Lesson 13 Lecture Notes

CS555

1 Recap

OWF $\Rightarrow$ PRG + PRF $\Rightarrow$ Stateless Symmetric-Key Encryption $\Rightarrow$ Stateless Public-Key Encryption (more costly)

1.1 Message Authentication Codes (MACs)

- Bob can ensure a message came from Alice.
- Tag is created and added using a shared secret key.
- Example: $\text{MAC}(sk, m) = f_s(m)$ where f is a PRF from family F_l.

1.2 Digital Signatures

- Public-key analog of MAC.
- Anyone can verify the source using:

$$(m, \sigma) \leftarrow \text{Sign}(sk, m), \text{ then } \text{Verify}(pk, m, \sigma)$$

.

- **Signature:** Requires n key pairs for n people
- **Properties:**
  - Publicly verifiable
  - Transferable
  - Non-repudiation
- **MAC:** Requires n^2 keys (every pair needs a unique secret key).
  - Privately verifiable
  - Not transferable $\rightarrow$ new tag for each new person
  - No non-repudiation

1.3 Applications of Signatures

1. **Certificates (PK directory):**
   - Issued by trusted **Certificate Authority**.
     - Prevent manipulation of others' public keys.
     - Prevent man-in-the-middle attacks.
   - **Example:** $\text{Cert} \leftarrow \text{Sign}(SK_{CA}, \text{Alice} \parallel pk||vk)$.
2. **Bitcoin/Cryptocurrency:**
   - Payments are anonymous but require verification.
   - Digital signatures ensure payment authenticity.

1.4 Digital Signature Definition

A digital signature scheme is a triple of PPT algorithms $(\text{Gen}, \text{Sign}, \text{Verify})$:

- $(vk, sk) \leftarrow \text{Gen}(1^n)$
- $\sigma \leftarrow \text{Sign}(sk, m)$
- $\text{Verify}(vk, m, \sigma) \in \{\text{Accept}, \text{Reject}\}$

2 Security

- Adversary can see polynomially many signatures but should not be able to produce a signature on a new message.
- **EUF-CMA:** Existentially Unforgeable under Chosen-Message Attack.

  - **Chosen-message attack:** adversary can query signatures on chosen messages.
  - **Existential forgery:** adversary wins if they produce a valid signature on a new message.
- **Strong EUF-CMA:** adversary wins if $\text{Verify}(vk, m', \sigma') = 1$ and $(m', \sigma') \notin \{(m_1, \sigma_1), (m_1, \sigma_1), \dots\}$
  - Don't need both m^* and σ^* to be new, but at least one must be.

3 Building Signature Schemes

3.1 Signing One Bit

Lamport One-Time Signatures

- $sk = \{x_0, x_1\}$, $vk = \{y_0 = f(x_0), y_1 = f(x_1)\}$.
  - f is a function such that you can't infer the sk from the vk (ie. one-way)
- Signature on bit b: $\sigma = x_b$.
- Verification: check $f(\sigma) = y_b$.
- Security relies on f being a one-way function: no PPT adversary can produce signature of $\bar{b}$ given signature of b because this implies given y_0, y_1 we can find pre-images x_0, x_1

