Digital signatures

Introduction to cryptology

Bruno Grenet

M1 INFO, MOSIG & AM

Université Grenoble Alpes – IM²AG

https://membres-ljk.imag.fr/Bruno.Grenet/IntroCrypto.html
https://membres-ljk.imag.fr/Pierre.Karpman/tea.html

Introduction

Goal: authenticity of a message, in the context of public key cryptography

- The sender *signs* a message m with a private key $sk \to$ *signature* σ
- Anyone, with the sender's public key pk, can *verify* the signature σ

Compare with MACs

- Public key/private key instead of a single key
- *tag* $\to$ *signature*

Advantages compared to MAC

Public verification: using the signer's public key

Transfer: a signed message can be forwarded with its signature

Non-repudiation: the signer cannot deny having signed

Examples of use

Vaccine pass

- Vaccination $\rightarrow$ signature (QR code) with the authorities' private key
- Verification $\rightarrow$ anyone can verify, with the authorities' public key

Authenticated email

- Alice publishes her public key pk_A
- When Alice sends an email, she sends it together with the corresponding signature
- The recipient can verify that the sender is Alice or… knows Alice's secret key!

Software distribution

- A software company distributes softwares with a signature
- Users (customers) download a software and check the signature before installing it

Certificates

- How can one be sure that pk_A really is Alice's public key?
- A *certificate authority* signs pk_A using its own secret key
- Web or tree of certificates

Contents

Digital signature scheme

Definition

A signature scheme is given by three algorithms:

$\mathsf{Gen}_n()$ generates a pair of keys (pk, sk) — n usually implicit

$\mathsf{Sign}_{sk}(m)$ computes a *signature* σ for m

$\mathsf{Vrfy}_{pk}(m, \sigma)$ returns 1 if the signature is *valid*, and 0 otherwise

Correction

The scheme is *correct* if for all $(pk, sk) \leftarrow \mathsf{Gen}()$ and $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$, $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$

Compare (again) with MACs

- Public key/private key instead of a single key
- *tag* $\to$ *signature*
- Mac $\to$ Sign

Security notions for digital signatures

Goals: unforgeability

Should be hard for an adversary to produce a valid signature without knowing the secret key

- Existential forgery: produce any pair (m, σ) such that $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$
- Universal forgery: given m, produce σ such that $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$

Means

- Key-Only Attack: the adversary only knows the public key
- Known Message Attack: the adversary knows some valid pairs (m_i, σ_i)
- Chosen Message Attacks: the adversary can query signatures for messages m_i
  - Generic: queries must be sent before knowing the public key
  - Non-adaptative: all queries must be sent before receiving any signature
  - Adaptative: queries can be made adaptively after receiving some signatures

Strongness

- Standard: Adversary must sign a message for which it does not know any signature
- Strong: Adversary must produce a new signature

A formal definition of security

Existential Unforgeability Game

Challenger $(pk, sk) \leftarrow \mathsf{Gen}()$
Adversary queries messages m_i and gets valid signatures $\sigma_i \leftarrow \mathsf{Sign}_{sk}(m_i), 1 \leq i \leq q$
Adversary outputs a candidate pair (m, σ) where $m \notin \{m_1, \ldots, m_q\}$

Advantage

- Advantage of A: $\mathsf{Adv}^{\mathsf{EUF-CMA}}_{\mathsf{Sign/Vrfy}}(A) = \Pr\left[\mathsf{Vrfy}_{pk}(m, \sigma) = 1\right]$
- Advantage function:

$$\mathsf{Adv}^{\mathsf{EUF-CMA}}_{\mathsf{Sign/Vrfy}}(q, t) = \max_{A_{q,t}} \mathsf{Adv}^{\mathsf{EUF-CMA}}_{\mathsf{Sign/Vrfy}}(A_{q,t})$$

where $A_{q,t}$ denotes an algorithm making $\leq q$ queries with running time $\leq t$

Note

- Exactly identical to the definition for a MAC!

