Lecture 10: Digital Signature

Viet Tung Hoang

The slides are loosely based on those of Prof. Mihir Bellare, UC San Diego.
Agenda

1. High-level Overview

2. Building Signature Scheme
Alice generates a pair of secret key and public key. She keeps $sk$ to herself, and stores $pk$ in a public, trusted database.
The adversary may replace Alice’s real key with its fake one
Solution

Alice has her $pk$ **certified** by CA

Tampering with $pk$ means one has to forge the CA’s digital signature
Digital Signature Scheme: Syntax

Key Gen

\[ \mathcal{K} \rightarrow \$ \rightarrow \{ pk, sk \} \]

Sign

\[ M \rightarrow \text{Sign} \rightarrow \$ \rightarrow \sigma \]

\[ \sigma \rightarrow sk \]

Verify

\[ \sigma \rightarrow M \rightarrow \text{Ver} \rightarrow 0/1 \]

\[ \text{Ver} \rightarrow pk \]
Digital Signature versus MAC

MAC
- Verifier needs to share a secret key with signer
- Verifier can impersonate signer

Digital Signature
- Verifier needs no secret
- Verifier cannot impersonate signer
Digital Signature: Unforgeability Security

- Similar to MAC security
- **Difference**: The adversary is given the public key

Again, digital signature doesn’t directly thwart replay attack.
Agenda

1. High-level Overview

2. Building Signature Scheme
A Bad Scheme: Plain RSA Signature

Key generation: Like RSA PKE
- Pick two large primes $p, q$ and compute $n = pq$
- Pick $e, d \in \mathbb{Z}^*_\varphi(n)$ such that $ed \equiv 1 \pmod{\varphi(n)}$
- Return $pk \leftarrow (n, e), sk \leftarrow (n, d)$

Sign:
- To sign message $x$ under $sk = (n, d)$, “decrypt” it: $\sigma \leftarrow x^d \mod n$

Question: How to verify?
Issues with Plain RSA Signature

- **Feasibility**: Can sign only short messages

- **Security**: Can easily break unforgeability security

No tag query needed!

\[ pk = (n, e) \]

\[ (M', \sigma') = (x^e \mod n, x) \]
Hash-then-Sign Paradigm

Plain RSA Signature $\rightarrow$ Full Domain Hash (FDH)

**Key generation:** Like Plain RSA

**Sign:** To sign message $M$

\[ H : \{0, 1\}^* \rightarrow \mathbb{Z}_N \]
Security Requirement for Hash Function

The hash must be collision-resistant

**Question:** Given a collision of the hash, break security of signature scheme

For the proof to go through, the hash has to be modeled as a random oracle
A Common Wrong Way to Hash

2048-bit string $Y$, viewed as a number in $\mathbb{Z}_N$

Broken by Desmedt and Odlyzko in 1985
How to Hash Properly

Use the first $m = \lfloor \log_2(N) \rfloor$ bits and take mod $N$
Hashing in PKCS#1

19 bytes to indicate hash info

2 bytes

padding

0001 FF FF ... FF 00 digest info

2048-bit string $Y$, viewed as a number in $\mathbb{Z}_N$
Verification in PKCS#1

Check if $Y$ is an encoding of $\text{SHA}(x)$

- 0001
- FF FF ... FF 00
- digest info
- $\text{SHA}(x)$

2 bytes
19 bytes
Length depends on the hash

Question: Write pseudocode for the encoding check
Bad Check for PKCS#1 Signature

Do not check if SHA output is the end of the encoded string

Totally 2048 bits

0001 FF FF ... FF 00 digest info SHA(x)
Attacking PKCS#1 Signature with Bad Check

Using \( e = 3 \) as verification key in RSA

**Forge signature for message \( x \):**

Infeasible, due to the RSA assumption

\[
\sigma^3 \equiv Y \mod N
\]

**Initial goal:** Find signature \( \sigma \) such that \( \sigma \leftarrow \text{PlainRSA-Sign}(Y) \)

Suffix length \( > \frac{2}{3} \cdot |Y| \)  
1368 bits
Attacking PKCS#1 Signature with Bad Check

Using $e = 3$ as verification key in RSA

Forge signature for message $x$:

Infeasible, due to the RSA assumption

Initial goal: Find signature $\sigma$ such that $\sigma \leftarrow \text{PlainRSA-Sign}(Y)$

Attack: Output signature $s = \lceil Y^{1/3} \rceil$
Why It Works

Theorem: For any $Y \leq 2^{2040}$, the number $Y' \leftarrow ([Y^{1/3}])^3$ satisfies $0 \leq Y' - Y < 2^{1368}$
Why It Works

\[ s^3 = Y' \]

\[ 0 \leq Y' - Y < 2^{1368} \]
Why It Works

$s$ is still regarded as a valid signature by the bad implementation

$s^3 = Y'$