

# Asymmetric Pairings

Alfred Menezes

(joint work with S. Chatterjee, D. Hankerson & E. Knapp)



# Overview

In their 2006 paper "Pairings for cryptographers", Galbraith, Paterson and Smart identified three kinds of pairings derived from elliptic curves:

- Type 1 pairings (symmetric, supersingular curves)

- Type 2 and 3 pairings (asymmetric, ordinary curves).

Protocol designers generally describe and analyze their protocols using Type 1 or Type 2 pairings.

The purpose of this talk is to give two examples that illustrate the following points:

1. There appears to be no advantage to using a Type 2 pairing over a Type 3 pairing.
2. When converting a protocol from Type 1 to Type 3, some tradeoffs may have to be made based on performance.

# Pairings

- ▶ Let  $n$  be a prime number.
- ▶ Let  $\mathbb{G}_1 = \langle P_1 \rangle$ ,  $\mathbb{G}_2 = \langle P_2 \rangle$ ,  $\mathbb{G}_T$  be groups of order  $n$ .

A **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  $U_1, U_2 \in \mathbb{G}_1$ ,  $V_1, V_2 \in \mathbb{G}_2$ :

$$e(U_1 + U_2, V_1) = e(U_1, V_1) \cdot e(U_2, V_1)$$

$$e(U_1, V_1 + V_2) = e(U_1, V_1) \cdot e(U_1, V_2).$$

2. [Non-degeneracy]  $e(P_1, P_2) \neq 1$ .
3. [Computability]  $e$  can be efficiently computed.

**Note:**  $e(aU, bV) = e(U, V)^{ab} = e(bU, aV) \forall U \in \mathbb{G}_1, V \in \mathbb{G}_2, a, b \in \mathbb{Z}$ .

The pairing is **symmetric** if  $\mathbb{G}_1 = \mathbb{G}_2$ ; else it is **asymmetric**.

# Boneh-Lynn-Shacham (BLS) Signatures

[Boneh, Lynn, Shacham; 2001]

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:  $X = xP_2 \in \mathbb{G}_2$ .

2. **Signature generation.** To sign  $M$ , Alice does:

Compute  $H = \text{Hash}(M)$ , where  $\text{Hash} : \{0, 1\}^* \rightarrow \mathbb{G}_1$ .

Compute  $\sigma = xH \in \mathbb{G}_1$ .

The signature on  $M$  is  $\sigma$ .

3. **Signature verification.** To verify  $(M, \sigma)$ , Bob does:

Compute  $H = \text{Hash}(M)$ .

Accept iff  $e(\sigma, P_2) = e(H, X)$ .

**Correctness:**  $e(\sigma, P_2) = e(xH, P_2) = e(H, xP_2) = e(H, X)$ .

# co-DHP

A **necessary** condition for the security of the BLS signature scheme is that **co-DHP** in  $(\mathbb{G}_1, \mathbb{G}_2)$  is hard:

Given  $H \in \mathbb{G}_1$  and  $xP_2 \in \mathbb{G}_2$ , compute  $xH$ .

[BLS] This condition is, in general, **not sufficient** for security.

**Example:** Let  $n \mid p - 1$ . Let  $\mathbb{G}_1 = (\mathbb{Z}_n, +)$ , and let  $\mathbb{G}_2$  be the order- $n$  subgroup of  $\mathbb{Z}_p^*$ . Define  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_2$  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 scheme is insecure – given a single signed message  $(M, \sigma)$ , the private key can easily be recovered (since  $\sigma = xH \bmod n$  where  $H = \text{Hash}(M)$ ).

# Type 2 and 3 Asymmetric Pairings

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

If an **efficiently-computable isomorphism**  $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ ,  $\psi : P_2 \mapsto P_1$ , is known, then  $e$  is called a **Type 2 pairing**.

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

**BLS-2**: BLS with a Type 2 pairing

**BLS-3**: BLS with a Type 3 pairing

# BLS-2 Security

**Theorem:** If co-DHP in  $(\mathbb{G}_1, \mathbb{G}_2)$  is hard and Hash 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  $e(\sigma, P_2) = e(H, X)$  for random  $H \in \mathbb{G}_1$ . However, the attacker can generate such  $(H, \sigma)$  itself by selecting random integers  $y$  and computing  $H = yP_1$  and  $\sigma = y\psi(X)$ .
2. The attacker's problem is reduced to computing  $\sigma = xH$  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  $H, xP_1 \in \mathbb{G}_1$  and  $xP_2 \in \mathbb{G}_2$ , compute  $xH$ .

