Digital Signature

Viet Tung Hoang

The slides are loosely based on those of Prof. Mihir Bellare, UC San Diego.
1. High-level Overview

2. Building Signature Scheme
Previously

Alice generates a pair of secret key and public key. She keeps $sk$ to herself, and stores $pk$ in a public, trusted database.
Problem

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

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

Sign

\[ M \rightarrow Sign \rightarrow \$\rightarrow \sigma \]

\[ sk \rightarrow Sign \rightarrow \sigma \]

Verify

\[ \sigma \rightarrow M \rightarrow Ver \rightarrow 0/1 \]

\[ pk \rightarrow Ver \rightarrow 0/1 \]
## Digital Signature versus MAC

<table>
<thead>
<tr>
<th>MAC</th>
<th>Digital Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Verifier needs to share a secret key with signer</td>
<td>- Verifier needs no secret</td>
</tr>
<tr>
<td>- Verifier can impersonate signer</td>
<td>- Verifier cannot impersonate signer</td>
</tr>
</tbody>
</table>
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 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!

\[ pk = (n, e) \]

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

Plain RSA Signature → 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 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 $\text{Hash}(x)$

- 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 $Y$
Skip $0^*1$; read the next $k$ bits to know Hash and output length $L$. Length depends on hash function.
Bad Check of Encoding

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

Allow some suffix

$L$ bits

0*1 info

2 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$^{13}$ || 1 | info = 1 | Hash($x$) = 0 |

Targeted encoding

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

$Y = 12,288$

Key idea: Suffix length $> 2n/3$
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 $s = \lceil Y^{1/3} \rceil = 24$
Verifying The Forged Signature

Targeted encoding

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

Obtained encoding

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

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

Theorem: For any $Y < 2^n / 3$ with $n \geq 16$, the number $Z \leftarrow ([Y^{1/3}])^3$ satisfies $0 \leq Z - Y < 2^{2/3n}$
Why It Works

\[ Y \]

\[ 0^*1 \quad \text{info} \quad \text{oo...oo} \]

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

\[ 0^*1 \quad \text{info} \quad Z - Y \]

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

Using $e = 3$ as verification key in RSA

Your Exercise