Next up: homomorphic signatures

Define unforgeability: adversary

\[ \sigma = \text{Sign}(vk, x) \]

\[
\begin{align*}
\sigma & \rightarrow \sigma_f \\
x, \sigma_f & \rightarrow y = f(x) \\
\sigma_y \leftarrow \text{Eval}(f, x, \sigma) \\
\end{align*}
\]

\[ \downarrow \]

Check that \( \sigma_y \) is a signature on \( y \) with respect to function \( f \).

- Can view as signature on pair \( (f, y) \) \( \leftarrow \) Why not just on \( y \) alone?

Requirements:

- Unforgeability: Cannot construct signature \( \sigma \) on \( (f, y) \) where \( y \neq f(x) \).
- Succinctness: Size of \( \sigma_y \) should be \( \|y\| \cdot \text{poly}(\lambda) \). In particular, should not depend on \( \|x\| \) or \( \|f\| \).
- Efficient verification: Can decompose verification algorithm as follows: \( \downarrow \)

- Preprocess \( (vk, f) \rightarrow vk_f \) (generates short function verification key \( vk_f = (vk, \text{poly}(\lambda, \|f\|)) \))
- Verify \( (vk_f, y, \sigma) \rightarrow \sigma/1 \) (runs in time \( \text{poly}(\lambda, d, \|y\|) \))

Homomorphic signatures allow computations on authenticated data.

Construct: relies on similar homomorphic structure as GSW (for message space \( \text{poly}(\lambda) \))

- KeyGen(\( \lambda \)):
  Set lattice parameters \( n = n(\lambda), g = g(\lambda) \).
  Sample \( (A, T) \leftarrow \text{TrapGen}(n, g) \)
  Sample \( B_1, \ldots, B_k \leftarrow \mathbb{Z}_g^\times \)
  Output \( vk = (A, B_1, \ldots, B_k), sk = R \)

- Sign(\( sk, x \)):
  Compute \( R_i \leftarrow A^i(B_i - x \cdot G) \) for \( i \in [k] \) using \( T \)
  In particular:
  \[
  A[R_1, \ldots, R_k] = [B_1 - x, G \ldots B_k - x \cdot G] \\
  = [B_1 \ldots B_k] - x \cdot G
  \]
  Output \( \sigma = (R_1, \ldots, R_k) \)

- Verify(\( vk, x, \sigma \)):
  Check that \( \|R_i\| \leq B \) and that \( A[R_1, \ldots, R_k] = [B_1 \ldots B_k] - x \cdot G \)
  \[ \downarrow \]
  Bound based on quality of trapdoor (lattice parameters)
Homomorphic evaluation:

\[ A[R_1, \ldots, R_k] = [B_i - x_i G, \ldots, B_e - x_e G] \]

To derive a signature on the sum of two bits \((x_i + x_j)\):

\[
\begin{align*}
R_i &= R_i + R_j \\
B_i &= B_i + B_j
\end{align*}
\]

Verification: \(AR_i = B_i - (x_i + x_j) G\) addition operation

To derive a signature on the product of two bits \((x_i x_j)\):

\[
\begin{align*}
AR_i &= B_i - x_i G \\
AR_j &= B_j - x_j G
\end{align*}
\]

AR \(x\) is a new verification component associated with

\[
AR(x) = B_x - x_i x_j G
\]

function of \(R_i, R_j\)

\[
\begin{align*}
AR_i &= B_i - x_i G \\
AR_j &= B_j - x_j G \\
AR_i G^{-1}(B_i) &= (B_i - x_i G) G^{-1}(B_i)
\end{align*}
\]

\[
B_j G^{-1}(B_i) = x_j B_i
\]

\[
B_j G^{-1}(B_i) - x_j B_i
\]

\[
= B_j G^{-1}(B_i) - A(x_j) B_i - x_i x_j G
\]

\[
\Rightarrow a(R_j G^{-1}(B_i) + x_j R_i) = B_j G^{-1}(B_i) - x_i x_j G
\]

\[
R_x = R_j G^{-1}(B_i) + x_j R_i
\]

\[
B_x = B_j G^{-1}(B_i)
\]

function of signature, input

\[ \|R_x\|_o \leq \|R_j\|_o + \|R_i\|_o \]

(\(h_1: \mathbb{Z}_q\) homomorphic multiplication)

Can depend on \(R_i, R_j, x\)

