Are pairings really dead? ☠

Chloe Martindale

Technische Universiteit Eindhoven

Ei/Ψ seminar, 17th June 2019

Why care about pairings?

- Building block of privacy protocols
- Allows for anonymous authentication.

Why care about pairings?

- Building block of privacy protocols
- Allows for anonymous authentication.

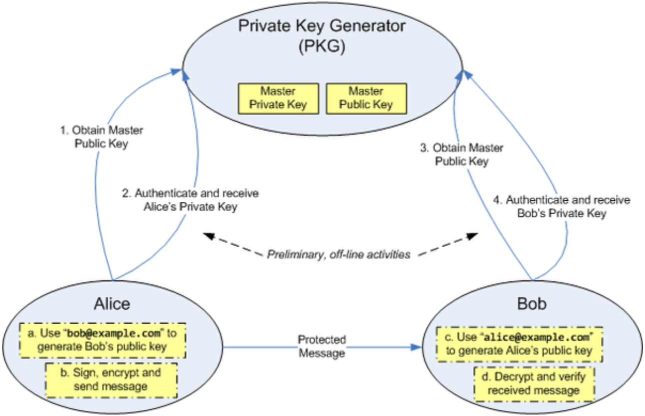


Image: Identity-based encryption; stolen shamelessly from Wikipedia

Why care about pairings?

- Building block of privacy protocols
- Allows for anonymous authentication. How?

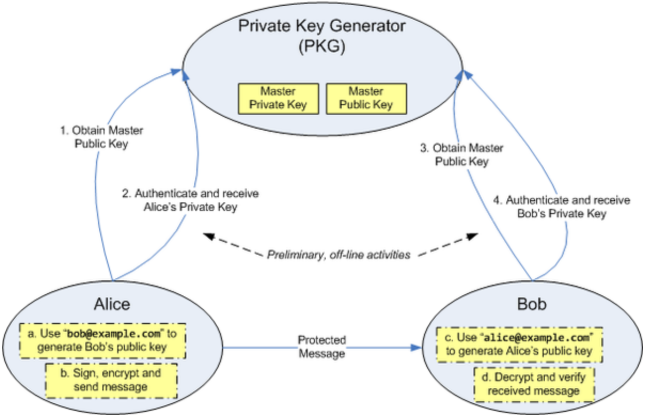


Image: Identity-based encryption; stolen shamelessly from Wikipedia

What is a pairing?

Pairings are maps of groups.

What is a pairing?

Pairings are maps of groups.

- A group $\mathbb{G}$ comes with a group operation $*$.

What is a pairing?

Pairings are maps of groups.

- A group $\mathbb{G}$ comes with a group operation $*$.
  - eg. $\mathbb{G} = \mathbb{Z}/p\mathbb{Z} - \{0\}$ with $*$ given by multiplication.

What is a pairing?

Pairings are maps of groups.

- A group $\mathbb{G}$ comes with a group operation $*$.
  - eg. $\mathbb{G} = \mathbb{Z}/p\mathbb{Z} - \{0\}$ with $*$ given by multiplication.

- If $g \in \mathbb{G}$ and $n \in \mathbb{Z}_{\geq 0}$, write $g^n = \underbrace{g * \cdots * g}_{n \text{ times}}$.

What is a pairing?

Pairings are maps of groups.

- A group $\mathbb{G}$ comes with a group operation $*$.
  - eg. $\mathbb{G} = \mathbb{Z}/p\mathbb{Z} - \{0\}$ with $*$ given by multiplication.
- If $g \in \mathbb{G}$ and $n \in \mathbb{Z}_{\geq 0}$, write $g^n = \underbrace{g * \cdots * g}_{n \text{ times}}$.
  - eg. $(3 \pmod 5)^2 = 3 \cdot 3 \pmod 5$.

What is a pairing?

Pairings are bilinear maps of groups.

What is a pairing?

