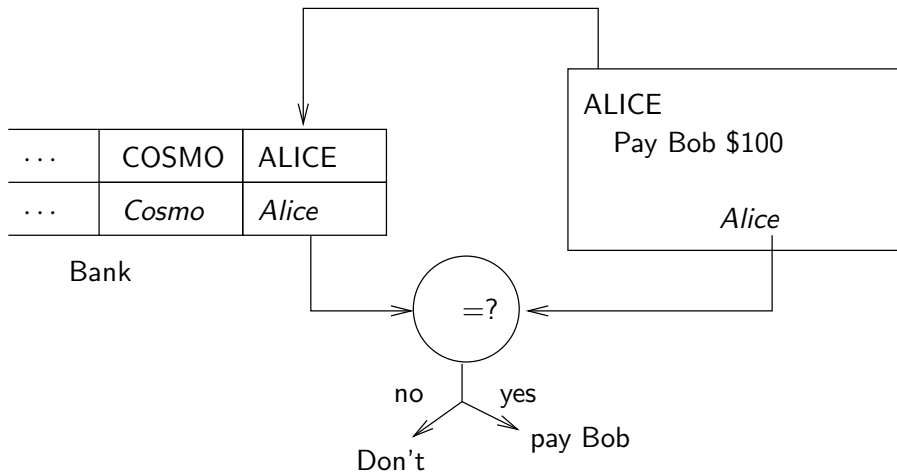
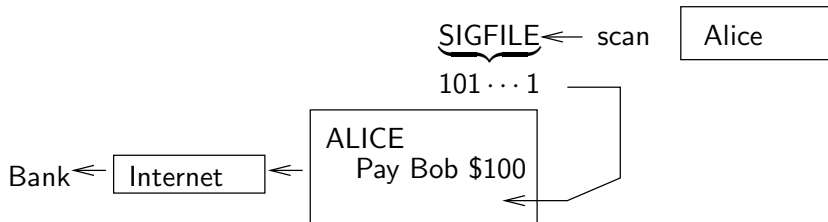


# DIGITAL SIGNATURES

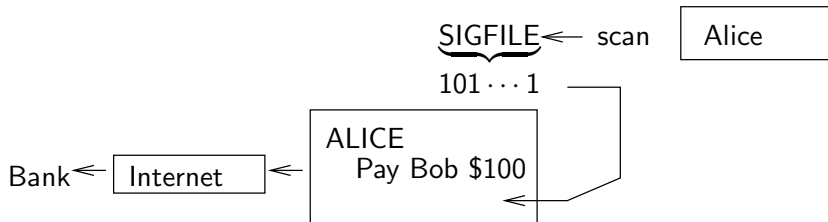
# Signing by hand



# Signing electronically



# Signing electronically

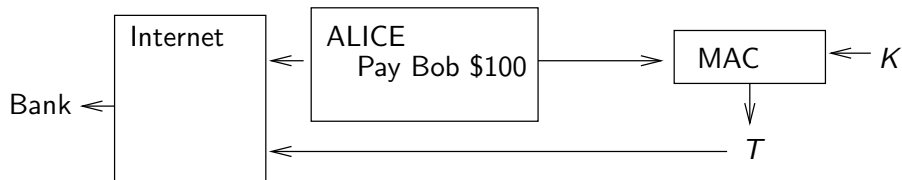


**Problem:** signature is easily copied

**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  $K$

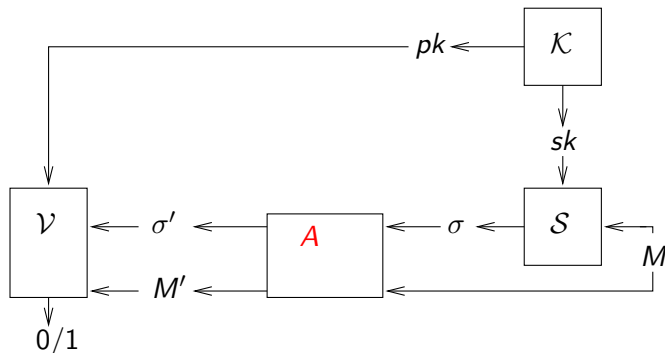


A digital signature will have additional attributes:

- 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  $\sigma$  of a document  $M$  using  $sk$ .

