

Digital Signatures

CS 171: Cryptography

Sanjam Garg

October 22, 2026

Overview

- 1 Digital Signatures
- 2 Public-Key Infrastructure
- 3 Textbook RSA Signatures
- 4 Hash-and-Sign
- 5 The Schnorr Identification Scheme
- 6 Summary

Public-Key Analogue of MAC

A digital signature is the public-key analogue of a MAC: the signer holds a secret key, and **anyone** holding the public key can verify.

Software update

- A device comes pre-loaded with the vendor's **public key**.
- No need to share a separate (symmetric) key with everyone.

Non-repudiation

- A party to a contract or a communication **cannot deny** the authenticity of their signature on a document.
- A judge can enforce it — something we could *not* do with MACs (the verifier could have produced the tag itself).

Definition 1 (Digital signature scheme)

A signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$:

- $\text{Gen}(1^n)$: outputs a key pair (pk, sk) .
- $\text{Sign}_{sk}(m)$: outputs a signature σ on the message m .
- $\text{Vrfy}_{pk}(m, \sigma)$: outputs 0 or 1.

Correctness

For all n and all (pk, sk) output by $\text{Gen}(1^n)$ (except for a negligible fraction), and all m :

$$\text{Vrfy}_{pk}(m, \text{Sign}_{sk}(m)) = 1.$$

Security?

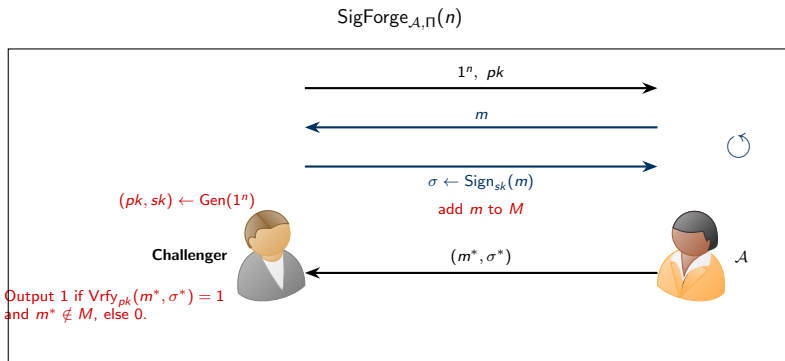
Attacker's power

- The **public key** is given to the attacker.
- *Adaptive chosen-message attack*: the attacker can request signatures on messages of its choice.

Security goal

Existential unforgeability: the attacker cannot forge a valid signature on **any** message not already signed by the signer.

SigForge _{$\mathcal{A}, \Pi$} : Challenger vs. Adversary



Compare with MacForge

The same game, except that $\mathcal{A}$ also gets the **public key** pk — and with it the power to verify by itself.

Unforgeability of Digital Signatures

Game $\text{SigForge}_{\mathcal{A}, \Pi}(n)$

- 1 $(pk, sk) \leftarrow \text{Gen}(1^n)$. $\mathcal{A}$ is given 1^n , pk , and **oracle access** to $\text{Sign}_{sk}(\cdot)$. Let M be the set of messages $\mathcal{A}$ queries.
- 2 $\mathcal{A}^{\text{Sign}_{sk}(\cdot)}$ outputs a pair (m^*, σ^*) .
- 3 Output 1 if $\text{Vrfy}_{pk}(m^*, \sigma^*) = 1$ and $m^* \notin M$, and 0 otherwise.

Definition 2 (Existential unforgeability)

Π is **existentially unforgeable under an adaptive chosen-message attack** if for all PPT $\mathcal{A}$:

$$\Pr [\text{SigForge}_{\mathcal{A}, \Pi}(n) = 1] \leq \text{negl}(n).$$

Security: Replay Attacks

What about replay attacks?

- Same issue as for MACs: a valid (m, σ) pair can be replayed.
- Not captured by the unforgeability definition — left to the **application designer** (e.g. include nonces, timestamps, or counters).