Pairings are bilinear maps of groups. In particular:

$$\begin{array}{ccc} \mathbb{G}_1 \times \mathbb{G}_2 & \to & \mathbb{G}_3 \\ (g, h) & \mapsto & P(g, h) \end{array}$$

What is a pairing?

Pairings are bilinear maps of groups. In particular:

$$
\begin{array}{rcl}
\mathbb{G}_1 \times \mathbb{G}_2 & \to & \mathbb{G}_3 \\
(g, h) & \mapsto & P(g, h) \\
(g^a, h^b) & \mapsto & P(g, h)^{ab}
\end{array}
$$

What is a pairing?

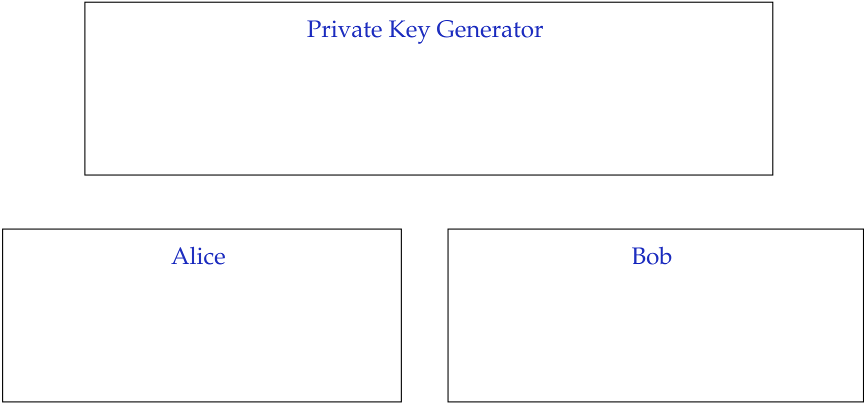
Pairings are bilinear maps of groups. In particular:

$$\begin{aligned} \mathbb{G}_1 \times \mathbb{G}_2 &\rightarrow \mathbb{G}_3 \\ (g, h) &\mapsto P(g, h) \\ (g^a, h^b) &\mapsto P(g, h)^{ab} \end{aligned}$$

Why is this useful?

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.



Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice

Bob

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$; Public $\mathsf{pub} \in \mathbb{G}_2$;
Master secret key $\mathsf{sk\text{-}m} \in \mathbb{Z}$; Master public key $\mathsf{pk\text{-}m} = \mathsf{pub}^{\mathsf{sk\text{-}m}} \in \mathbb{G}_2$.

Alice

Secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$

Bob

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\textsf{id-a} \in \mathbb{G}_1$; Public $\textsf{pub} \in \mathbb{G}_2$;
Master secret key $\textsf{sk-m} \in \mathbb{Z}$; Master public key $\textsf{pk-m} = \textsf{pub}^{\textsf{sk-m}} \in \mathbb{G}_2$.
Computes $\textsf{sk-b} = \textsf{id-a}^{\textsf{sk-m}}$...

Alice

Secret identity $\textsf{id-a} \in \mathbb{G}_1$
Choose random $r \in \mathbb{Z}$...
Compute $\textsf{enc-id-a} = P(\textsf{id-a}, \textsf{pk-m})$...

Bob

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\textsf{id-a} \in \mathbb{G}_1$; Public $\textsf{pub} \in \mathbb{G}_2$;

Master secret key $\textsf{sk-m} \in \mathbb{Z}$; Master public key $\textsf{pk-m} = \textsf{pub}^{\textsf{sk-m}} \in \mathbb{G}_2$.

Computes $\textsf{sk-b} = \textsf{id-a}^{\textsf{sk-m}}$...

Sends $\textsf{sk-b}$ to Bob

Alice

Secret identity $\textsf{id-a} \in \mathbb{G}_1$

Choose random $r \in \mathbb{Z}$...

Compute $\textsf{enc-id-a} = P(\textsf{id-a}, \textsf{pk-m})$...

