Implementing Pairing-Based Protocols

Alfred Menezes

University of Waterloo

September 4 2008

Abstract

In recent years, bilinear pairings have been used to design ingenious protocols for various tasks including identity-based encryption, homomorphic encryption, group signatures, and non-interactive zero-knowledge proof systems. Researchers have also made many discoveries that have dramatically improved the efficiency of these protocols. This tutorial will cover efficiency and security issues when implementing pairing-based protocols. For concreteness, the presentation will focus on the software implementation of pairing-based signature schemes due to Boneh-Lynn-Shacham (BLS) and Waters, and their aggregation signature scheme counterparts, using Barreto-Naehrig (BN) elliptic curves.

Outline

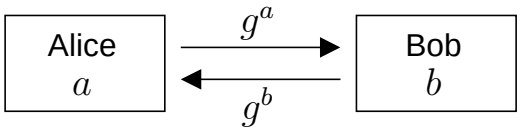
1. Symmetric Pairings

Discrete Logarithm and Diffie-Hellman Problems

- Let $\mathbb{G} = \langle g \rangle$ be a multiplicatively-written group of prime order n.
- The *discrete logarithm problem* (DLP) in $\mathbb{G}$ is:
  Given n, g, and $h \in \langle g \rangle$, find the integer $a \in [0, n-1]$ such that $h = g^a$.
- The *Diffie-Hellman problem* (DHP) in $\mathbb{G}$ is:
  Given n, g, g^a and g^b, find g^{ab}.
- If DLP can be efficiently solved, then so can DHP.
  That is, *DHP* $\leq$ *DLP*.
- Hence DHP is no harder than DLP.
- It is generally assumed (and has been proven in some cases) that DLP is no harder than DHP,
  i.e., DLP $\leq$ DHP.

Diffie-Hellman Key Agreement Scheme

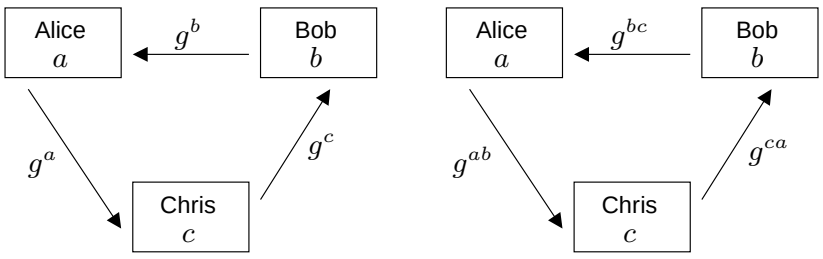
$\mathbb{G} = \langle g \rangle$ is a group of prime order n.



The shared secret is $K = g^{ab}$.

Secure against *passive attacks* if DHP in $\mathbb{G}$ is hard.

Three-Party Two-Round Diffie-Hellman



Shared secret is $K = g^{abc}$.

Security against *passive* attacks:
Given g, g^a, g^b, g^c, g^{ab}, g^{ca}, g^{bc}, find g^{abc}.

Question: Is there a one-round protocol for this task?

Symmetric Bilinear Pairings

- Let n be a prime number.
- Let $\mathbb{G} = \langle g \rangle$ be a group of order n with identity 1.
- Let $\mathbb{G}_T$ be a group of order n with identity 1.

Definition: A *symmetric bilinear pairing* [*Type 1 pairing*] on $(\mathbb{G}, \mathbb{G}_T)$ is a function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that:

1. [*Bilinearity*] For all $u, v, w \in \mathbb{G}$:
$$e(u \cdot v, w) = e(u, w) \cdot e(v, w)$$
$$e(u, v \cdot w) = e(u, v) \cdot e(u, w).$$
2. [*Non-degeneracy*] $e(g, g) \neq 1$.
3. [*Computability*] e can be efficiently computed.

Note: $e(u^a, v^b) = e(u, v)^{ab} = e(u^b, v^a)$ for all $u, v \in \mathbb{G}$, $a, b \in \mathbb{Z}$.
Note: $e(u, v) = e(v, u)$ for all $u, v \in \mathbb{G}$.

The DLP in $\mathbb{G}$ and $\mathbb{G}_T$

The discrete logarithm problem (DLP) in $\mathbb{G}$ is no harder than the DLP in $\mathbb{G}_T$.

Justification:

- *DLP*$_{\mathbb{G}}$: Given $g, g^a \in \mathbb{G}$, find a.
- Note that $e(g, g^a) = e(g, g)^a$.
  Let $\alpha = e(g, g)$, $\beta = e(g, g^a)$. Then $a = \log_\alpha \beta$.
- Thus instances of the DLP in $\mathbb{G}$ can be efficiently reduced to instances of the DLP in $\mathbb{G}_T$.
  That is, *DLP*$_{\mathbb{G}} \leq$ *DLP*$_{\mathbb{G}_T}$.

Bilinear Diffie-Hellman Problem (BDHP)

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a symmetric pairing.
BDHP: Given g, g^a, g^b, g^c, compute $e(g, g)^{abc}$.

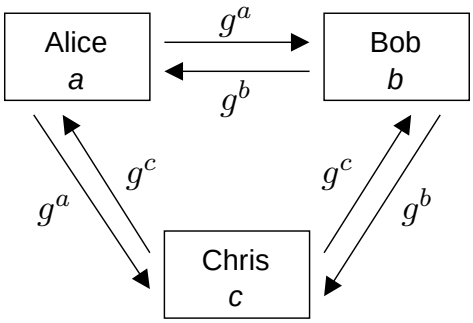
1. *BDHP* $\leq$ *DHP*$_{\mathbb{G}}$.
   - Compute g^{ab}.
   - Compute $e(g^{ab}, g^c) = e(g, g)^{abc}$.
2. *BDHP* $\leq$ *DHP*$_{\mathbb{G}_T}$.
   - Compute $\alpha = e(g, g)$.
   - Compute $\alpha^{ab} = e(g^a, g^b)$.
   - Compute $\alpha^c = e(g, g^c)$.
   - Then compute α^{abc}.

Nothing much else is known about the hardness of BDHP.

Three-Party One-Round Diffie-Hellman

[Joux; 2000]

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a symmetric pairing.



Shared secret is $K = e(g^b, g^c)^a = e(g, g)^{abc}$.

Secure against *passive* attacks if BDHP is hard.

2. BLS Signature Scheme

Short Signatures

[Boneh, Shacham, Lynn; 2001]

Let $\mathbb{G} = \langle g \rangle$ be a group of prime order n.

In ElGamal-type signature schemes, the signature is generally comprised of two integers modulo n.

Question: Are there discrete log signature schemes where signatures are comprised of only one integer modulo n or one group element?

BLS-1 Signature Scheme

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a symmetric pairing.

1. *Key generation*. Alice does:
   Private key: $x \in_R [1, n-1]$; Public key: $X = g^x$.
2. *Signature generation*. To sign M, Alice does:
   Compute $h = H(M)$, where $H : \{0,1\}^* \to \mathbb{G}$.
   Compute $\sigma = h^x$.
   The signature on M is σ.
3. *Signature verification*. To verify (M, σ), Bob does:
   Compute $h = H(M)$.
   Accept iff $e(\sigma, g) = e(h, X)$.
   (That is, (g, X, h, σ) is a valid DH-quadruple.)

Correctness: $e(\sigma, g) = e(h^x, g) = e(h, g^x) = e(h, X)$.

Security of Signature Schemes

Chosen-message attack: The attacker has access to a signing oracle which it can use to obtain Alice's signatures on any messages of its choosing.