Small linear function of \(R_i\) and \(R_j\)

Composition to compute signature on \(R_{x,x}\) on evaluation \(f(x)\)

By above analysis, multiplication scales noise by a factor of \(t\) so if \(f\) can be computed by a circuit of depth \(d\), \(\|R_{x,x}\|_o \leq t^{O(d)}\)

To verify a signature \(R_{x,x}\) on \((f(x), y = f(x))\), verifier computes \(B_j\) from \(B_i, \ldots, B_e\) and checks that \(\|R_{x,x}\|_o \leq t^{O(d)}\)

More generally:

\[ R_{x,x} = [R_1, \ldots, R_k] H_{f,x} \]

where \(H_{f,x} \in \mathbb{Z}_q^{2^{k+1}}\) and \(\|R_{x,x}\|_o \leq t^{O(d)} = (n \log q)^{O(d)}\)

where \(d\) is the (multiplicative) depth of the circuit computing \(f\)

Now, if \(AR_i = B_i - x_i G\), then from the above,

\[ AR_{x,x} = B_x - f(x) G \]

where \(B_j\) is the matrix obtained by evaluating \(f\) on \(B_i, \ldots, B_e\)

This can be expanded as

\[
AR_{x,x} = A[R_1, \ldots, R_k] H_{f,x} = [B_i - x_i G, \ldots, B_e - x_e G] H_{f,x}
\]

\[ = B_x - f(x) G \]
Decouple into two equations:
- Input-independent evaluation: \([B_1 \cdots B_k]: H_y = B_y\)
- Input-dependent evaluation: \([B_i - x_i G : \cdots : B_k - x_k G] H_{f,x} = B_y - f(x) : G\)

Unforgeability: Will consider a weaker (selective) notion of security where the message that is signed is independent of the verification key [not difficult to get full adaptive security, but somewhat tedious]

---

Unforgeability: Will consider a weaker (selective) notion of security where the message that is signed is independent of the verification key [not difficult to get full adaptive security, but somewhat tedious]

\[
\begin{aligned}
\text{challenge} & : \chi \\
\text{adversary} & : (vk, sk) \leftarrow \text{Key Gen}(1^k) \\
& : \sigma_x \leftarrow \text{Sign}(sk, \chi) \\
& : f, y, \sigma_{f,y} \leftarrow \text{Output} 1 \text{ if } y \neq f(x) \text{ and } V_{vk_f}(vk, f) \\
\end{aligned}
\]

Proof of unforgeability.

Observe: If correctly simulates verification key by LH1, suppose A succeeds: then \(AR_x = B_y - y : G\)
\(AR_x = B_y - f(x) : G\)
\(\Rightarrow A(R-R^*) = (f(x)-y) : G\)
\[d(x) \neq y \Rightarrow f(x)-y \in \{-1,1\}\]

\(R^*\) is short since signature verifies \(R^*\) is short since \(R_1, H_{f, x}\) are small

R is short since signature verifies \(R-R^*\) is a trapdoor for A
Context-hiding for homomorphic signatures:

- In many settings, we also want the computed signature to hide information about the input to the computation.

\[
\text{Alice} \xrightarrow{\chi, \sigma} \text{Server} \xrightarrow{f} \text{Bob}
\]

Bob wants to check signature on \( y = f(x) \) but should not learn anything about \( x \).

We will see one application of this type of property to (designated-prover) NIZKs.

We say a homomorphic signature scheme is context-hiding if there exists an efficient simulator \( S \) where for all \((vk, sk) = \text{KeyGen}(1^n)\), \( x \in \{0,1\}^n \), and \( f: \{0,1\}^n \rightarrow \{0,1\} \):

\[
\{ vk, \text{Eval}(vk, f, \sigma) \} \approx \{ vk, S(sk, vk, f, f(x)) \}
\]

\(\leftarrow\) simulator needs to simulate valid signatures so it needs to know the signing key; however, it does not know the input \( x \), only the value \( f(x) \).

\(\rightarrow\) this means signature reveals no information about \( x \) other than \((f, f(x))\).

Current construction is not context-hiding:

\[
R_{f,x} := [R_1 \ldots R_k] \cdot H_{f,x}
\]

\(\leftarrow\) this is a function of \( x \)!

To achieve context-hiding, we need a way to re-randomize a signature.

Suppose \( AR_{f,x} = By - y \cdot G \) where \( y \in \{0,1\} \).