Sends $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ to Bob

Bob

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$; Public $\mathsf{pub} \in \mathbb{G}_2$;
Master secret key $\mathsf{sk\text{-}m} \in \mathbb{Z}$; Master public key $\mathsf{pk\text{-}m} = \mathsf{pub}^{\mathsf{sk\text{-}m}} \in \mathbb{G}_2$.
Computes $\mathsf{sk\text{-}b} = \mathsf{id\text{-}a}^{\mathsf{sk\text{-}m}}$...
Sends $\mathsf{sk\text{-}b}$ to Bob

Alice

Secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$
Choose random $r \in \mathbb{Z}$...
Compute $\mathsf{enc\text{-}id\text{-}a} = P(\mathsf{id\text{-}a}, \mathsf{pk\text{-}m})$...
Sends $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ to Bob

Bob

Receives secret key $\mathsf{sk\text{-}b} \in \mathbb{G}_1$ from PKG
Receives $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ from Alice

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\textsf{id-a} \in \mathbb{G}_1$; Public $\textsf{pub} \in \mathbb{G}_2$;
Master secret key $\textsf{sk-m} \in \mathbb{Z}$; Master public key $\textsf{pk-m} = \textsf{pub}^{\textsf{sk-m}} \in \mathbb{G}_2$.
Computes $\textsf{sk-b} = \textsf{id-a}^{\textsf{sk-m}}$...
Sends $\textsf{sk-b}$ to Bob

Alice

Secret identity $\textsf{id-a} \in \mathbb{G}_1$
Choose random $r \in \mathbb{Z}$...
Compute $\textsf{enc-id-a} = P(\textsf{id-a}, \textsf{pk-m})$...
Sends $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ to Bob

Bob

Receives secret key $\textsf{sk-b} \in \mathbb{G}_1$ from PKG
Receives $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ from Alice
Compute $\textsf{ver} = P(\textsf{sk-b}, \textsf{pub}^r)$
Verify that $\textsf{ver} = \textsf{enc-id-a}^r$

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\textsf{id-a} \in \mathbb{G}_1$; Public $\textsf{pub} \in \mathbb{G}_2$;

Master secret key $\textsf{sk-m} \in \mathbb{Z}$; Master public key $\textsf{pk-m} = \textsf{pub}^{\textsf{sk-m}} \in \mathbb{G}_2$.

Computes $\textsf{sk-b} = \textsf{id-a}^{\textsf{sk-m}}$...

Sends $\textsf{sk-b}$ to Bob

Alice

Secret identity $\textsf{id-a} \in \mathbb{G}_1$

Choose random $r \in \mathbb{Z}$...

Compute $\textsf{enc-id-a} = P(\textsf{id-a}, \textsf{pk-m})$...

Sends $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ to Bob

Bob

Receives secret key $\textsf{sk-b} \in \mathbb{G}_1$ from PKG

Receives $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ from Alice

Compute $\textsf{ver} = P(\textsf{sk-b}, \textsf{pub}^r)$

Verify that $\textsf{ver} = \textsf{enc-id-a}^r$ †

† Bilinearity:

$P(\textsf{sk-b}, \textsf{pub}^r) = P(\textsf{id-a}^{\textsf{sk-m}}, \textsf{pub}^r)$

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\textsf{id-a} \in \mathbb{G}_1$; Public $\textsf{pub} \in \mathbb{G}_2$;
Master secret key $\textsf{sk-m} \in \mathbb{Z}$; Master public key $\textsf{pk-m} = \textsf{pub}^{\textsf{sk-m}} \in \mathbb{G}_2$.
Computes $\textsf{sk-b} = \textsf{id-a}^{\textsf{sk-m}}$...
Sends $\textsf{sk-b}$ to Bob

