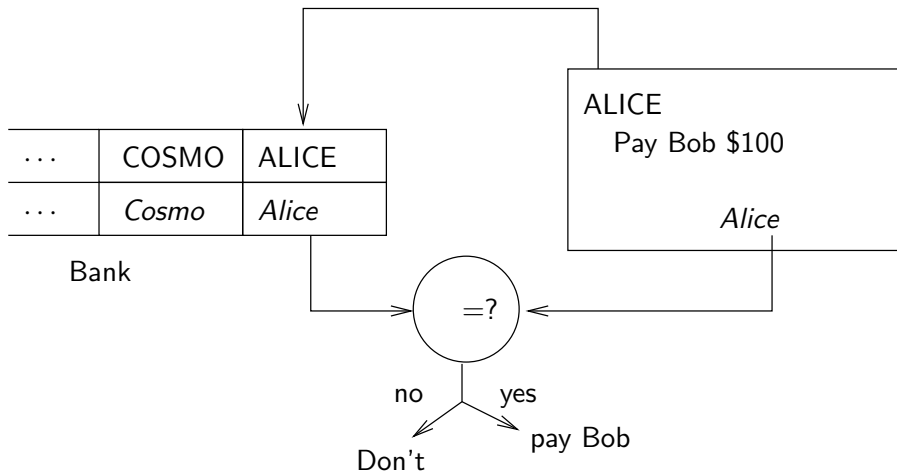
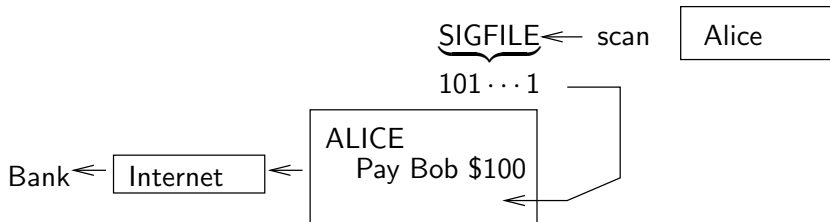


DIGITAL SIGNATURES

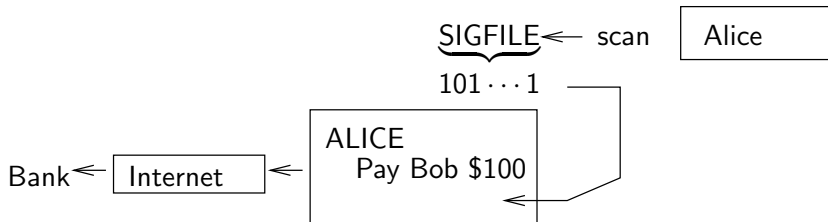
Signing by hand



Signing electronically



Signing electronically

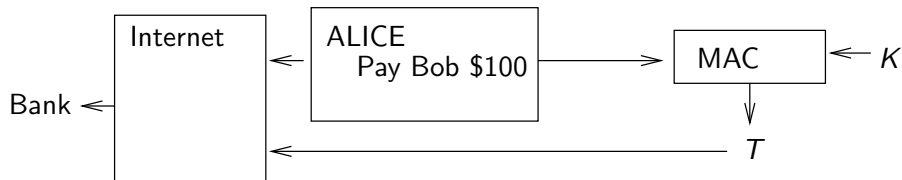


Problem: signature is easily copied

Inference: signature must be a function of the message that only Alice can compute

What about a MAC?

Let Bank and Alice share a key K

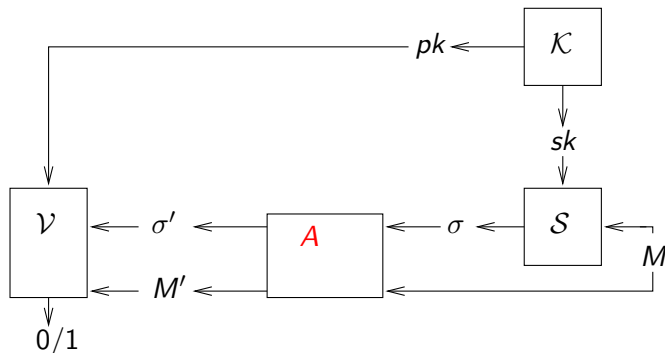


A digital signature will have additional attributes:

- Even the bank cannot forge
- Verifier does not need to share a key with signer or, indeed, have any secrets

Digital signatures

A digital signature scheme $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is a triple of algorithms where



Correctness: $\mathcal{V}(pk, M, \mathcal{S}(sk, M)) = 1$ with probability one for all M .

