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
The Need For Signing Is Ubiquitous
How To Sign Electronically?

Problem: A digitized signature is easily copied → forgery

Lots of apps to digitize signatures
Digital Signature Scheme: Syntax

Key Gen

\[ \mathcal{K} \rightarrow pk \xrightarrow{} sk \]

Sign

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

\[ sk \rightarrow \sigma \rightarrow M \]

Verify

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

\[ pk \rightarrow \text{Ver} \]
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.
1. High-level Overview

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

Key generation: Like RSA encryption

Sign:
- To sign a message, “decrypt” it:

Verify:
- To verify a signature, “encrypt” it and compare with the message
Issues with Plain RSA Signature

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

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

No sign query needed!

\[ pk = (n, e) \quad \rightarrow \quad (M', \sigma') = (x^e \ mod \ n, x) \]
Exercise: Forging Plain RSA For Targeted Msg

Goal: The forged message must be a **specific** one.

\[ (M' = M_1 \cdot M_2, \sigma') \]
Hash-then-Sign Paradigm

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

Key generation: Like Plain RSA

Sign: To sign message $M$
How to verify?
Security Requirement for Hash Function

What intuition suggests: Hash must be collision-resistant

If \( H(M) = H(M') \) then \( M \) and \( M' \) have the same signature

What proof requires: Hash is modeled as a random oracle
A Gap of **Demand** and **Supply**

2048 bits of output

\[ M \]

\[ H \]

Plain RSA Signing

512 bits of output

\[ M \]

SHA
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 what hash function and its output length

2 bytes padding

0001 FF FF ... FF 00 hash info

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

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

$\sigma$ \rightarrow \text{Plain RSA Verification} \rightarrow 2048$-bit $Y$ \rightarrow Encoding checking

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

- 0001: 2 bytes
- FF FF ... FF 00: 19 bytes
- hash info
- $\text{Hash}(x)$: Length depends on hash

Encoding check must be done carefully, otherwise there is an attack
Breaking PKCS#1 Signature With Bad Check

WarmUp: A Toy Example

$k$ bits for hash function and its output length

$n$-bit string $Y$

Plain RSA Signing
Bad Check of Encoding

Skip 0*1; read the next $k$ bits to know Hash and output length $L$
Bad Check of Encoding

Compare the next $L$ bits with $\text{Hash}(x)$

$L$ bits

Allow some suffix

$0^*1$ info $k$ bits Length depends on hash function
Attacking Toy Variant With Bad Check
An Illustration for $n = 16$ and $k = 1$

Forge signature for an arbitrary message $x$

Correct encoding

| $0^{13} \parallel 1$ | info = 1 | Hash($x$) = 0 |

Targeted encoding

| 001 | info = 1 | Hash($x$) = 0 | $0^{11}$ |

$Y = 12, 288$
Attacking Toy Variant With Bad Check

Using $e = 3$ as verification key in RSA

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

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

For $e = 3$ and $Y = 12288$: Find $\sigma$ such that $\sigma^3 \equiv 12288 \mod N$

**Attack:** Approximate $\sigma$ via $s = \left\lfloor Y^{1/3} \right\rfloor = 24$
Verifying The Forged Signature

Targeted encoding

- $s$
- $x$
- Plain RSA Verification
- 16-bit $Z$
- Encoding checking

Targeted encoding

- info = 1
- Hash($x$) = 0
- $0^{11}$

Obtained encoding

- info = 1
- Hash($x$) = 0
- 11000000000

$Z = 24^3 \mod N = 13824$

$Z$ passes the bad encoding checking $\Rightarrow s = 24$ is a valid signature
**Why It Works**

\[ s = \left\lfloor Y^{1/3} \right\rfloor \]

Plain RSA Verification

\[ Z = s^3 \]

**Theorem:** For any \( Y < 2^n/3 \) with \( n \geq 16 \), the number

\[ Z \leftarrow (\left\lfloor Y^{1/3} \right\rfloor)^3 \]

satisfies

\[ 0 \leq Z - Y < 2^{2/3n} \]
Why It Works

\[ 0 \leq Z - Y < 2^{2n/3} \]

\[ s^3 = Z \]
Attacking PKCS#1 Signature with Bad Check

Using $e = 3$ as verification key in RSA

Your Exercise