Today: Pairing-based cryptography and applications

1. Definition

2. 3-way key agreement [Joux 2000]

3. Short Signature scheme [Boneh-Lynn-Shacham 2001]

4. Identity-based encryption scheme [Boneh-Franklin 2001]

Definition: Let $G$ and $G_T$ be two groups of prime order $q$. Let $g \in G$ be a generator; i.e., $G = \{g, g^2, \ldots, g^{q-1}, 1\}$. A pairing (or a bilinear map) is an efficiently computable bilinear function $e: G \times G \rightarrow G_T$ such that for every $a, b \in \mathbb{Z}_q$,

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

and $e(g, g) \neq 1$.

Corollary: $e(g, g)$ is a generator of $G_T$

This follows from the fact that a prime order group has only two subgroups the entire group and the trivial group consisting only of the identity. Since $e(g, g) \neq 1$ and since $G_T$ is a prime order group it must hold that the group that is generated by $e(g, g)$ is the entire group $G_T$. 

Claim: Let $G$ be a prime order group and let $e: G \times G \to G_T$ be a bilinear map. Then $e(g^a, g^b) = e(g^{ab}, g) = e(g, g^{ab}) = e(g, g)^{ab}$

Claim: Let $G$ be a prime order group and let $e: G \times G \to G_T$ be a bilinear map. Then the DDH assumption on $G$ is false.

Proof: Consider the following algorithm that given a triplet $g^a, g^b, g^c$ decides if $c = ab$ or if $c$ is randomly distributed in $Z_q$.

The algorithm checks if $e(g^a, g^b) = e(g, g^c)$. If this equality holds it outputs $c = ab$ and otherwise it predicts that $c$ is uniformly distributed in $Z_q$.

Claim: Let $G$ be a prime order group and let $e: G \times G \to G_T$ be a bilinear map. Then if the Discrete Log assumption is false in $G_T$ is must also be false in $G$.

Proof: Let $A$ be an algorithm that breaks the Discrete Log in $G_T$ Namely, $\Pr_{h_T \leftarrow G_T}[A(h_T) = a \text{ s.t. } e(g, g)^a = h]$ is non-negligible.

We construct an algorithm $B$ that has approximately the same runtime as $A$ and breaks the discrete log in $G$ with approximately the same probability as $A$ does.
Define $B(h) = A(e(g,h))$

It remains to note that if $h = g^a$ then $e(g,h) = e(g,g)^a$ and thus $B$ succeeds whenever $A$ succeeds.

**Why are groups with bilinear maps useful??**

1. We believe that the **CDH Assumption** holds in $G$. Namely, given $(g^a, g^b)$ for random $a, b \leftarrow Z_q$ it is hard to compute $g^{ab}$

2. We believe the **(decisional) bilinear Diffie-Hellman Assumption**: 

   $$(g^a, g^b, g^c, g^{abc}) \approx (g^a, g^b, g^c, g^u)$$

   where $a, b, c, u \leftarrow Z_q$

3. We know how to construct groups with bilinear maps based on elliptic curves, for which non-trivial algorithms are not known for breaking the above two assumptions, and thus we can use short keys.

4. These groups has many applications!
Application 1: 3-Way Key Agreement [Joux 2000]

This is a generalization of the Diffie-Hellman key agreement. Recall that the DH key agreement allows 2 parties to agree on a secret key non-interactively in the presence of a passive adversary that listens to the communication.

We will see how to extend this to 3 parties using bilinear maps:

Let $G$ be a group of prime order $q$ with a bilinear map

$$e: G \times G \rightarrow G_T$$

Consider 3 parties: Alice, Bob and Charlie.

Alice chooses a random $a \leftarrow Z_q$ and sends $g^a$

Bob chooses a random $b \leftarrow Z_q$ and sends $g^b$

Charlie chooses a random $c \leftarrow Z_q$ and sends $g^c$

The secret is $e(g, g)^{abc}$.

Alice computes the secret by computing $e(g^b, g^c)^a = e(g, g)^{abc}$. Bob and Charlie compute it analogously.
This scheme is strongly secure against passive attacks assuming
\[(g^a, g^b, g^c, g^{abc}) \approx (g^a, g^b, g^c, g^u)\]
Which is precisely the decisional bilinear DH assumption.

