Short Signatures from the Weil Pairing

Ryan Gibson
Motivation for Short Signatures

• Some domains have strong bandwidth constraints where short, secure digital signatures are desirable
  - Bar-codes, product keys, etc.

• Common signature schemes produce long signatures. For $\sim 80$ bits of security (i.e. taking $2^{80}$ operations to break)
  - RSA needs a 1024-bit modulus $\rightarrow$ 1024 bit signatures
  - DSA needs a 1024-bit modulus $\rightarrow$ 320 bit signatures
  - “A 320-bit signature is too long to be keyed in by a human”
    • $uq\&%=!j`+<z-cHC1DGi8fq=-a?!OY+m}_U`DB-ljw+WG(4Pu<p$
That is,

- $G_1$ is generated by $g_1$, $G_2$ is generated by $g_2$
- $\psi : G_2 \rightarrow G_1$ is an isomorphism with $\psi(g_2) = g_1$
- That is, $\forall a, b \in G_2$, $\psi(a \cdot b) = \psi(a) \cdot \psi(b)$
- $e : G_1 \times G_2 \rightarrow G_T$ is a bilinear map where $|G_1| = |G_2| = |G_T|$
  - That is,
    - **Bilinear**: $\forall u \in G_1$, $v \in G_2$, $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$
    - **Nondegenerate**: $e(g_1, g_2) \neq 1$
Cryptography Background/Notation

- With \((G_1, G_2)\), we have natural generalizations of CDH, DDH:

- Computational co-Diffie-Hellman (co-CDH):
  - Given \(g_2, g_2^a \in G_2, h \in G_1\), compute \(h^a \in G_1\)

- Decision co-Diffie-Hellman (co-DDH):
  - Given \(g_2, g_2^a \in G_2, h, h^b \in G_1\), determine if \(a = b\)

- When \(G_1 = G_2\), these are the same as standard CDH, DDH
Experiments for co-CDH

Experiment $\text{Expt}_{G_1,G_2,g_1,g_2}^{\text{co-cdh}}(A)$

\begin{align*}
    a & \leftarrow_R \mathbb{Z}_p \\
    h & \leftarrow_R G_1 \\
    X & \leftarrow A(g_2, g_2^a, h) \\
    \text{if } h^a = X & \text{ then return 1 else return 0}
\end{align*}

$\text{Adv}_{G_1,G_2,g_1,g_2}^{\text{co-cdh}}(A) = \Pr \left[ \text{Expt}_{G_1,G_2,g_1,g_2}^{\text{co-cdh}}(A) = 1 \right]$

$\text{Adv}_{G_1,G_2,g_1,g_2}^{\text{co-cdh}}(t) = \max_A \left\{ \text{Adv}_{G_1,G_2,g_1,g_2}^{\text{co-cdh}}(A) \right\}$

where the max is taken over all adversaries running in time $t$
Experiments for co-DDH

Experiment $\text{Expt}_{G_1, G_2, g_1, g_2}^{\text{co-ddh-1}}(A)$

\[
\begin{align*}
    a & \leftarrow_R \mathbb{Z}_p \\
    h & \leftarrow_R G_1 \\
    b' & \leftarrow A(g_2, g_2^a, h, h^a) \\
    \text{if } b' = 1 \text{ then return } 1 \text{ else return } 0
\end{align*}
\]

Experiment $\text{Expt}_{G_1, G_2, g_1, g_2}^{\text{co-ddh-0}}(A)$

\[
\begin{align*}
    a & \leftarrow_R \mathbb{Z}_p; b \leftarrow_R \mathbb{Z}_p \\
    h & \leftarrow_R G_1 \\
    b' & \leftarrow A(g_2, g_2^a, h, h^b) \\
    \text{if } b' = 0 \text{ then return } 1 \text{ else return } 0
\end{align*}
\]

\[
\text{Adv}_{G_1, G_2, g_1, g_2}^{\text{co-ddh}}(A) = \Pr \left[ \text{Expt}_{G_1, G_2, g_1, g_2}^{\text{co-ddh-1}}(A) = 1 \right] - \Pr \left[ \text{Expt}_{G_1, G_2, g_1, g_2}^{\text{co-ddh-0}}(A) = 1 \right]
\]

\[
\text{Adv}_{G_1, G_2, g_1, g_2}(t) = \max_A \left\{ \text{Adv}_{G_1, G_2, g_1, g_2}^{\text{co-ddh}}(A) \right\}
\]

where the max is taken over all adversaries running in time $t$
Gap co-Diffie-Hellman group pairs

- We will be interested in pairs of groups \((G_1, G_2)\) where co-DDH is “easy” but co-CDH is intractable.