Step 1: key generation

Alice lets $(pk, sk) \xleftarrow{\$} \mathcal{K}$ and stores sk (securely).

Step 2: pk dissemination

Alice enables any potential verifier to get pk .

Step 3: sign

Alice can generate a signature σ of a document M using sk .

Step 4: verify

Anyone holding pk can verify that σ is Alice's signature on M .

Dissemination of public keys

The public key does not have to be kept secret but a verifier needs to know it is authentic, meaning really Alice's public key and not someone else's.

Could put (Alice, pk) on a trusted, public server (cryptographic DNS.)

Common method of dissemination is via certificates as discussed later.

Signatures versus MA schemes

In a MA scheme:

- Verifier needs to share a secret with sender
- Verifier can “impersonate” sender!

In a digital signature scheme:

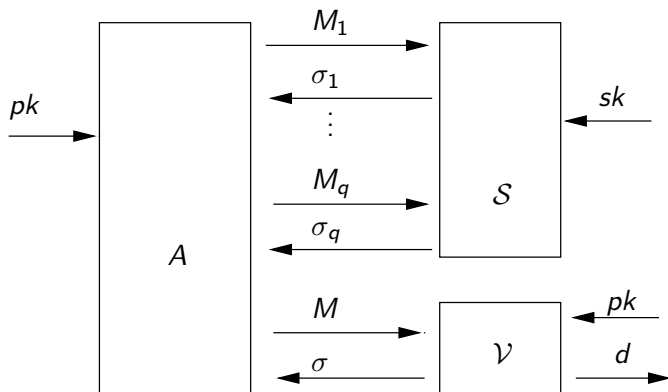
- Verifier needs no secret
- Verifier cannot “impersonate” sender

Possible adversary goals

- find sk
- Forge

Possible adversary abilities

- can get pk
- known message attack
- chosen message attack



A wins if

- $d = 1$
- $M \notin \{M_1, \dots, M_q\}$

Interpretation: adversary cannot get a verifier to accept σ as Alice's signature of M unless Alice has really previously signed M , even if adversary can obtain Alice's signatures on messages of the adversary's choice.

As with MA schemes, the definition does **not** require security against replay. That is handled on top, via counters or time stamps.

Formalization: UF-CMA

Let $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ be a signature scheme and A an adversary.

Game $\text{UF-CMA}_{\mathcal{DS}}$

procedure Initialize

$(pk, sk) \xleftarrow{\$} \mathcal{K}; S \leftarrow \emptyset$

return pk

procedure Finalize(M, σ)

$d \leftarrow \mathcal{V}(pk, M, \sigma)$

return $(d = 1 \wedge M \notin S)$

procedure Sign(M):

$\sigma \xleftarrow{\$} \mathcal{S}(sk, M)$

$S \leftarrow S \cup \{M\}$

return σ

The uf-cma advantage of A is

$$\text{Adv}_{\mathcal{DS}}^{\text{uf-cma}}(A) = \Pr [\text{UF-CMA}_{\mathcal{DS}}^A \Rightarrow \text{true}]$$

A difference with MACs

The UF-CMA game for MA schemes gave the adversary a verification oracle which is not given in the DS case.

Why?

A difference with MACs

The UF-CMA game for MA schemes gave the adversary a verification oracle which is not given in the DS case.

Why? Verification in a MA scheme relies on the secret key but in a DS scheme, the adversary can verify on its own anyway with the public key, so the oracle would not provide an extra capability.

Fix an RSA generator \mathcal{K}_{rsa} and let the key generation algorithm be

Alg \mathcal{K}

$(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{rsa}$

$pk \leftarrow (N, e); sk \leftarrow (N, d)$

return pk, sk

We will use these keys in all our RSA-based schemes and only describe signing and verifying.