Alice

Secret identity $\textsf{id-a} \in \mathbb{G}_1$
Choose random $r \in \mathbb{Z}$...
Compute $\textsf{enc-id-a} = P(\textsf{id-a}, \textsf{pk-m})$...
Sends $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ to Bob

Bob

Receives secret key $\textsf{sk-b} \in \mathbb{G}_1$ from PKG
Receives $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ from Alice
Compute $\textsf{ver} = P(\textsf{sk-b}, \textsf{pub}^r)$
Verify that $\textsf{ver} = \textsf{enc-id-a}^r$ †

† Bilinearity:

$$P(\textsf{sk-b}, \textsf{pub}^r) = P(\textsf{id-a}^{\textsf{sk-m}}, \textsf{pub}^r) = P(\textsf{id-a}, \textsf{pub})^{\textsf{sk-m} \cdot r}$$

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\textsf{id-a} \in \mathbb{G}_1$; Public $\textsf{pub} \in \mathbb{G}_2$;

Master secret key $\textsf{sk-m} \in \mathbb{Z}$; Master public key $\textsf{pk-m} = \textsf{pub}^{\textsf{sk-m}} \in \mathbb{G}_2$.

Computes $\textsf{sk-b} = \textsf{id-a}^{\textsf{sk-m}}$...

Sends $\textsf{sk-b}$ to Bob

Alice

Secret identity $\textsf{id-a} \in \mathbb{G}_1$

Choose random $r \in \mathbb{Z}$...

Compute $\textsf{enc-id-a} = P(\textsf{id-a}, \textsf{pk-m})$...

Sends $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ to Bob

Bob

Receives secret key $\textsf{sk-b} \in \mathbb{G}_1$ from PKG

Receives $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ from Alice

Compute $\textsf{ver} = P(\textsf{sk-b}, \textsf{pub}^r)$

Verify that $\textsf{ver} = \textsf{enc-id-a}^r$ †

† Bilinearity:

$$P(\textsf{sk-b}, \textsf{pub}^r) = P(\textsf{id-a}^{\textsf{sk-m}}, \textsf{pub}^r) = P(\textsf{id-a}, \textsf{pub})^{\textsf{sk-m}\cdot r} = P(\textsf{id-a}, \textsf{pub}^{\textsf{sk-m}})^r$$

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\textsf{id-a} \in \mathbb{G}_1$; Public $\textsf{pub} \in \mathbb{G}_2$;
Master secret key $\textsf{sk-m} \in \mathbb{Z}$; Master public key $\textsf{pk-m} = \textsf{pub}^{\textsf{sk-m}} \in \mathbb{G}_2$.
Computes $\textsf{sk-b} = \textsf{id-a}^{\textsf{sk-m}}$...
Sends $\textsf{sk-b}$ to Bob

Alice

Secret identity $\textsf{id-a} \in \mathbb{G}_1$
Choose random $r \in \mathbb{Z}$...
Compute $\textsf{enc-id-a} = P(\textsf{id-a}, \textsf{pk-m})$...
Sends $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ to Bob

Bob

Receives secret key $\textsf{sk-b} \in \mathbb{G}_1$ from PKG
Receives $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ from Alice
Compute $\textsf{ver} = P(\textsf{sk-b}, \textsf{pub}^r)$
Verify that $\textsf{ver} = \textsf{enc-id-a}^r$ †

† Bilinearity:

$$P(\textsf{sk-b}, \textsf{pub}^r) = P(\textsf{id-a}^{\textsf{sk-m}}, \textsf{pub}^r) = P(\textsf{id-a}, \textsf{pub})^{\textsf{sk-m}\cdot r} = P(\textsf{id-a}, \textsf{pub}^{\textsf{sk-m}})^r = P(\textsf{id-a}, \textsf{pk-m})^r.$$

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).
- Hard discrete logarithms problems in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).
- Hard discrete logarithms problems in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.
  - Bilinearity of $P \rightsquigarrow$ complexity of DLP in each of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ is the fastest algorithm for solving DLP in any of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_3$.

