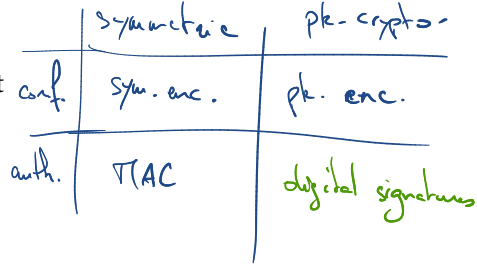
Lecture 8. Digital signatures

Active adversaries, no shared secret

Bruno Grenet



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

Introduction to cryptology
Université Grenoble Alpes – IM²AG
M1 INFO, MOSIG & AM

Introduction

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

- The sender *signs* a message m with a signing key sk $\rightarrow$ *signature* σ
- Anyone, with the sender's verification key vk, can *verify* the signature σ

Compare with MACs

- Pair (signing key, verification key) instead of a single key
- *tag* $\rightarrow$ *signature*

Advantages compared to MAC

Public verification: using the signer's verification 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 → signature (QR code) with the authorities' signing key
- Verification → anyone can verify, with the authorities' verification key

Authenticated email

- Alice publishes her verification key vk_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 signing 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 vk_A really is Alice's verification key?
- A *certificate authority* signs vk_A using its own signing key
- Web or tree of certificates

Contents

Digital signature scheme

Definition

A signature scheme is given by three algorithms:

$\mathrm{Gen}_n()$	generates a pair of keys $(\mathrm{vk}, \mathrm{sk})$	n usually implicit
$\mathrm{Sign}_{\mathrm{sk}}(m)$	computes a *signature* σ for m	
$\mathrm{Vrfy}_{\mathrm{vk}}(m, \sigma)$	returns 1 if the signature is *valid*, and 0 otherwise	

Correction

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

Compare (again) with MACs

- Signing key/verification key instead of a single key
- *tag* $\rightarrow$ *signature*
- Mac $\rightarrow$ Sign

Public and private/secret keys

- Signing key $\leftrightarrow$ *private/secret* key
- Verification key $\leftrightarrow$ *public* key

Security notions for digital signatures

Goals: unforgeability

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

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

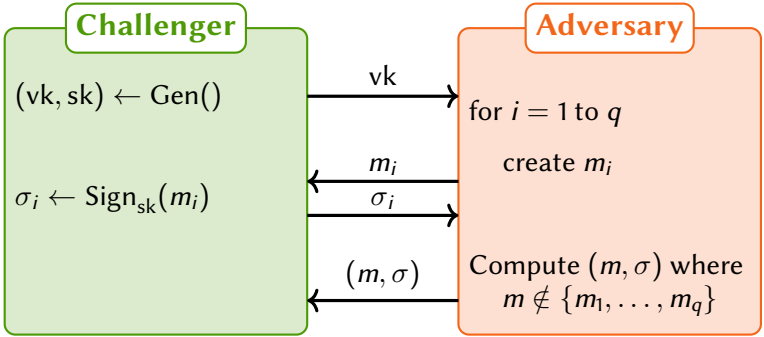
Means

- ~~Key-Only Attack: the adversary only knows the verification 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 verification 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

EUF-CMA: Existential UnForgeability under Chosen Message Attack



Advantages

- Advantage of $\mathcal{A}$: $\mathrm{Adv}^{\text{EUF-CMA}}_{\text{Sign/Vrfy}}(\mathcal{A}) = \Pr\left[\mathrm{Vrfy}_k(m, \sigma) = 1\right]$
- Advantage function: $\mathrm{Adv}^{\text{EUF-CMA}}_{\text{Sign/Vrfy}}(q, t) = \max_{\mathcal{A}_{q,t}} \mathrm{Adv}^{\text{EUF-CMA}}_{\text{Sign/Vrfy}}(\mathcal{A}_{q,t})$

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

- Same definitions as for Mac security

A common misconception

Signatures are not "encryption with the decryption key"

The bad idea

Given (Enc, Dec) with the same encryption and decryption key spaces, define

- $\mathsf{Sign}_{\mathsf{sk}}(m) = \mathsf{Enc}_{\mathsf{sk}}(m)$ $\qquad \mathsf{sk} = \mathsf{dk}$
- $\mathsf{Vrfy}_{\mathsf{vk}}(m, \sigma) = [\mathsf{Dec}_{\mathsf{vk}}(\sigma) = m?]$ $\qquad \mathsf{vk} = \mathsf{ek}$

Existential forgery

- Given σ, everybody can compute $m = \mathsf{Dec}_{\mathsf{vk}}(\sigma)$
- Existential forgery as long as σ is a well-formed ciphertext

Inefficiency

- Signature longer than the message!