Plain RSA signatures: Idea

Signer $pk = (N, e)$ and $sk = (N, d)$

Let $f, f^{-1}: \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ be the RSA function (encryption) and inverse (decryption) defined by

$$f(x) = x^e \bmod N \quad \text{and} \quad f^{-1}(y) = y^d \bmod N.$$

Sign by “decrypting” the message y :

$$x = \mathcal{S}_{N,d}(y) = f^{-1}(y) = y^d \bmod N$$

Verify by “encrypting” signature x :

$$\mathcal{V}_{N,e}(x) = 1 \text{ iff } f(x) = y \text{ iff } x^e \equiv y \bmod N.$$

Plain RSA signature scheme

Signer $pk = (N, e)$ and $sk = (N, d)$

Alg $S_{N,d}(y)$:

$x \leftarrow y^d \pmod N$

return x

Alg $V_{N,e}(y, x)$:

if $x^e \equiv y \pmod N$ **then return** 1

return 0

Here $y \in \mathbb{Z}_N^*$ is the message and $x \in \mathbb{Z}_N^*$ is the signature.

Security of plain RSA signatures

To forge signature of a message y , the adversary, given N, e but not d , must compute $y^d \bmod N$, meaning invert the RSA function f at y .

But RSA is 1-way so this task should be hard and the scheme should be secure.

Correct?

Security of plain RSA signatures

To forge signature of a message y , the adversary, given N, e but not d , must compute $y^d \bmod N$, meaning invert the RSA function f at y .

But RSA is 1-way so this task should be hard and the scheme should be secure.

Correct?

Of course not...

Attacks on plain RSA

Existential forgery under no-message attack: Given $pk = (N, e)$ adversary outputs

- message $y = 1$ and signature $x = 1$
- message $y = x^e \bmod N$ and signature x for any $x \in \mathbb{Z}_N^*$ of its choice

Adversary wins because in both cases we have

$$x^e \equiv y \pmod{N}$$

Homomorphic properties of RSA

Let $pk = (N, e)$ and $sk = (N, d)$ be RSA keys. Then $\forall x_1, x_2 \in \mathbb{Z}_N^*$ and $\forall y_1, y_2 \in \mathbb{Z}_N^*$

- $(x_1 x_2)^e \equiv x_1^e \cdot x_2^e \pmod N$
- $(y_1 y_2)^d \equiv y_1^d \cdot y_2^d \pmod N$

That is

- $f(x_1 x_2) \equiv f(x_1) \cdot f(x_2) \pmod N$
- $f^{-1}(y_1 y_2) \equiv f^{-1}(y_1) \cdot f^{-1}(y_2) \pmod N$

where

$$f(x) = x^e \pmod N \quad \text{and} \quad f^{-1}(y) = y^d \pmod N$$

are the RSA function and its inverse respectively.

Another attack on plain RSA

For all messages $y_1, y_2 \in \mathbb{Z}_N^*$ we have

$$\mathcal{S}_{N,d}(y_1 y_2) = \underbrace{\mathcal{S}_{N,d}(y_1)}_{x_1} \cdot \underbrace{\mathcal{S}_{N,d}(y_2)}_{x_2}$$

So given x_1, x_2 one can forge signature of message $y_1 y_2 \bmod N$

Adversary $A(N, e)$:

Pick some distinct $y_1, y_2 \in \mathbb{Z}_N^* - \{1\}$

$x_1 \leftarrow \mathbf{Sign}(y_1)$; $x_2 \leftarrow \mathbf{Sign}(y_2)$

return $(y_1 y_2 \bmod N, x_1 x_2 \bmod N)$

When Diffie and Hellman introduced public-key cryptography they suggested the DS scheme

$$\begin{aligned}S(sk, M) &= D(sk, M) \\ \mathcal{V}(pk, M, \sigma) &= 1 \text{ iff } E(pk, \sigma) = M\end{aligned}$$

where (E, D) is a public-key encryption scheme.

But

- This views public-key encryption as deterministic; they really mean trapdoor permutations in our language
- Plain RSA is an example
- It doesn't work!

Nonetheless, many textbooks still view digital signatures this way.

In plain RSA, the message is an element of \mathbb{Z}_N^* . We really want to be able to sign strings of arbitrary length.

Throwing in a hash function

Let $H: \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$ be a public hash function and let $pk = (N, e)$ and $sk = (N, d)$ be the signer's keys. The hash-then-decrypt scheme is

Alg $\mathcal{S}_{N,d}(M)$:

$y \leftarrow H(M)$

$x \leftarrow y^d \bmod N$

return x

Alg $\mathcal{V}_{N,e}(M, x)$:

$y \leftarrow H(M)$

if $x^e \equiv y \pmod{N}$ **then return** 1

return 0

Succintly,

$$\mathcal{S}_{N,d}(M) = H(M)^d \bmod N$$

Different choices of H give rise to different schemes.

