CS 355 Lecture 13 : Pairings-based Cryptography

Last week: Elliptic curves

→ size of group elements
→ complexity of group operation

Goal: group $\mathbb{G}$ with better tradeoff between efficiency and hardness of DLog than $\mathbb{F}_p^*$

↳ non-generic attacks

Today: Pairing-based cryptography

$\Rightarrow$ exploiting additional structure of elliptic curve groups

Many applications:
- DLog attacks
- 3-party key exchange
- short signatures
- Identity-based encryption
- ...

$$\boxed{\begin{array}{l}\text{Logistics: HW 4 out today!} \\ \text{Due May } 24^{th}\end{array}}$$

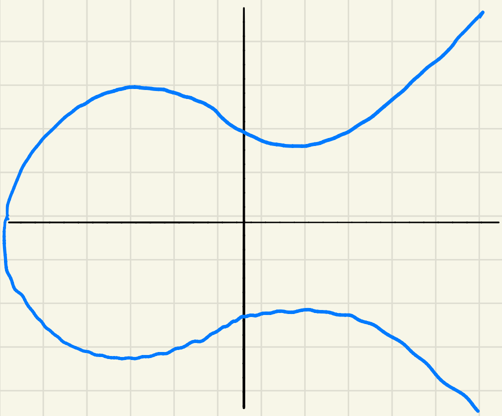
Brief recap on Elliptic curves

For an elliptic curve $E: y^2 = x^3 + Ax + B$

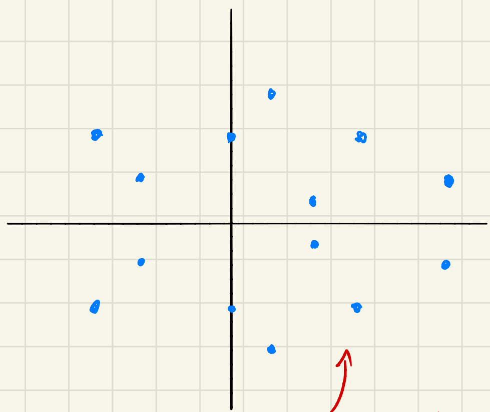
the points on E over $\mathbb{F}_p$ form a group $E(\mathbb{F}_p)$

of order $\# E(\mathbb{F}_p) \approx p$ (Hasse's theorem)

The curve E over $\mathbb{R}$:



$E(\mathbb{F}_p)$



integer solutions of E taken mod p

- Point representation

a point $P \in E(\mathbb{F}_p)$ is of the form $P=(x,y)$ where $x, y \in \mathbb{F}_p$

↳ we need $2\log(p)$ bits to represent P

↳ Point compression: Given x, the coordinate y is determined by E up to a sign ($y = \pm\sqrt{x^3 + Ax + B}$)

$$\boxed{\text{We can represent } P \text{ as } (x, \operatorname{sgn}(y)) \text{ using } \log(p) + 1 \text{ bits}}$$

	$\mathbb{F}_p^*$		$E(\mathbb{F}_p)$
element size	$\log(p)$	$\approx$	$\log(p)+1$
complexity of the group operation	1 multiplication	$\approx$	cte # of multiplications in $\mathbb{F}_p^*$
best Dlog algorithm	$2^{\tilde{O}(\sqrt[3]{\log p})}$	$\ll$	$O(\sqrt{p})$

- A change of notation

assume $\#E(\mathbb{F}_p) = q$

abstract cyclic group of order q

group	$E(\mathbb{F}_p)$	G
element	$P=(x,y)$	g^a
group operation	$P \boxplus Q$	$g^a \cdot g^b = g^{a+b}$