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).
- Hard discrete logarithms problems in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.
  - Bilinearity of $P \rightsquigarrow$ complexity of DLP in each of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ is the fastest algorithm for solving DLP in any of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_3$.
- An explicit pairing formula.

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).
- Hard discrete logarithms problems in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.
  - Bilinearity of $P \rightsquigarrow$ complexity of DLP in each of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ is the fastest algorithm for solving DLP in any of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_3$.
- An explicit pairing formula.
  - Example: the Weil pairing with $\mathbb{G}_1$ and $\mathbb{G}_2$ as elliptic curve groups and $\mathbb{G}_3$ as a finite field group.

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).
- Hard discrete logarithms problems in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.
  - Bilinearity of $P \rightsquigarrow$ complexity of DLP in each of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ is the fastest algorithm for solving DLP in any of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_3$.
- An explicit pairing formula.
  - Example: the Weil pairing with $\mathbb{G}_1$ and $\mathbb{G}_2$ as elliptic curve groups and $\mathbb{G}_3$ as a finite field group.
- Fast pairing computation.

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).
- Hard discrete logarithms problems in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.
  - Bilinearity of $P \rightsquigarrow$ complexity of DLP in each of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ is the fastest algorithm for solving DLP in any of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_3$.
- An explicit pairing formula.
  - Example: the Weil pairing with $\mathbb{G}_1$ and $\mathbb{G}_2$ as elliptic curve groups and $\mathbb{G}_3$ as a finite field group.
- Fast pairing computation.
  - Instances of the Weil pairing can be efficiently computed with Miller's algorithm.

How hard is the discrete logarithm problem?

$g \in \mathbb{G}$ (any): $\underbrace{g * \cdots * g}_{n \text{ times}} \longrightarrow$ **DLP solver** $\longrightarrow n$

Complexity: depends on $\mathbb{G}$

How hard is the discrete logarithm problem?

$g \in \mathbb{G}$ (any): $\underbrace{g * \cdots * g}_{n \text{ times}} \longrightarrow$ **DLP solver** $\longrightarrow n$

Complexity: depends on $\mathbb{G}$

$g \in \mathbb{F}_{p^k}^*$: $\underbrace{g * \cdots * g}_{n \text{ times}} \longrightarrow$ **Index calculus+** $\longrightarrow n$

Complexity:
if p large, non-special, and k small, $L_{p^k}(1/2, c)$
for most pairing instances, p is special, giving $L_{p^k}(1/3, c)$

How hard is the discrete logarithm problem?

$g \in \mathbb{G}$ (any): $\underbrace{g * \cdots * g}_{n \text{ times}} \longrightarrow$ **DLP solver** $\longrightarrow n$

Complexity: depends on $\mathbb{G}$

$g \in \mathbb{F}_{p^k}^*$: $\underbrace{g * \cdots * g}_{n \text{ times}} \longrightarrow$ **Index calculus+** $\longrightarrow n$

Complexity:
if p large, non-special, and k small, $L_{p^k}(1/2, c)$
for most pairing instances, p is special, giving $L_{p^k}(1/3, c)$

$g \in E(\mathbb{F}_{p^k})$: $\underbrace{g * \cdots * g}_{n \text{ times}} \longrightarrow$ **Pollard-ρ** $\longrightarrow n$

Complexity:
if p large, non-special, and k small, for most E,
$O(\sqrt{r})$, where r is the largest prime dividing $\#E(\mathbb{F}_{p^k})$

Balancing pairings for efficiency

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

Balancing pairings for efficiency

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- 128-bit security: 2 favourite choices (called BN and BLS) with complexity DLP in $\mathbb{G}_1 \approx$ complexity of DLP in $\mathbb{G}_3$.

