

  $t \leftarrow H_q(pk_A \| pk_B)$

  $pk_{AB} \leftarrow pk_A^t \cdot pk_B$

  return $\text{Verify}(pk_{AB}, m, \sigma_{AB})$

Question: Why does this work?

How do we build $H: \{0,1\}^* \to E(\mathbb{F}_p)$?

In 2 steps:

(1) $H_p : \{0,1\}^* \longrightarrow \mathbb{F}_p$

(2) $M : \mathbb{F}_p \longrightarrow E(\mathbb{F}_p)$

Step (1): hashing to $\mathbb{F}_p$

Idea: use a hash function

$$H_{p'} : \{0,1\}^* \longrightarrow \{0,1\}^{\lambda + \log p}$$

extra λ bits ensures H_p's output is close to uniform $\in \mathbb{F}_p$

$H_p(m) \triangleq$

$h \leftarrow H_{p'}(m)$

return $\mathrm{Int}(h) \bmod p$

interpret h as an integer

Step 2: Mapping from $\mathbb{F}_p$ to $E(\mathbb{F}_p)$

$\Rightarrow$ Goal: construct a point (x,y) on $E(\mathbb{F}_p)$ from $t \in \mathbb{F}_p$

$\hookrightarrow E: y^2 = x^3 + Ax + B$

Attempt #1

$M_{h\&i}(t \in \mathbb{F}_p)$:

  $ySq \leftarrow t^3 + At + B$

  if ySq is square $\in \mathbb{F}_p$:

    return $(t, \sqrt{ySq} \in \mathbb{F}_p)$

  else:

    return $M_{h\&i}(t+1 \in \mathbb{F}_p)$

Problem:

Running time of $M_{h\&i}$ depends on t.

$\Rightarrow$ side channels ! e.g. WPA3 password leak [VR'20]

Number theory background: quadratic reciprocity

Def (Legendre symbol):

$$\left(\frac{a}{p}\right) \triangleq \begin{cases} 0 & \text{if } a \equiv 0 \bmod p \\ 1 & \text{if } a \text{ is square mod } p \\ -1 & \text{otherwise.} \end{cases}$$

Fact $\left(\frac{a \cdot b}{p}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{b}{p}\right)$

So, $\forall a, b \in \mathbb{F}_p$:

a sq, b nsg $\Rightarrow$ ab nsg	$1 \cdot -1 = -1$
a sq, b sq $\Rightarrow$ ab sq	$1 \cdot 1 = 1$
a nsq, b nsq $\Rightarrow$ ab sq	$-1 \cdot -1 = 1$

"is non-square" "is square"

Attempt #2 [SW'06, Ulas '07, BCIMRT'10]

$$E : y^2 = f(x) \triangleq x^3 + Ax + B$$

Idea: pick z s.t. $f(uz) = u^3 f(z)$ for u any non-square $\in \mathbb{F}_p$.

$\Rightarrow$ By quadratic reciprocity, if $f(z)$ is non-square, $f(uz) = u^3 f(z)$ is square,

$\Rightarrow$ either z or $u \cdot z$ must be an x-coordinate on $E(\mathbb{F}_p)$!

Solve for z w.r.t. u:

$$f(uz) = u^3 f(z)$$

$$u^3 z^3 + Auz + B = u^3(z^3 + Az + B)$$

$$\Rightarrow z = -\frac{B}{A}\left(1 + \frac{1}{u^2 + u}\right)$$

$$Z = -\frac{B}{A}\left(1 + \frac{1}{u^2+u}\right)$$ for u any non-square $\in \mathbb{F}_p$

But: we want to map from $\mathbb{F}_p$, not from non-squares!

Idea #2: fix a non-square β.

Then $\forall t \in \mathbb{F}_p$, βt^2 is non-square

by quadratic reciprocity

$M_{swu}(t \in \mathbb{F}_p)$:

$u \leftarrow \beta t^2$

$Z \leftarrow -\frac{B}{A}\left(1 + \frac{1}{u^2+u}\right)$

if $f(Z)$ is square $\in \mathbb{F}_p$:

return $(Z, \sqrt{f(Z)} \in \mathbb{F}_p)$

else:

return $(uZ, \sqrt{f(uZ)} \in \mathbb{F}_p)$

not too hard to eliminate side-channels in M_{swu}!

Putting it all together:

$\Rightarrow$ We can build a hash

$$H : \{0,1\}^* \to E(\mathbb{F}_p)$$

via $M_{swu}(H_p(m))$.

Issue: is this uniformly distributed?

$\Rightarrow$ No, but we can fix that [FFSTV'13]

$$H(m) \triangleq M_{swu}(H_p(0\|m)) \boxplus M_{swu}(H_p(1\|m))$$

H_p is a R.O. $\Rightarrow$ $H_p(0\|m)$ & $H_p(1\|m)$ are independent

5 Identity-based encryption

[Boneh & Franklin '01]

Took 17 years to solve!