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

**Security argument.** Given  $X \in \mathbb{G}_2$  and  $\psi(X)$ :  $[\psi(xP_2) = xP_1]$

1. The signing oracle is useless: It gives  $\sigma \in \mathbb{G}_1$  such that  $e(\sigma, P_2) = e(H, X)$  for random  $H \in \mathbb{G}_1$ . However, the attacker can generate such  $(H, \sigma)$  itself by selecting random integers  $y$  and computing  $H = yP_1$  and  $\sigma = y\psi(X)$ .
2. The attacker's problem is reduced to computing  $\sigma = xH$  given  $H \in \mathbb{G}_1, X \in \mathbb{G}_2$  and  $\psi(X)$ . This is precisely co-DHP\* in  $(\mathbb{G}_1, \mathbb{G}_2)$ .

# Type 2 or Type 3?

**co-DHP**: Given  $Q \in \mathbb{G}_1$  and  $zP_2 \in \mathbb{G}_2$ , compute  $zQ$ .

**co-DHP\***: Given  $Q, zP_1 \in \mathbb{G}_1$  and  $zP_2 \in \mathbb{G}_2$ , compute  $zQ$ .

[BLS] co-DHP\* is a “stronger complexity assumption” than co-DHP.

[BLS] “Since  $\psi$  naturally exists in all the group pairs  $(\mathbb{G}_1, \mathbb{G}_2)$  we are considering, there is no reason to rely on this stronger complexity assumption.”

# Barreto-Naehrig (BN) Curves

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

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.

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

( $k = 12$  is the smallest positive integer with  $n \mid p^k - 1$ )

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

# Pairings from BN Curves

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

The (full) **Tate pairing**  $\hat{e} : E[n] \times E[n] \rightarrow \mathbb{G}_T$  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

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

Let  $\mathbb{G}_1 = E(\mathbb{F}_p)$ ; let  $\mathbb{G}_2$  be any other order- $n$  subgroup of  $E[n]$ .

The (restricted) **Tate pairing**  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  is:

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

To allow ‘denominator elimination’, one selects  $\mathbb{G}_2$  to be the Trace-0 subgroup of  $E[n]$ .

# Trace-0 Subgroup

$E$  has a degree-6 twist  $\tilde{E} : Y^2 = X^3 + b'$  over  $\mathbb{F}_{p^2}$  with  $n \parallel \#\tilde{E}(\mathbb{F}_{p^2})$ .

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 there is an efficiently-computable monomorphism  $\phi : \tilde{\mathbb{G}}_2 \rightarrow E(\mathbb{F}_{p^{12}})$ .

Let  $Q = \phi(\tilde{Q})$ , and let  $\mathbb{G}_2 = \langle Q \rangle$ .

Then  $\mathbb{G}_2 \neq \mathbb{G}_1$ , and  $\phi : \tilde{\mathbb{G}}_2 \rightarrow \mathbb{G}_2$  is a group isomorphism.

We can thus identify  $\mathbb{G}_2$  and  $\tilde{\mathbb{G}}_2$ , so elements of  $\mathbb{G}_2$  are essentially defined over  $\mathbb{F}_{p^2}$  (instead of over  $\mathbb{F}_{p^{12}}$ ).

$\mathbb{G}_2$  is the **Trace-0 subgroup** of  $E[n]$  (all points  $P$  in  $E[n]$  satisfying  $\text{Tr}(P) = \sum_{i=0}^{11} \pi^i(P) = \infty$  where  $\pi$  is the  $p$ -th power Frobenius).

# Type 2 and 3 Pairings from BN Curves

The fastest pairing known is the **R-Ate pairing**  $e_3 : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ :

$$e_3(P, Q) = \hat{e}(Q, P)^L \text{ for some fixed integer } L \text{ (with } n \nmid L\text{)}.$$

[Lee, Lee, Park; 2007]

It is a **Type 3** pairing because no efficiently computable isomorphism  $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$  is known.

---

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

Define  $e_2 : \mathbb{G}_1 \times \mathbb{G}'_2 \rightarrow \mathbb{G}_T$  by  $e_2(P, Q) = \hat{e}(Q, P)^{2L}$ .