Existential forger: The attacker forges Alice's signature on a single message – any message that it did not give to its signing oracle.

Definition: A signature scheme is said to be *secure* if it is existentially unforgeable by a computationally bounded adversary who launches a chosen-message attack.

BLS-1 Security

Theorem: If DHP in $\mathbb{G}$ is hard and H is a random function, then the BLS-1 signature scheme is secure.

Security argument. Given $X \in \mathbb{G}$:

1. The signing oracle is useless: It gives σ such that $e(\sigma, g) = e(h, X)$ for random $h \in \mathbb{G}$. However, the attacker can generate such (h, σ) itself by selecting random integers y and computing $h = g^y$ and $\sigma = X^y$.
2. Since H is random, the attacker may as well select M first. Its task then is to computing $\sigma = h^x$ given h and X. This is precisely DHP in $\mathbb{G}$.

3. BGLS Aggregate Signature Scheme

Aggregate Signatures

[Boneh, Gentry, Lynn, Shacham; 2003]

Desirable property: Given signatures $\sigma_1, \sigma_2, \ldots, \sigma_k$ on messages $M_1, M_2, \ldots, M_k$ generated by k entities $A_1, A_2 \ldots A_k$, anyone can compute one signature σ which can convince a verifier of the authenticity of $M_1, M_2, \ldots, M_k$.

Application: Reduce the size of certificate chains.

Application: Secure BGP protocol.

Note: Aggregate signature schemes are different from multisignature schemes where k entities sign the same message M to produce one signature.

BGLS-1 Aggregate Signature Scheme

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a Type 1 pairing.

1. *Key generation*. User A_i has BLS private key x_i and public key $X_i = g^{x_i}$.
2. *Signature generation*. User A_i generates its BLS signature $\sigma_i = h_i^{x_i}$ on M_i, where $h_i = H(M_i)$.
3. *Aggregation*. The aggregated signature is $\sigma = \prod \sigma_i$.
4. *Aggregate verification*. Given the public keys X_i, the messages M_i, and σ, a verifier does:
   Verify that the M_i are pairwise distinct.
   Compute $h_i = H(M_i)$.
   Accept iff $e(\sigma, g) = \prod e(h_i, X_i)$.

Correctness: $e(\sigma, g) = e(\prod \sigma_i, g) = \prod e(\sigma_i, g) = \prod e(h_i, X_i)$.

Security of Aggregate Signature Schemes

Chosen-message attack: The attacker is given a public key X_1, and a signing oracle w.r.t. X_1.

Attacker's goal: The attacker's task is to produce $k - 1$ public keys $X_2, X_3, \ldots, X_k$; k pairwise distinct messages M_1, M_2, $\ldots M_k$; and an aggregate signature σ such that $e(\sigma, g) = \prod e(h_i, X_i)$, where $h_i = H(M_i)$. Of course, the attacker cannot query the signing oracle with M_1.

Definition: An aggregate signature scheme is said to be *secure* if a computationally bounded attacker cannot achieve the goal described above.

BGLS-1 Security

Theorem: If DHP in $\mathbb{G}$ is hard and H is a random function, then the BGLS-1 signature scheme is secure.

Security argument (for $k = 2$). Given $X_1 \in \mathbb{G}$:

1. The signing oracle is useless.
2. Since H is random, the attacker may as well select M_1 and M_2 first (with $M_1 \neq M_2$, so $h_1 \neq h_2$). Its task then is to find X_2 and σ such that $e(\sigma, g) = e(h_1, X_1) \cdot e(h_2, X_2)$. Call this problem P.

   Then DHP $\leq$ P: Given $X_1, h_1 \in \mathbb{G}$, select random y_2 and compute $h_2 = g^{y_2}$. Use the P-oracle to compute X_2 and σ such that $\sigma = h_1^{x_1} h_2^{x_2} = h_1^{x_1} X_2^{y_2}$. Then compute $h_1^{x_1} = \sigma / X_2^{y_2}$.

4. Asymmetric Pairings

Asymmetric Bilinear Pairing

- Let n be a prime number.
- Let $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, $\mathbb{G}_T$ be groups of order n (with $\mathbb{G}_1 \neq \mathbb{G}_2$).
- Note: Elements of $\mathbb{G}_1$ are "shorter" than elements of $\mathbb{G}_2$.

Definition An *asymmetric bilinear pairing* on $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ such that;

1. [*Bilinearity*] For all $u_1, u_2 \in \mathbb{G}_1$, $v_1, v_2 \in \mathbb{G}_2$:
$$e(u_1 \cdot u_2, v_1) = e(u_1, v_1) \cdot e(u_2, v_1)$$
$$e(u_1, v_1 \cdot v_2) = e(u_1, v_1) \cdot e(u_1, v_2).$$
2. [*Non-degeneracy*] $e(g_1, g_2) \neq 1$.
3. [*Computability*] e can be efficiently computed.

Note: $e(u^a, v^b) = e(u, v)^{ab}$ for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}$.

BLS-2 Signature Scheme

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be an asymmetric pairing.

1. *Key generation*. Alice does:
   Private key: $x \in_R [1, n-1]$; Public key: $X = g_2^x \in \mathbb{G}_2$.
2. *Signature generation*. To sign M, Alice does:
   Compute $h = H(M)$, where $H : \{0,1\}^* \rightarrow \mathbb{G}_1$.
   Compute $\sigma = h^x \in \mathbb{G}_1$.
   The signature on M is σ.
3. *Signature verification*. To verify (M, σ), Bob does:
   Compute $h = H(M)$.
   Accept iff $e(\sigma, g_2) = e(h, X)$.

Correctness: $e(\sigma, g_2) = e(h^x, g_2) = e(h, g_2^x) = e(h, X)$.

co-DHP

A *necessary* condition for the security of the BLS-2 signature scheme is that *co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$ is hard: Given g_2^x and $h \in \mathbb{G}_1$, compute h^x.

This condition is, in general, *not sufficient*.

Example: Let $\mathbb{G}_1 = (\mathbb{Z}_n, +)$, and let $\mathbb{G}_2$ be the order-n subgroup of $\mathbb{Z}_p^*$. Define $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_2$ by $e(x, y) = y^x$.

- e is bilinear.
- The co-DHP in $(\mathbb{G}_1, \mathbb{G}_2)$ must be hard, because otherwise the DLP in $\mathbb{G}_2$ is easy.
- However, the BLS-2 scheme is insecure – given a single signed message (M, σ), the private key can easily be recovered (since $\sigma = hx \bmod n$ where $h = H(M)$).

Type 2 and 3 Asymmetric Pairings

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an asymmetric pairing.

If an *efficiently computable isomorphism* $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, $\psi : g_2 \mapsto g_1$, is known, then e is called a *Type 2 pairing*.

If no such isomorphism ψ is known, then e is called a *Type 3 pairing*.

BLS-2: BLS with a Type 2 asymmetric pairing

BLS-3: BLS with a Type 3 asymmetric pairing

BLS-2 Security

Theorem: If co-DHP in $(\mathbb{G}_1, \mathbb{G}_2)$ is hard and H is a random function, then the BLS-2 signature scheme is secure.

Security argument. Given $X \in \mathbb{G}_2$:

1. The signing oracle is useless: It gives $\sigma \in \mathbb{G}_1$ such that $e(\sigma, g_2) = e(h, X)$ for random $h \in \mathbb{G}_1$. However, the attacker can generate such (h, σ) itself by selecting random integers y and computing $h = g_1^y$ and $\sigma = \psi(X)^y$.
2. The attacker's problem is reduced to computing $\sigma = h^x$ given $h \in \mathbb{G}_1$ and $X \in \mathbb{G}_2$. This is precisely co-DHP in $(\mathbb{G}_1, \mathbb{G}_2)$.