Evaluator knows \( y \) so it can compute the matrix

\[
V := [A \mid By + (y-1) \cdot G] = [A \mid AR_{f,x} + (2y-1) \cdot G]
\]

Now, since \( y \in \{0,1\}, 2y-1 \in \{-1,1\} \). Then \( R_{f,x} \) is a trapdoor for \( V \):

\[
V \cdot \begin{bmatrix} -R_{f,x} \\ I \end{bmatrix} = (2y-1) \cdot G = G - G
\]

The public key then includes a random target \( z \in \mathbb{Z}_q^n \) and the signature is formed by sampling a short vector \( t \) such that \( Vt = z \):

\[
t \leftarrow V^{-1}(z) \text{ using trapdoor } [-R_{f,x}]
\]

To verify a signature, the verifier computes \( By \) from \( B_1, \ldots, B_k \) constructs \( V \) from the verification key and checks that \( Vt = z \) and \( \|t\|_\infty \leq \beta \) where \( \beta = (n \log q)^{o(1)} \) is the noise bound.

\(\leftarrow\) quality of trapdoor is \( \| [-R_{f,x}] \|_1 \) which is \( (n \log q)^{o(1)} \) so norm bound is also \( (n \log q)^{o(1)} \).
Recap:

**Homomorphic Encryption**

- **Public Key**: \( A = \left[ \frac{A}{s^T A + e^T} \right] \)
- **Ciphertext**: \( C = AR + \mu \cdot G \)

**Homomorphic Signatures**

- **Private Key**: \( A \leftarrow \mathbb{Z}_p \)
- **Signature**: \( AR = B - \mu \cdot G \)

**Ciphertext Evaluation**

\( C_1, \ldots, C_e, f \mapsto C_f \)

\( [C_1 - x, 1 \ldots C_e - x_e \cdot G] H_y = C_f - f(x) \cdot G \)

\( A [R_1 \ldots 1R_e] H_y \mapsto [R_1 \ldots 1R_e] H_y x = R_y x \)

**Homomorphic on Randomness**

GSW homomorphisms are homomorphic on both messages and on randomness.

**Signature Evaluation**

\( C_f = AR_f + f(x) \cdot G \)

\( C \) homomorphic on randomness

**Verification**

He: cipher text evaluation

He: signature evaluation

He: verification
Another view: We can view GSW/homomorphic signatures as homomorphic commitment scheme:

$$ \mathbf{pp} : \mathbf{A} \in \mathbb{Z}_{}^{m \times n} $$

to commit to a message \( x \in \{0,1\}^n \), sample \( R \in \mathbb{Z}_{}^m \) and output \( C \leftarrow AR + x G \)

to open a commitment to message \( \mu \), reveal \( R \), and check that

$$ C = AR + \mu G \quad \text{and} \quad \| R \|_2 < \beta \quad \text{(for some noise bound \( \beta \))} $$

Observe: commitment is just GSW ciphertext, so supports arbitrary computation

$$ C = AR + \mu G \quad \Rightarrow \quad C_j = AR_{j,x} + f(x) \cdot G $$

where \( R_{j,x} = [R_1, \ldots, R_L] \cdot H_{j,x} \)

To see this, sample \( (A, T) \leftarrow \text{TrapGen}(n, g) \). Then \( A \) is statistically close to uniform.

To generate opening for commitment \( C \) to message \( \mu \in \{0,1\}^n \),

$$ R \leftarrow \text{SamplePre}(A, T, C - \mu G, \varepsilon) $$

This yields short \( R \) where

$$ \text{AR} = C - \mu G \quad \Rightarrow \quad C = AR + \mu G $$

Succinct homomorphic commitments (i.e., functional commitments):

Commitment to \( x \): \( C_i = AR_i + x_i G \)

\( C_L \) grows with the input length \( L \)

$$ C_L = AR_L + x_L G $$

Can we compress further? Yes, but will need a stronger assumption.

\( l \)-succinct SIS: SIS with respect to \( A \in \mathbb{Z}_{}^{m \times n} \) holds even given a trapdoor for the related matrix

$$ B = \left[ \begin{array}{cc|c} A & \cdots & W_1 \\ A & \cdots & W_2 \\ \vdots & \ddots & \vdots \\ A & \cdots & W_l \end{array} \right] \quad \text{where} \quad W_i \in \mathbb{Z}_{}^{n \times t} \)