Introduction to Cryptography

F. Koeune – O. Pereira

MAT2450 – Slides 8

Message integrity revisited

Goal: ensure integrity and origin of message

We had a solution with MACs
However, same key used to generate and check MACs
$\Rightarrow$ Anyone who can check a MAC can also *forge* one

- Cannot work if *all* participants do not trust each other
- Cannot be used to prove a commitment to a third party (see why?)
- ...

Couldn't asymmetric cryptography provide us with a better solution?

Digital signature

Principle

- Bob generates a key pair pk, sk
- sk is kept secret, and used to *sign* messages
- pk is made public, and used to *verify* signatures

So

- Only Bob can produce signatures
- Anyone (who has Bob's public key) can verify that Bob's signature is authentic

Advantages over MAC (1)

Simpler key management

- Signature: with one key pair, Bob can send authenticated messages to as many users as he wants
- MAC: typically, Bob will need one key per contact

Publicly verifiable

- Signature: if Alice receives a message signed by Bob, she knows that everyone else will also consider it authentic
- MAC: Bob could have sent a valid $\mathrm{Mac}_k(m)$ to Alice, but an invalid $\mathrm{Mac}_{k'}(m)$ to Steve

Advantages over MAC (2)

Transferable

- Signature: Alice knows she can bring a signed message to a third party (e.g. a judge) and convince him that Bob signed the message
- MAC:
  - Alice would need to reveal the key to the third party
  - Even if she does, Alice could have generated the MAC herself

Non repudiable

- Bob cannot later deny that he had signed the message

So, why would we use MACs?

Because electronic signature is more expensive

Same as "symmetric vs. asymmetric encryption" argument: 100-1000 times less efficient (for short messages, at least)

Because they are very useful to strengthen symmetric encryption: non-malleable/authenticated encryption, ...

Definition

A signature scheme is a triple $\Pi := \langle \mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy} \rangle$

- Gen probabilistically selects $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$
  pk/sk are the public/private key
- Sign provides $\sigma \leftarrow \mathrm{Sign}_{sk}(m)$
- Vrfy outputs a bit $b := \mathrm{Vrfy}_{pk}(m, \sigma)$
  (1 meaning valid, 0 meaning invalid)

s.t. $\forall n, (pk, sk) \leftarrow \mathrm{Gen}(1^n), m$:

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

(except, possibly, with negligible probability)

Remark: can be defined

- For fixed-length messages
- For arbitrary-length messages

Usage

1. Bob uses Gen (once and for all) to generate a key pair (pk, sk)
2. Bob advertises pk (website, directory, ...)
3. When he wants to transmit m, Bob computes $\sigma \leftarrow \mathrm{Sign}_{sk}(m)$ and sends (m, σ)
4. Receiver retrieves pk
5. Receiver checks that $\mathrm{Vrfy}_{pk}(m, \sigma) = 1$. This ensures that
   - m originates from Bob
   - m has not been modified

Remarks

- Does not say *when* m was emitted
- Nor that m is not a replay

$\Rightarrow$ Specific measures have to be added if this is a concern

Key distribution

This scheme assumes that recipient can obtain a valid and authentic copy of *pk*

- Difficulty not to be underestimated
- We will come back to it

But, at least, we need to do it only once

Secure signature

Now we need to define security

Intuition: no adversary can produce a valid signature for a message that was not previously signed, even if he can obtain signatures of messages of his choice.

This would be called a *forgery*

Secure signature

Define the signature forgery experiment $\text{Sig-forge}_{\mathcal{A},\Pi}(n)$

1. Choose $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$
2. $\mathcal{A}$ receives pk and oracle access to $\mathrm{Sign}_{sk}(\cdot)$ for messages of his choice (denote by $\mathcal{Q}$ the set of these messages)
3. $\mathcal{A}$ outputs (m, σ)
4. Define $\text{Sig-forge}_{\mathcal{A},\Pi}(n) := 1$ iff $\mathrm{Vrfy}_{pk}(m, \sigma) = 1$ and $m \notin \mathcal{Q}$

Secure signature

A signature scheme $\Pi = \langle \mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy} \rangle$ is *existentially unforgeable under an adaptive chosen-message attack* (EUF-CMA) if $\forall$ PPT $\mathcal{A}$, $\exists$ negl. ϵ:

$$\Pr[\text{Sig-forge}_{\mathcal{A},\Pi}(n) = 1] \leq \epsilon(n)$$

Schnorr's Signature Scheme

NI Schnorr can be turned into a signature scheme:

- just hash message together with a!