Signatures from One-Way Functions

Theorem

Digital signatures can be realized based on **any one-way function**.

(We will not see this construction. Instead: practical schemes from RSA and from DLog.)

Public-Key Infrastructure (PKI)

The problem

How does Alice get **Bob's** public key (authentically)?

- Use digital signatures for secure public-key distribution.
- Suppose everyone knows the public key of a trusted party: a **Certification Authority** (CA).

Certification Authority

Public key PK_{CA} , secret key SK_{CA} .

What Does Bob Do?

- Bob goes to the CA with his public key PK_{Bob} .
- The CA issues a **certificate**

$$cert_{CA \rightarrow Bob} = \text{Sign}_{SK_{CA}}(\text{Bob} \parallel PK_{Bob}).$$

- Given this certificate and PK_{CA} , Alice can verify that PK_{Bob} is indeed Bob's public key.
- How does Alice get the CA's key in the first place?

How Does Alice Get the CA's Key in the First Place?

Bootstrapping trust

- PK_{CA} must be obtained through a **trusted out-of-band channel**.
- In practice it is **pre-installed**: browsers and operating systems ship with a set of trusted root CA public keys.
- Trust is hierarchical: root CAs sign intermediate CAs, which sign end entities (a certificate chain).

Construction

- $\text{Gen}(1^n)$: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$; $pk = (N, e)$, $sk = d$.
- $\text{Sign}_{sk}(m)$, for $m \in \mathbb{Z}_N^*$: output $\sigma := m^d \bmod N$.
- $\text{Vrfy}_{pk}(m, \sigma)$: output 1 iff $\sigma^e = m \bmod N$.
- **Correctness**: $\sigma^e = m^{ed} = m \bmod N$ (lecture 15).
- **Intuition**: only the holder of d can compute e -th roots modulo N — the RSA assumption.

Is Textbook RSA Secure?

No! Two forgeries

- ① **No-message forgery:** pick any $\sigma \in \mathbb{Z}_N^*$ and output $(m := \sigma^e \bmod N, \sigma)$. It is valid. $\mathcal{A}$ cannot choose m — but existential unforgeability rules out a forgery on *any* new message.
- ② **Multiplicative forgery:** given signatures σ_1, σ_2 on m_1, m_2 , the product $\sigma_1\sigma_2$ is a valid signature on m_1m_2 :

$$(\sigma_1\sigma_2)^e = m_1m_2 \bmod N.$$

Querying m_1 and $m_2 := m/m_1$ gives a signature on **any** target m .

Lesson

The signer should take the e -th root of something the adversary **cannot control**.

Hash-and-Sign

A method to sign messages of arbitrary length

Ingredients

- A signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ for **short** messages (length $\ell(n)$).
- A hash function $\Pi_H = (\text{Gen}_H, H)$ with output length $\ell(n)$.

New signature scheme Π'

$\text{Gen}'(1^n): (pk, sk) \leftarrow \text{Gen}(1^n), s \leftarrow \text{Gen}_H(1^n);$ public key (pk, s) , secret key sk .

$$\begin{aligned}\text{Sign}'_{sk}(m) &:= \text{Sign}_{sk}(H_s(m)), \\ \text{Vrfy}'_{pk,s}(m, \sigma) &:= \text{Vrfy}_{pk}(H_s(m), \sigma).\end{aligned}$$

Hash-and-Sign: Proof of Security

Theorem 3

If Π is existentially unforgeable and Π_H is collision resistant, then Π' is existentially unforgeable.

$\mathcal{A}$ queries $m_1, \dots, m_q$ and outputs (m^*, σ^*) with $m^* \notin \{m_1, \dots, m_q\}$. Let Coll :
 $H_s(m^*) = H_s(m_i)$ for some i .

Case Coll: $\mathcal{B}_1$ breaks H

Given s : run Gen itself, answer $\text{Sign}_{sk}(H_s(m))$; output (m^*, m_i) — a collision.