Contents

General principle

Identification protocol: prove one's identity to an interlocutor

Players: A *prover*: owns a secret key *sk*

A *verifier*: knows the corresponding public key *pk*

Goals for the prover:

- convince the verifier that they knows the secret key *sk*
- without revealing *anything* about *sk* to the verifier

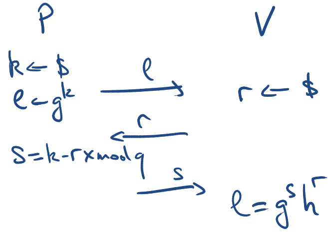
Fiat-Shamir construction

- Given an identification protocol, we can build a signature scheme

Schnorr's protocols

- Identification protocol
- Signature scheme *via* the Fiat-Shamir construction
- Example: DSA & ECDSA are variants of Schnorr's scheme

Schnorr identification protocol (1989)



Protocol definition

Public: a (cyclic) group G of *prime* order q, with generator g

Keys: $sk = x \in \{0, \ldots, q-1\}$ and $pk = h = g^x$ (public)

Protocol:

1. Prover: $k \twoheadleftarrow \{0, ..., q-1\}$; $\ell \leftarrow g^k$; Send ℓ
2. Verifier: $r \twoheadleftarrow \{0, ..., q-1\}$; Send r — r: the *challenge*
3. Prover: $s \leftarrow (k - r \cdot x) \bmod q$; Send s — using $sk = x$
4. Verifier: accept iff $\ell = g^s \cdot h^r$ — using $pk = h$

Correction

$$\ell = g^k \qquad g^s h^r = g^s g^{xr} = g^{s + xr \bmod q} = g^k = \ell$$

Security definition

Game: an adversary observes several *transcripts*, and tries to impersonate a Prover

Advantage: probability for the adversary to convince a verifier

Schnorr identification security: proof sketch

Theorem

If the discrete logarithm problem is hard in G, Schnorr identification protocol is secure:
If an adversary is able to convince a verifier, it can compute discrete logarithms in G

Hyp: A_0 is able to convince the verifier

→ A_0 runs the protocol twice, with the same value of k
and gets two challenges r_1 and r_2 from the verifier
and answers with s_1 and s_2

→ the verifier accepts both s_1 and s_2

$$\Rightarrow g^k = g^{s_1} h^{r_1} = g^{s_2} h^{r_2} \Rightarrow g^{s_1 + r_1 x} = g^{s_2 + r_2 x} \Rightarrow s_1 - s_2 = x(r_2 - r_1) \bmod q$$

$$\Rightarrow x = (s_1 - s_2)(r_2 - r_1)^{-1} \bmod q$$

Missing in the proof: technically involved argument to show that if A_0 has prob. ε to convince the verifier, then A_0 has prob. $\geq \varepsilon^2 - \varepsilon/q$ to compute x.

Fiat-Shamir construction (1986)

Build a signature scheme from an identification protocol

Requires: an identification protocol and a hash function

Builds: a signature scheme

$\mathsf{Sign}_{sk}(m)$: simulation of the identification protocol where the challenge is produced by the hash function; the signature is the challenge and the answer

$\mathsf{Vrfy}_{pk}(\sigma)$: check that the answer is consistent with the challenge

Theorem (admitted) *Pointcheval, Stern (1996)*

If the identification protocol is secure and *H is random*, the resulting signature scheme is EUF-CMA secure

Remarks

- An identification protocol is an interactive *zero-knowledge proof* ZKP
- Fiat-Shamir construction turns any ZKP into a *non-interactive* one NIZKP

Schnorr signature scheme (1989)

Protocol description

Public: A cyclic group G of order $q \simeq 2^n$ and generator g, $H : \{0,1\}^* \to G$

Keys: $sk = x \twoheadleftarrow \{0, \ldots, q-1\}$ and $pk = h \leftarrow g^x$

$\text{Sign}_{sk}(m)$: *Simulation of the identification protocol:* $m \in \{0,1\}^*$

x

1. $k \twoheadleftarrow \{0, \ldots, q-1\}$; $\ell \leftarrow g^k$
2. $r \leftarrow H(\ell \| m)$; $s \leftarrow k - rx \bmod q$ — challenge and answer
3. Return the signature (r, s)

$\text{Vrfy}_{pk}(m, r, s)$:

h

1. $\ell \leftarrow g^s \cdot h^r$
2. Accept iff $H(\ell \| m) = r$

Correction

$\ell = g^s h^r$ as before + "H(ℓ‖m) = H(ℓ‖m)"

Theorem

Pointcheval, Stern (1996)

If the DLP is hard in G and *H is random*, Schnorr signature is EUF-CMA secure

Contents

Hash-and-sign

Rationale

- Signature schemes are less efficient than MACs
- Some signature schemes are designed for fixed-length messages only

Obvious idea

- Compute the signature of a hash of the message, rather than the message
- Remark: used in Schnorr's signature scheme