$\in \mathbb{Z}_q$

generator of G

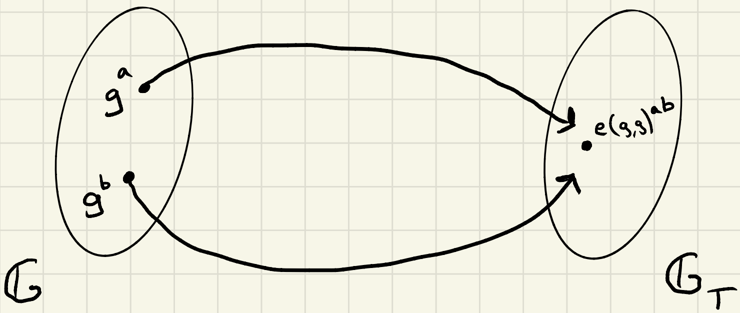
Pairings

we can also define asymmetric pairings $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$

cyclic groups of order q

Definition: A (symmetric) pairing $e: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a mapping with the following properties:

- Bilinearity: $\forall a, b \in \mathbb{Z}_q, \ g \in \mathbb{G}: e(g^a, g^b) = e(g,g)^{ab}$
- Non-degenerate: if g generates $\mathbb{G}$, then $e(g,g)$ generates $\mathbb{G}_T$
- Efficiency: the mapping e can be efficiently computed



Why non-degenerate: the mapping $e(g^a, g^b) = 1$ is bilinear

Why efficient: the CDH mapping $e(g^a, g^b) = g^{ab}$ is bilinear but usually assumed to be hard to compute

Q: If a pairing $e: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ exists, what can you say about the hardness of DDH in $\mathbb{G}$?

$\longrightarrow \quad (g^a, g^b, g^{ab}) \not\approx^c (g^a, g^b, g^r)$

Why pairings?

- Originally: attacks on discrete log over $E(\mathbb{F}_p)$

  For some elliptic curve groups $E(\mathbb{F}_p)$, there exists a bilinear map from $E(\mathbb{F}_p)$ to $\mathbb{G}_T$, where $\mathbb{G}_T$ is a subgroup of $\mathbb{F}_{p^\alpha}$ for a small constant α (e.g. $\alpha = 2$)

  [Menezes, Okamoto, Vanstone '93] DLog over $E(\mathbb{F}_p)$ $\leftarrow O(\sqrt{p})$ can be mapped to DLog over $\mathbb{F}_{p^\alpha}$ $\leftarrow 2^{\tilde{O}(\sqrt[3]{\alpha \log p})}$

  If α is small enough, mapping DLog to $\mathbb{F}_{p^\alpha}$ gives a faster attack both asymptotically and in practice.

- "Bug $\Rightarrow$ Feature": [Joux, '00], [Boneh, Franklin '01]

  $\hookrightarrow$ if p (or α) is large enough, security is preserved and we can exploit the additional structure of the pairing to build new schemes for which we know no constructions from non-pairing groups (e.g. $\mathbb{F}_p$)

Application 1: 3-party key-exchange [Joux, '00]

Recall classic Diffie-Hellman key exchange:

Alice $[a \xleftarrow{R} \mathbb{Z}_q]$ $\qquad\qquad$ Bob $[b \xleftarrow{R} \mathbb{Z}_q]$

$$\xrightarrow{\quad g^a \quad}$$

$$\xleftarrow{\quad g^b \quad}$$

$\downarrow$ $\qquad\qquad\qquad\qquad$ $\downarrow$

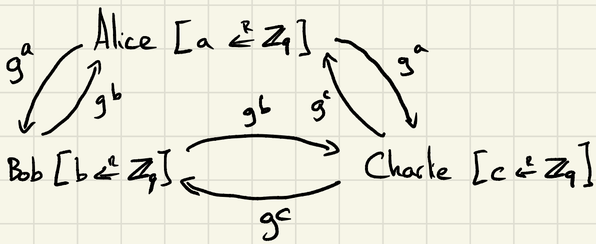
$(g^b)^a = g^{ab}$ $\qquad\qquad$ $(g^a)^b = g^{ab}$

Security:

$(g^a, g^b, g^{ab}) \overset{c}{\approx} (g^a, g^b, g^r)$ by DDH

Essentially relies on the group operation being "1-linear": it is easy to compute linear relations in the exponent but difficult to compute quadratic relations

What about 3 parties?