## Step 4: verify

Anyone holding  $pk$  can verify that  $\sigma$  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  $(\text{Alice}, pk)$  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:

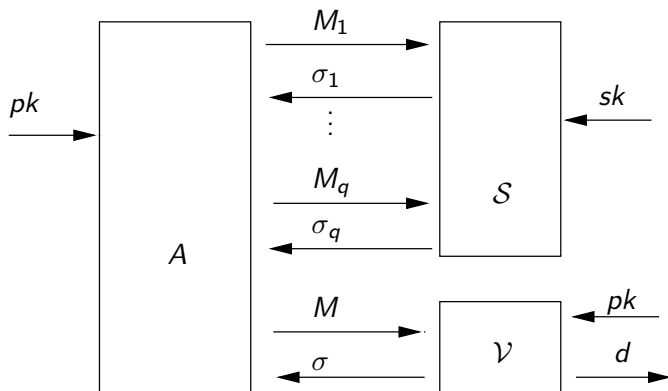
- Verifier needs no secret
- Verifier cannot “impersonate” sender

## 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  $\sigma$  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**( $M, \sigma$ )

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

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

**procedure Sign**( $M$ ):

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

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

return  $\sigma$

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.

# Strong unforgeability

Adversary can't get receiver to accept  $\sigma$  as Alice's signature on  $M$  unless

- UF: Alice previously signed  $M$
- SUF: Alice previously signed  $M$  and produced signature  $\sigma$

Adversary wins if it gets receiver to accept  $\sigma$  as Alice's signature on  $M$  and

- UF: Alice did not previously sign  $M$
- SUF: Alice may have previously signed  $M$  but the signature(s) produced were different from  $\sigma$



# Formalization: SUF-CMA

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

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

**procedure Initialize:**

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

**return**  $pk$

**procedure Finalize** $(M, \sigma)$ :

**return**  $\mathcal{V}(pk, M, \sigma) = 1$  and  $(M, \sigma) \notin S$

**procedure Sign** $(M)$ :

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

$S \leftarrow S \cup \{(M, \sigma)\}$

**return**  $\sigma$

The suf-cma advantage of  $A$  is

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

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

$x \leftarrow y^d \bmod N$

**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  $y_1 y_2 \pmod N$

**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**  $(y_1 y_2 \pmod N, x_1 x_2 \pmod N)$

# DH signatures

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

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

But with hash-then-decrypt RSA

- $H(1)^d \neq 1$  so 1 is not a signature of 1
- $\mathcal{S}_{N,d}(M_1 M_2) = H(M_1 M_2)^d \neq H(M_1)^d \cdot H(M_2)^d \pmod{N}$

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

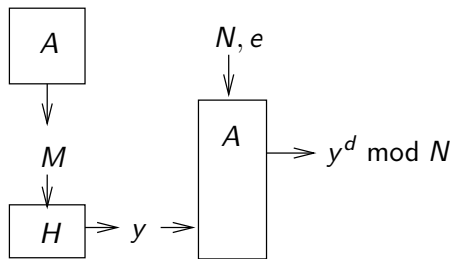
$$S_{N,d}(M) = H_{PKCS}(M)^d \pmod N$$

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

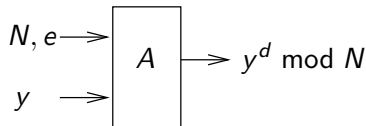
# Does 1-wayness prevent forgery?

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 **no matter what we assume about  $h$**  and even if  $h$  is a random oracle.

We will validate the hash-then-decrypt paradigm

$$S_{N,d}(M) = H(M)^d \bmod N$$

by showing the signature scheme is provably UF-CMA assuming RSA is 1-way as long as  $H$  is a RO.

This says the paradigm has no “structural weaknesses” and we should be able to get security with “good” choices of  $H$ .

A “good” choice of  $H$  might be something like

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

# Full-Domain-Hash (FDH) [BR96]

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

algorithm $\mathcal{S}_{N,d}^H(M)$		algorithm $\mathcal{V}_{N,e}^H(M, x)$
<b>return</b> $H(M)^d \bmod N$		<b>if</b> $x^e \equiv H(M) \pmod{N}$ <b>then return</b> 1
		<b>else return</b> 0

Here  $H: \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$  is a random oracle.

