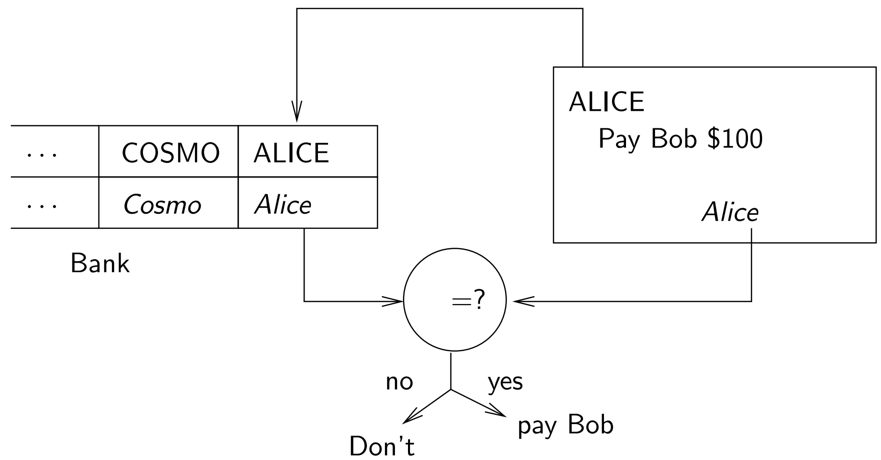
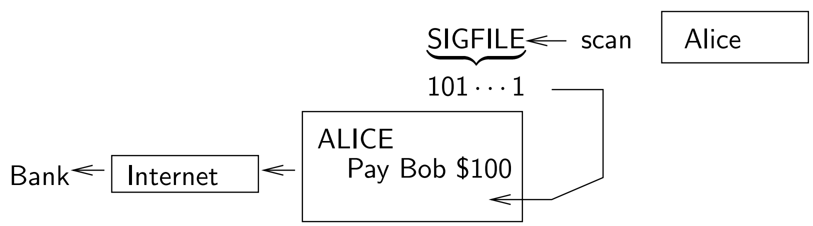
DIGITAL SIGNATURES

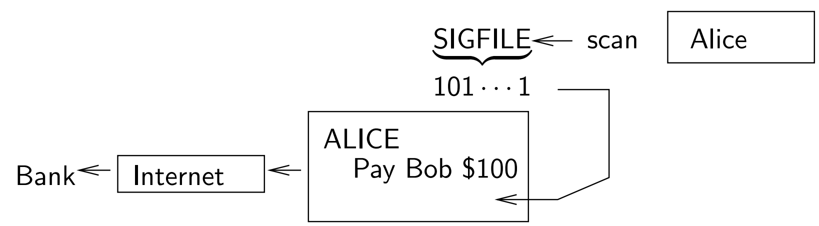
Signing by hand



Signing electronically



Signing electronically

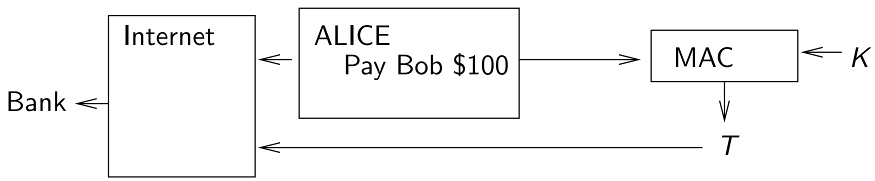


Problem: signature is easily copied

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

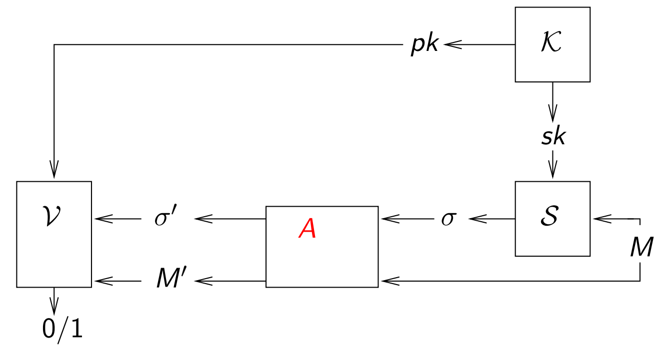


A digital signature will have additional attributes:

- 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.

Usage

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.

Step 2: 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.

Alice could put her public key pk on her webpage, her Facebook, a key server or include it as an email attachment.

Common method of dissemination is via certificates as discussed later.

UF-CMA Security of a DS scheme

Intent: 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.