Case $\neg\text{Coll}$: $\mathcal{B}_2$ breaks Π

Given pk and a Sign_{sk} oracle: sample s , sign $H_s(m)$ via the oracle; output $(H_s(m^*), \sigma^*)$ — a new message.

$$\begin{aligned}\Pr [\text{SigForge}_{\mathcal{A}, \Pi'}(n) = 1] &\leq \Pr [\text{Hash-coll}_{\mathcal{B}_1, \Pi_H}(n) = 1] + \Pr [\text{SigForge}_{\mathcal{B}_2, \Pi}(n) = 1] \\ &\leq \text{negl}_1(n) + \text{negl}_2(n).\end{aligned}$$

The analogue of the **hash-and-MAC** paradigm (lecture 11), with the same two reductions.

Hash-and-Sign for Textbook RSA?

- Textbook RSA signs only $m \in \mathbb{Z}_N^*$. Apply hash-and-sign: $\sigma := H_s(m)^d \bmod N$?
- The theorem does **not** apply: it needs a scheme that is secure for short messages, and textbook RSA is not.
- Yet hashing seems to block both forgeries: picking σ first gives σ^e , but a forgery needs an m with $H_s(m) = \sigma^e$; and $H_s(m_1) \cdot H_s(m_2)$ has nothing to do with $H_s(m_1 m_2)$.

Next lecture

A proof — in an idealized model in which H is a truly random function (the **random-oracle model**).

Schnorr Signature Scheme: A Detour

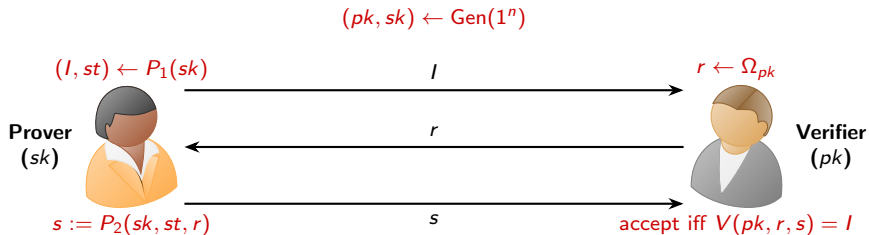
Plan

We first study the **Schnorr identification scheme**. Next lecture, the **Fiat–Shamir** transform turns it into a signature scheme.

Identification scheme

- Allows a **prover** to convince a **verifier** of its identity: that it knows the sk matching pk .
- Consists of four algorithms $(\text{Gen}, P_1, P_2, V)$.

Identification Scheme



Correctness

An honest prover can always (or almost always) convince the verifier.

Security of Identification Schemes

An adversary that does **not** know sk should not fool the verifier — even after **passively eavesdropping** on many honest executions.

Game $\text{Ident}_{\mathcal{A}, \Pi}(n)$

- ① $(pk, sk) \leftarrow \text{Gen}(1^n)$. $\mathcal{A}$ is given 1^n , pk , and oracle access to Trans_{sk} , which outputs the transcript (l, r, s) of a fresh honest execution.
- ② $\mathcal{A}$ outputs l^* .
- ③ $r^* \leftarrow \Omega_{pk}$ is given to $\mathcal{A}$, which outputs s^* .
- ④ Output 1 if $V(pk, r^*, s^*) = l^*$, and 0 otherwise.

Definition 4 (Passive security)

$\Pi = (\text{Gen}, P_1, P_2, V)$ is **secure under a passive attack** if for all PPT $\mathcal{A}$:

$$\Pr [\text{Ident}_{\mathcal{A}, \Pi}(n) = 1] \leq \text{negl}(n).$$

The Schnorr Identification Scheme

Gen(1^n)

$(G, q, g) \leftarrow \mathcal{G}(1^n)$ (prime order q); $x \leftarrow \mathbb{Z}_q$, $h := g^x$. Output $pk = (G, q, g, h)$ and $sk = x$.

$P_1(sk)$

Sample $k \leftarrow \mathbb{Z}_q$. Output $I := g^k$ and $st := (I, k)$.

