Intro to Bilinear Maps

John Bethencourt
bethenco@cs.cmu.edu

Computer Sciences Department
Carnegie Mellon University
Outline

Introduction
   Definitions
   Commentary
   Real Instances

Problems and Assumptions
   Basics
   New Problems

History of Usage
   Early Usage
   Recent Usage

Using Bilinear Maps
   Intuition
   Examples of Pairing-Based Constructions

Conclusion
Motivation

Why bilinear maps?

- Bilinear maps are the tool of *pairing-based crypto*
  - Hot topic started with an identity based encryption scheme by Boneh and Franklin in 2001
  - Really useful in making new schemes, lots of low hanging fruit
  - Over 200 papers and counting as of March 2006

- What do they basically do?
  - Establish relationship between cryptographic groups
  - Make DDH easy in one of them in the process
  - Let you solve CDH “once”
Definition of a Bilinear Map

Let $G_1$, $G_2$, and $G_t$ be cyclic groups of the same order.

Definition

A bilinear map from $G_1 \times G_2$ to $G_t$ is a function $e : G_1 \times G_2 \rightarrow G_t$ such that for all $u \in G_1, v \in G_2, a, b \in \mathbb{Z},$

$$e(u^a, v^b) = e(u, v)^{ab} .$$

Bilinear maps are called pairings because they associate pairs of elements from $G_1$ and $G_2$ with elements in $G_t$. Note that this definition admits degenerate maps which map everything to the identity of $G_t$. 
Definition of an Admissible Bilinear Map

Let \( e : G_1 \times G_2 \rightarrow G_t \) be a bilinear map.
Let \( g_1 \) and \( g_2 \) be generators of \( G_1 \) and \( G_2 \), respectively.

**Definition**

The map \( e \) is an *admissible bilinear map* if \( e(g_1, g_2) \) generates \( G_t \) and \( e \) is efficiently computable.

These are the only bilinear maps we care about. Sometimes such a map is denoted \( \hat{e} \); we continue to use \( e \). Also, from now on we implicitly mean admissible bilinear map when we say bilinear map.
Relationships Between $G_1$, $G_2$, and $G_t$

- $G_1$, $G_2$, and $G_t$ are all isomorphic to one another since they have the same order and are cyclic
- They are different groups in the sense that we represent the elements and compute the operations differently
- Normally, however, $G_1 = G_2$ (in addition to being isomorphic)
  - From now on we assume this unless otherwise noted
  - Denote both by $G = G_1 = G_2$
- $G$ and $G_t$ may have either composite or prime order
  - Makes a difference in how they work / are used
  - Most often prime order
- If $G = G_t$ called a self-bilinear map
  - Very powerful
  - No known examples, open problem to make one
The Other Notation

- Sometimes $G$ is written additively
  - In this case $P, Q$ normal names for elements of $G$
  - Bilinear property expressed as $\forall P, Q \in G, \forall a, b \in \mathbb{Z},$
    \[
e(aP, bQ) = e(P, Q)^{ab}\]
- I prefer notation of both $G$ and $G_t$ written multiplicatively
- Will continue to use it
What Groups to Use?

- Typically $G$ is an elliptic curve (or subgroup thereof)
  - The elliptic curve defined by $y^2 = x^3 + 1$ over the finite field $F_p$ (simple example)
  - Supersingular curves
  - MNT curves
  - Choosing between supersingular curves and MNT curves has performance implications
- More generally, $G$ is typically an abelian variety over some field
  - Elliptic curves are abelian varieties of dimension 1
  - Other abelian varieties have had some consideration
- $G_t$ is normally a finite field
What Bilinear Maps to Use?

- (Modified) Weil pairing and Tate pairing are more or less only known examples
  - Very complicated math
  - Non-trivial to compute
  - No need to understand it to use them
- Weil and Tate pairings computed using Miller’s algorithm
  - Computationally expensive
  - Common to be very explicit about how many pairings are needed for operations in some scheme
  - Tate pairing normally somewhat faster than Weil
  - Making these faster still is current research
Decisional Diffie-Hellman

First thing to know about bilinear maps is their effect on the Decisional Diffie-Hellman (DDH) problem. Review definition:

Definition

Let $G$ be a group of order $q$ with generator $g$. The advantage of an probabilistic algorithm $A$ in solving the Decisional Diffie-Hellman problem in $G$ is

$$\text{Adv}_{A,G}^{\text{DDH}} = \left| \Pr[A(g, g^a, g^b, g^{ab}) = 1] - \Pr[A(g, g^a, g^b, g^z) = 1] \right|$$

where $a, b, z$ are drawn from the uniform distribution on $\mathbb{Z}_q$ and the probability is taken over the choices of $a, b, z$ and $A$’s coin flips.
...is Easy with a Bilinear Map!