- Formally, two groups \((G_1, G_2)\) are a \((\tau, t, \varepsilon)\)-co-GDH group pair if:
  - The group operations on \(G_1\) and \(G_2\) and the map \(\psi\) between the groups can be computed in time at most \(\tau\).
  - In \((G_1, G_2)\), the co-DDH problem can be solved in time at most \(\tau\).
  - \(\text{Adv}^{\text{co-cdh}}_{G_1, G_2, g_1, g_2}(t) < \varepsilon\).

- We want \(\tau\) to be small and \(t/\varepsilon\) to be large.
Why co-GDH group pairs?

- When co-DDH is “easy” and co-CDH is “hard”, given \( g_2, g_2^a \in G_2, h \in G_1 \)
  - It is easy to determine if a group element is \( h^a \in G_1 \), but
  - It is hard to compute \( h^a \in G_1 \)

- This is exactly what we want from a “easy-to-verify, hard-to-forge” signature scheme!
Why bilinear maps?

- The only known examples of GDH groups arise from bilinear maps!

- Let \( e : G_1 \times G_2 \rightarrow G_T \) be an efficiently computable bilinear map

- Then, for any co-DDH tuple \( (g_2, g_2^a, h, h^b) \),

\[
e(h, g_2^a) = e(h^b, g_2) \iff e(h, g_2)^a = e(h, g_2)^b \iff a = b \mod p
\]
A Signature Scheme from GDH Groups

• Let \((G_1, G_2)\) be a co-GDH group pair with \(|G_1| = |G_2| = p\)
• Let \(H : \{0, 1\}^* \rightarrow G_1\) be a random oracle
• Consider the following signature scheme \(\mathcal{DS} = \langle \mathcal{K}, S, V \rangle\)

\[\begin{array}{ccc}
\text{Algorithm } \mathcal{K}() & \text{Algorithm } S_x(M) & \text{Algorithm } V_v(M, \sigma) \\
\begin{align*}
x & \leftarrow_R \mathbb{Z}_p \\
v & \leftarrow g_2^x \\
\text{return } (v, x)
\end{align*} & \begin{align*}
h & \leftarrow H(M) \\
\sigma & \leftarrow h^x \\
\text{return } \sigma
\end{align*} & \begin{align*}
h & \leftarrow H(M) \\
\text{if } (g_2, v, h, \sigma) \text{ is a co-DH tuple} \\
\text{return } 1 \\
\text{else} \\
\text{return } 0
\end{align*}
\end{array}\]

• A signature is a single group element!
Proof of Security

• Proposition: Let $\mathcal{DS} = \langle \mathcal{K}, \mathcal{S}, \mathcal{V} \rangle$ be the signature scheme as previously described. Then,

$$\text{Adv}_{\mathcal{DS}}^{\text{uf-cma}}(t, q_s, q_H) \leq e(q_s + 1) \cdot \text{Adv}_{G_1,G_2,g_1,g_2}^{\text{co-cdh}}(t')$$

• where $t' = t + (q_H + 2q_s) \cdot (\text{time for an exponentiation in } G_1)$

• Proof: Let $A$ be a uf-cma adversary attacking $\mathcal{DS}$ (in the random oracle model), running in time $t$, making at most $q_s$ signature queries, and at most $q_H$ hash queries

• We’ll construct a co-CDH adversary $B$, running in time $t'$, with the desired advantage
Proof of Security (Adversary Summary)

- co-CDH Adversary $B$
  - Given: $g_2 \in G_2$, $u = g_2^a \in G_2$, $h \in G_1$
  - Goal: compute $h^a \in G_1$
  - Has access to $A$, but no oracles

- uf-cma Adversary $A$
  - Given: public key $v$
  - Goal: forge valid, unqueried $(M, \sigma)$ pair
  - Has access to signing/hashing oracles: $S_x(\cdot)$, $H(\cdot)$
Proof of Security

- The basic strategy is to faithfully implement $H(M_i)$ in such a way that $B$ knows how to compute $S_x(M_i)$

- However, we need to respond differently for one “special” query
  - Set up one query so that a forgery for this $M_i$ will give us $h^a$
  - Hope that $B$ actually forges a signature for this message since we’ll have to fail and terminate otherwise (we can’t respond to a $S_x(M_i)$ query here)

- It is nontrivial to get both types of queries working for the same public key $g_2^x$
Proof of Security

• Randomly choose \( r \leftarrow R \mathbb{Z}_p \) and give \( B \) the public key \( u \cdot g_2^r = g_2^{a+r} \in G_2 \)

• For “normal” queries:

  Algorithm \( H(M_i) \)
  
  If \( M_i \) has not already been queried
  
  \[ b_i \leftarrow R \mathbb{Z}_p \]
  
  \[ w_i \leftarrow \psi(g_2)^{b_i} \]
  
  store \( \langle M_i, w_i, b_i \rangle \)
  
  return \( w_i \)

  else
  
  return \( w_i \) from last query

  Algorithm \( S_x(M_i) \)
  
  \[ \sigma_i \leftarrow \psi(u)^{b_i} \cdot \psi(g_2)^{rb_i} \]
  
  return \( \sigma_i \)