3.2 Expanding to Longer Messages

Step 0: SIgning polynomially many bits with fixed verification keys

- need **Collision Resistant Hash Functions (CRHF):**
  - Compressing family of functions for which it is computationally hard to produce a collision
  - $H = \{h : \{0,1\}^m \to \{0,1\}^n\}$ where $m > n$
  - for every PPT alg. A:
$$\Pr_{h \leftarrow H}[A(1^n, h) = (x, y) : x \neq y,\ h(x) = h(y)] = \text{negl}(n)$$
  - Do CRHFs exist? theoretical → DLOG assumption gives it; practically → SHA3
  - **Domain Extension Theorem:** if there exists compressing $(n+1)$ to n bits, then there exists full hash functions to compress poly(n) bits to n bits.
    - given $\{h : \{0,1\}^{n+1} \to \{0,1\}^n\}$, find $\{h' : \{0,1\}^m \to \{0,1\}^n\}$
    - x is m-bits → m=poly(n)
    - have public n-bit string (P)
    - Run $y'_1 = h(P, x_1),\ y'_2 = h(y'_1, x_2), \dots y'_m = h(y'_{m-1}, x_m)$
    - Have a proof by contradiction: that is, assume you can find (x, z) such that $h'(x) = h'(z)$. then $h(y'_{m-1}, x_m) = h(y'_{m-1}, z_m)$ which gives some $h(x') = h(m')$ begin a collision in our base hash.
- Hash message into n bits, then sign the hash.
- For each hash bit, you need SK and VK bits:
$$SK = [(x_{1,0}, x_{1,1})\ (x_{2,0}, x_{2,1})\ \cdots\ (x_{n,0}, x_{n,1})] \quad \text{and} \quad VK = [(y_{1,0}, y_{1,1})\ (y_{2,0}, y_{2,1})\ \cdots\ (y_{n,0}, y_{n,1})]$$
- **Signing:**
$$sign(m) \leftarrow computez(h(m)) \to sign = (x_{1,z_1}), (x_{2,z_2}), \cdots (x_{n,z_n})$$
- Verification: recompute hash and check signatures of corresponding bits.
- **Security:** if $h(m) = h(m')$, then signatures collide ⇒ contradiction unless CRHF exists.

3.3 Back to Collision-Resistant Hash Functions (CRHF)

- Example construction based on the discrete logarithm problem:
  - Let $p = 2q + 1$ be a safe prime.
  - $H : \{h : (\mathbb{Z}_q)^2 \to \mathbb{QR}_p\}$
  - Choose generators $g_1, g_2 \in \mathbb{QR}_p$.
  - Define hash function:
$$h_{g_1,g_2}(x_1, x_2) = g_1^{x_1} g_2^{x_2} \pmod{p}.$$
- Suppose an adversary finds a collision:
$$h(x_1, y_1) = h(x_2, y_2).$$
  Then:
$$g_1^{x_1} g_2^{y_1} \equiv g_1^{x_2} g_2^{y_2} \pmod{p}.$$
  Rearranging:
$$g_1^{x_1 - x_2} \equiv g_2^{y_2 - y_1} \pmod{p}.$$
  This reveals the discrete logarithm of g_1 with respect to g_2.
- Therefore, finding collisions is as hard as solving the discrete logarithm problem.

3.4 Other Constructions

- **From OWFs/OWPs:** no universal black-box construction of CRHF exists.
- Lamport one-time signatures only achieve one-time security (require new key pair for each signature).

Known: EUF-CMA secure signature schemes exist assuming CRHFs.

3.5 Many-Time Signature Schemes

- **Step 1: Signature Chains (stateful, growing signature).**
  - Alice starts with (vk_0, sk_0).
  - To sign m_1, generate (vk_1, sk_1).
  - Output (m_1, σ_1, vk_1) where $\sigma_1 = \text{Sign}(sk_0, m_1 \parallel vk_1)$.
  - Verification: check $\text{Verify}(vk_0, m_1 \parallel vk_1, \sigma_1)$, then move to vk_1.
  - **Optimization:** only need to remember past key, not past messages. $\rightarrow$ Use part of vk_i to sign m_{i+1} and rest to sign vk_{i+1}
- **Step 2: Signature Trees.**
  - Organize verification keys in a binary tree.
  - Each node signed by its parent.
  - Reduces growth of signature size.
- **Step 3: Pseudorandom Trees.**
  - Use PRFs to generate child keys from parent key.
  - Shrinks Alice's storage requirements.
- **Step 4: Stateless Signatures.**
  - Use randomization and deterministic derivation to eliminate state.
  - Achieve many-time signatures without requiring Alice to track evolving state.