UF-CMA Security of a DS scheme

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

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

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

procedure Sign(M)
$\sigma \xleftarrow{\$} \mathcal{S}(sk, M)$
$S \leftarrow S \cup \{M\}$
return σ

procedure Finalize(M, σ)
$d \leftarrow \mathcal{V}(pk, M, \sigma)$
return $(d = 1 \wedge M \notin S)$

The uf-cma advantage of A is

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

UF-CMA: Explanations

The "return pk" statement in **Initialize** means the adversary A gets the public key pk as input. It does not get sk.

It can call **Sign** with any message M of its choice to get back a correct signature $\sigma \xleftarrow{\$} \mathcal{S}(sk, M)$ of M under sk. Notation indicates signing algorithm may be randomized.

To win, it must output a message M and a signature σ that are

- Correct: $\mathcal{V}(pk, M, \sigma) = 1$
- New: $M \notin S$, meaning M was not a query to **Sign**

Interpretation: Sign represents the signer and **Finalize** represents the verifier. Security means that the adversary can't get the verifier to accept a message that is not authentic, meaning was not already signed by the sender.

RSA signatures

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

Alg $\mathcal{K}$
$(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{\text{rsa}}$
$pk \leftarrow (N, e)$; $sk \leftarrow (N, d)$
Return (pk, sk)

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

Plain RSA signature scheme

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

Alg $\mathcal{S}_{N,d}(y)$
$x \leftarrow y^d \bmod N$
Return x

Alg $\mathcal{V}_{N,e}(y, x)$
If $(x^e \bmod N = y)$ then return 1
Else 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

adversary $A(N, e)$
Return $(1, 1)$

$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS}}(A) = 1$ because $1^d \equiv 1 \pmod{N}$

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

$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS}}(A) = 1$ because $(y_1 y_2)^d \equiv y_1^d y_2^d \pmod{N}$

DH signatures

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

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

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.

Other issues

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: $\{0,1\}^* \to \mathbb{Z}_N^*$ 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 $(x^e \bmod N = y)$ then return 1
Else return 0

Succinctly,

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

Different choices of H give rise to different schemes.

What we need from H

Suppose we have an adversary C that can find a collision for H. Then we can break DS via

adversary $A(N, e)$
$(M_1, M_2) \stackrel{\$}{\leftarrow} C$
$\sigma_1 \leftarrow \textbf{Sign}(M_1)$
Return (M_2, σ_1)

This works because $H(M_1) = H(M_2)$ implies M_1, M_2 have the same signatures:

$$\sigma_1 = \mathcal{S}_{N,d}(M_1) = H(M_1)^d \bmod N = H(M_2)^d \bmod N = \mathcal{S}_{N,d}(M_2)$$

Conclusion: H needs to be collision-resistant

RSA PKCS#1 signatures

Signer has $pk = (N, e)$ and $sk = (N, d)$ where $|N| = 1024$. Let $h\colon \{0,1\}^* \to \{0,1\}^{160}$ be a hash function (like SHA1) and let $n = 1024/8 = 128$.

Then

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

And

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

RSA PKCS#1 signatures

Signer has $pk = (N, e)$ and $sk = (N, d)$ where $|N| = 1024$. Let $h\colon \{0,1\}^* \to \{0,1\}^{160}$ be a hash function (like SHA1) and let $n = 1024/8 = 128$.

Then

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

And

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

But first $n - 20 = 108$ bytes of $H_{PKCS}(M)$ are fixed, so $H_{PKCS}(M)$ does not look "random."

Full-Domain-Hash (FDH) [BR96]

Signer public key is $pk = (N, e)$ and secret key is $sk = (N, d)$

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

Alg $\mathcal{V}_{N,e}(M, x)$
If $(x^e \bmod N = H(M))$ then return 1
Else return 0

Public hash function $H\colon \{0,1\}^* \to \mathbb{Z}_N^*$ is defined for example by letting $H(M)$ be the first $|N|$ bits of

$$\text{SHA1512}(0^8\|M)\|\text{SHA1512}(0^7 1\|M)\|\cdots$$

.