# UF-CMA in RO model

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**( $M, \sigma$ ):

**return**  $\mathcal{V}^H(pk, M, \sigma) = 1$  and  $M \notin S$

**procedure Sign**( $M$ ):

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

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

**return**  $\sigma$

**procedure**  $H(M)$ :

**if**  $H[M] = \perp$  **then**  $H[M] \xleftarrow{\$} \mathcal{R}$

**return**  $H[M]$

Here  $\mathcal{R}$  is the range of  $H$ .

The uf-cma advantage of  $A$  is

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

# Security of FDH in RO model

**Theorem:** [BR96] Let  $\mathcal{K}_{rsa}$  be a RSA generator and  $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$  the associated FDH RO-model signature scheme. Let  $A$  be a uf-cma adversary making  $q_s$  signing queries and  $q_H$  queries to the RO  $H$  and having running time at most  $t$ . Then there is an inverter  $I$  such that

$$\mathbf{Adv}_{\mathcal{DS}}^{\text{uf-cma}}(A) \leq (q_s + q_H + 1) \cdot \mathbf{Adv}_{\mathcal{K}_{rsa}}^{\text{owf}}(I).$$

Furthermore the running time of  $I$  is that of  $A$  plus the time for  $\mathcal{O}(q_s + q_H + 1)$  computations of the RSA function.

There is a “crucial” hash query  $Q$  such that

- If  $A$  does not query  $Q$  it has 0 advantage
- If  $A$  queries  $Q$  an overlying algorithm can “see” it and solve some presumed hard computational problem

**Example:** In the RO EG KEM,  $Q = g^{xy}$  where  $pk = g^x$  and  $g^y$  is in challenge ciphertext.

For signatures we exploit the RO model in a new way by replying to RO queries with carefully constructed objects. In particular the inverter  $I$  that on input  $y$  aims to compute  $y^d \bmod N$  might reply to a RO query  $M$  made by  $A$  via

- $y$  or some function thereof
- $x^e \bmod N$  for  $x \xleftarrow{\$} \mathbb{Z}_N^*$  chosen by  $I$

Thus  $I$  is “programming”  $H(M)$  to equal values of  $I$ 's choice.



# The case $q_S = 0$ and $q_H = 1$

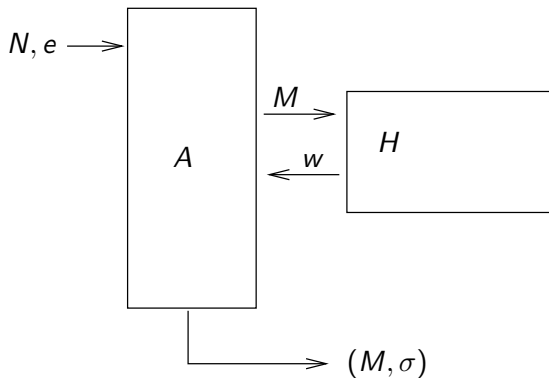
Assume  $A$

- Makes no **Sign** queries
- Makes exactly one  $H$ -query  $M$
- Then outputs a forgery  $(M, \sigma)$

Let us see how to build  $I$  so that

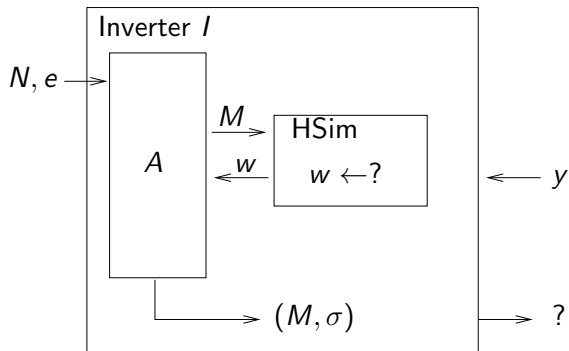
$$\mathbf{Adv}_{DS}^{\text{uf-cma}}(A) = \mathbf{Adv}_{\mathcal{K}_{rsa}}^{\text{owf}}(I).$$