BLS-3 Security

*co-DHP** in $(\mathbb{G}_1, \mathbb{G}_2)$: Given g_1^x, g_2^x, $h \in \mathbb{G}_1$, compute h^x.

Theorem: If co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$ is hard and H is a random function, then the BLS-3 signature scheme is secure.

Security argument. Given $X \in \mathbb{G}_2$ and $\psi(X)$: [$\psi(g_2^z) = g_1^z$]

1. The signing oracle is useless: It gives $\sigma \in \mathbb{G}_1$ such that $e(\sigma, g_2) = e(h, X)$ for random $h \in \mathbb{G}_1$. However, the attacker can generate such (h, σ) itself by selecting random integers y and computing $h = g_1^y$ and $\sigma = \psi(X)^y$.
2. The attacker's problem is reduced to computing $\sigma = h^x$ given $h \in \mathbb{G}_1$, $X \in \mathbb{G}_2$ and $\psi(X)$. This is precisely co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$.

BGLS-2

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a Type 2 pairing.

1. *Key generation*. User A_i has BLS-2 private key x_i and public key $X_i = g_2^{x_i} \in \mathbb{G}_2$.
2. *Signature generation*. User A_i generates its BLS-2 signature $\sigma_i = h_i^{x_i} \in \mathbb{G}_1$ on M_i, where $h_i = H(M_i)$.
3. *Aggregation*. The aggregated signature is $\sigma = \prod \sigma_i$.
4. *Aggregate verification*. Given the public keys X_i, the messages M_i, and σ, a verifier does:
   Verify that the M_i are pairwise distinct.
   Compute $h_i = H(M_i)$.
   Accept iff $e(\sigma, g_2) = \prod e(h_i, X_i)$.

Correctness: $e(\sigma, g_2) = e(\prod \sigma_i, g_2) = \prod e(\sigma_i, g_2) = \prod e(h_i, X_i)$.

BGLS-2 Security

Theorem: If co-DHP in $(\mathbb{G}_1, \mathbb{G}_2)$ is hard and H is a random function, then the BGLS-2 signature scheme is secure.

Security argument (for $k = 2$). Given $X_1 \in \mathbb{G}_2$:

1. The signing oracle is useless.
2. Since H is random, the attacker may as well select M_1 and M_2 first (with $M_1 \neq M_2$, so $h_1 \neq h_2$). Its task then is to find $X_2 \in \mathbb{G}_2$ and $\sigma \in \mathbb{G}_1$ such that $e(\sigma, g_2) = e(h_1, X_1) \cdot e(h_2, X_2)$. Call this problem P.
   Then co-DHP $\leq$ P: Given $X_1 \in \mathbb{G}_2$, $h_1 \in \mathbb{G}$, select random y_2 and compute $h_2 = g_1^{y_2}$. Use the P-oracle to compute $X_2 \in \mathbb{G}_2$ and $\sigma \in \mathbb{G}_1$ such that $\sigma = h_1^{x_1} h_2^{x_2} = h_1^{x_1} \psi(X_2)^{y_2}$. Then compute $h_1^{x_1} = \sigma / \psi(X_2)^{y_2}$.

BGLS-3 Security

BGLS-3: A_i's public key is ($X_i = g_2^{x_i}$, $W_i = g_1^{x_i} = \psi(X_i)$).

Theorem: If co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$ is hard and H is a random function, then the BGLS-3 signature scheme is secure.

Security argument (for $k = 2$). Given $X_1 \in \mathbb{G}_2, W_1 \in \mathbb{G}_1$:

1. The signing oracle is useless.
2. Since H is random, the attacker may as well select M_1 and M_2 first (with $M_1 \neq M_2$, so $h_1 \neq h_2$). Its task then is to find $X_2 \in \mathbb{G}_2$, $W_2 = \psi(X_2)$, and $\sigma \in \mathbb{G}_1$ such that $e(\sigma, g_2) = e(h_1, X_1) \cdot e(h_2, X_2)$. Call this problem P. Then co-DHP* $\leq$ P: Given $X_1 \in \mathbb{G}_2$, $W_1 = \psi(X_1)$, $h_1 \in \mathbb{G}_1$, select random y_2 and compute $h_2 = g_1^{y_2}$. Use the P-oracle to compute $X_2 \in \mathbb{G}_2$, $W_2 = \psi(X_2)$, and $\sigma \in \mathbb{G}_1$ such that $\sigma = h_1^{x_1} h_2^{x_2} = h_1^{x_1} W_2^{y_2}$. Then compute $h_1^{x_1} = \sigma / W_2^{y_2}$.

5. Waters Signature Scheme

Waters-1 Signature Scheme

[Waters, 2005]

An efficient pairing-based signature scheme with a security proof *that does not use random oracles*.

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a symmetric pairing with $\mathbb{G} = \langle g \rangle$, and let k denote the security parameter.

Let $H_1 : \{0,1\}^* \to \{0,1\}^k$ be a collision-resistant hash function.

Let u_0, u_1, ..., u_k be randomly selected elements of $\mathbb{G}$. Denote $U = (u_0, u_1, \ldots, u_k)$.

Define a hash function $H : \{0,1\}^* \to \mathbb{G}$ as follows:
Let $H_1(M) = (m_1, m_2, \ldots, m_k)$ (where each $m_i \in \{0,1\}$).
Then $H(M) = u_0 \prod u_i^{m_i}$.

Waters-1 Signature Scheme (2)

1. *Key generation*. Alice does:
   Private key $Z = g^z$; Public key: $\zeta = e(g,g)^z$.
2. *Signature generation*. To sign M, Alice does:
   Compute $h = H(M)$, and select $r \in_R [1, n-1]$.
   Compute $\alpha = Zh^r$ and $\beta = g^r$.
   The signature on M is $\sigma = (\alpha, \beta)$.
3. *Signature verification*. To verify (M, σ), Bob does:
   Compute $h = H(M)$.
   Accept iff $e(\alpha, g) = \zeta \cdot e(\beta, h)$.

Correctness:
$e(\alpha, g) = e(Zh^r, g) = e(Z, g) \cdot e(h^r, g) = e(g,g)^z \cdot e(\beta, h)$.

Hash Function Parameters U

Note: The public parameters u_0, u_1, ..., u_k should be generated verifiably at random (by a third party). That is, the third party should not know the discrete logarithms of the u_i.

This is because if a third party knew the $x_i = \log_g u_i$, then that party could recover Alice's private key Z given a single signed message (M, σ) as follows:

1. Compute $H_1(M) = (m_1, m_2, \ldots, m_k)$.
2. Compute $c = x_0 + \sum m_i x_i \bmod n$. [Then $h = H(M) = u_0 \prod u_i^{m_i} = g^{x_0} \prod g^{x_i m_i} = g^{x_0 + \sum m_i x_i} = g^c$]
3. Compute $Z = \alpha / \beta^c$. [Recall: $\alpha = Z h^r$, $\beta = g^r$]

Waters-1 Security

Theorem: If DHP in $\mathbb{G}$ is hard, and H_1 is collision-resistant, then the Waters-1 signature scheme is secure.

Security argument: Suppose we are given an instance (g^x, g^y) of DHP in $\mathbb{G}$. We show how an attacker of the Waters-1 scheme can be used to compute g^{xy}.