Construction

Given a signature scheme $(\mathrm{Sign}, \mathrm{Vrfy})$ for fixed-length messages $m \in \mathcal{M}$
a hash function $H : \{0,1\}^* \to \mathcal{M}$

Build a signature scheme $(\mathrm{Sign}', \mathrm{Vrfy}')$ for messages in $\{0,1\}^*$:
$\mathrm{Sign}'_{sk}(m)$: $\mathrm{Sign}_{sk}(H(m))$
$\mathrm{Vrfy}'_{pk}(m, \sigma)$: $\mathrm{Vrfy}_{pk}(H(m), \sigma)$

Hash-and-sign security

Theorem

If (Sign, Vrfy) is EUF-CMA secure and H is collision resistant, then (Sign′, Vrfy′) is EUF-CMA secure

Hyp: $\mathcal{A}$ an adversary against (Sign',Vrfy')

- Queries m_i ↝ signatures σ_i = Sign'_sk(m_i) = Sign_sk(H(m_i))
- Produces a valid pair (m,σ)

Case 1 : there exists m_i such that H(m) = H(m_i) ↝ H is not collision resistant

Case 2 : for all m_i, H(m) ≠ H(m_i).

Let h = H(m) and h_i = H(m_i) for all i.

$\mathcal{A}$ knows (h_i, σ_i) and computes (h,σ) such that Vrfy_pk(h,σ) = 1

⇒ (Sign,Vrfy) is not EUF-CMA secure

Remark: Add probabilities for a real proof

Signcryption

Combine signature and public-key encryption

A problem with *Encrypt-then-sign*

Keys: (pk_S, sk_S) for the Sender and (pk_R, sk_R) for the Recipient

Sender computes $c \leftarrow \mathrm{Enc}_{pk_R}(m)$ and $\sigma \leftarrow \mathrm{Sign}_{sk_S}(c)$

Recipient decrypts c using $\mathrm{Dec}_{sk_R}(c)$ and verifies it with $\mathrm{Vrfy}_{pk_S}(\sigma)$

Adversary intercepts c and computes $\sigma_A \leftarrow \mathrm{Sign}_{sk_A}(c)$

$\rightarrow$ the adversary can pretend to be the sender

Workaround

- Each user X has a unique *identity* id_X
- Each participant can obtain the public-key pk_X associated to id_X
- Signature of the message or ciphertext *and the identity*

Secure *signcryption*

Two examples

Encrypt-then-sign: $c \leftarrow \mathsf{Enc}_{pk_R}(m); \sigma \leftarrow \mathsf{Sign}_{sk_S}(c \| id_S)$
Sign-then-encrypt: $\sigma \leftarrow \mathsf{Sign}_{sk_S}(m); c \leftarrow \mathsf{Enc}_{pk_R}(m \| \sigma \| id_S)$

Security definition

IND-CCA: standard game/advantage, but including the signature
INT-CTXT: game of *ciphertext forgery* ciphertext integrity

Result (informally)

Both *Encrypt-then-Sign* and *Sign-then-Encrypt* are secure if the encryption scheme and the signature schemes are (sufficiently) secure

Public-Key Infrastructures

Where do I find public-keys? How to be sure of the real owner of a key?

Certificates

- $\text{cert}_{B\to C} = \text{Sign}_{sk_B}(id_C \| pk_C)$: B certifies that C's public-key is pk_C
- If A trusts B:
  - C can send pk_C together with $\text{cert}_{B\to C}$
  - A can verify $\text{cert}_{B\to C}$ and accept pk_C as the public-key of C

Certificate authorities and chains

Certificate authority: trusted entities, used as roots in certificate chains *e.g* DigiCert

Certificate chains: trees of certifications, from authorities to end users

Certificate revokation

- Short-lived certificates: add an expiration date $cert_{B\to C} = \text{Sign}_{sk_B}(id_C \| pk_C \| T)$
- Certification revokation lists, using a serial number for each certificate

Conclusion

Signature scheme

- Goals:
  - Authenticity: *identity of the sender*
  - Non-repudiation: *commitment of the sender*
- Asymmetric (and more powerful!) version of MACs

Constructions

- Based on the same problems as asymmetric encryption (discrete log., RSA, LWE, ...)
- Combination with hashing for efficiency
- Links with zero-knowledge proofs
- Public-key infrastructures: a whole subject!

Authentication without encryption can be useful...

... encryption without authentication is useless!