- Basic property of bilinear map is making DDH easy in $G$
  - With bilinear map $e : G \times G \rightarrow G_t$, a polynomial time $A$ may gain advantage one
  - Given $g, g^a, g^b, g^c$, determine whether $c \equiv ab \mod q$ by just checking whether $e(g^a, g^b) = e(g, g^c)$
- However if the map is from distinct groups $G_1$ and $G_2$, DDH may still be hard in $G_1$ and / or $G_2$ (XDH assumption)
  - Believed to be the case with some MNT curves (and only those)
  - Only possible if there is no efficiently computable isomorphism between $G_1$ and $G_2$
  - A few schemes use this assumption
Computational Diffie-Hellman

- Note that Computational Diffie-Hellman (CDH) could still be hard in $G$
- That is, a bilinear map is not known to be useful for solving CDH
- A prime order group $G$ is called a gap Diffie-Hellman (GDH) group if DDH is easy in $G$ but CDH is hard
  - Definition is independent of presence of bilinear map
  - Bilinear maps may be viewed as an attempt to make GDH groups
Discrete Log

Next thing to know is the following fact about discrete logs with a bilinear map.

**Theorem**

*If there exists a bilinear map $e : G \times G \to G_t$, then the discrete log problem in $G$ is no harder than the discrete log problem in $G_t$.***

Also straightforward. Given $g \in G$ and $g^a \in G$, we can compute $e(g, g) \in G_t$ and $e(g, g^a) = e(g, g)^a \in G_t$. Then we can use a discrete log solver for $G_t$ to obtain $a$. This is called the MOV reduction.
Most Common New Problems

Some new problems have been defined and assumed hard in the new bilinear context.

**Bilinear Diffie-Hellman** Given $g, g^a, g^b, g^c$, compute $e(g, g)^{abc}$

(something like a “three-way” CDH but across the two groups)

**Decisional Bilinear Diffie-Hellman** Distinguish $g, g^a, g^b, g^c, e(g, g)^{abc}$ from $g, g^a, g^b, g^c, e(g, g)^z$

**$k$-Bilinear Diffie-Hellman Inversion** Given $g, g^y, g^{y^2}, \ldots, g^{y^k}$, compute $e(g, g)^{1/y}$

**$k$-Decisional Bilinear Diffie-Hellman Inversion** Distinguish $g, g^y, g^{y^2}, \ldots, g^{y^k}, e(g, g)^{1/y}$ from $g, g^y, g^{y^2}, \ldots, g^{y^k}, e(g, g)^z$
More New Problems

If we have a map from distinct groups $G_1$ and $G_2$, then we can make the “Co” assumptions.

**Computational Co-Diffie-Hellman** Given $g_1, g_1^a \in G_1$ and $g_2, g_2^b \in G_2$, compute $g_2^{ab}$

**Decisional Co-Diffie-Hellman** Distinguish $g_1, g_1^a \in G_1$ and $g_2, g_2^b, g_2^{ab} \in G_2$ from $g_1, g_1^a \in G_1$ and $g_2, g_2^b, g_2^z \in G_2$

**Co-Bilinear Diffie-Hellman** Given $g_1, g_1^a, g_1^b \in G_1$ and $g_2 \in G_2$, compute $e(g_1, g_2)^{ab}$

**Decisional Co-Bilinear Diffie-Hellman** Distinguish $g_1, g_1^a, g_1^b, g_2, e(g_1, g_2)^{ab}$ from $g_1, g_1^a, g_1^b, g_2, e(g_1, g_2)^z$
Introduction of Pairings to Cryptography

- **1993**: used to break crypto
  - Weil and Tate pairings first used in cryptographic context in efforts to break ECC
  - Idea was to reduce DLP in elliptic curves to DLP in finite fields (MOV reduction)
- **2000**: first “good” use
  - Joux’s protocol for one-round 3-party Diffie-Hellman
  - Previous multi-round schemes for 3-party Diffie-Hellman existed, but showed how bilinear maps could be useful
- **2001**: Boneh and Franklin’s identity-based encryption scheme
  - First practical IBE scheme
  - Showed bilinear maps allowed dramatic new constructions, very influential
2001 to Present (2006)

- Many schemes for new primitives and improved schemes for existing primitives based on bilinear maps
- IBE related stuff
  - Hierarchical identity based encryption (HIBE)
  - Dual-HIBE
  - IBE, HIBE without random oracles
  - IBE with threshold decryption
  - Identity based signatures (also ID-based blind signatures, ring signatures, hierarchical ID-based signatures)
  - Identity based chameleon hashes
  - Identity based “signcryption”
2001 to Present (2006)

- Signatures
  - Short signatures (also without random oracles)
  - Blind signatures
  - Multi-signatures
  - Aggregate signatures
  - Verifiable encrypted signatures
  - Ring signatures
  - Threshold signatures
  - Unique signatures without random oracles
  - Authentication-tree based signatures without random oracles
2001 to Present (2006)

- Other stuff
  - BGN cryptosystem, which is sort of doubly homomorphic
  - Threshold decryption
  - $k$-party key agreement
  - Identification scheme

- Much more
Intuition

- Informally, why are bilinear maps so useful?
- Lets you “cheat” and solve a computational Diffie-Hellman problem
  - But only once!
- After that, you are stuck in the group $G_t$
- Seems to be just the right level of power
  - Enough to be useful in making a construction work
  - But not enough to make it insecure
- Now several examples of pairing-based constructions to hopefully illustrate this
Joux’s 3-Party Diffie-Hellman

This is a simple protocol; you could almost come up with it yourself on the spot.
Let $G$ be a group with prime order $q$, $e : G \times G \rightarrow G_t$ be a bilinear map, and $g$ be a generator of $G$. Let $\hat{g} = e(g, g) \in G_t$.