Then  $e_2$  is an asymmetric pairing. It is a **Type 2** pairing because the Trace map is an efficiently-computable isomorphism from  $\mathbb{G}'_2$  to  $\mathbb{G}_1$ .

# $e_2$ or $e_3$ ?

**Functionality:** Some pairing-based protocols use  $\psi$  in the protocol itself. Can all these protocols be implemented with Type 3 pairings?

**Security:** Some protocols use  $\psi$  in their reductionist security proofs. Can these proofs be converted to the Type 3 setting?

**Efficiency:** Elements of  $\mathbb{G}'_2$  are defined over  $\mathbb{F}_{p^{12}}$  whereas elements of  $\mathbb{G}_2$  are defined over  $\mathbb{F}_{p^2}$ , so arithmetic in  $\mathbb{G}'_2$  is roughly 15 times as expensive as in  $\mathbb{G}_2$ .

How do the costs of computing  $e_2$  and  $e_3$  compare?

**Bandwidth:** Elements of  $\mathbb{G}'_2$  are 6 times larger than elements of  $\mathbb{G}_2$ .

# Computing Type 2 Pairings

Let  $P \in \mathbb{G}_1$  and  $Q \in \mathbb{G}'_2$ . We wish to compute  $e_2(P, Q)$ .

Define  $\hat{Q} = Q - \pi^6(Q)$ .

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

$$\begin{aligned} \text{Now, } e_2(P, Q) &= \hat{e}(Q, P)^{2L} \\ &= \hat{e}(2Q, P)^L \\ &= \hat{e}(Q + \hat{Q} + \pi^6(Q), P)^L \\ &= \hat{e}(\hat{Q}, P)^L \cdot \hat{e}(Q + \pi^6(Q), P)^L \\ &= e_3(P, \hat{Q}). \end{aligned}$$

Thus the task of computing  $e_2(P, Q)$  is easily reduced to the task of computing an R-ate pairing  $e_3(P, \hat{Q})$ .

# Defining $\mathbb{G}_1$ , $\mathbb{G}_2$ and $\mathbb{G}'_2$

- ▶ Let  $P'_2$  be an arbitrary point in  $E[n] \setminus (\mathbb{G}_1 \cup \mathbb{G}_2)$ , and set  $\mathbb{G}'_2 = \langle P'_2 \rangle$ .

- ▶ Define  $P_1 = \frac{1}{k} \text{Tr}(P'_2)$  so that the map

$$\psi : \mathbb{G}'_2 \rightarrow \mathbb{G}_1, \quad Q \mapsto \frac{1}{k} \text{Tr}(Q)$$

is an efficiently-computable isomorphism with  $\psi(P'_2) = P_1$ .

- ▶ Finally, set  $P_2 = c^{-1}(P'_2 - P_1)$  for an arbitrary integer  $c \in \mathbb{Z}_n^*$ .

- ▶ Then  $P_2 \in \mathbb{G}_2$  and the map

$$\rho : \mathbb{G}'_2 \rightarrow \mathbb{G}_2, \quad Q \mapsto Q - \psi(Q)$$

is an efficiently-computable isomorphism with  $\rho(P'_2) = cP_2$ .

# Efficient Representation for $\mathbb{G}'_2$

Given a point  $Q \in \mathbb{G}'_2$ , one can efficiently determine the unique points  $Q_1 \in \mathbb{G}_1$  and  $Q_2 \in \mathbb{G}_2$  such that  $Q = Q_1 + Q_2$ ; namely,  $Q_1 = \psi(Q)$  and  $Q_2 = \rho(Q) = Q - Q_1$ .

Writing  $D(Q) = (\psi(Q), \rho(Q))$ , and letting  $\mathbb{H}'_2 \subseteq \mathbb{G}_1 \times \mathbb{G}_2$  denote the range of  $D$ , we have an efficiently-computable isomorphism  $D : \mathbb{G}'_2 \rightarrow \mathbb{H}'_2$  whose inverse is also efficiently computable.

Hence, **without loss of generality**, points  $Q \in \mathbb{G}'_2$  can be represented by a pair of points  $(Q_1, Q_2)$  with  $Q_1 \in \mathbb{G}_1$  and  $Q_2 \in \mathbb{G}_2$ .