$\Longrightarrow$ Alice computes $e(g^b, g^c)^a$

Bob computes $e(g^a, g^c)^b$

Charlie computes $e(g^a, g^b)^c$

Shared key: $e(g,g)^{abc}$

Security: Bilinear DDH (BDDH) assumption

$$(g, g^a, g^b, g^c, e(g,g)^{abc}) \overset{c}{\approx} (g, g^a, g^b, g^c, e(g,g)^r)$$

Pairings make it easy to compute quadratic relations in the exponent, but computing cubic relations should be hard.

Open problems:

* N-party key exchange for $N > 3$.
  would require a multilinear map (or indistinguishability obfuscation)
  ↳ Some candidates but questionable security and far from practical
* 3-party key exchange from other assumptions (e.g. lattices)

Application 2: Short Signatures [Boneh, Lynn, Shacham '01]

Existing signature candidates: (128-bit level security)

Scheme	Group	Best attack	Group size	Signature	length
RSA	$\mathbb{Z}_N$	$2^{\tilde{O}(\sqrt[3]{\log N})}$	2048 bits	1 group element	2048 bits
ECDSA	$E(\mathbb{F}_p)$	$O(\sqrt{p})$	256 bits	2 group elements	512 bits
Schnorr	$E(\mathbb{F}_p)$	$O(\sqrt{p})$	256 bits	1 group element, 1 hash	384 bits
BLS	$E(\mathbb{F}_q)$	$O(\sqrt{q})$	256 bits	1 group element	256 bits

the field order q is not prime but of the form $q = 3^r$

the curve is chosen so that the pairing maps $E(\mathbb{F}_q)$ to a subgroup of $\mathbb{F}_{q^6}^*$ (i.e., $\alpha = 6$). For these concrete parameters, the generic Dlog attack in $E(\mathbb{F}_q)$ is estimated to be faster than the best non-generic Dlog attack in $\mathbb{F}_{q^6}^*$

using point compression

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

$\mathrm{Sign}(sk, m) \to \sigma$: $\quad \sigma = H(m)^a$ $\quad$ where $H: \{0,1\}^* \to \mathbb{G}$ is a hash function (modeled as a random oracle)

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

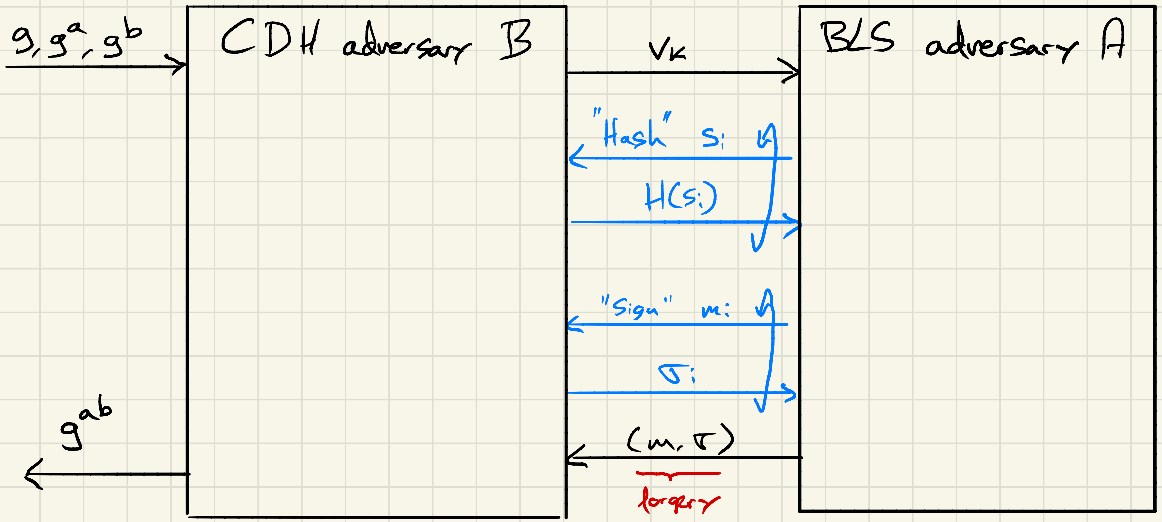
Correctness:

$$e(\sigma, g) = e(H(m)^a, g) \overset{H(m) = g^x \text{ for some } x \in \mathbb{Z}_q}{=} e(g^{xa}, g) \overset{\text{by bilinearity}}{=} e(g,g)^{xa}$$

$$\underset{\text{by bilinearity}}{=} e(g^x, g^a) = e(H(m), g^a)$$

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

Security proof



Challenge: give consistent responses to A's R.O. and signing queries while somehow embedding the CDH challenge into them