What we need from H

Suppose an adversary can find a collision for H , meaning distinct M_1, M_2 with $H(M_1) = H(M_2)$.

Then

$$H(M_1)^d \equiv H(M_2)^d \pmod{N}$$

meaning M_1, M_2 have the same signature.

So forgery is easy:

- Obtain from signing oracle the signature $x_1 = H(M_1)^d \pmod{N}$ of M_1
- Output M_2 and its signature x_1

Conclusion: H needs to be collision-resistant

Preventing previous attacks

For plain RSA

- 1 is a signature of 1
- $\mathcal{S}_{N,d}(y_1y_2) = \mathcal{S}_{N,d}(y_1) \cdot \mathcal{S}_{N,d}(y_2)$

But with hash-then-decrypt RSA

- $H(1)^d \neq 1$ so 1 is not a signature of 1
- $\mathcal{S}_{N,d}(y_1y_2) = H(y_1y_2)^d \neq H(y_1)^d \cdot H(y_2)^d \pmod{N}$

A “good” choice of H prevents known attacks.

RSA PKCS#1 signatures

Signer has $pk = (N, e)$ and $sk = (N, d)$ where $|N| = 1024$. Let $h: \{0, 1\}^* \rightarrow \{0, 1\}^{160}$ be a hash function (like SHA-1) and let $n = |N|_8 = 1024/8 = 128$.

Then

$$H_{PKCS}(M) = 00||01|| \underbrace{FF|| \dots || FF}_{n-22} || \underbrace{h(M)}_{20}$$

And

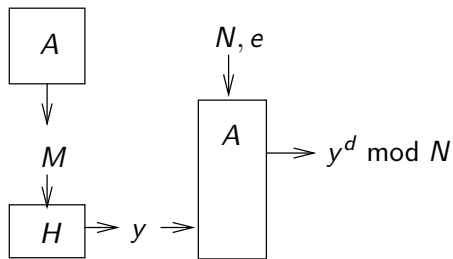
$$S_{N,d}(M) = H_{PKCS}(M)^d \pmod N$$

Then

- H_{PKCS} is CR as long as h is CR
- $H_{PKCS}(1) \not\equiv 1 \pmod N$
- $H_{PKCS}(y_1 y_2) \not\equiv H_{PKCS}(y_1) \cdot H_{PKCS}(y_2) \pmod N$
- etc

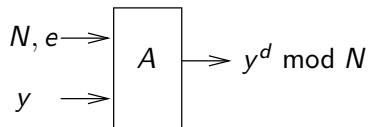
Does 1-wayness prevent forgery?

Forger's goal



y here need not be random

Inverter's goal



y here is **random**

Problem: 1-wayness of RSA does not imply hardness of computing $y^d \bmod N$ if y is not random

Recall

$$H_{PKCS}(M) = 00\|01\|FF\| \dots \|FF\|h(M)$$

But first $n - 20 = 108$ bytes out of n are fixed so $H_{PKCS}(M)$ does not look “random” **even if h is a RO or perfect.**

We cannot hope to show RSA PKCS#1 signatures are secure assuming (only) that RSA is 1-way.

A “better” choice of H might be something like

$$H(M)^d = \text{first } n \text{ bytes of} \\ \text{SHA1}(1 \parallel M) \parallel \text{SHA1}(2 \parallel M) \parallel \dots \parallel \text{SHA1}(11 \parallel M)$$

ElGamal Signatures / DSA

Let $G = \mathbf{Z}_p^* = \langle g \rangle$ where p is prime.

Signer keys: $pk = X = g^x \in \mathbf{Z}_p^*$ and $sk = x \leftarrow^{\$} \mathbf{Z}_{p-1}$

Algorithm $\mathcal{S}_X(m)$

$k \leftarrow^{\$} \mathbf{Z}_{p-1}^*$

$r \leftarrow g^k \bmod p$

$s \leftarrow (m - xr) \cdot k^{-1} \bmod (p - 1)$

return (r, s)

Algorithm $\mathcal{V}_X(m, (r, s))$

if $(r \notin G \text{ or } s \notin \mathbf{Z}_{p-1})$ **return** 0

if $(X^r \cdot r^s \equiv g^m \bmod p)$

then return 1

else return 0

Correctness check: If $(r, s) \leftarrow^{\$} \mathcal{S}_X(m)$ then:

$$\begin{aligned} X^r \cdot r^s &= g^{xr} g^{ks} = g^{xr+ks} \\ &= g^{xr+k(m-xr)k^{-1} \bmod (p-1)} \\ &= g^{xr+m-xr} = g^m \end{aligned}$$

so $\mathcal{V}_X(m, (r, s)) = 1$.