1. Set $\zeta = e(g^x, g^y) = e(g, g)^{xy}$; the corresponding (unknown) private key is $Z = g^{xy}$.
2. Let q be an upper bound on the number of signing queries made by the attacker, and select $t \in_R [0, k]$.
3. Select $a_0, a_1, \ldots, a_k \in_R [0, q-1]$ and $b_0, b_1, \ldots, b_k \in_R [0, n-1]$.
4. Compute $u_0 = (g^x)^{a_0 - tq} g^{b_0}$ and $u_i = (g^x)^{a_i} g^{b_i}$ for $1 \le i \le k$. Note that $H(M) = u_0 \prod u_i^{m_i} = g^{xF + J}$ where $F = F(M) = -tq + a_0 + \sum a_i m_i \bmod n$ and $J = J(M) = b_0 + \sum b_i m_i \bmod n$.

Waters-1 Security (2)

5. Now, run the attacker with public key ζ and hash function parameters $U = (u_0, u_1, \ldots, u_k)$.
6. Answer a signing query for M as follows:
   Compute $F = F(M)$ and $J = J(M)$. If $F = 0$ then abort.
   Otherwise, select random $\hat{r} \in \mathbb{Z}_n$ and compute $\alpha = (g^y)^{-J/F} h^{\hat{r}}$ and $\beta = g^{\hat{r}} (g^y)^{-1/F}$.
   Note: Let $r = \hat{r} - y/F$. Then $\beta = g^{r+y/F} g^{-y/F} = g^r$.
   And $\alpha = g^{-yJ/F} g^{(xF+J)(r+y/F)} = g^{xy} g^{(xF+J)r} = Z h^r$.
7. Suppose that the attacker eventually outputs a valid signature σ on a (new) message M. If $F(M) \neq 0$ then abort. Otherwise, we must have $\beta = g^r$ and $\alpha = Z h^r = Z (g^r)^J$. Hence, compute $Z = \alpha / \beta^J$.

The probability of not aborting is at least

$$\left(1 - \frac{1}{q}\right)^q \frac{1}{(k+1)q} \approx \frac{1}{(k+1)q}.$$

Waters-3a Signature Scheme

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a Type 3 pairing.

Case 1: Suppose we insist on short signatures, i.e., $\alpha, \beta \in \mathbb{G}_1$.

1. *Key generation*. Alice does:
   Private key $Z = g_1^z$; Public key: $\zeta = e(g_1, g_2)^z$.
2. *Signature generation*. To sign M, Alice does:
   Compute $h' = \psi(H(M))$, and select $r \in_R [1, n-1]$.
   Compute $\alpha = Z (h')^r$ and $\beta = g_1^r$.
   The signature on M is $\sigma = (\alpha, \beta)$.
3. *Signature verification*. To verify (M, σ), Bob does:
   Compute $h = H(M)$, and accept iff $e(\alpha, g_2) = \zeta \cdot e(\beta, h)$.

Problem: There is no ψ for a Type 3 pairing.
Solution: Each user selects $x_i \in_R [0, n-1]$ and computes $u_i = g_2^{x_i}$ and $v_i = g_1^{x_i}$. The user's public key includes $U = (u_0, u_1, \ldots, u_k)$, while $V = (v_0, v_1, \ldots, v_k)$ is kept private. [Then $h' = v_0 \prod v_i^{m_i}$]

Waters-3a Signature Scheme

Problem: The public key is now very large (1 element in $\mathbb{G}_T$ and $k+1$ elements in $\mathbb{G}_2$).

Recall co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$: Given g_1^x, g_2^x, $h \in \mathbb{G}_1$, compute h^x.

Theorem: If co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$ is hard, and H_1 is collision-resistant, then the Waters-3a signature scheme is secure.

Waters-3b Signature Scheme

Case 2: Suppose we insist on short and shared U,
i.e., $U \in \mathbb{G}_1^{k+1}$ and is generated by a third party.

1. *Key generation*. Alice does:
   Private key $Z = g_1^z$; Public key: $\zeta = e(g_1, g_2)^z$.
2. *Signature generation*. To sign M, Alice does:
   Compute $h = H(M)$, and select $r \in_R [1, n-1]$.
   Compute $\alpha = Zh^r$, $\beta = g_2^r$, and $\gamma = g_1^r$.
   The signature on M is $\sigma = (\alpha, \beta, \gamma)$.
3. *Signature verification*. To verify (M, σ), Bob does:
   Compute $h = H(M)$, and accept iff $e(\alpha, g_2) = \zeta \cdot e(h, \beta)$ and $e(\gamma, g_2) = e(g_1, \beta)$.

Problem: The signature consists of 2 $\mathbb{G}_1$ elements and 1 $\mathbb{G}_2$ element.

Waters-3b Signature Scheme

Why is the extra signature component γ needed?

In the security reduction, when the attacker returns a forgery $M, (\alpha, \beta, \gamma)$, we have $\alpha = Zh^r \in \mathbb{G}_1$, $\beta = g_2^r \in \mathbb{G}_2$, and $\gamma = g_1^r \in \mathbb{G}_1$. So, the solver can compute $Z = \alpha/\gamma^J$.

Theorem: If co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$ is hard, and H_1 is collision-resistant, then the Waters-3b signature scheme is secure.

6. LOSSW Aggregate Signature Scheme

Main Idea

[Lu, Ostrovsky, Sahai, Shacham, Waters; 2006]

Signature generation/aggregation is sequential:

- A_1 generates a Waters-1 signature (α_1, β_1) on M_1.
- A_2 sets $\beta_2 = \beta_1$ and generates its signature component α_2' for message M_2. (For this, A_2 should know the discrete logs of the hash function parameters, and therefore should have her own hash function parameters.)
  A_2 sets $\alpha_2 = \alpha_1 \cdot \alpha_2'$.
  A_2 then randomizes β_2 and α_2.
- A_3 then signs and aggregates ....

Note: Signature verification is not sequential.

LOSSW-1 Key Generation

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a symmetric pairing with $\mathbb{G} = \langle g \rangle$.

User A_j does:

- Select $x_{j,i} \in_R [0, n-1]$ for $0 \le i \le k$.
  Denote $X_j = (x_{j,0}, x_{j,1}, \ldots, x_{j,k})$.
- Compute $u_{j,i} = g^{x_{j,i}}$ for $0 \le i \le k$.
  Denote $U_j = (u_{j,0}, u_{j,1}, \ldots, u_{j,k})$.
- Select $Z_j = g^{z_j}$ and compute $\zeta_j = e(g,g)^{z_j}$.
- A_j's *public key* is (U_j, ζ_j).
  Note that U_j defines A_j's own hash function $H_j : \{0,1\}^* \to \mathbb{G}$.
- A_j's *private key* is (X_j, Z_j).

LOSSW-1 Signature Generation/Aggregation

1. A_1 signs M_1: $\alpha_1 = Z_1 h_1^{r_1}$, $\beta_1 = g^{r_1}$, where $h_1 = H_1(M_1)$.
   A_1 forwards $(A_1, M_1, \alpha_1, \beta_1)$ to A_2.
2. A_2 signs M_2 and aggregates as follows:
   Verify A_1's signature on M_1.
   Set $\beta_2 = \beta_1$.
   Compute $\alpha_2' = Z_2 h_2^{r_1}$, where $h_2 = H_2(M_2)$.
   Compute $\alpha_2 = \alpha_1 \cdot \alpha_2'$.
   Select $r_2 \in_R [0, n-1]$.
   Compute $\beta_2 = \beta_1 \cdot g^{r_2}$ and $\alpha_2 \leftarrow \alpha_2 h_1^{r_2} h_2^{r_2}$.
   (Note that $\beta_2 = g^{r_1+r_2}$ and $\alpha_2 = Z_1 h_1^{r_1+r_2} Z_2 h_2^{r_1+r_2}$)
   A_2 forwards $(A_1, A_2, M_1, M_2, \alpha_2, \beta_2)$ to A_3.
3. A_3 verifies the aggregate signature, signs M_3, and aggregates .....