Assume: * A queries the R.O. for the message m for which it forges σ
* A makes no duplicate queries

these are without loss of generality

Adversary B

the secret key ($sk = a$) is unknown to B

- send $v_k = (g, g^a)$ to A
- Guess which of A's R.O. queries is for the forged message (index i^*)
- For the i^*-th R.O. query, respond with g^b
- For other R.O. queries on msg m_i, respond with g^{b_i} for $b_i \xleftarrow{\$} \mathbb{Z}_q$
- For a sign query on msg m_i, respond with $(g^a)^{b_i}$ (if A requests a signature on m_{i^*}, abort)
- If we guessed correctly and get a forgery (m_{i^*}, σ):

$$e(\sigma, g) = e(H(m_{i^*}), g^a) = e(g^b, g^a) = e(\underbrace{g^{ab}}_{\sigma}, g)$$

Since A can make at most $poly(\lambda)$ R.O. queries,

$$\text{CDH-ADV}[\mathcal{B}, \mathbb{G}] \geq \frac{1}{poly(\lambda)} \cdot \text{Sig-Adv}[\mathcal{A}, S_{BLS,\mathbb{G}}]$$

Application 3: Identity-based encryption (IBE) [Boneh, Franklin '01]

Goal: Instead of needing to know someone's RSA public key (2048 bits... hard to remember) to send them an encrypted message what if the public key could be an arbitrary string (e.g. email address, username, phone number, ...)

IBE [Shamir '84]: encrypt with respect to identies:

global public parameters

master secret key

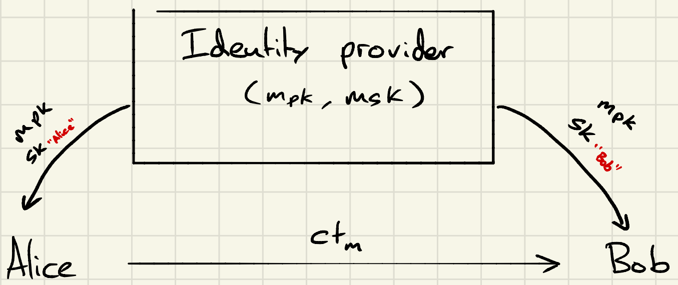
$\text{Setup}(1^\lambda) \rightarrow (mpk, msk)$

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

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

$\text{Decrypt}(sk_{id}, ct_m) \rightarrow m / \perp$ [decrypts m if ct_m is an encryption to id]

↳ challenge of IBE is to compress an exponential number of (public/secret) key pairs (one per identity) into a single master (public/private) key pair



$ct_m = \text{Encrypt}(mpk, \text{"Bob"}, m)$

$m = \text{Decrypt}(sk_{\text{"Bob"}}, ct_m)$

IBE was a major open problem solved by Boneh-Franklin in 2001 using pairings (and also concurrently by Cocks)

Very exciting recent result: IBE can be constructed from CDH or factoring! [Döttling, Garg '17] (but far from practical)

Boneh-Franklin IBE Scheme:

$\text{Setup}(1^\lambda) \to (\text{mpk}, \text{msk})$: $\quad s \xleftarrow{R} \mathbb{Z}_q$ $\quad$ mpk: $h = g^s$, msk: s

$\text{Encrypt}(\text{mpk}, \text{id}, m) \to ct_m$: $\quad r \xleftarrow{R} \mathbb{Z}_q$, $ct_m = (g^r, m \cdot e(h^r, H(\text{id})))$

How to decrypt?

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

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

Security follows from the Bilinear DDH assumption if $H: \{0,1\}^* \to \mathbb{G}$ is modeled as a random oracle.