# Constructing Digital Signatures

Slides by Prof. Jonathan Katz.
Lightly edited by me.

# Hash-and-sign paradigm

- Given
  - A signature scheme $\Pi$ = (Gen, Sign, Vrfy) for "short" messages of length n
  - Hash function H: $\{0,1\}^* \rightarrow \{0,1\}^n$
- Construct a signature scheme $\Pi$'=(Gen, Sign', Vrfy') for arbitrary-length messages:
  - $\text{Sign}'_{sk}(m) = \text{Sign}_{sk}(H(m))$
  - $\text{Vrfy}'_{pk}(m, \sigma) = \text{Vrfy}_{pk}(H(m), \sigma)$

# Hash-and-sign paradigm

- Theorem: If $\Pi$ is secure and H is collision-resistant, then $\Pi'$ is secure
- Proof: Say the sender signs $m_1$, $m_2$, …
  - Let $h_i = H(m_i)$

- Attacker outputs forgery $(m, \sigma)$, $m \neq m_i$ for all i
- Two cases:
  - $H(m) = h_i$ for some i
    - Collision in H!
  - $H(m) \neq h_i$ for all i
    - Forgery in the underlying signature scheme

# Hash-and-sign paradigm

- Same idea as in the hash-and-MAC paradigm

- Can be viewed as analogous to hybrid encryption
  - The *functionality* of digital signatures at the asymptotic cost of a *symmetric-key* operation

# Signature schemes

- We will discuss how to construct signature schemes for “short” messages
  - Using hash-and-sign, this implies signatures for arbitrary length messages

# Signature schemes in practice

- RSA-based signatures
  - Can be proven secure (based on RSA assumption, in random-oracle model)
- Dlog-based signatures
  - Shorter signatures, faster signing than RSA-based signatures
  - (EC)DSA
    - Widely used, no proof of security
  - Schnorr
    - Can be prove secure (based on dlog assumption, in random-oracle model)

# RSA-based signatures

# Recall…

- Choose random, equal-length primes p, q
- Compute modulus N=pq
- Choose e, d such that $e \cdot d = 1 \bmod \phi(N)$

- The $e^{th}$ root of m modulo N is $[m^d \bmod N]$

  $$(m^d)^e = m^{de} = m^{[ed \bmod \phi(N)]} = m \bmod N$$

- *RSA assumption*: given N, e only, hard to compute the $e^{th}$ root of a uniform $m \in \mathbb{Z}^*_N$

# “Plain” RSA signatures

N, e

m, $\sigma$

$m \stackrel{?}{=} [\sigma^e \bmod N]$

$(N, e, d) \leftarrow \text{RSAGen}(1^n)$

$pk = (N, e)$

$sk = d$

$\sigma = [m^d \bmod N]$

# Security?

- Intuition
  - Signature of m is the $e^{th}$ root of m – supposedly hard to compute!

# Attack 1

- Can sign *specific* messages
  - E.g., easy to compute the $e^{th}$ root of m = 1, or the cube root of m = 8

# Attack 2

- Can generate signatures on "random" messages
  - Choose arbitrary $\sigma$; set $m = [\sigma^e \bmod N]$

# Attack 3

- Can combine two signatures to obtain a third
  - Say $\sigma_1$, $\sigma_2$ are valid signatures on $m_1$, $m_2$ with respect to public key N, e
  - Then $\sigma' = [\sigma_1 \cdot \sigma_2 \bmod N]$ is a valid signature on the message $m' = [m_1 \cdot m_2 \bmod N]$
    - $(\sigma_1 \cdot \sigma_2)^e = \sigma_1^e \cdot \sigma_2^e = m_1 \cdot m_2 \bmod N$

# RSA-FDH

- Main idea: apply a "cryptographic transformation" to messages before signing

- Public key: (N, e)  private key: d
- $\text{Sign}_{sk}(m) = H(m)^d \bmod N$
  - H must map onto all of $\mathbb{Z}^*_N$
- $\text{Vrfy}_{pk}(m, \sigma)$: output 1 iff
  $\sigma^e = H(m) \bmod N$
- (This also handles long messages without additional hashing)

# Intuition for security?

- Look at the three previous attacks…
  - Not easy to compute the $e^{th}$ root of H(1), …
  - Choose $\sigma$…, but how do you find an m such that $H(m) = \sigma^e \bmod N$?
    - Computing inverses of H should be hard
  - $H(m_1) \cdot H(m_2) = \sigma_1^e \cdot \sigma_2^e = (\sigma_1 \cdot \sigma_2)^e \neq H(m_1 \cdot m_2)$

# Security of RSA-FDH

- If the RSA assumption holds, and H is modeled as a random oracle (mapping onto $\mathbb{Z}^*_N$), then RSA-FDH is secure

- In practice, H is instantiated with a (modified) cryptographic hash function
  - Must ensure that the range of H is large enough!

# RSA-FDH in practice

- The RSA PKCS #1 v2.1 standard includes a signature scheme inspired by RSA-FDH
  - Essentially a randomized variant of RSA-FDH

# dlog-based signatures

# Digital signature standard (DSS)

- US government standard for digital signatures
  - DSA, based on discrete-logarithm problem in subgroup of $\mathbb{Z}_p^*$
  - ECDSA, based on elliptic-curve groups
  - See book for details

- Compared to RSA-based signatures
  - Shorter signatures and public keys (for EDCSA)
  - Can have faster signing
  - Slower verification