Speaker's Notes: Digital Signatures (Schnorr Scheme)

Introduction - Digital Signature Systems (0-2:30)

[BOARD] Write: (G, S, V) - Key Gen, Sign, Verify

Core Definition

- Digital signature = cryptographic analog of handwritten signature
- Three algorithms form the system:
  - G: probabilistic key generation, takes security parameter k, produces (pk, sk)
  - S: signing algorithm, takes message m and secret key sk, produces signature $S_{sk}(m)$
  - V: verification algorithm, takes signature s, message m, public key pk
  - Outputs $V_{pk}(s, m) \in \{\text{accept}, \text{reject}\}$

[BOARD] Write: Correctness requirement: $V_{pk}(S_{sk}(m), m) = \text{accept}$

- Must ALWAYS accept legitimate signatures
- Anyone with pk can verify, only holder of sk can sign

[TRANSITION] Now we need to define what "secure" means for signatures. Like MACs, we consider adaptive chosen message attacks...

CMA Security Definition (2:30-5:00)

[BOARD] Draw oracle game diagram:

```
Adversary E ⟷ Oracle (has sk)
         ↑ gives pk
```

The Security Game

- Oracle generates (pk, sk) using $G(k)$
- Adversary E receives pk only
- E can query oracle: submit any message m_i, get back $S_{sk}(m_i)$
- Can make polynomial number of queries
- **Win condition**: Produce (m_0, s_0) where:
  - m_0 was NEVER queried to oracle (existential forgery)
  - $V_{pk}(s_0, m_0) = \text{accept}$

[BOARD] Write: **Definition 12.1**: CMA-secure if $\mathrm{Adv}_E(k)$ negligible for all PPT adversaries

- Strongest possible security notion
- Even seeing many valid signatures doesn't help forge new one
- Models real-world scenario: attacker sees legitimate signatures

[TRANSITION] The Schnorr signature scheme achieves CMA security based on discrete log hardness. Let me introduce the setup...

Schnorr Scheme - Parameters (5:00-7:00)

[BOARD] Write:

- Primes p, q where $q|(p-1)$
- $\alpha \in \mathbb{Z}_p^*$ with $|\alpha| = q$ (order q)

Setting Up the Group

- Work in **subgroup** of $\mathbb{Z}_p^*$, not full group
- Subgroup has order q (should be large prime, e.g., 256 bits)
- p much larger (e.g., 2048 bits) for security

[BOARD] Write: Get α: Start with generator α_0, compute $\alpha = \alpha_0^{(p-1)/q} \bmod p$

- This gives element of order **exactly** q
- By Lagrange's theorem: $\alpha^q = 1$ in the subgroup
- Working in subgroup makes signatures shorter

[TRANSITION] Before seeing the actual signature scheme, understand the interactive protocol it's based on - a zero-knowledge proof...

Interactive Schnorr Protocol (7:00-10:30)

[BOARD] Write: **Goal**: Prove knowledge of s where $\beta = \alpha^s$, without revealing s

The Three-Round Protocol

[BOARD] Draw protocol flow:

```
Signer (knows s)          Verifier
    |------ c = α^r ------>|
    |<------- e ------------|  (random challenge)
```

```
    |------ z = r+es ------>|
    |                       | Check: α^z ?= c·β^e
```

- **Round 1**: Signer commits to random $r \in \mathbb{Z}_q$, sends $c = \alpha^r$
- **Round 2**: Verifier sends random challenge $e \in \mathbb{Z}_q$
- **Round 3**: Signer responds with $z = (r + es) \bmod q$
- **Verification**: Check $\alpha^z \stackrel{?}{=} c\beta^e \bmod p$

Why It Works

[BOARD] Write correctness: $c\beta^e = \alpha^r(\alpha^s)^e = \alpha^{r+es} = \alpha^z \pmod p$

- If signer knows s, verification always passes
- If signer doesn't know s: must guess e before seeing it
    - Success probability only $1/q$ (negligible!)
- Zero-knowledge: verifier learns nothing about s itself

[TRANSITION] This is interactive - requires back-and-forth. For actual signatures, need non-interactive version using Fiat-Shamir transform...

Non-Interactive Signature Scheme (10:30-14:00)

[BOARD] Write: **Key idea**: Replace random challenge e with $e = h(c, m)$

The Fiat-Shamir Transform

- Hash function $h : \{0,1\}^* \to \mathbb{Z}_q$ acts as "verifier"
- Challenge now depends on commitment c AND message m
- In Random Oracle Model: h outputs effectively random
- Non-interactive: no verifier needed during signing

The Complete Scheme

[BOARD] Write:

- **Keys**: $pk = (h, p, q, \alpha, \beta = \alpha^s)$, $sk = s$ (random in $\mathbb{Z}_q$)
- **Sign**(m):
    1. Pick random $r \in \mathbb{Z}_q$
    2. Compute $c = \alpha^r$
    3. Output (e, z) where $e = h(c, m)$, $z = (r + es) \bmod q$
- **Verify**$(m, (e, z))$:
    1. Compute $c = \alpha^z \beta^{-e}$
    2. Check $e \stackrel{?}{=} h(c, m)$

Verification Correctness

[BOARD] Write: $\alpha^z\beta^{-e} = \alpha^{r+es} \cdot \alpha^{-es} = \alpha^r = c$

- Since c matches, $h(c, m)$ must equal e
- Signature is just pair (e, z) - compact!
- Note: $\beta^{-e} = (\alpha^s)^{-e} = \alpha^{-es}$

[TRANSITION] Finally, let's discuss the security guarantee this provides...

Security Analysis (14:00-15:00)

[BOARD] Write: **Theorem**: If DL hard in subgroup, then Schnorr is CMA-secure (in ROM)

Security Reduction

- Proof by contrapositive
- Assume Schnorr NOT CMA-secure → adversary can forge
- Then can use adversary to compute discrete logarithms
- Therefore: DL hardness implies Schnorr security

Key Requirements

- Random Oracle Model for hash function h
- Discrete log must be hard in subgroup generated by α
- In practice: use standardized groups (e.g., NIST curves)
- Provides strongest security: CMA-secure (Definition 12.1)

TOTAL TIME: ~15 minutes