LOSSW-1 Signature Verification

Given (A_j, M_j) for $1 \le j \le \ell$, and aggregate signature $(\alpha_\ell, \beta_\ell)$, a verifier does the following:

Verify that the users A_j are pairwise distinct.
Compute $h_j = H_j(M_j)$ for $1 \le j \le \ell$.
Accept iff $e(\alpha_\ell, g) = (\prod \zeta_j) \cdot e(\beta_\ell, \prod h_j)$.

Correctness: (for $\ell = 2$)

$$\begin{aligned} e(\alpha_2, g) &= e(Z_1 h_1^{r_1+r_2} Z_2 h_2^{r_1+r_2}, g) \\ &= e(Z_1, g) \cdot e(Z_2, g) \cdot e((h_1 h_2)^{r_1+r_2}, g) \\ &= \zeta_1 \cdot \zeta_2 \cdot e(\beta_2, h_1 h_2). \end{aligned}$$

LOSSW-1 Security

The security proof is in the *certified-key model*, where a user has to present its private key (X, Z) to the CA in order to get its public key (U, ζ) certified.

- The LOSSW-1 attacker is given a target public key (U, ζ).
- The attacker can provide key pairs (X_j, Z_j) for certification.
- The attacker can request sequential aggregate signatures w.r.t. the target public key (U, ζ). For this, it also has to supply a list of (certified) public keys, a list of messages, and an aggregated signature.
- The attacker's goal is to produce a list of public keys (including the target key), a list of messages, and a valid aggregated signature. Of course, the message corresponding to the target key cannot have been queried to its signing oracle.

LOSSW-1 Security (2)

Theorem: If an LOSSW-1 attacker can meet the goal described above, then the Waters-1 signature scheme is insecure.

The proof shows how to construct a Waters-1 attacker, given an LOSSW-1 attacker. The proof makes crucial use of the requirement that the LOSSW-1 attacker operate in the certified-key model.

7. Barreto-Naehrig Elliptic Curves

Barreto-Naehrig (BN) Curves

BN curves were introduced in 2005.

Let z be an integer so that $n = 36z^4 + 36z^3 + 18z^2 + 6z + 1$ and $p = 36z^4 + 36z^3 + 24z^2 + 6z + 1$ are prime. Let $t = 6z^2 + 1$.

Then there is an ordinary elliptic curve $E/\mathbb{F}_p : Y^2 = X^3 + b$, with $\#E(\mathbb{F}_p) = n = p + 1 - t$ and *embedding degree* $k = 12$.

"Embedding degree $k = 12$" means that $k = 12$ is the smallest positive integer such that n divides $p^k - 1$.

It follows that $E[n] \subseteq E(\mathbb{F}_{p^{12}})$, where $E[n]$ denotes the set of all n-torsion points on E. (Recall that $E[n] \cong \mathbb{Z}_n \oplus \mathbb{Z}_n$)

Pairings from BN Curves

Define $\mathbb{G}_1 = E(\mathbb{F}_p)$.

Let $Q \in E[n]$, $Q \notin \mathbb{G}_1$, and define $\mathbb{G}_2 = \langle Q \rangle$.

Let $\mathbb{G}_T$ be the unique order-n subgroup of $\mathbb{F}_{p^{12}}^*$.

$\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of order n.

The Tate pairing yields an asymmetric bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. This pairing can be of Type 2 or Type 3, depending on the choice of $\mathbb{G}_2$.

Note: $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive groups, while $\mathbb{G}_T$ is a multiplicative group.

BN Curves at the 128-Bit Security Level

BN curves are ideal for the *128-bit* security level.

Let p be a 256-bit prime number.

Let $\mathbb{G}_1 = E(\mathbb{F}_p)$.

Then the DLP in $\mathbb{G}_1$ can be solved in $\approx \sqrt{p} \approx 2^{128}$ time using *Pollard rho's method*.

Also, the Tate pairing can be used to reduce the DLP in $\mathbb{G}_1$ to the DLP in $\mathbb{F}_{p^{12}}$; the latter can be solved in $\approx 2^{128}$ time using the *Number Field Sieve algorithm*.

A Particular BN Curve

Let $z = 0x6000000000001F2D$.

For this choice of BN parameter, p is a 256-bit prime of Hamming weight 87.

The BN curve $E/\mathbb{F}_p : Y^2 = X^3 + 3$ has order n, a 256-bit prime of Hamming weight 91.

We have $n = p + 1 - t$ where $t - 1$ is a 128-bit integer of Hamming weight 28.

Note that $p \equiv 7 \pmod 8$ (whence -2 is a nonsquare modulo p) and $p \equiv 1 \pmod 6$.

We will only describe the Tate, Ate, and R-ate pairings for this particular BN curve.

8. Tate, Ate and R-ate pairings

Extension Field Arithmetic

The extension field $\mathbb{F}_{p^{12}}$ is represented using the tower extensions:

$\mathbb{F}_{p^2} = \mathbb{F}_p[u]/(u^2+2)$.
$\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3-\xi)$, where $\xi = -u-1$.
$\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[w]/(w^2-v)$.

We also have $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^2}[W]/(W^6-\xi)$, where $W = w$.

Cost of arithmetic operations in $\mathbb{F}_p$:

- Let m, s, i denote the cost of multiplication, squaring, inversion in $\mathbb{F}_p$.
- We have $s \approx 0.9m$; we will use $s \approx m$
- $i \approx 41m$ [on a Pentium 4].

Arithmetic in $\mathbb{F}_{p^2}$

$\mathbb{F}_{p^2} = \mathbb{F}_p[u]/(u^2+2)$.
An element of $\mathbb{F}_{p^2}$ is represented as $a+bu$, where $a, b \in \mathbb{F}_p$.

Let $\tilde{m}$, $\tilde{s}$, $\tilde{i}$ denote the cost of multiplication, squaring, inversion in $\mathbb{F}_{p^2}$.

Multiplication: $\tilde{m} \approx 3m$ [Karatsuba]

Squaring: $\tilde{s} \approx 2m$
$[(a+bu)^2 = (a-b)(a+2b) - ab + (2ab)u]$

Inversion: $\tilde{i} \approx i + 2m + 2s \approx i + 4m$
$[(a+bu)^{-1} = (a-bu)/(a^2+2b^2)]$

p-th powering: free since $(a+bu)^p = a - bu$.

Arithmetic in $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{12}}$

$\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3 - \xi)$, where $\xi = -u - 1$.

Multiplication in $\mathbb{F}_{p^6}$: $6\tilde{m} \approx 18m$ [Karatsuba]
Squaring in $\mathbb{F}_{p^6}$: $2\tilde{m} + 3\tilde{s} \approx 12m$ [Chung-Hasan]
Inversion in $\mathbb{F}_{p^6}$: $\tilde{i} + 9\tilde{m} + 3\tilde{s} \approx i + 37m$ [Scott]

$\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[w]/(w^2 - v)$.

Let M, S, I denote the cost of multiplication, squaring, inversion in $\mathbb{F}_{p^{12}}$.

Multiplication: $M \approx 54m$ [Karatsuba]
Squaring: $S \approx 36m$
Inversion: $I \approx i + 97m$.