**Open problem:** Extend to more than 3 parties!

Can be done via an interactive protocol. Any function can be computed securely via an interactive protocol. This is known as secure multi-party computation (and is taught in 6.857).

1. Application 2: Short signature scheme [Boneh-Lynn-Shacham01]

In what follows we construct a signature scheme using groups with bilinear maps. The advantage of this scheme over previous schemes is that it produces extremely short signature schemes, consisting of only a **single group element**!

Moreover, since we use elliptic curve groups which do not have any non-trivial attacks (beyond the baby-step giant-step algorithm) we can take a relatively small security parameter.
Let $G, G_T$ be cyclic groups of prime order $q$.
Let $g \in G$ be a generator and let $e: G \times G \to G_T$ be a bilinear map.
Let $H: M \to G$ be a hash function modeled as a Random Oracle, where $M$ is the message space.

**Gen:** Sample a random $x \leftarrow Z_q$. Let $pk = u = g^x$ and $sk = x$.

**Sign**(sk, m): outputs $H(m)^{sk}$

**Ver**(pk, m, σ): outputs 1 if and only if $e(g, \sigma) = e(pk, H(m))$

**Theorem:** This signature scheme is secure (existentially unforgeable against adaptive chosen message attacks), assuming the CDH in $G$ and assuming $H$ is a Random Oracle.

**Proof Idea:** First note that this scheme is existentially unforgeable assuming the adversary does not see any signatures. This is the case since o.w., the fact that $H$ is a RO implies that the adversary given a random $r \leftarrow G$ and the public key $g^x$ can generate a signature $r^x$. This breaks the CDH assumption.
Next, we argue that the signature oracle is of no help to the adversary. This is the case, since when the adversary asks for a signature of a message \( m \in M \) he obtains \( r^x \) for \( r = H(m) \).

Since \( H \) is a RO this signature can be efficiently simulated by choosing \( \sigma = pk^u = g^{xu} \) and then “programming” the RO to satisfy \( H(m) = g^u \).

**Note:** This signature is extremely short since it consists of a single group element which consists of only 256 bits (since we don’t have non-trivial attacks on CDH in elliptic curves we can take small groups that consist of only \( 2^{256} \) elements.

**Application 3: Identity-Based Encryption** [Boneh-Franklin 2001]

In public key cryptography we assume that each party has a \( pk \).

**How do we know the other user’s \( pk \)?**

This is a big problem with no good solution.

The way we deal with this problem in practice is using **certification authorities** (CA) that authorize public keys, but this does not work very well. There are many CA’s. Which do we trust? How do they check the user’s \( pk \)?
Identity-based encryption (IBE):

Use “natural” public keys, such as the user’s email address.

The question is: How do we generate a corresponding \( sk \)?

This is precisely what IBE does.

An IBE assume a Trusted Third Party (TTP).

IBE Scheme:

TTP:

1. Choose a group \( G \) of prime order \( q \) that has a bilinear map \( e: G \times G \rightarrow G_T \), and choose a generator \( g \) of \( G \).
2. Choose 2 hash functions: \( H_1: names \rightarrow G \) and \( H_2: G_T \rightarrow M \), where \( M \) is the message space. Both \( H_1 \) and \( H_2 \) are modelled as Random Oracles.
3. Choose a random secret \( s \leftarrow Z_q \)
4. Publish \( (G, G_T, e, g, H_1, H_2) \) as public parameters along with a master public key \( mpk = g^s \).
**Goal:** Allow anyone to encrypt a msg to Alice given only her “name” and \( mpk \).

\[ Enc(pp, mpk, name, m) : \]

Let \( h_A = e(H_1(name), mpk) = e(H_1(name), g)^s \).

Choose a random \( r \leftarrow Z_q \) and output \((g^r, m \oplus H_2(h_A^r))\)

Similar to El-Gamal with \( pk_A = h_A \)

To decrypt Alice needs a corresponding \( sk_A \) which she gets from TTP:

\[ sk_A = H_1("Alice")^s \]

\[ Dec(pp, sk_A, (u, v)) : \]

Compute \( m = v \oplus H_2(e(sk_A, u)) \)

**Correctness:** Follows from \( e(sk_A, u) = h_A^r = e(H_1(name), g)^{sr} \)

**Security:** follows from the bilinear DH assumption.