Introduction to Cryptography
Part IV: Digital Signatures

André Schrottenloher

Inria Rennes
Team CAPSULE

[0]

Hash Functions

Hash functions

A **hash function** is a public function that takes a variable-length message and outputs a fixed-length digest: $H : \{0,1\}^* \to \{0,1\}^n$.

"General" hash functions are used whenever you need a random-looking function:

- Hash tables;
- Randomized algorithms (e.g., Pollard's rho method).

They are **not enough** for cryptography: we need **cryptographic** hash functions.

Hash functions (ctd.)

In the context of symmetric cryptography, a hash function is **secure** if it offers a **generic** security level against:

- Preimage attacks;
- Second preimage attacks;
- Collision attacks.

Generic = there should be no better attack than those we have against a truly random function.

In other words we want the behavior to be ideal (typical requirement in symmetric crypto).

Preimage resistance

Fix $H : \{0,1\}^* \to \{0,1\}^n$.

Preimage resistance

For $t \leftarrow \{0,1\}^n$, it should be difficult to find m such that $t = H(m)$.

- By brute force, this takes time $\mathcal{O}(2^n)$ (to succeed with constant probability)
- So it should take time $\mathcal{O}(2^n)$

Example: password authentication.

- One stores only $H(\text{password})$.
- An attacker having access to the database cannot find the passwords.

Second preimage resistance

Fix $H : \{0,1\}^* \to \{0,1\}^n$.

For $x \leftarrow \{0,1\}^m$, it should be difficult to find $y \neq x$ such that $H(y) = H(x)$.

- By brute force, this takes time $\mathcal{O}(2^n)$ (to succeed with constant probability)

Example: signatures (this lecture).

Collision resistance

Collision resistance

- Producing a collision (pair $x \neq y$ such that $H(x) = H(y)$) should take time $\mathcal{O}(2^{n/2})$ (birthday paradox!)

This is the same as long as the input size is $\geq n$ bits.

On the existence of collisions / preimages

There exists collisions & preimages (the message space is much bigger than the hash space).

There **exists** an algorithm that returns in **constant time** a collision for **any** hash function.

$\Longrightarrow$ however, we don't know how to write it down.

Some examples

MD5 (broken)

- 128-bit hash (RFC 1321, Rivest, 1992)
- Collisions found (Wang, Yu, 2005)
- Forgery of certificates (Stevens et al., 2009)

SHA-0 (broken)

- 160-bit hash (NSA, 1993)
- Collisions (theoretical) in 1998 (Joux, Chabaud)

SHA-1 (broken)

- 160-bit hash
- Theoretical collisions in 2005 (Wang et al.)
- Practical collisions in 2017 (Stevens et al., 2009)
- Chosen-prefix collisions (Leurent, Peyrin, 2020)
- Still used a lot . . .

Current standards

SHA-2

- Published by NSA in 2001
- Family of hash functions of 224, 256, 384, 512 bits

SHA-3

- a.k.a. Keccak, winner of an open competition organized by NIST
- Sponge function, published in 2015
- Outputs of 224, 256, 384, 512 bits

Digital signatures

Motivation

IND-CCA2 asymmetric encryption offers only **confidentiality** of messages.

Digital signatures (DS) offer:

- **authenticity**
- **integrity**

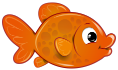
What are some constraints associated to a digital signature?

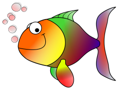
- It should depend on the signed message (otherwise you can copy it)
- It should depend on some secret
- Everybody should be able to verify it

Definition

$$\begin{cases} \text{KeyGen}: & 1^n & \mapsto & \text{sk}, \text{pk} \\ \text{Sign}: & \text{sk}, \text{h} & \mapsto & \sigma \\ \text{Verify}: & \text{pk}, \sigma, \text{h} & \mapsto & \{0,1\} \end{cases}$$

Correctness: $\forall m,\ \text{Verify}(\text{pk}, m, \text{Sign}(m, \text{sk})) = 1.$





$\text{sk}, \text{pk} = \text{KeyGen}(1^\lambda)$

$\xrightarrow{\quad\quad \text{pk} \quad\quad}$

$\sigma = \text{Sign}(\text{sk}, \text{h})$

$\xrightarrow{\quad\quad \sigma \quad\quad}$

$b = \text{Verify}(\text{pk}, \sigma, \text{h})$

Accepte si $b = 1$

Breaking authenticity

An attacker's power: **chosen message attack**.

- The attacker can obtain signatures $\sigma_i = \mathrm{Sign}(\mathsf{sk}, h_i)$ for chosen messages h_i

An attacker's goal: **existential forgery**.