Pairings

Recall that $E/\mathbb{F}_p : Y^2 = X^3 + 3$ and $\mathbb{G}_1 = E(\mathbb{F}_p)$.

Consider $\tilde{E}/\mathbb{F}_{p^2} : Y^2 = X^3 + 3/\xi$.
$\tilde{E}$ is a degree-6 twist of E over $\mathbb{F}_{p^2}$, and $n \mid \#\tilde{E}(\mathbb{F}_{p^2})$.

Let $\tilde{Q} \in \tilde{E}(\mathbb{F}_{p^2})$ be a point of order n, and let $\tilde{\mathbb{G}}_2 = \langle \tilde{Q} \rangle$.
Then $\phi : \tilde{\mathbb{G}}_2 \to E(\mathbb{F}_{p^{12}})$ defined by $(x, y) \mapsto (xW^2, yW^3)$ is a group monomorphism.

Let $Q = \phi(\tilde{Q})$, and let $\mathbb{G}_2 = \langle Q \rangle$. Then $\mathbb{G}_2 \neq \mathbb{G}_1$, and $\phi : \tilde{\mathbb{G}}_2 \to \mathbb{G}_2$ is a group isomorphism.

$\mathbb{G}_2$ is the *Trace-0 subgroup* of $E[n]$.

Note: No efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ is known.

Tate Pairing

Let $P \in \mathbb{G}_1$. A *Miller function* $f_{n,P}$ is a function whose only zeros/poles in E are a zero of order n at P, and a pole of order n at ∞.

The (full)*Tate pairing* $\hat{e} : E[n] \times E[n] \to \mathbb{G}_T$ is a bilinear pairing defined as follows: Let $P, Q \in E[n]$, and let $R \in E(\mathbb{F}_{p^{12}})$ with $R \notin \{\infty, P, -Q, P - Q\}$. Then

$$\hat{e}(P, Q) = \big(f_{n,P}(Q + R)/f_{n,P}(R)\big)^{(p^{12}-1)/n}.$$

The (restricted) *Tate pairing* $t_n : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is defined by

$$t_n(P, Q) = \big(f_{n,P}(Q)\big)^{(p^{12}-1)/n}.$$

t_n is an asymmetric bilinear pairing.

Since no efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ is known, t_n is a *Type 3* pairing.

Computing the Tate Pairing

INPUT: $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.
OUTPUT: $t_n(P, Q)$.

1. Write n in binary: $n = \sum_{i=0}^{L-1} n_i 2^i$.
2. $T \leftarrow P$, $f \leftarrow 1$.
3. For i from $L - 2$ downto 0 do: {Miller operation}
   - 3.1 Let ℓ be the tangent line at T.
   - 3.2 $T \leftarrow 2T$, $f \leftarrow f^2 \cdot \ell(Q)$.
   - 3.3 If $n_i = 1$ and $i \neq 0$ then
     Let ℓ be the line through T and P.
     $T \leftarrow T + P$, $f \leftarrow f \cdot \ell(Q)$.
4. Return($f^{(p^{12}-1)/n}$). {Final exponentiation}

Ate Pairing

[Hess, Smart, Vercauteren; 2006]

The *Ate pairing* $a_n : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is defined by

$$a_n(P,Q) = \left(f_{t-1,Q}(P)\right)^{(p^{12}-1)/n}.$$

a_n is a *Type 3* asymmetric bilinear pairing.

Note: $a_n(P,Q) = \hat{e}(Q,P)^M$ for some fixed integer M.

Note: $t \approx \sqrt{n}$.

Computing the Ate Pairing

INPUT: $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.
OUTPUT: $a_n(Q,P)$.

1. Write $t-1$ in binary: $t-1 = \sum_{i=0}^{L-1} t_i 2^i$.
2. $T \leftarrow Q$, $f \leftarrow 1$.
3. For i from $L-2$ downto 0 do: {Miller operation}
   - 3.1 Let ℓ be the tangent line at T.
   - 3.2 $T \leftarrow 2T$, $f \leftarrow f^2 \cdot \ell(P)$.
   - 3.3 If $t_i = 1$ then
     Let ℓ be the line through T and Q.
     $T \leftarrow T+Q$, $f \leftarrow f \cdot \ell(P)$.
4. Return($f^{(p^{12}-1)/n}$). {Final exponentiation}

R-ate Pairing

[Lee, Lee, Park; 2008]

The *R-Ate pairing* $R_n : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is defined by

$$R_n(P,Q) = \left(f \cdot \left(f \cdot \ell_{aQ,Q}(P)\right)^p \cdot \ell_{\pi(aQ+Q),aQ}(P)\right)^{(p^{12}-1)/n},$$

where $a = 6z+2$, $f = f_{a,Q}(P)$, $\ell_{A,B}$ denotes the line through A and B, and $\pi : (x,y) \mapsto (x^p, y^p)$ is the Frobenius map.

R_n is a *Type 3* asymmetric bilinear pairing.

Note: $R_n(P,Q) = \hat{e}(Q,P)^L$ for some fixed integer L.

Note: $a \approx \sqrt{t} \approx n^{1/4}$.

Computing the R-ate Pairing

INPUT: $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.
OUTPUT: $R_n(Q,P)$.

1. Write $a = 6z+2$ in binary: $a = \sum_{i=0}^{L-1} a_i 2^i$.
2. $T \leftarrow Q$, $f \leftarrow 1$.
3. For i from $L-2$ downto 0 do: {Miller operation}
   3.1 Let ℓ be the tangent line at T.
   3.2 $T \leftarrow 2T$, $f \leftarrow f^2 \cdot \ell(P)$.
   3.3 If $a_i = 1$ then
   Let ℓ be the line through T and Q.
   $T \leftarrow T+Q$, $f \leftarrow f \cdot \ell(P)$.
4. $f \leftarrow f \cdot (f \cdot \ell_{T,Q}(P))^p \cdot \ell_{\pi(T+Q),T}(P)$.
5. Return($f^{(p^{12}-1)/n}$). {Final exponentiation}

Comparisons

Parameter	Bitlength	Hamming weight
n	256	91
$t-1$	128	28
a	66	9

Pairing	Miller operation	Final exp	Total	Timings*
Tate	27934m	7246m+i	35180m+i	–
Ate	15801m	7246m+i	23047m+i	81
R-ate	7847m+i	7246m+i	15093m+$2i$	54

*Timings: 10^6 clock cycles on a 2.8 GHz Pentium 4 using general purpose registers.
($m \approx 2,200$ cycles; $54 \cdot 10^6$ clock cycles ≈ 0.05 seconds)

9. Type 2 Pairings versus Type 3 Pairings

Type 2 Pairings

Recall that $\mathbb{G}_1 = E(\mathbb{F}_p)$ and $\mathbb{G}_2 = \langle Q \rangle$, where $Q = \phi(\tilde{Q})$ and $\tilde{Q} \in \tilde{E}(\mathbb{F}_{p^2})[n]$. The R-ate pairing $R_n : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a Type 3 pairing.

Recall also that $R_n(P, Q) = \hat{e}(Q, P)^L$ for some fixed integer L.

Let $R \in E(\mathbb{F}_{p^{12}})[n]$ with $R \notin \mathbb{G}_1$ and $R \notin \mathbb{G}_2$. Define $\mathbb{G}_2' = \langle R \rangle$.

Define $e_n : \mathbb{G}_1 \times \mathbb{G}_2' \to \mathbb{G}_T$ by $e_n(P, Q) = \hat{e}(Q, P)^{2L}$.