Balancing pairings for efficiency

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- 128-bit security: 2 favourite choices (called BN and BLS) with complexity DLP in $\mathbb{G}_1 \approx$ complexity of DLP in $\mathbb{G}_3$.
- Most common choice implemented in practise: BN

Balancing pairings for efficiency

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- 128-bit security: 2 favourite choices (called BN and BLS) with complexity DLP in $\mathbb{G}_1 \approx$ complexity of DLP in $\mathbb{G}_3$.
- Most common choice implemented in practise: BN
- Balancing all three groups impractical.

Balancing pairings for efficiency

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- 128-bit security: 2 favourite choices (called BN and BLS) with complexity DLP in $\mathbb{G}_1 \approx$ complexity of DLP in $\mathbb{G}_3$.
- Most common choice implemented in practise: BN
- Balancing all three groups impractical.

For these two choices:

Balancing pairings for efficiency

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- 128-bit security: 2 favourite choices (called BN and BLS) with complexity DLP in $\mathbb{G}_1 \approx$ complexity of DLP in $\mathbb{G}_3$.
- Most common choice implemented in practise: BN
- Balancing all three groups impractical.

For these two choices:

- Complexity of DLP in $\mathbb{G}_1$ is $O(2^{128})$.

Balancing pairings for efficiency

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- 128-bit security: 2 favourite choices (called BN and BLS) with complexity DLP in $\mathbb{G}_1 \approx$ complexity of DLP in $\mathbb{G}_3$.
- Most common choice implemented in practise: BN
- Balancing all three groups impractical.

For these two choices:

- Complexity of DLP in $\mathbb{G}_1$ is $O(2^{128})$.
- Complexity of DLP in $\mathbb{G}_3$ is $O(2^{103})$ (BN) and $O(2^{126})$ (BLS) respectively.

Balancing pairings for efficiency

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- 128-bit security: 2 favourite choices (called BN and BLS) with complexity DLP in $\mathbb{G}_1 \approx$ complexity of DLP in $\mathbb{G}_3$.
- Most common choice implemented in practise: BN
- Balancing all three groups impractical.

For these two choices:

- Complexity of DLP in $\mathbb{G}_1$ is $O(2^{128})$.
- Complexity of DLP in $\mathbb{G}_3$ is $O(2^{103})$ (BN) and $O(2^{126})$ (BLS) respectively.
- ... wait what?

Rebalancing pairings for efficiency

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

Rebalancing pairings for efficiency

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- 2016: new improvements/refinements to the attack methods. See eg. [BD17] for an overview.

Rebalancing pairings for efficiency

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- 2016: new improvements/refinements to the attack methods. See eg. [BD17] for an overview.
- Worst-case asymptotic complexity went from $L_{p^k}[1/3, 1.923]$ to $L_{p^k}[1/3, 1.526]$.

Rebalancing pairings for efficiency

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- 2016: new improvements/refinements to the attack methods. See eg. [BD17] for an overview.
- Worst-case asymptotic complexity went from $L_{p^k}[1/3, 1.923]$ to $L_{p^k}[1/3, 1.526]$.
- With new understanding, different parameters will rule them all.

Rebalancing pairings for efficiency

3 concrete approaches so far:

- Just increase the parameters for BN and BLS until they are secure [BD16].

Rebalancing pairings for efficiency

3 concrete approaches so far:

- Just increase the parameters for BN and BLS until they are secure [BD16].
- Guillevic, Masson, and Thomé [GMT19]:
  - Construct pairings with a different method, where attacks don't apply (Cocks-Pinch).

Rebalancing pairings for efficiency

3 concrete approaches so far:

- Just increase the parameters for BN and BLS until they are secure [BD16].
- Guillevic, Masson, and Thomé [GMT19]:
  - Construct pairings with a different method, where attacks don't apply (Cocks-Pinch).
  - Pro: safe against further improvements to known attack methods.

