CS 55500 - Lesson 13

Recap

OWF → PRGs + PRFs → stateless SK encryption
↘ stateless PK encryption (more costly)

MACs:

- Bob can ensure message came from Alice
- tag is created + added by shared secret key

PRF F_k → $MAC(SK, m) = f_s(m)$

Digital Signatures → Public-Key analog of MAC

- anyone can verify source

$m, \sigma \leftarrow Sign(SK, m)$ and then $verify(PK, m, \sigma)$
(σ ↑ tag)

- **Signature** requires n key pairs for n ppl (each person just needs 1 pair) VS **MAC** needs n^2 keys (every pair needs unique SK)

Signature	MAC
• publicly verifiable	• privately verifiable
• transferable	• not transferable ↳ new tag for each new person
• Non-repudiation	• doesn't give Non-rep

Signature Applications

1. Certificates (PK directory):
   - come from Trusted certificate Authority → prevent manipulation of other ppl's PK
     ↘ prevent Man in the middle attack
   - issue certificate $\sigma \leftarrow Sign(SK, Alice \| pK \| vK)$

2. Bitcoin / cryptocurrency:
   - payment is anonymous but need verification payment was made by you

Dig Sig defn:

triple of PPT algs $(Gen, sign, verify)$ ∋

$(vK, SK) \leftarrow Gen(1^n)$

$\sigma \leftarrow Sign(SK, m)$

$Acc(1)/Rej(0) \leftarrow verify(vK, m, \sigma)$

Security

- adversary can see poly many sig on msgs and not be able to produce sig on new msg

1. Adversary can query on chosen messages (Chosen-message attack)
2. Adversary wins if she can produce valid sig for message outside of seen ones (Existential Forgery)

★ EUF-CMA Security

(Existentially unforgeable against a chosen message attack)

Strong EUF-CMA security

- at end Eve wins if $\text{Verify}(vk, m^*, \sigma^*) = 1$ and

$(m^*, \sigma^*) \notin \{(m_1, \sigma_1), (m_2, \sigma_2), \dots\}$ ← we require m^* and σ^* don't both have to be new

Lamport (one-time) signatures → signing a bit

SK: $[x_0, x_1]$ VK: $[y_0 = f(x_0), y_1 = f(x_1)]$ — func st you can't infer SK from VK

Signature: $\sigma = x_b$ Verify (b, σ): Check if $f(\sigma) = y_b$

Assume f is a OWF: no PPT Adv can produce signature of $\bar{b}$ given signature of b

bc this implies given y_0, y_1 we can find pre-images x_0, x_1

Expanding to longer messages → Signature Scheme

Step 0: Signing polynomially many bits w/ fixed verification key

★ Collision-Resistant Hash functions: compressing family of functions for which it is comp. hard to produce a collision

$\mathcal{H} = \{h : \{0,1\}^m \to \{0,1\}^n\}$ w/ $m > n$

↳ for every PPT Alg A:

$$\Pr_{h \leftarrow \mathcal{H}}[A(1^n, h) = (x, y) : x \neq y, h(x) = h(y)] = \text{negl}(n).$$

$h \leftarrow \mathcal{H}$: rand selected hash function from family $\mathcal{H}$

- Do CRHFs exist? theoretical → assuming discrete log, yes!
  practical → SHA3
- domain extension theorem: ∃ compressing (+1) to (n) bits ⇒ ∃ hash func.

given $h: \{0,1\}^{n+1} \to \{0,1\}^n$, find $h': \{0,1\}^m \to \{0,1\}^n$ compress poly(n) to n

- x is m-bits → m = poly(n)

have public n-bit string (P)

1. Run $h(P, x_1) = y_1'$

Proof by contradiction

assume you can find x, z

2. Run $h(y'_1, x_2) = y'_2$

:

m. Run $h(y'_{m-1}, x_m)$

gets to $h(y'_{m-1}, x_m) = h(y'_{m-1}, z_m)$ $\exists\ h'(x) = h'(z)$ then you can have $h(x') = h(z')$

Signing Poly many bits

- hash message into n bits + sign the hash
- for each hash bit, you need SK and VK bits

$$SK = \begin{bmatrix} x_{1,0} & x_{2,0} & \cdots & x_{n,0} \\ x_{1,1} & x_{2,1} & & x_{n,1} \end{bmatrix} \quad VK = \begin{bmatrix} y_{1,0} & y_{2,0} & \cdots & y_{n,0} \\ y_{1,1} & y_{2,1} & & y_{n,1} \end{bmatrix}$$

$sign(m) \rightarrow comp\ z(h(m)) \rightarrow sign = (x_{1,z_1}, x_{2,z_2}, \ldots x_{n,z_n})$

verify (m, σ) : recompute hash $z = h(m)$

check if $\forall i$: $f(\sigma_i) = y_{i,z_i}$

if you can find m and m' $\ni h(m) = h(m')$ then you can find m and m' to make same sig (contradiction) so have to have CRHF

Back to CRHF

$p = 2q+1$ is a "safe" prime $\quad H = \{h : (\mathbb{Z}_q)^2 \rightarrow QR_p\}$

two residues to a quad. res.

$2\log q \rightarrow \log(2q+1)$ → Compress!

- Each $h_{g_1,g_2} \in H$ is parameterized by gens g_1, g_2 of QR_p (group of order q)

$$h_{g_1,g_2}(x_1, x_2) = g_1^{x_1} g_2^{x_2} \pmod p$$

suppose adv find (x_1, y_1) and (x_2, y_2) collision!

$$g_1^{x_1} g_2^{x_2} = g_1^{y_1} g_2^{y_2}$$

$$g_1^{x_1 - y_1} = g_2^{y_2 - x_2}$$

$$g_1 = g_2^{(y_2 - x_2)(x_1 - y_1)^{-1}}$$ ← have found $DLOG_{g_2}(g_1)$

* Broken DLP meaning can't find collision

Other Constr:

- OWF/OWP: does not exist universal black-box to create CRHF

*only at one-time security (requires new SK/VK for each new σ)

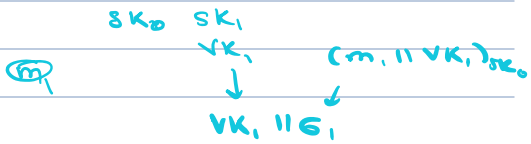
Know: EUF-CMA-secure sign schemes exist assuming CRHFs exist

(Many-Time) Sign Schemes

Step1: Stateful, growing signatures: Signature Chain
Step2: Shrink this signature: Signature Trees
Step3: shrink Alice's storage: Pseudorandom trees
Step4: use randomization to get to stateless
Step5: Make it stateless + deterministic

STEP 1

- Alice starts w/ SK_0 (publish VK_0)
- Signing m_1
  - Gen pair (VK_1, SK_1)
  - $\sigma_1 \leftarrow Sign(SK_0, m_1 || VK_1)$



OUTPUT $VK_1 || \sigma_1$ next step output $VK_1 || m_1 || \sigma_1 || VK_2 || \sigma_2$

- remember $VK_1 || m_1 || \sigma_1$ as well as SK_1

- verify
  - Verify$(VK_0, m_1 || VK_1, \sigma_1)$ ← gets next VK

*optimization: need to memorize only past key not past messages
↳ Use part of r_{K_i} to sign m_{i+1} and rest to sign VK_{i+1}
- only carry signature of key going forward