Then e_n is an asymmetric bilinear pairing.

The *Trace map* $\mathsf{Tr} : \mathbb{G}_2' \to \mathbb{G}_1$ defined by $\mathsf{Tr}(Z) = Z + \sigma(Z) + \cdots + \sigma^{11}(Z)$ is an efficiently computable isomorphism. [σ is p-th power Frobenius, $(x, y) \mapsto (x^p, y^p)$]

Thus $e_n : \mathbb{G}_1 \times \mathbb{G}_2' \to \mathbb{G}_T$ is a *Type 2* pairing.

Computing Type 2 Pairings

Let $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2'$. We wish to compute $e_n(P, Q)$.

Define $\hat{Q} = Q - \sigma^6(Q)$. Then $\hat{Q} \neq \infty$.
And $\mathsf{Tr}(\hat{Q}) = \mathsf{Tr}(Q) - \mathsf{Tr}(\sigma^6(Q)) = \infty$, so $\hat{Q} \in \mathbb{G}_2$.

Now, $e_n(P, Q) = \hat{e}(Q, P)^{2L} = \hat{e}(2Q, P)^L$
$= \hat{e}(Q + \hat{Q} + \sigma^6(Q), P)^L$
$= \hat{e}(\hat{Q}, P)^L \cdot \hat{e}(Q + \sigma^6(Q), P)^L$
$= \hat{e}(\hat{Q}, P)^L \cdot \hat{e}(\mathsf{Tr}_{\mathbb{F}_{p^{12}}/\mathbb{F}_{p^6}}(Q), P)^L$
$= R_n(P, \hat{Q})$.

Thus the task of computing $e_n(P, Q)$ is easily reduced to the task of computing an R-ate pairing $R_n(P, \hat{Q})$.

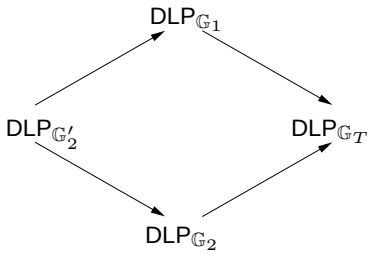
Type 2 versus Type 3: Security

Type 2: $e_n : \mathbb{G}_1 \times \mathbb{G}'_2 \to \mathbb{G}_T$. Type 3: $R_n : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

co-DHP: Given g_2^x and $h \in \mathbb{G}_1$, compute h^x.
co-DHP*: Given g_1^x, g_2^x, and $h \in \mathbb{G}_1$, compute h^x.

For Type 2, Tr : $\mathbb{G}'_2 \to \mathbb{G}_1$ is an efficiently computable isomorphism. Hence DLP$_{\mathbb{G}'_2} \leq$ DLP$_{\mathbb{G}_1} \leq$ DLP$_{\mathbb{G}_T}$.

Also, $\phi : \mathbb{G}'_2 \to \mathbb{G}_2$ defined by $Q \to Q - \frac{1}{k}\mathsf{Tr}(Q)$ is an efficiently computable isomorphism. Hence DLP$_{\mathbb{G}'_2} \leq$ DLP$_{\mathbb{G}_2} \leq$ DLP$_{\mathbb{G}_T}$.



Decisional DHP is easy in $\mathbb{G}'_2$, but not known to be easy in $\mathbb{G}_1$/$\mathbb{G}_2$.

Type 2 versus Type 3: Efficiency

	Type 2	*Type 3*
Bitlength of elements in $\mathbb{G}_1$	256	256
Bitlength of elements in $\mathbb{G}'_2/\mathbb{G}_2$	6144	512
Bitlength of elements in $\mathbb{G}_T$	3072	3072
Decompressing elements in $\mathbb{G}_1$	315m	315m
Decompressing elements in $\mathbb{G}'_2/\mathbb{G}_2$	46,764m	674m
Exponentiation in $\mathbb{G}_1$	2,345m	2,345m
Exponentiation in $\mathbb{G}'_2/\mathbb{G}_2$	105,462m	5,859m
Fixed-base exp in $\mathbb{G}_1$	718m	718m
Fixed-base exp in $\mathbb{G}'_2/\mathbb{G}_2$	34,312m	1,906m
e_N/R_N Pairing	15,175m	15,175m
Hashing into $\mathbb{G}_1$	315m	315m
Hashing into $\mathbb{G}'_2/\mathbb{G}_2$	—	6,533m

Efficiency Notes

Representing points

- Elements in $\mathbb{G}_1$: A point $Q = (x, y) \in E(\mathbb{F}_p)$ can be represented by x (plus one bit of y). The value y can be recovered by solving $y^2 = x^3 + 3$ over $\mathbb{F}_p$ via $y = \pm\sqrt{x^3 + 3} = (x^3 + 3)^{(p+1)/4}$. This exponentiation can be done using sliding windows of width 5.
- Elements in $\mathbb{G}_2$: A point $\tilde{Q} = (x, y) \in \tilde{E}(\mathbb{F}_{p^2})$ can be represented by x (plus one bit of y). The y-coordinate $y = \pm\sqrt{x^3 + 3}$ can be recovered at a cost of $2\sqrt{\cdot} + i + m + 2s$.
- Elements in $\mathbb{G}_2'$: A point $Q = (x, y) \in E(\mathbb{F}_{p^{12}})$ can be represented by x and y (point decompression is too expensive).

Efficiency Notes (2)

Exponentiation

- 5-NAF method. Cost of w-NAF method with ℓ-bit multipliers is

$$1D + (2^{w-2} - 1)A + \frac{\ell}{w+1}A + \ell D.$$

  Here, $D = 3m + 4s$ and $A = 8m + 3s$.
- Fixed-base: Comb method with two tables.

Hashing

- Into $\mathbb{G}_1$: Hash to an x-coordinate of $E(\mathbb{F}_p)$; then compute the y-coordinate.
- Into $\mathbb{G}_2$: Hash to an x-coordinate of $\tilde{E}(\mathbb{F}_{p^2})$; then compute the y-coordinate; then multiply by $\#\tilde{E}(\mathbb{F}_{p^2})/n$.
- Into $\mathbb{G}_2'$: Cannot be done efficiently.

10. BLS versus Waters

BLS versus Waters

	BLS-2	*BLS-3*	*Waters-3a*	*Waters-3b*
Security	RO	RO	coll-res	coll-res
	co-DHP	co-DHP*	co-DHP*	co-DHP*
Public key size	6144	512	32.5KB	3072
Signature size	256	256	512	1024
Sig. generation	2,660m	2,660m	4,471m	6,377m
Sig. verification	30,980m	31,654m	34,820m	63,412m

RO = random oracle, KB = Kilobytes

Example: Waters-3b signature verification = 1 Waters-hash + 2 decompressions in $\mathbb{G}_1$ + 1 decompression in $\mathbb{G}_2$ + 4 pairings = (128 $\times$ 11) + (2 $\times$ 315) + 674 + (4 $\times$ 15175) = 63,412m.

Products of R-ate Pairings

Objective: Compute $R_n(P_1, Q_1) \cdot R_n(P_2, Q_2) \cdot R_n(P_3, Q_3) \cdots$

Cost of a single R-ate pairing:

- Miller operation:
  - [65 times] Compute ℓ, $2T$, $f \leftarrow f \cdot \ell(P)$: 71$m$.
  - [65 times] $f \leftarrow f^2$: 36m.
  - [8 times] Compute ℓ, $T \leftarrow T + P$, $f \leftarrow f \cdot \ell(P)$: 79$m$
- Step 4: 260m + i.
- Final exponentiation $f^{(p^{12}-1)/n}$: 7246m + i.