- Produce some **new** valid message / signature pair (h, σ): $h \notin \{h_1, \ldots, h_q\}$

The new message does not need to have any meaning, for it to be a meaningful forgery.

EUF-CMA

Existential unforgeability against chosen-message attacks (EUF-CMA) is defined by a security game played by $\mathcal{C}$ and $\mathcal{A}$.

- **Initialization**: $\mathcal{C}$ generates a pair $\mathsf{pk}, \mathsf{sk} \leftarrow \mathsf{KeyGen}(1^n)$ and gives pk to $\mathcal{A}$
- **Queries:** at any point, $\mathcal{A}$ can choose h_i and obtain the signature $\sigma_i = \mathsf{Sign}(\mathsf{sk}, h_i)$
- **Forgery:** $\mathcal{A}$ sends a pair h^*, σ^* to $\mathcal{C}$ and wins if:

$$\begin{cases} \mathsf{Verify}(\mathsf{sk}, h^*, \sigma^*) = 1 \\ h^* \notin \{h_1, \ldots, h_q\} \end{cases}$$

The EUF-CMA advantage of $\mathcal{A}$ is defined as:

$$\mathrm{Adv}^{EUF-CMA}(\mathcal{A}) = \left| \Pr\left[\mathcal{A} \text{ wins}\right] \right| \ .$$

The DS scheme is EUF-CA secure iff any PPT adversary has a negligible advantage.

Theorem: domain extension with hash-and-sign

Theorem

Let $S := (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is a secure signature for short messages in $\{0,1\}^n$. Let H be a collision-resistant hash. Define S':

$$\begin{cases} \mathsf{KeyGen}' = \mathsf{KeyGen} \\ \mathsf{Sign}'(\mathsf{sk}, m) = \mathsf{Sign}(\mathsf{sk}, H(m)) \\ \mathsf{Verify}'(\mathsf{pk}, m, \sigma) = \mathsf{Verify}(\mathsf{pk}, H(m), \sigma) \end{cases}$$

Hash-and-sign security

What happens if:

- The adversary can find second preimages in the hash function?

$\Longrightarrow$ Choose a message m. Ask for $\sigma = \mathrm{Sign}(\mathrm{sk}, H(m))$; find second preimage m' such that $H(m') = H(m)$; now (m', σ) is a forgery.

- The adversary can find preimages?

$\Longrightarrow$ it's stronger than second preimages anyway

- The adversary can find collisions?

$\Longrightarrow$ find (m, m') such that $H(m) = H(m') = t$. Ask for $\sigma = \mathrm{Sign}(\mathrm{sk}, H(m))$. Now (m', σ) is a forgery. impersonate

Example The Flame malware (2012) used a chosen-prefix collision on MD5 to sign some of its components by impersonating a Microsoft certificate.

Constructing signatures

Contrary to PKE, a **one-way function** is enough to construct signatures.

- Hash-based signatures: SPHINCS+ (post-quantum)

More practical: all standard public-key assumptions like RSA? DLOG and also the post-quantum ones.

RSA Signatures

Basic RSA signature

KeyGen:

- Choose primes P, Q, $N = PQ$, choose e, d with $ed = 1 \pmod{\phi(N)}$.
- $\mathrm{sk} = (N, d)$
- $\mathrm{pk} = (N, e)$

Sign $m \in \mathbb{Z}_N^*$

- Return $\sigma = m^d \pmod{N}$

Verify $m \in \mathbb{Z}_N^*, \sigma$

- Check that $\sigma^e = m \pmod{N}$

This is not secure.

Attacks on basic RSA signature

Attack 1 Take any value t, then (t^e, t) is a valid message-signature pair $\Longrightarrow$ a "no-message" attack.

Attack 2 For any $m \in \mathbb{Z}_N^*$, we can forge a signature of m.

- Ask to sign $m_1 \in \mathbb{Z}_N^*$
- Ask to sign $m_2 = m(m_1)^{-1} \mod N$
- Compute $\sigma = \sigma_1 \sigma_2$

RSA-FDH

KeyGen:

- Generate $N = PQ$, and e, d
- Construct a CRHF $H \ : \{0,1\}^* \to \mathbb{Z}_N$
- $\mathsf{sk} = (N, d)$, $\mathsf{pk} = (N, e)$

Sign $m \in \{0,1\}^*$

- Return $\sigma = H(m)^d \mod N$

Verify $m \in \{0,1\}^*, \sigma$

- Check that $\sigma^e = H(m) \mod N$

Previous attacks do not apply:

- Signatures are not malleable anymore
- If we take t and compute t^e, we would need to find m such that $H(m) = t^e$: a preimage problem.