# The case $q_S = 0$ and $q_H = 1$



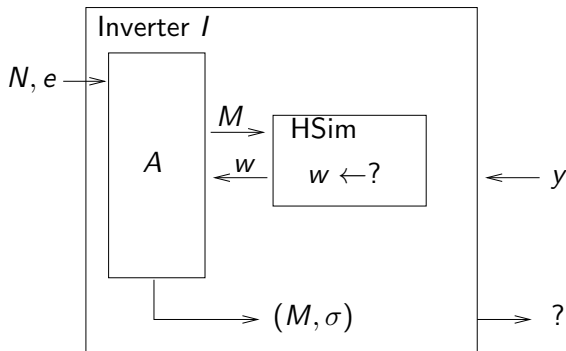
$$\mathbf{Adv}_{DS}^{\text{uf-cma}}(A) = \Pr[\sigma^e \equiv w \pmod{N}]$$

# The inverter for the case $q_S = 0$ and $q_H = 1$



Q: How should  $I$  choose  $w$  and what should it output?

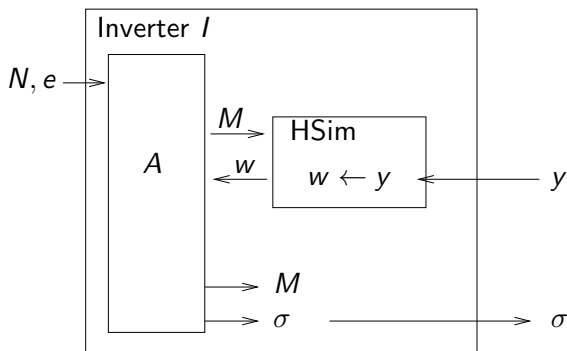
# The inverter for the case $q_S = 0$ and $q_H = 1$



Q: How should  $I$  choose  $w$  and what should it output?

A: Let  $w = y$  and output  $\sigma$ !

# The inverter for the case $q_S = 0$ and $q_H = 1$



$$\begin{aligned}\mathbf{Adv}_{DS}^{\text{uf-cma}}(A) &= \Pr[\sigma^e \equiv w \pmod{N}] \\ &= \Pr[\sigma^e \equiv y \pmod{N}] \\ &= \mathbf{Adv}_{\mathcal{K}_{rsa}}^{\text{owf}}(I)\end{aligned}$$

# The inverter for the case $q_S = 0$ and $q_H = 1$

**Inverter**  $I(N, e, y)$ :  
 $(M, \sigma) \leftarrow A^{\text{HSim}}(N, e)$   
**return**  $\sigma$

**subroutine**  $\text{HSim}(M)$ :  
**return**  $y$

Then

$$\mathbf{Adv}_{\mathcal{DS}}^{\text{uf-cma}}(A) = \mathbf{Adv}_{\mathcal{K}_{rsa}}^{\text{owf}}(I).$$

# The case $q_S = 0$ and $q_H > 1$

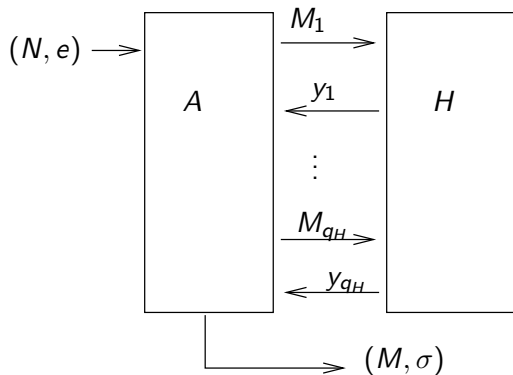
Assume  $A$

- Makes no **Sign** queries
- Makes  $H$ -queries  $M_1, \dots, M_{q_H}$
- Then outputs a forgery  $(M, \sigma)$  such that  $M \in \{M_1, \dots, M_{q_H}\}$

Let us see how to build  $I$  so that

$$\mathbf{Adv}_{\mathcal{DS}}^{\text{uf-cma}}(A) = q_H \cdot \mathbf{Adv}_{\mathcal{K}_{rsa}}^{\text{owf}}(I).$$

The case  $q_s = 0$  and  $q_H > 1$



Let  $i$  be such that  $M = M_i$ .