If the product of t pairings is desired, then the steps can be interleaved, with all instances sharing the same accumulator f.

Thus, the $f \leftarrow f^2$ operation and the final exponentiation can be shared by all t pairing instances.

So, the cost of computing the product of t pairings is (15093m+2i) + ($t - 1$)(5507m + i).

BLS versus Waters (revised)

	BLS-2	*BLS-3*	*Waters-3a*	*Waters-3b*
Security	RO	RO	coll-res	coll-res
	co-DHP	co-DHP*	co-DHP*	co-DHP*
Public key size	6144	512	32.5KB	3072
Signature size	256	256	512	1024
Sig. generation	2,660m	2,660m	4,471m	6,377m
Sig. verification	21,353m	22,027m	25,193m	44,158m

11. BGLS versus LOSSW

BGLS-3

Public key: $X_i \in \mathbb{G}_2$.
Note: $W_i = \psi(X_i) \in \mathbb{G}_1$ is needed during public-key certification, but is not used in signature generation or verification.

Signature generation: $\sigma_i = h_i^{x_i} \in \mathbb{G}_1$, where $h_i = H(M_i)$.

Aggregate verification (ℓ users): Decompress σ + Decompress X_i + hash M_i + product of $\ell + 1$ pairings: $e(\sigma, g_2) = \prod e(h_i, X_i)$.

BGLS-2 versus BGLS-3

	BGLS-2	*BGLS-3*
Security	RO	RO
	co-DHP	co-DHP*
Public key size	6144	512
Signature size	256	256
Sig. generation	2,660	2,660
Sig. verification	15,490	15,490
	+ 5,863ℓ	+ 6,537ℓ

LOSSW-3a Key Generation

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a Type 3 pairing.

User A_j does:

- Select $x_{j,i} \in_R [0, n-1]$ for $0 \leq i \leq k$.
  Denote $X_j = (x_{j,0}, x_{j,1}, \ldots, x_{j,k})$.
- Compute $u_{j,i} = g_2^{x_{j,i}}$ and $v_{j,i} = g_1^{x_{j,i}}$ for $0 \leq i \leq k$.
  Denote $U_j = (u_{j,0}, u_{j,1}, \ldots, u_{j,k})$ and
  $V_j = (v_{j,0}, v_{j,1}, \ldots, v_{j,k})$.
- Select $Z_j = g_1^{z_j}$ and compute $\zeta_j = e(g, g)^{z_j}$.
- A_j's *public key* is (U_j, V_j, ζ_j).
  Note that U_j defines A_j's own hash function
  $H_j : \{0,1\}^* \rightarrow \mathbb{G}_2$ and V_j defines $\psi(H_j)$.
- A_j's *private key* is (X_j, Z_j).

LOSSW-3a Signature Generation/Aggregation

1. A_1 signs M_1: $\alpha_1 = Z_1(h_1')^{r_1}$, $\beta_1 = g_1^{r_1}$, where $h_1' = \psi(H_1(M_1))$.
   A_1 forwards $(A_1, M_1, \alpha_1, \beta_1)$ to A_2.
2. A_2 signs M_2 and aggregates as follows:
   Verify A_1's signature on M_1.
   Set $\beta_2 = \beta_1$.
   Compute $\alpha_2' = Z_2(h_2')^{r_1}$, where $h_2' = \psi(H_2(M_2))$.
   Compute $\alpha_2 = \alpha_1 \cdot \alpha_2'$.
   Select $r_2 \in_R [0, n-1]$.
   Compute $\beta_2 = \beta_1 \cdot g_1^{r_2}$ and $\alpha_2 \leftarrow \alpha_2 (h_1')^{r_2} (h_2')^{r_2}$.
   (Note that $\beta_2 = g_1^{r_1+r_2}$ and $\alpha_2 = Z_1(h_1')^{r_1+r_2} Z_2(h_2')^{r_1+r_2}$)
   A_2 forwards $(A_1, A_2, M_1, M_2, \alpha_2, \beta_2)$ to A_3.
3. A_3 verifies the aggregate signature, signs M_3, and aggregates .....

LOSSW-3a Signature Verification

Given (A_j, M_j) for $1 \le j \le \ell$, and aggregate signature $(\alpha_\ell, \beta_\ell)$, a verifier does the following:

Verify that the users A_j are pairwise distinct.
Compute $h_j = H_j(M_j)$ for $1 \le j \le \ell$.
Accept iff $e(\alpha_\ell, g_2) = (\prod \zeta_j) \cdot e(\beta_\ell, \prod h_j)$.

BGLS versus LOSSW

	BGLS-2	*BGLS-3*	*LOSSW-3a*	*LOSSW-3b*
Security	RO co-DHP	RO co-DHP*	coll-res co-DHP* CKM	coll-res co-DHP* CKM
Public key size	6144	512	48.6KB	16.4KB
Signature size	256	256	512	1024
Sig. generation	2,660	2,660	28,169 + 5,248d	51,472 + 1408d
Sig. verification	15,490 + 5,863ℓ	15,490 + 6,537ℓ	21,353 + 3,840ℓ	42,750 + 1408ℓ

CKM = Certified-key model, KB = kilobytes, $d \in [1, \ell - 1]$

Other Aggregate Signature Schemes

- Lysyanskaya, Micali, Reyzin, Shacham (EUROCRYPT 2004)
  Sequential aggregate signatures from trapdoor permutations.
- Gentry, Ramzan (PKC 2006)
  Identity-based aggregate signature schemes.
- Neven (EUROCRYPT 2008)
  Sequential aggregate signed data.

12. Further Reading (1)

1. A. Menezes, “An introduction to pairing-based cryptography”, www.cacr.math.uwaterloo.ca/∼ajmeneze/publications/pairings.pdf.
2. [*BLS*] D. Boneh, B. Lynn and H. Shacham, “Short signatures from the Weil pairing”, *Journal of Cryptology*, 17 (2004), 297-319.
3. [*BGLS*] D. Boneh, C. Gentry, H. Shacham and B. Lynn, “Aggregate and verifiably encrypted signatures from bilinear maps”, *EUROCRYPT 2003*. Full paper available from crypto.stanford.edu/∼dabo/abstracts/aggreg.html.
4. [*Waters*] B. Waters, “Efficient identity-based encryption without random oracles”, *EUROCRYPT 2005*, LNCS 3494, pp. 114-127.
5. [*LOSSW*] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham and B. Waters, “Sequential aggregate signatures and multisignatures without random oracles”, *EUROCRYPT 2006*, LNCS 4004, pp. 465-485.

Further Reading (2)

6. S. Galbraith “Pairings”, Ch. IX of I. Blake, G. Seroussi and N. Smart, eds., *Advances in Elliptic Curve Cryptography*, Cambridge University Press, 2005.
7. S. Galbraith, K. Paterson and N. Smart, “Pairings for cryptographers”, eprint.iacr.org/2006/165.
8. M. Scott, “Implementing cryptographic pairings”, *Pairing-Based Cryptography – Pairing 2007*, LNCS 4575 (2007), 177–196.
9. A. Devegili, M. Scott and R. Dahab, “Implementing cryptographic pairings over Barreto-Naehrig curves”, *Pairing-Based Cryptography – Pairing 2007*, LNCS 4575 (2007), 197–207.
10. D. Hankerson, A. Menezes and M. Scott, “Software implementation of pairings”, draft chapter, 2008. www.cacr.math.uwaterloo.ca/∼ajmeneze/publications/pairings_software.pdf.