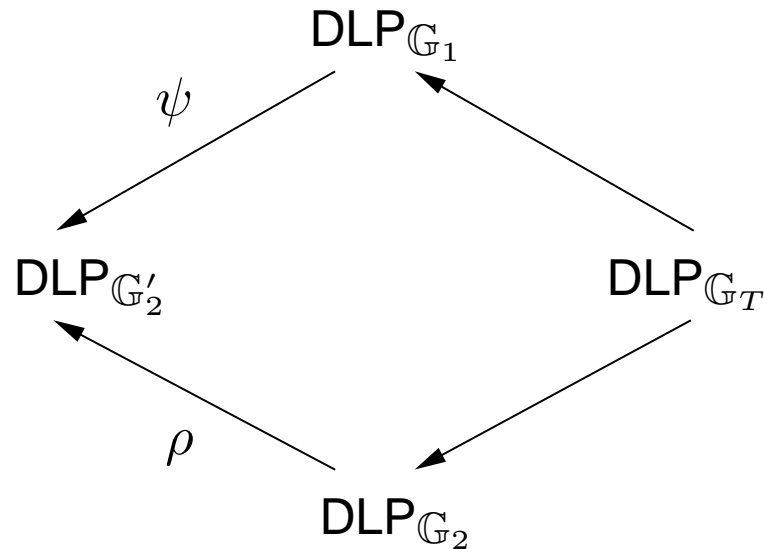
Arithmetic in  $\mathbb{G}'_2$  with this representation is component-wise.

**Bandwidth:**  $\mathbb{G}'_2$  elements are 1.5 times larger than  $\mathbb{G}_2$  elements.

**Efficiency:** Arithmetic in  $\mathbb{G}'_2$  is only 4/3 times as expensive as in  $\mathbb{G}_2$ .

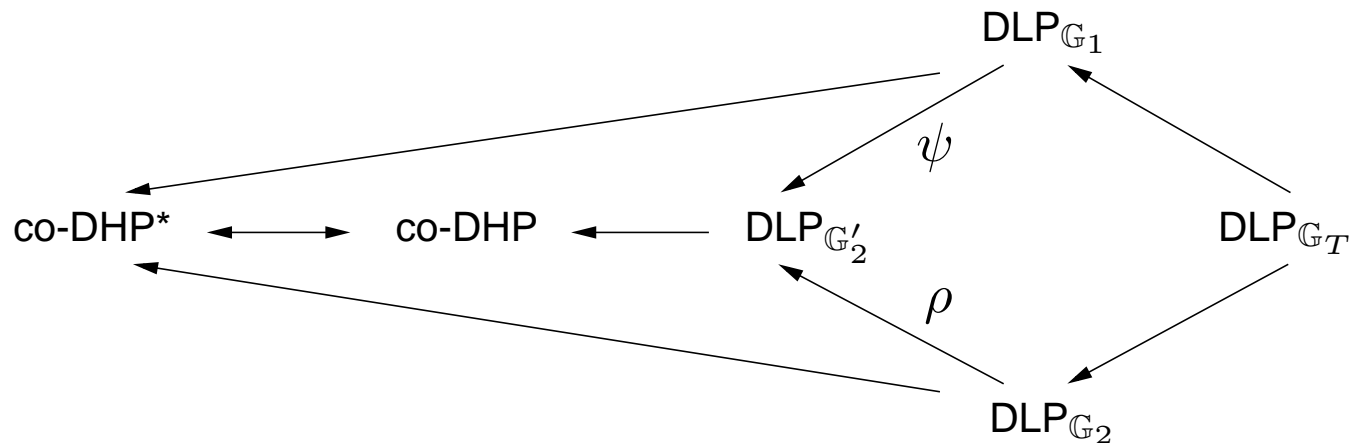
# Type 2 versus Type 3: Security

Known relationships between the DLP in  $\mathbb{G}_1$ ,  $\mathbb{G}_2$ ,  $\mathbb{G}'_2$  and  $\mathbb{G}_T$ .



Reductions in the opposite directions are not known.

# Type 2 versus Type 3: Security



**co-DHP:** Given  $Q \in \mathbb{G}_1$  and  $zP'_2 \in \mathbb{G}'_2$ , compute  $zQ$ .

**co-DHP\*:** Given  $Q, zP_1 \in \mathbb{G}_1$  and  $zP_2 \in \mathbb{G}_2$ , compute  $zQ$ .

**Fact:** If  $c$  is known, then  $\text{co-DHP} \leq \text{co-DHP}^*$  and  $\text{co-DHP}^* \leq \text{co-DHP}$ .  
(So  $\text{co-DHP}$  and  $\text{co-DHP}^*$  are provably equivalent.)