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 verification key vk

Goals for the prover:

- convince the verifier that it knows the secret (key) sk
- without revealing *anything* about sk to the verifier

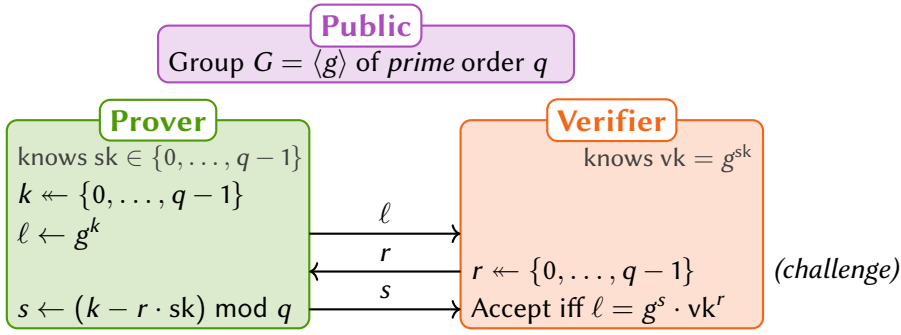
Fiat-Shamir construction

- 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)



Correction

$$g^s \cdot \text{vk}^r = g^{s + r \cdot \text{sk}} = g^{(k - r \cdot \text{sk}) + r \cdot \text{sk}} = g^k = \ell$$

Security definition

Game: an adversary plays the role of the Prover, without knowing sk

Advantage: probability for the adversary to convince the verifier

Schnorr identification security: proof sketch

Theorem

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

Proof sketch You are given $h \in G$ and want to compute x s.t. $h = g^x$.

- Play the identification game against A_0 with $vk = h$ (known) and $sk = x$ (unknown)
- Run the adversary A_0 twice with the same value $t = g^k$.
  - Let r_1 and r_2 be the two challenges and s_1, s_2 the resp. answers.
  - If A_0 is twice successful, $t = g^{s_1} \cdot vk^{r_1} = g^{s_2} \cdot vk^{r_2}$
    then $g^{s_1 + x \cdot r_1} = g^{s_2 + x r_2}$, or $s_1 + x r_1 = s_2 + x s_2$ in $\mathbb{F}_q$
    then you can compute $x = (s_1 - s_2)(r_2 - r_1)^{-1}$

Fiat-Shamir construction (1986)

Build a signature scheme from an identification protocol

Requires: an identification protocol and a hash function
Builds: a signature scheme
$\mathrm{Sign}_{\mathrm{sk}}(m)$: simulation of the identification protocol where the challenge is produced by the hash function; the signature is the challenge with its answer
$\mathrm{Vrfy}_{\mathrm{vk}}(\sigma)$: check that the answer is consistent with the challenge

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

If the identification protocol is secure and *H is a random oracle*, 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 $G = \langle g \rangle$ of order $q \simeq 2^n$ and $H : \{0,1\}^* \to G$

Keys: $\mathsf{sk} \twoheadleftarrow \{0, \ldots, q-1\}$ and $\mathsf{vk} \leftarrow g^{\mathsf{sk}}$

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

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

$\mathsf{Vrfy}_{\mathsf{vk}}(m, r, s)$: 1. $\ell \leftarrow g^s \cdot \mathsf{vk}^r$

2. Accept iff $H(\ell \| m) = r$

Correction

$g^s \cdot \mathsf{vk}^r = g^{s + \mathsf{sk} \cdot r} = g^k$ as before, and $H(\ell \| m) = H(\ell \| m)$

Theorem

Pointcheval, Stern (1996)

If the DLP is hard in G and H *is a random oracle*, 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 $(\mathsf{Sign}, \mathsf{Vrfy})$ for fixed-length messages $m \in \mathcal{M}$
a hash function $H : \{0,1\}^* \to \mathcal{M}$

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

Hash-and-sign security

Theorem

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

Proof sketch Assume that we are given an adversary A against $(\text{Sign}', \text{Vrfy}')$.