Rebalancing pairings for efficiency

3 concrete approaches so far:

- Just increase the parameters for BN and BLS until they are secure [BD16].
- Guillevic, Masson, and Thomé [GMT19]:
  - Construct pairings with a different method, where attacks don't apply (Cocks-Pinch).
  - Pro: safe against further improvements to known attack methods.
  - Con: not as fast (less choices for parameters).

Rebalancing pairings for efficiency

3 concrete approaches so far:

- Just increase the parameters for BN and BLS until they are secure [BD16].
- Guillevic, Masson, and Thomé [GMT19]:
  - Construct pairings with a different method, where attacks don't apply (Cocks-Pinch).
  - Pro: safe against further improvements to known attack methods.
  - Con: not as fast (less choices for parameters).
- Fotiadis and me [FM19]:

Rebalancing pairings for efficiency

3 concrete approaches so far:

- Just increase the parameters for BN and BLS until they are secure [BD16].
- Guillevic, Masson, and Thomé [GMT19]:
  - Construct pairings with a different method, where attacks don't apply (Cocks-Pinch).
  - Pro: safe against further improvements to known attack methods.
  - Con: not as fast (less choices for parameters).
- Fotiadis and me [FM19]:
  - Take many families constructed with previous favourite method (Brezing-Weng).

Rebalancing pairings for efficiency

3 concrete approaches so far:

- Just increase the parameters for BN and BLS until they are secure [BD16].
- Guillevic, Masson, and Thomé [GMT19]:
  - Construct pairings with a different method, where attacks don't apply (Cocks-Pinch).
  - Pro: safe against further improvements to known attack methods.
  - Con: not as fast (less choices for parameters).
- Fotiadis and me [FM19]:
  - Take many families constructed with previous favourite method (Brezing-Weng).
  - Search for the family for which the attack is least effective.

Rebalancing pairings for efficiency

3 concrete approaches so far:

- Just increase the parameters for BN and BLS until they are secure [BD16].
- Guillevic, Masson, and Thomé [GMT19]:
  - Construct pairings with a different method, where attacks don't apply (Cocks-Pinch).
  - Pro: safe against further improvements to known attack methods.
  - Con: not as fast (less choices for parameters).
- Fotiadis and me [FM19]:
  - Take many families constructed with previous favourite method (Brezing-Weng).
  - Search for the family for which the attack is least effective.
  - Find a family member for which the attack has no effect.

Rebalancing pairings for efficiency

3 concrete approaches so far:

- Just increase the parameters for BN and BLS until they are secure [BD16].
- Guillevic, Masson, and Thomé [GMT19]:
  - Construct pairings with a different method, where attacks don't apply (Cocks-Pinch).
  - Pro: safe against further improvements to known attack methods.
  - Con: not as fast (less choices for parameters).
- Fotiadis and me [FM19]:
  - Take many families constructed with previous favourite method (Brezing-Weng).
  - Search for the family for which the attack is least effective.
  - Find a family member for which the attack has no effect.
  - Pros: Most efficient results, can use pre-attack optimization tricks.

Rebalancing pairings for efficiency

3 concrete approaches so far:

- Just increase the parameters for BN and BLS until they are secure [BD16].
- Guillevic, Masson, and Thomé [GMT19]:
  - Construct pairings with a different method, where attacks don't apply (Cocks-Pinch).
  - Pro: safe against further improvements to known attack methods.
  - Con: not as fast (less choices for parameters).
- Fotiadis and me [FM19]:
  - Take many families constructed with previous favourite method (Brezing-Weng).
  - Search for the family for which the attack is least effective.
  - Find a family member for which the attack has no effect.
  - Pros: Most efficient results, can use pre-attack optimization tricks.
  - Con: If new improvements to known attacks are found, ☠.

More candidates

There are many more choices!

More candidates

There are many more choices!
Barbulescu, El Mrabet, and Ghammam [BEG19] recently computed a large database of choices for $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ with the new security requirements in mind.

More candidates

There are many more choices!

Barbulescu, El Mrabet, and Ghammam [BEG19] recently computed a large database of choices for $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ with the new security requirements in mind.