Goal [Shamir '84]

Instead of needing to know someone's public key, encrypt to an arbitrary string

e.g. 2048-bit N for RSA

e.g. "rsw@jfet.org"

IBE uses an identity provider to generate & distribute keys.

⇒ Need to trust the provider!

→ This can make sense, say, on corporate networks.

⇒ How does this compare to Kerberos?

IBE Syntax

$$\mathrm{Setup}(1^\lambda) \to (ippk, ipsk)$$

global params

identity provider's secret key

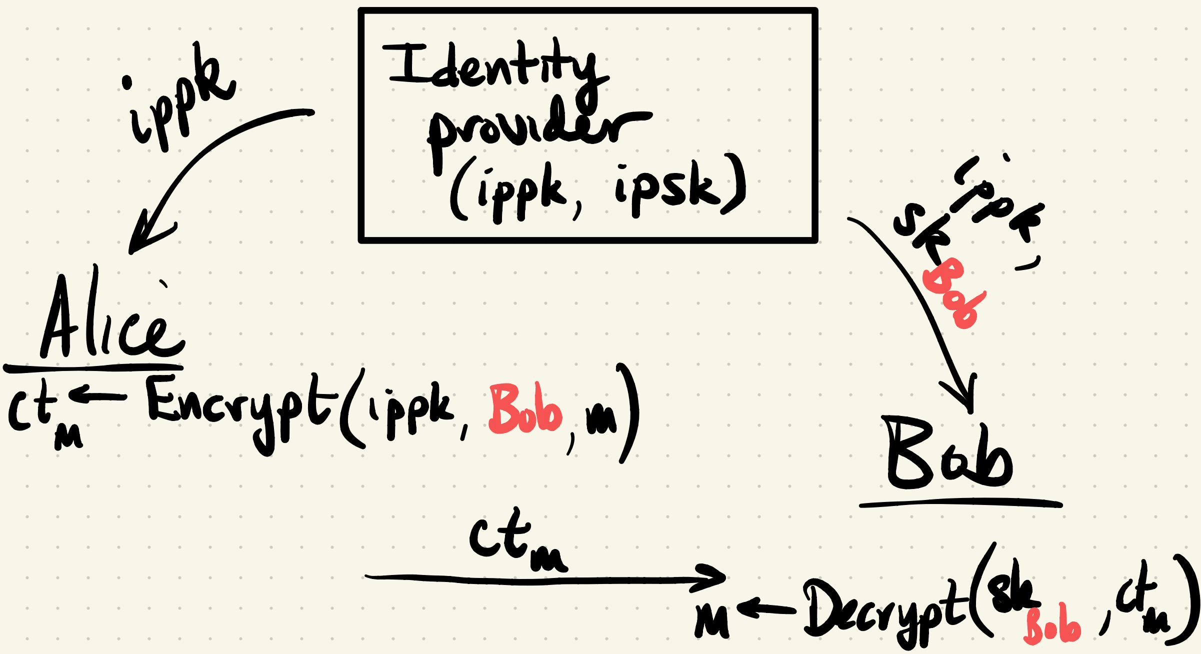
$$\mathrm{KeyGen}(ipsk, id) \to sk_{id}$$

provider generates key for id

$$\mathrm{Encrypt}(ippk, id, m) \to ct_m$$

encrypt m to id

$$\mathrm{Decrypt}(sk_{id}, ct_m) \to m$$



IBE from pairings

Fix $\mathbb{G}, H, e, \ldots$ as before

- $\text{Setup}(\cdot) \to (ippk, ipsk) \in \mathbb{G} \times \mathbb{Z}_q$:

  $sk \xleftarrow{\$} \mathbb{Z}_q$

  return (g^{sk}, sk)

- $\text{KeyGen}(ipsk, id) \to sk_{id} \in \mathbb{G}$:

  return $H(id)^{ipsk}$

$m \in \mathbb{G}_T$ e.g. a random key for hybrid enc.

- $\text{Encrypt}(ippk, id, m) \to ct_m \in \mathbb{G} \times \mathbb{G}_T$:

  $r \xleftarrow{\$} \mathbb{Z}_q$

  return $(g^r, m \cdot e(ippk^r, H(id)))$

- $\text{Decrypt}(sk_{id}, ct_m) \to m \in \mathbb{G}_T$:

  $(R, c) \leftarrow ct_m$

  $k \leftarrow e(R, sk_{id})$

  return $c \cdot k^{-1}$

$= e(g^r, H(id)^{ipsk})$
$= e(g^{ipsk \cdot r}, H(id))$
$= e(ippk^r, H(id))$

Security: Bilinear DDH, modeling H as a R.O.