Exercise

Let $\mathcal{K}_{\rm rsa}$ be a RSA key generator with security parameter $k \geq 2048$. Let the algorithms of signature scheme $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ be defined as follows, with notation explained on the next slide:

Alg $\mathcal{K}$
$(N, p, q, e, d) \stackrel{\$}{\leftarrow} \mathcal{K}_{\rm rsa}$; $pk \leftarrow (N, e)$; $sk \leftarrow (N, d)$; Return (pk, sk)

Alg $\mathcal{S}_{N,d}(M)$
If $|M| \neq 4096$ then return $\perp$
$M[1]M[2] \leftarrow M$
$x_1 \leftarrow \langle 1\rangle \,||\, M[1]$; $x_2 \leftarrow \langle 2\rangle \,||\, M[2]$
$y \leftarrow H(x_1) \cdot H(x_2) \bmod N$
$s \leftarrow y^d \bmod N$
Return s

Alg $\mathcal{V}_{N,e}(M, s)$
If $|M| \neq 4096$ then return 0
$M[1]M[2] \leftarrow M$
$x_1 \leftarrow \langle 1\rangle \,||\, M[1]$; $x_2 \leftarrow \langle 2\rangle \,||\, M[2]$
If $s^e \equiv H(x_1) \cdot H(x_2) \pmod{N}$
then return 1 else return 0

Exercise

Above, $H\colon \{0,1\}^* \to \mathbf{Z}_N^*$ is a public, collision resistant hash function. A valid message M is a 4096 bit string and is viewed as a pair of 2048 bit blocks, $M = M[1]M[2]$. By "$||$" we denote concatenation, and by $\langle i \rangle$ we denote the encoding of integer i as a binary string of exactly two bits.

Present in pseudocode a $\mathcal{O}(k^3)$-time adversary A making at most three queries to its **Sign** oracle and achieving $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS}}(A) = 1$.

ElGamal Signatures

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

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

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

$k \xleftarrow{\$} \mathbf{Z}_{p-1}^*$
$r \leftarrow g^k \mod p$
$s \leftarrow (m - xr) \cdot k^{-1} \mod (p-1)$
Return (r, s)

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

If ($r \notin G$ or $s \notin \mathbf{Z}_{p-1}$)
  then return 0
If ($X^r \cdot r^s \equiv g^m \mod p$)
  then return 1
else return 0

Correctness check: If $(r, s) \xleftarrow{\$} \mathcal{S}_x(m)$, then, in the group G we have:

$$X^r \cdot r^s = g^{xr} g^{ks} = g^{xr+ks} = g^{xr+k(m-xr)k^{-1} \mod (p-1)} = g^{xr+m-xr} = g^m$$

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

Security of ElGamal Signatures

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

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

$k \xleftarrow{\$} \mathbf{Z}_{p-1}^*$
$r \leftarrow g^k \mod p$
$s \leftarrow (m - xr) \cdot k^{-1} \mod (p-1)$
Return (r, s)

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

If ($r \notin G$ or $s \notin \mathbf{Z}_{p-1}$)
  then return 0
If ($X^r \cdot r^s \equiv g^m \mod p$)
  then return 1
else return 0

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

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

Forgery of ElGamal Signatures

Adversary has better luck if it picks m itself:

Adversary $A(X)$
$r \leftarrow gX \mod p$; $s \leftarrow (-r) \mod (p-1)$; $m \leftarrow s$
Return $(m, (r, s))$

Then:

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

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 $sk = x \xleftarrow{\$} \mathbf{Z}_{p-1}$
$H : \{0,1\}^* \to \mathbf{Z}_{p-1}$ a hash function.

Alg $\mathcal{S}_x(M)$
$m \leftarrow H(M)$
$k \xleftarrow{\$} \mathbf{Z}_{p-1}^*$
$r \leftarrow g^k \mod p$
$s \leftarrow (m - xr) \cdot k^{-1} \mod (p-1)$
Return (r, s)

Alg $\mathcal{V}_X(M, (r, s))$
$m \leftarrow H(M)$
If ($r \notin G$ or $s \notin \mathbf{Z}_{p-1}$)
then return 0
If ($X^r \cdot r^s \equiv g^m \mod p$)
then return 1
else return 0

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 $sk = x \xleftarrow{\$} \mathbf{Z}_{p-1}$
$H : \{0,1\}^* \to \mathbf{Z}_{p-1}$ a hash function.

Alg $\mathcal{S}_x(M)$
$m \leftarrow H(M)$
$k \xleftarrow{\$} \mathbf{Z}_{p-1}^*$
$r \leftarrow g^k \mod p$
$s \leftarrow (m - xr) \cdot k^{-1} \mod (p-1)$
Return (r, s)

Alg $\mathcal{V}_X(M, (r, s))$
$m \leftarrow H(M)$
If ($r \notin G$ or $s \notin \mathbf{Z}_{p-1}$)
then return 0
If ($X^r \cdot r^s \equiv g^m \mod p$)
then return 1
else return 0

Requirements on H:

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

DSA

Let p be a 1024-bit prime. For DSA, let q be a 160-bit prime dividing $p - 1$.