- A queries signatures for $m_1, \ldots, m_q$ and gets $\begin{cases} h_i = H(m_i) \\ \sigma_i = \text{Sign}_{sk}(h_i) \end{cases}$
- At the end, A produces m, h, σ s.t. $h = H(m)$, $\sigma = \text{Sign}_{sk}(h)$.

Two cases:

1. $h = h_i$ for some i: A produced a collision for H $\Rightarrow C$
2. $h \neq h_i$ for all i: in this case, A produced an existential forgery (h, σ) against $(\text{Sign}, \text{Vrfy})$. $\Rightarrow B$

With more work: $\text{Adv}^{\text{EUF-CMA}}_{\text{Sign}'}(A) \leq \text{Adv}^{\text{EUF-CMA}}_{\text{Sign}}(B) + \text{Adv}^{\text{coll}}_{H}(C)$

Signcryption

Combine signature and public-key encryption

A problem with *Encrypt-then-sign*

Keys: $(\mathsf{vk}_S, \mathsf{sk}_S)$ for the Sender and $(\mathsf{ek}_R, \mathsf{dk}_R)$ for the Recipient

Sender computes $c \leftarrow \mathrm{Enc}_{\mathsf{ek}_R}(m)$ and $\sigma \leftarrow \mathrm{Sign}_{\mathsf{sk}_S}(c)$

Recipient decrypts c using $\mathrm{Dec}_{\mathsf{dk}_R}(c)$ and verifies it with $\mathrm{Vrfy}_{\mathsf{vk}_S}(\sigma)$

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

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

Workaround

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

Secure *signcryption*

Two examples

Encrypt-then-sign: $c \leftarrow \mathrm{Enc}_{ek_R}(m); \sigma \leftarrow \mathrm{Sign}_{sk_S}(c \| id_S)$ $\xrightarrow{c,\sigma}$ $m \leftarrow \mathrm{Dec}_{dk_R}(c)$; $\mathrm{Vrfy}_{vk_S}(c \| id_S, \sigma)$

Sign-then-encrypt: $\sigma \leftarrow \mathrm{Sign}_{sk_S}(m); c \leftarrow \mathrm{Enc}_{ek_R}(m \| \sigma \| id_S)$ $\xrightarrow{c}$ $m \| \sigma \| i \leftarrow \mathrm{Dec}_{dk_R}(c)$

$\mathrm{Vrfy}_{vk_S}(m, \sigma)$ and $i = id_S$

Security definition

(*Cf. Authenticated Encryption with Associated Data*)

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

Remark

- Encrypt-then-MAC was secure, but MAC-then-Encrypt was not…
- … but the reason is that it was based on IND-C**P**A secure encryption

Additional properties

Forward secrecy

- Knowing the Sender's signing key should not help decrypt the ciphertext
  - If the sender is corrupted, encrypted messages remain safe

Non-repudiation

- Knowing the Recipient's decryption key should not help signing a ciphertext
  - If the recipient is corrupted, the sender cannot be impersonated
  - The sender cannot pretend it was impersonated

Examples

- Sign-then-encrypt satisfies forward secrecy
- Encrypt-then-sign satisfies non-repudiation
- Possible to get both, but requires some care

Public-Key Infrastructures

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

Certificates

- $\text{cert}_{B \to C} = \text{Sign}_{\text{sk}_B}(id_C \| \text{vk}_C)$: B certifies that C's verification key is vk_C
- If A trusts B:
  - C can send vk_C together with $\text{cert}_{B \to C}$
  - A can verify $\text{cert}_{B \to C}$ and accept vk_C as the verification 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 $cert_{B \to C} = \text{Sign}_{\text{sk}_B}(id_C \| \text{vk}_C \| T)$
- 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 mostly useless!