Digital Signature

Hyoungshick Kim
Department of Software
College of Software
Sungkyunkwan University

Digital signatures

- Developed to verify that a message comes from the claimed sender
- Opposite of public key encryption
  - Private key used to sign message
  - Public key used to verify signature

Encryption and Digital signature

- Encryption
  - Suppose we **encrypt** M with Bob's public key
  - Bob's private key can **decrypt** to recover M
- Digital Signature
  - **Sign** by "encrypting" with your private key
  - Anyone can **verify** signature by "decrypting" with public key
  - But only you could have signed
  - Like a handwritten signature, but way better…

"Plain" RSA signatures

- Public key (N, e); private key (N, d)
- To sign message $m \in Z_N^*$, compute $\sigma = m^d \bmod N$
- To verify signature σ on message m, check whether $\sigma^e = m \bmod N$
- Correctness holds…
- What about security?

Security of "plain" RSA signature?

- Attacker can sign specific messages
  - E.g., easy to compute the e^{th} root of $m = 1$
- Attacker can sign "random" messages
  - Choose arbitrary σ; set $m = [\sigma^e \bmod N]$
- Attacker can combine two signatures to obtain a third
  - Say σ_1, σ_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 (Full Domain Hash)

- Apply a “cryptographic transformation” to messages before signing

- Public key: (N, e) Private key: d
- $\text{Sign}_{sk}(m) = H(m)^d \bmod N$
- $\text{Verify}_{pk}(m, \sigma)$: output 1 iff $\sigma^e = H(m) \bmod N$
- This also handles long messages

Security of RSA-FDH

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

DSA/DSS signatures

- Another popular signature scheme, based on the hardness of the discrete logarithm problem
  - Introduced by NIST in 1992
  - US government standard

- ECDSA (based on ECDLP)
  - Used for Bitcoin

Discrete-logarithm problem

- Fix cyclic group G of order m, and generator g
- We know that $\{g^0, g^1, \ldots, g^{m-1}\} = G$
  - For every $h \in G$, there is a unique $x \in \mathbb{Z}_m$ s.t. $g^x = h$
  - Define $\log_g h$ to be this x – *the discrete logarithm of h with respect to g* (in the group G)

- <u>Dlog problem in G:</u> Given g, h, compute $\log_g h$
- <u>Dlog assumption in G:</u> Solving the discrete log problem in G is hard

Diffie-Hellman problems

- Fix group G with generator g
- Define $DH_g(h_1, h_2) = DH_g(g^x, g^y) = g^{xy}$
- *Computational* Diffie-Hellman (CDH) problem:
  - Given g, h_1, h_2, compute $DH_g(h_1, h_2)$
- *Decisional* Diffie-Hellman (DDH) problem:
  - Given g, h_1, h_2, distinguish the correct $DH_g(h_1, h_2)$ from a uniform element of G

Relating the Diffie-Hellman problems

- If the discrete-logarithm problem is easy, so is the CDH problem
- If the CDH problem is easy, so is the DDH problem

Group selection

- For cryptographic applications, best to use *prime-order* groups
  - The dlog problem is “easier” if the order of the group has small prime factors

- Two common choices of groups
  - Prime-order subgroup of $\mathbb{Z}^*_p$, p prime
    - E.g., $p = tq + 1$ where q is also a prime
    - Take the subgroup of t^{th} powers, i.e., $G = \{ [x^t \bmod p] \mid x \in \mathbb{Z}^*_p \}$
  - Prime-order subgroup of an *elliptic curve* group

Questions?


Sungkyunkwan
Campus
Social
Graduate
campus
Sciences
University
Seoul
Natural
Suwon
School
Korean
university
edit
established
constructed
English
Global
Building
Institute
Professors
Confucian
schools
Humanities
lecturers