Scheme	signing cost	verification cost	signature size
ElGamal	1 1024-bit exp	1 1024-bit exp	2048 bits
DSA	1 160-bit exp	1 160-bit exp	320 bits

By a "e-bit exp" we mean an operation $a, n \mapsto a^n \mod p$ where $a \in \mathbf{Z}_p^*$ and n is an e-bit integer. A 1024-bit exponentiation is more costly than a 160-bit exponentiation by a factor of $1024/160 \approx 6.4$.

DSA is in FIPS 186.

DSA

- Fix primes p, q such that q divides $p - 1$
- Let $G = \mathbf{Z}_p^* = \langle h \rangle$ and $g = h^{(p-1)/q}$ so that $g \in G$ has order q
- H: $\{0,1\}^* \to \mathbf{Z}_q$ a hash function
- Signer keys: $pk = X = g^x \in \mathbf{Z}_p^*$ and $sk = x \xleftarrow{\$} \mathbf{Z}_q$

Alg $\mathcal{S}_x(M)$
$m \leftarrow H(M)$
$k \xleftarrow{\$} \mathbf{Z}_q^*$
$r \leftarrow (g^k \mod p) \mod q$
$s \leftarrow (m + xr) \cdot k^{-1} \mod q$
Return (r, s)

Alg $\mathcal{V}_X(M, (r, s))$
$m \leftarrow H(M)$
$w \leftarrow s^{-1} \mod q$
$u_1 \leftarrow mw \mod q$
$u_2 \leftarrow rw \mod q$
$v \leftarrow (g^{u_1} X^{u_2} \mod p) \mod q$
If ($v = r$) then return 1 else return 0

Details: Signature is regenerated if $s = 0$.

Discussion

DSA as shown works only over the group of integers modulo a prime, but there is also a version ECDSA of it for elliptic curve groups.

In ElGamal and DSA/ECDSA, the expensive part of signing, namely the exponentiation, can be done off-line.

No proof that ElGamal or DSA is UF-CMA under a standard assumption (DL, CDH, ...) is known. Proofs known for variants.

The Schnorr scheme works in an arbitrary (prime-order) group. When implemented in a 160-bit elliptic curve group, it is as efficient as ECDSA. It can be proven UF-CMA in the random oracle model under the discrete log assumption [PS,AABN]. The security reduction, however, is quite loose.

Exercise

Let p be a prime of bit length $k \geq 1024$ such that $(p-1)/2$ is also prime, and let g be a generator of the group $G = \mathbf{Z}_p^*$. (Here p, g are public quantities.) Let $q = p - 1$ be the order of G. Consider the digital signature scheme $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ whose component algorithms are depicted below, where the message m is in $\mathbf{Z}_q^*$:

Alg $\mathcal{K}$

$x \stackrel{\$}{\leftarrow} \mathbf{Z}_q$; $X \leftarrow g^x$; $y \stackrel{\$}{\leftarrow} \mathbf{Z}_q$; $Y \leftarrow g^y$

return $((X, Y), (x, y))$

Alg $\mathcal{S}((x, y), m)$

If $m \notin \mathbf{Z}_q^*$ then return $\perp$

$z \leftarrow (y + xm) \bmod q$

return z

Alg $\mathcal{V}((X, Y), m, z)$

if $m \notin \mathbf{Z}_q^*$ then return 0

if $z \notin \mathbf{Z}_q$ then return 0

if $(g^z \equiv YX^m \pmod p)$ then return 1

else return 0

Exercise

1. Prove that $\mathcal{V}((X, Y), m, z) = 1$ for any key-pair $((X, Y), (x, y))$ that might be output by $\mathcal{K}$, any message $m \in \mathbf{Z}_q^*$, and any z that might be output by $\mathcal{S}((x, y), m)$.
2. Present in pseudocode a $\mathcal{O}(k^2)$-time adversary A making at most two queries to its **Sign** oracle and achieving $\mathbf{Adv}_{\mathcal{DS}}^{\text{uf-cma}}(A) = 1$.

Randomization in signatures

We have seen many randomized signature schemes: PSS, ElGamal, DSA/ECDSA, Schnorr, ...

Re-using coins across different signatures is not secure, but there are (other) ways to make these schemes deterministic without loss of security.