This signature is valid since

\[
\psi(u)^{b_i} \cdot \psi(g_2)^{rb_i} = \psi\left(g_2^a\right)^{b_i} \cdot \psi(g_2)^{rb_i} = \psi(g_2)^{b_i(a+r)} = w_i^{a+r}
\]
Proof of Security

- For “special” queries, just add \( h \) onto the result of \( H(M_i) \)

\[
\begin{align*}
\text{Algorithm } H(M_i) & \\
& b_i \leftarrow_R \mathbb{Z}_p \\
w_i & \leftarrow h \cdot \psi(g_2)^{b_i} \\
\text{store } \langle M_i, w_i, b_i \rangle \\
& \text{return } w_i
\end{align*}
\]

\[
\begin{align*}
\text{Algorithm } S_x(M_i) & \\
& \text{fail and terminate}
\end{align*}
\]

- If \( B \) successfully forges a \( (M_i, \sigma_i) \) pair here, we know

\[
\sigma_i = H(M_i)^{a+r} = (h \cdot \psi(g_2)^{b_i})^{a+r} = h^a \cdot h^r \cdot \psi(g_2)^{ab_i} \cdot \psi(g_2)^{rb_i}
\]

\[
= h^a \cdot h^r \cdot \psi(u)^{b_i} \cdot \psi(g_2)^{rb_i}
\]

- so \( A \) recovers \( h^a = \frac{\sigma_i}{h^r \cdot \psi(u)^{b_i} \cdot \psi(g_2)^{rb_i}} \)
The reduction in the paper actually uses random “coin flips” to determine if each query should be treated as a “normal” or “special” one.

Each query is “special” with probability \(1/(q_s + 1)\).

Otherwise, the analysis is identical.
Proof of Security

- For $B$ to succeed, we need three events to occur:
  - $\mathcal{E}_1$: $B$ does not abort in any of $A$’s signature queries
  - $\mathcal{E}_2$: $A$ successfully forges an (unqueried) $(M, \sigma)$ pair
  - $\mathcal{E}_3$: $A$ “chooses” to forge our “special” query

- Since $H(\cdot)$ gets called $q_s$ times, we have

\[
\Pr[\mathcal{E}_1] = \left(1 - \frac{1}{q_s + 1}\right)^{q_s} \geq \frac{1}{e}
\]

\[
\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] = \text{Adv}_{DS}^{\text{uf-cma}}(A)
\]

\[
\Pr[\mathcal{E}_3 \mid \mathcal{E}_1 \land \mathcal{E}_2] = \frac{1}{q_s + 1}
\]
Proof of Security

• Finally, this yields

\[ \Pr[\mathcal{E}_1 \land \mathcal{E}_2 \land \mathcal{E}_3] = \text{Adv}_{G_1, G_2, g_1, g_2}^{\text{co-cdh}}(B) \]

\[ \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \Pr[\mathcal{E}_3 \mid \mathcal{E}_1 \land \mathcal{E}_2] = \text{Adv}_{G_1, G_2, g_1, g_2}^{\text{co-cdh}}(B) \]

\[ \frac{1}{e} \cdot \text{Adv}_{\text{DS}}^{\text{uf-cma}}(A) \cdot \frac{1}{q_s + 1} \leq \text{Adv}_{G_1, G_2, g_1, g_2}^{\text{co-cdh}}(B) \]

\[ \text{Adv}_{\text{DS}}^{\text{uf-cma}}(A) \leq e(q_s + 1) \cdot \text{Adv}_{G_1, G_2, g_1, g_2}^{\text{co-cdh}}(B) \]

• which completes the proof
The Weil pairing is a bilinear map on groups arising from elliptic curves. This gives us the co-GDH group pairs needed for the signature scheme.

Signature generation in this scheme is faster than RSA, but verification is slower than RSA.

- DSA signatures required two elements of $\mathbb{Z}_q$
- These signatures require only one element of $\mathbb{Z}_p$
  - The authors explicitly construct many “secure” elliptic curves, one of which provides 171-bit signatures that have the security of 320-bit DSA signatures.
  - They also provide details on the types of elliptic curves that would provide signatures that are always half the size of DSA, but it is an open problem whether or not such curves can be constructed.
The Other Half of the Paper

The authors briefly describe extensions of the signature scheme:

- An aggregation scheme where many message, signature pairs from distinct parties (each with their own public/private keys) can be combined into a single short signature.

- A method in which many signatures on the same message (each from a party with distinct public/private keys) can be verified “as a batch” – i.e. much faster than verifying them one-by-one.

- A scheme in which many parties possess a share of a private key and can only sign messages when a certain fraction of parties agree to do so.