Protocol

1. Alice picks $a \xleftarrow{R} \mathbb{Z}_q$, Bob picks $b \xleftarrow{R} \mathbb{Z}_q$, and Carol picks $c \xleftarrow{R} \mathbb{Z}_q$.
2. Alice, Bob, and Carol broadcast $g^a$, $g^b$, and $g^c$ respectively.
3. Alice computes $e(g^b, g^c)^a = \hat{g}^{abc}$, Bob computes $e(g^c, g^a)^b = \hat{g}^{abc}$, and Carol computes $e(g^a, g^b)^c = \hat{g}^{abc}$.
Intuition

- From Alice’s perspective, map lets you “cheat” to get $\hat{g}^{bc}$ from $g^b$ and $g^c$.
- Then regular exponentiation gets you the rest of the way to $\hat{g}^{abc}$.
- Note that you can’t use $e$ to get $\hat{g}^{abc}$ from $g^a, g^b, g^c$.
  - $e(g^a, e(g^b, g^c)) = e(g^a, \hat{g}^{bc}) \neq \hat{g}^{abc}$ ($\hat{g}^{bc}$ not in $G$).
- Only one cheat allowed!
Boneh and Franklin’s IBE Scheme

Let $G$ be a group with prime order $q$, $e : G \times G \rightarrow G_t$ be a bilinear map, and $g$ be a generator of $G$. Let $\hat{g} = e(g, g) \in G_t$. Let $h_1 : \{0, 1\}^* \rightarrow G$ and $h_2 : G_t \rightarrow \{0, 1\}^*$ be hash functions. These are all public parameters.

**Setup**

PKG picks $s \leftarrow \mathbb{Z}_q$. Then $g^s$ is the public key of PKG.
### Boneh and Franklin’s IBE Scheme

#### Encryption

If Alice wants to send a message $m$ to Bob, she picks $r \leftarrow \mathbb{Z}_q$ then computes the following.

$$\text{Encrypt} \left( g, g^s, \text{“Bob”}, m \right) = \left( g^r, m \oplus h_2(e(h_1(\text{“Bob”}), g^s)^r) \right) = \left( g^r, m \oplus h_2(e(h_1(\text{“Bob”}), g)^{rs}) \right)$$

#### Making a Private Key

PKG may compute the private key of Bob as follows.

$$\text{MakeKey} \left( s, \text{“Bob”} \right) = h_1(\text{“Bob”})^s$$
Boneh and Franklin’s IBE Scheme

Decryption

Given an encrypted message
\((u, v) = (g^r, m \oplus h_2(e(h_1(“Bob”), g)^{rs}))\) and a private key
\(w = h_1(“Bob”)^s\), Bob may decrypt as follows.

\[
\text{Decrypt} (u, v, w) = v \oplus h_2(e(w, u)) \\
= m \oplus h_2(e(h_1(“Bob”), g)^{rs}) \\
\quad \oplus h_2(e(h_1(“Bob”)^s, g^r)) \\
= m \oplus h_2(e(h_1(“Bob”), g)^{rs}) \\
\quad \oplus h_2(e(h_1(“Bob”)^s, g^r)) \\
= m
\]
Boneh and Franklin’s IBE Scheme

- How to understand this?
- Let $t$ be the discrete log of $h_1(“Bob”) \text{ base } g$
- We don’t know what it is, but it is well defined
- Now the situation is like 3-party Diffie-Hellman
  - Alice has public $g^r$, private $r$
  - PKG has public $g^s$, private $s$
  - Bob has public $g^t$, unknown (!) $t$
- $e(h_1(“Bob”), g)^{rs} = e(g^t, g)^{rs} = \hat{g}^{rst}$ is like session key for encryption
Boneh and Franklin’s IBE Scheme

- Alice and PKG could compute $\hat{g}^{rst}$ just like in Joux’s scheme
- But what about Bob?
  - PKG helps him over previously authenticated, secure channel
  - PKG computes $(g^t)^s = g^{st}$ and sends it to Bob
  - Bob can now compute $e(g^{st}, g^r) = \hat{g}^{rst}$
- The point is that Bob gets $g^{st}$ rather than $\hat{g}^{st}$
  - With $g^{st}$, still one cheat left
  - If it was $\hat{g}^{st}$ (which anyone can compute), couldn’t apply $e$ anymore
Questions?

- Best reference is a website called the *The Pairing-Based Crypto Lounge*
- Huge list of papers relating to bilinear maps
- To get the URL just Google for it