If  $c$  is unknown, then  $\text{co-DHP}$  and  $\text{co-DHP}^*$  appear to be unrelated.

# BLS-2 versus BLS-3

Let  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  be either  $e_2$  or  $e_3$ .

1. **Key generation.** Alice does:

Private key:  $x \in_R [1, n - 1]$ ; Public key:  $X = xP_2 \in \mathbb{G}_2$ .

2. **Signature generation.** To sign  $M$ , Alice does:

Compute  $H = \text{Hash}(M)$ , where  $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$ .

Compute  $\sigma = xH \in \mathbb{G}_1$ .

The signature on  $M$  is  $\sigma$ .

3. **Signature verification.** To verify  $(M, \sigma)$ , Bob does:

Compute  $H = \text{Hash}(M)$ .

Accept iff  $e(\sigma, P_2) = e(H, X)$ .

$$[e_2(H, X) = e_3(H, X - \pi^6(X)) = e_3(H, 2X_2)]$$

The only difference between BLS-2 and BLS-3 is that the public key  $X$  in the former is slightly larger (because it includes an extra  $\mathbb{G}_1$  component). This component is not used in signature gen. or ver. so it can be dropped – then BLS-2 and BLS-3 are identical.

# More Generally.....

We have arguments in support of the following claims:

Given any protocol, Protocol-2, described using a Type-2 pairing, and a security proof for Protocol-2 with respect to some problem  $\mathcal{P}$ -2, there is a natural transformation of Protocol-2 to a Protocol-3 that uses a Type-3 pairing, a natural transformation of  $\mathcal{P}$ -2 to  $\mathcal{P}$ -3, and a natural transformation of the security proof to one for Protocol-3 with respect to  $\mathcal{P}$ -3.

Moreover, Protocol-3 is at least as efficient as Protocol-2, and  $\mathcal{P}$ -3 is equally as hard as  $\mathcal{P}$ -2 (for appropriately chosen parameters).

In other words,  $\psi$  does not play any cryptographically significant role and hence there is no reason to use Protocol-2 instead of Protocol-3.

# Waters-1 Signature Scheme

[Waters; 2005]

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

Let  $e_1 : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  be a symmetric pairing with  $\mathbb{G} = \langle P \rangle$ , and let  $k$  denote the security parameter.

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

Let  $U_0, U_1, \dots, U_k$  be randomly selected elements of  $\mathbb{G}$ . Denote  $U = (U_0, U_1, \dots, U_k)$ .

Define a hash function  $\text{Hash} : \{0, 1\}^* \rightarrow \mathbb{G}$  as follows:

Let  $H_1(M) = (m_1, m_2, \dots, m_k)$  (where each  $m_i \in \{0, 1\}$ ).

Then  $\text{Hash}(M) = U_0 + \sum m_i U_i$ .

# Waters-1 Signature Scheme

1. **Key generation.** Alice does:  
Private key  $Z = zP$ ; Public key:  $\zeta = e_1(P, P)^z$ .
2. **Signature generation.** To sign  $M$ , Alice does:  
Compute  $H = \text{Hash}(M)$ , and select  $r \in_R [1, n - 1]$ .  
Compute  $\alpha = Z + rH$  and  $\beta = rP$ .  
The signature on  $M$  is  $\sigma = (\alpha, \beta)$ .
3. **Signature verification.** To verify  $(M, \sigma)$ , Bob does:  
Compute  $H = \text{Hash}(M)$ .  
Accept iff  $e_1(\alpha, P) = \zeta \cdot e_1(\beta, H)$ .

## Correctness:

$$e_1(\alpha, P) = e_1(Z + rH, P) = e_1(Z, P) \cdot e_1(rH, P) = e_1(P, P)^z \cdot e_1(\beta, H).$$

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

# Hash Function Parameters $U$

The public parameters  $U_0, U_1, \dots, U_k$  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  $u_i = \log_P U_i$ , then that party could recover Alice's private key  $Z$  given a single signed message  $(M, \sigma)$  as follows:

1. Compute  $H_1(M) = (m_1, m_2, \dots, m_k)$ .
2. Compute  $c = u_0 + \sum m_i u_i \pmod n$ .

[Then  $H = \text{Hash}(M) = U_0 + \sum m_i U_i = (u_0 + \sum m_i u_i)P = cP$ .]