Security of ElGamal Signatures

Signer keys: $pk = X = g^x \in \mathbf{Z}_p^*$ and $sk = x \leftarrow^s \mathbf{Z}_{p-1}$

Algorithm $\mathcal{S}_x(m)$

$k \leftarrow^s \mathbf{Z}_{p-1}^*$

$r \leftarrow g^k \bmod p$

$s \leftarrow (m - xr) \cdot k^{-1} \bmod (p - 1)$

return (r, s)

Algorithm $\mathcal{V}_X(m, (r, s))$

if $(r \notin G \text{ or } s \notin \mathbf{Z}_{p-1})$ **return** 0

if $(X^r \cdot r^s \equiv g^m \bmod p)$

then return 1

else return 0

Suppose given $X = g^x$ and m the adversary wants to compute r, s so that $X^r \cdot r^s \equiv g^m \bmod p$. It could:

- Pick r and try to solve for $s = \text{DLog}_{\mathbf{Z}_p^*, r}(g^m X^{-r})$
- Pick s and try to solve for r ...?

Existential Forgery of ElGamal Signatures

Adversary has better luck if it picks m itself:

Adversary $A(X)$

$r \leftarrow g^X \pmod{p}$; $s \leftarrow (-r) \pmod{p-1}$; $m \leftarrow s$

return $(m, (r, s))$

Then:

$$\begin{aligned} X^r \cdot r^s &= X^{g^X} (g^X)^{-g^X} = X^{g^X} g^{-g^X} X^{-g^X} = g^{-g^X} \\ &= g^{-r} = g^m \end{aligned}$$

so (r, s) is a valid forgery on m .

ElGamal with hashing

Let $G = \mathbf{Z}_p^* = \langle g \rangle$ where p is a prime.

Signer keys: $pk = X = g^x \in \mathbf{Z}_p^*$ and $sk = x \leftarrow^{\$} \mathbf{Z}_{p-1}$

Algorithm $\mathcal{S}_X(M)$

$m \leftarrow H(M)$

$k \leftarrow^{\$} \mathbf{Z}_{p-1}^*$

$r \leftarrow g^k \pmod p$

$s \leftarrow (m - xr) \cdot k^{-1} \pmod{(p-1)}$

return (r, s)

Algorithm $\mathcal{V}_X(M, (r, s))$

$m \leftarrow H(M)$

if $(r \notin G \text{ or } s \notin \mathbf{Z}_{p-1})$ **return** 0

if $(X^r \cdot r^s \equiv g^m \pmod p)$

then return 1

else return 0

Here $H : \{0, 1\}^* \rightarrow \mathbf{Z}_{p-1}$ is a hash function.

ElGamal with hashing

Let $G = \mathbf{Z}_p^* = \langle g \rangle$ where p is a prime.

Signer keys: $pk = X = g^x \in \mathbf{Z}_p^*$ and $sk = x \stackrel{\$}{\leftarrow} \mathbf{Z}_{p-1}$

Algorithm $\mathcal{S}_X(M)$

$m \leftarrow H(M)$

$k \stackrel{\$}{\leftarrow} \mathbf{Z}_{p-1}^*$

$r \leftarrow g^k \pmod p$

$s \leftarrow (m - xr) \cdot k^{-1} \pmod{(p-1)}$

return (r, s)

Algorithm $\mathcal{V}_X(M, (r, s))$

$m \leftarrow H(M)$

if $(r \notin G \text{ or } s \notin \mathbf{Z}_{p-1})$ **return** 0

if $(X^r \cdot r^s \equiv g^m \pmod p)$

then return 1

else return 0

Here $H : \{0, 1\}^* \rightarrow \mathbf{Z}_{p-1}$ is a hash function.

Requirements on H :

- Collision-resistant
- One-way to prevent previous attack

Security of ElGamal and other DL Signatures

- NO proof that ElGamal is UF-CMA under a standard assumption (DL, CDH, ...) is known
- Proofs are known for variants
- There are other DL-based signatures with proofs such as Schnorr