Schnorr's signature scheme:

- $\mathrm{Gen}(1^n)$ runs $\mathcal{G}(1^n)$ to obtain $\langle \mathbb{G}, q, g \rangle$, then picks $x \leftarrow \mathbb{Z}_q$, sets $h = g^x$ and returns $(pk, sk) = (h, x)$
- $\mathrm{Sign}_x(m)$ picks $r \leftarrow \mathbb{Z}_q$, sets $a = g^r$, $e = \mathcal{H}(a, m)$, $f = r + ex$, and returns (e, f)
- $\mathrm{Vrfy}_h(m, (e, f))$ computes $a = g^f / h^e$ and checks if $\mathcal{H}(a, m) = e$.

Observe: h is not included in the inputs of $\mathcal{H}$

- not needed if h is published prior any signing

Schnorr's Signature Scheme

Security:
Schnorr's signature scheme is EUF-CMA secure in the ROM, assuming that the DL problem is hard wit respect to $\mathcal{G}$.

Let:

- $\mathcal{A}$ be an EUF-CMA adversary against Schnorr's *signature* making at most t queries to the RO (all distinct) and winning with probability ϵ
- P^* be an adversary against the soundness of Schnorr's protocol

We show that P^* can win with probability $\epsilon/t +$ negl.

If ϵ is non-negligible, then:

- The DL problem is not hard w.r.t. $\mathcal{G}$, or
- Schnorr's protocol is not sound

Schnorr's Signature Scheme

P^* proceeds as follows:

1. Receives proof statement $h = g^x$
   Submit it as public key to $\mathcal{A}$
2. Picks a random $j \leftarrow \{1, \ldots, t\}$
3. When $\mathcal{A}$ makes its i-th RO query, on (g^r, m):
   - If $i = j$, submits g^r to the Schnorr verifier, get e, set $\mathcal{H}(g^r, m) = e$ and return e to $\mathcal{A}$
   - If $i \neq j$, returns a random value
4. When $\mathcal{A}$ asks for a signature on m, runs Schnorr's simulator to get (a, e, f) and sets $\mathcal{H}(a, m) = e$
5. When $\mathcal{A}$ outputs a forgery (m^*, e^*, f^*), outputs f^* to the Schnorr verifier

Schnorr's Signature Scheme

This strategy wins if:

- $\mathcal{A}$ indeed produces a forgery ($\Pr = \epsilon$)
- P^* did not define $\mathcal{H}(g^r, m)$ before the j-th query (Pr overwhelming)
- P^* made a correct guess, i.e., $g^{f^*}/h^{e^*} = g^r$ and $m^* = m$ with (g^r, m) as in the j-th query ($\Pr \geq 1/q$)

Since P^* cannot solve the DL and can only break soundness with negligible probability, ϵ must be negligible.

DSA – ECDSA [1991]

- $\mathrm{Gen}(1^n)$ runs $\mathcal{G}(1^n)$ to obtain $\langle \mathbb{G}, q, g \rangle$, then picks $x \leftarrow \mathbb{Z}_q$, sets $h = g^x$ and returns $(pk, sk) = (h, x)$
- $\mathrm{Sign}_x(m)$ picks $k \leftarrow \mathbb{Z}_q^*$, sets $r = F(g^k)$, define $s = k^{-1}(\mathcal{H}(m) + xr) \bmod q$, and returns (r, s)
- $\mathrm{Vrfy}_h(m, (r, s))$ checks if $r = F\left(g^{\mathcal{H}(m)s^{-1}} h^{rs^{-1}}\right)$.

$F(x)$ (resp. $F(x, y)$) defined as $x \bmod q$ in DSA (resp. ECDSA)

Security:

- Secure if $F, \mathcal{H}$ are modeled as RO
- Very little known for actual F (clearly very far from RO)

Widely used standard since 1993, despite expiration of Schnorr's patent (2008)