3. Compute  $Z = \alpha - c\beta$ . [Recall:  $\alpha = Z + rH$ ,  $\beta = rP$ ]

# Waters-3a Signature Scheme

Let  $e_3 : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  be a Type 3 pairing.

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

1. **Key generation.** Alice does:

Private key  $Z = zP_1$ ; Public key:  $\zeta = e(P_1, P_2)^z$ .

2. **Signature generation.** To sign  $M$ , Alice does:

Compute  $H' = \psi(\text{Hash}(M))$ , and select  $r \in_R [1, n - 1]$ .

Compute  $\alpha = Z + rH'$  and  $\beta = rP_1$ .

The signature on  $M$  is  $\sigma = (\alpha, \beta)$ .

3. **Signature verification.** To verify  $(M, \sigma)$ , Bob does:

Compute  $h = \text{Hash}(M)$ , and accept iff  $e_2(\alpha, P_2) = \zeta \cdot e(\beta, H)$ .

**Problem:** There is no  $\psi$  for a Type 3 pairing.

**Solution:** Each user selects  $u_i \in_R [0, n - 1]$  and computes  $U_i = u_i P_2$  and  $V_i = u_i P_1$ . The user's public key includes  $U = (U_0, U_1, \dots, U_k)$ , while  $V = (V_0, V_1, \dots, V_k)$  is kept private. [Then  $H' = V_0 + \sum m_i V_i$ ]

# Waters-3a Signature Scheme

**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.

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

# Waters-3b Signature Scheme

**Case b:** 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  $Z = zP_1$ ; Public key:  $\zeta = e(g_1, g_2)^z$ .
2. **Signature generation.** To sign  $M$ , Alice does:  
Compute  $H = \text{Hash}(M)$ , and select  $r \in_R [1, n - 1]$ .  
Compute  $\alpha = Z + rH$ ,  $\beta = rP_2$ , and  $\gamma = rP_1$ .  
The signature on  $M$  is  $\sigma = (\alpha, \beta, \gamma)$ .
3. **Signature verification.** To verify  $(M, \sigma)$ , Bob does:  
Compute  $H = \text{Hash}(M)$ , and accept iff  
 $e(\alpha, P_2) = \zeta \cdot e(H, \beta)$  and  $e(\gamma, P_2) = e(P_1, \beta)$ .

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

Also signature generation and verification are more expensive.

# Waters-3b Signature Scheme

Why is the extra signature component  $\gamma$  needed?

In the security reduction, when the attacker returns a forgery  $M, (\alpha, \beta, \gamma)$ , we have  $\alpha = Z + rH \in \mathbb{G}_1$ ,  $\beta = rP_2 \in \mathbb{G}_2$ , and  $\gamma = rP_1 \in \mathbb{G}_1$ . So, the solver can compute  $Z = \alpha - h\gamma$ , where  $H = hP_1$ .

**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.

# BLS versus Waters

Estimates for a particular BN curve:

|                   | BLS-2      | BLS-3      | Waters-3a  | Waters-3b  |
|-------------------|------------|------------|------------|------------|
| Security          | ROM        | ROM        | coll-res   | coll-res   |
|                   | co-DHP     | co-DHP*    | co-DHP*    | co-DHP*    |
| Public key size   | 770        | 513        | 32.1KB     | 1,024      |
| Signature size    | 257        | 257        | 514        | 1,027      |
| Sig. generation   | 1,848 $m$  | 1,848 $m$  | 1,447 $m$  | 5,576 $m$  |
| Sig. verification | 22,342 $m$ | 22,027 $m$ | 25,193 $m$ | 47,120 $m$ |

ROM: Random oracle model

$m$ :  $\mathbb{F}_p$  multiplication

# Conclusions

The map  $\psi$  does not appear to play any cryptographically significant role in pairing-based protocols.

**Question:** Is there a (natural) protocol in the Type 2 setting that has no counterpart in the Type 3 (and Type 1) settings?

When converting a protocol from the Type 1 setting to the Type 3 setting, some tradeoffs may have to be made that can impact performance.

Protocol designers who are interested in the performance of their protocols should describe and analyze their protocols using Type 3 pairings.