Challenge space

$\Omega_{pk} = \mathbb{Z}_q$: $r \leftarrow \mathbb{Z}_q$.

Correctness: $\frac{g^s}{h^r} = \frac{g^{rx+k}}{g^{xr}} = g^k = I$.

$P_2(sk, st, r)$

Output $s := rx + k \bmod q$.

$V(pk, r, s)$

Output g^s/h^r (accept iff it equals I).

The Transcript Oracle Trans_{sk} Is Useless

Simulating transcripts without sk

- 1 Sample $r, s \leftarrow \mathbb{Z}_q$.
- 2 Set $l := g^s / h^r$.
- 3 Output (l, r, s) .

Verification passes, and sk was **not** used.

- **Same distribution** as a real transcript: there, r is uniform, $s = rx + k$ is uniform (because k is) and independent of r , and $l = g^s / h^r$ is then determined.

Consequence

A reduction can answer Trans_{sk} queries by itself. So it suffices to prove security **without** the Trans_{sk} oracle.

Proof Without Trans_{sk} : Rewinding

Rewinding

Once $\mathcal{A}$ outputs I , we can run it on two **different** challenges $r_1 \neq r_2$, from the same state.

Extracting the secret key

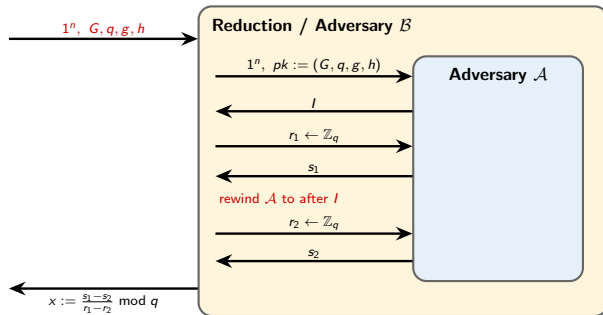
Suppose $\mathcal{A}$ produces accepting responses s_1 and s_2 for both challenges. Then

$$\frac{g^{s_1}}{h^{r_1}} = I = \frac{g^{s_2}}{h^{r_2}} \implies g^{s_1 - r_1 x} = g^{s_2 - r_2 x} \implies s_1 - r_1 x = s_2 - r_2 x \pmod{q}.$$

Solve for x (since $r_1 \neq r_2$ and q is prime):

$$x = \frac{s_1 - s_2}{r_1 - r_2} \pmod{q}.$$

Schnorr Is Secure: Proof by Reduction



Success probability (without Trans_{sk})

If $\mathcal{A}$ wins Ident with probability $\epsilon(n)$, both runs succeed with $r_1 \neq r_2$ with probability at least $\epsilon(n)^2 - 1/q$ (averaging over $\mathcal{A}$'s state after I). Then $\mathcal{B}$ solves DLog : non-negligible if ϵ is. $\square$

Summary

- **Digital signatures** are the public-key analogue of MACs: public verification and **non-repudiation**.
- Syntax (Gen, Sign, Vrfy); security is existential unforgeability under an adaptive chosen-message attack (game $\text{SigForge}_{\mathcal{A}, \Pi}(n)$).
- **PKI**: a trusted CA certifies (identity, public key) pairs — using only the definition. Signatures exist from any one-way function.
- **Textbook RSA signatures** $\sigma = m^d \bmod N$ are insecure: no-message and multiplicative forgeries.
- **Hash-and-sign** extends short-message signatures to any length. *Proved*: a forgery is a collision ($\mathcal{B}_1$) or a forgery on a new hash value ($\mathcal{B}_2$). It does not repair textbook RSA by itself.
- **Schnorr identification**: passively secure under DLog. Transcripts can be simulated without sk ; rewinding on two challenges extracts $x = \frac{s_1 - s_2}{r_1 - r_2}$.
- *Next lecture*: the **random-oracle model**; **RSA-FDH** $\sigma = H(m)^d$; and **Fiat-Shamir**, which turns the identification scheme into **Schnorr signatures**.