Why is this not a ‘concrete approach’?

More candidates

There are many more choices!
Barbulescu, El Mrabet, and Ghammam [BEG19] recently computed a large database of choices for $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ with the new security requirements in mind.
Why is this not a 'concrete approach'?

- Concrete security level not yet calculated.
- Concrete timings not yet integrated.
- May be a faster candidate, but currently unknown!

So where are we at?

The computation of a pairing like those above can be boiled down to multiplications in $\mathbb{F}_p$, where $\mathbb{G}_3 = \mathbb{F}_{p^k}^*$.

$\mathbf{m}$ = one $\mathbb{F}_p$-multiplication.

So where are we at?

The computation of a pairing like those above can be boiled down to multiplications in $\mathbb{F}_p$, where $\mathbb{G}_3 = \mathbb{F}_{p^k}^*$.
$\mathbf{m}$ = one $\mathbb{F}_p$-multiplication.

Pairing choice	$\log(p)$	Pairing cost	Clock cycles
BN	462	17871$\mathbf{m}$	2966586
$k = 6$ [GMT]	672	8472$\mathbf{m}$	2660208
KSS	339	25926$\mathbf{m}$	2566674
$k = 8$ [GMT]	544	11636$\mathbf{m}$	2443560
BLS	461	13878$\mathbf{m}$	2303748
Family 17a [FM]	398	16189$\mathbf{m}$	2088381
Family 17b [FM]	407	16172$\mathbf{m}$	2086188

Table: Choices for 128-bit security

The number of clock cycles is based on a generic Montgomery-schoolbook algorithm for multiplication mod p on a 64-bit processor.

So where are we at?

- The fastest 128-bit-secure example so far is about $\times 2$ as slow as the fastest (now non-secure) example that was previously being used in practise (BN).

So where are we at?

- The fastest 128-bit-secure example so far is about $\times 2$ as slow as the fastest (now non-secure) example that was previously being used in practise (BN).
- Further optimizations can improve the situation:

So where are we at?

- The fastest 128-bit-secure example so far is about $\times 2$ as slow as the fastest (now non-secure) example that was previously being used in practise (BN).
- Further optimizations can improve the situation:
  - Optimization of finite field multiplication for the specific modulus

So where are we at?

- The fastest 128-bit-secure example so far is about $\times 2$ as slow as the fastest (now non-secure) example that was previously being used in practise (BN).
- Further optimizations can improve the situation:
  - Optimization of finite field multiplication for the specific modulus
  - High-level parallelization for eg. Cortex M4 chips

So where are we at?

- The fastest 128-bit-secure example so far is about $\times 2$ as slow as the fastest (now non-secure) example that was previously being used in practise (BN).
- Further optimizations can improve the situation:
  - Optimization of finite field multiplication for the specific modulus
  - High-level parallelization for eg. Cortex M4 chips
  - Hardware optimizations

So where are we at?

- The fastest 128-bit-secure example so far is about $\times 2$ as slow as the fastest (now non-secure) example that was previously being used in practise (BN).
- Further optimizations can improve the situation:
  - Optimization of finite field multiplication for the specific modulus
  - High-level parallelization for eg. Cortex M4 chips
  - Hardware optimizations

Thank you!

References

BD17 Barbulescu and Duquesne: Updating key size estimations for pairings. http://eprint.iacr.org/2017/334.

BEG19 Barbulescu, El Mrabet, and Ghammam: A taxonomy of pairings, their security, their complexity. https://eprint.iacr.org/2019/485.

FM19 Fotiadis and Martindale: Optimal TNFS-secure pairings on elliptic curves with composite embedding degree. https://eprint.iacr.org/2019/555

GMT19 Guillevic, Masson, and Thomé: Cocks-pinch curves of embedding degrees five to eight and optimal ate pairing computation. https://eprint.iacr.org/2019/431.