$$\mathbf{Adv}_{DS}^{\text{uf-cma}}(A) = \Pr[\sigma^e \equiv y_i \pmod{N}]$$



# Inverter for the case $q_s = 0$ and $q_H > 1$

As before, return  $y$  in response to a  $H$ -query:

**Inverter**  $I(N, e, y)$ :

$(M, \sigma) \xleftarrow{\$} A^{\text{HSim}}(N, e)$

**return**  $\sigma$

**subroutine**  $\text{HSim}(M)$ :

**return**  $y$

# Inverter for the case $q_s = 0$ and $q_H > 1$

As before, return  $y$  in response to a  $H$ -query:

<b>Inverter</b> $I(N, e, y)$ : $(M, \sigma) \xleftarrow{\$} A^{\text{HSim}}(N, e)$ <b>return</b> $\sigma$		<b>subroutine</b> $\text{HSim}(M)$ : <b>return</b> $y$
---	--	---

Say  $A$ 's queries are  $M_1, \dots, M_q$  and  $M = M_i$ . Then if  $\sigma^e \equiv H(M_i) \pmod{N}$  we have  $\sigma^e \equiv y \pmod{N}$  so  $I$  wins so

$$\mathbf{Adv}_{\mathcal{DS}}^{\text{uf-cma}}(A) = \Pr[\sigma^e \equiv y_i \pmod{N}] = \mathbf{Adv}_{\mathcal{K}_{rsa}}^{\text{owf}}(I).$$

# Inverter for the case $q_s = 0$ and $q_H > 1$

As before, return  $y$  in response to a  $H$ -query:

<b>Inverter</b> $I(N, e, y)$ : $(M, \sigma) \stackrel{\$}{\leftarrow} A^{\text{HSim}}(N, e)$ <b>return</b> $\sigma$	<b>subroutine</b> $\text{HSim}(M)$ <b>return</b> $y$
---	---

This is wrong because the answers to  $A$ 's queries are not independent, meaning  $\text{HSim}$  does not look like a "real" RO.

What if  $A$  made queries  $M_1 \neq M_2$  and on getting back  $y_1, y_2$  aborted if  $y_1 = y_2$ ?  $A$ 's advantage in the simulation could be 0.

# Inverter for the case $q_s = 0$ and $q_H > 1$

We could return  $y$  in response to a random query and random values in response to the rest:

**Inverter**  $I(N, e, y)$ :

$g \xleftarrow{\$} \{1, \dots, q_H\}; j \leftarrow 0$

$(M, \sigma) \xleftarrow{\$} A^{\text{HSim}}(N, e)$

**return**  $\sigma$

**subroutine**  $\text{HSim}(M)$

$j \leftarrow j + 1$

**if**  $j = g$  **then**  $y_i \leftarrow y$

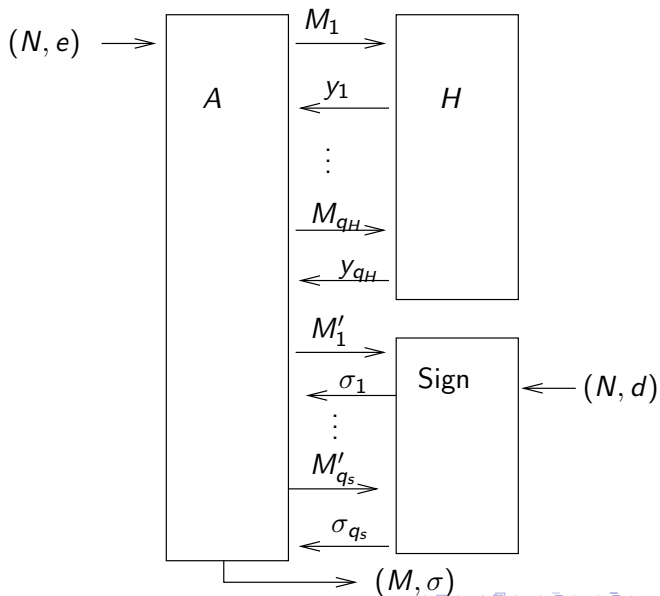
**else**  $y_j \xleftarrow{\$} \mathbb{Z}_N^*$

**return**  $y_j$

Say  $A$ 's queries are  $M_1, \dots, M_{q_H}$  and  $M = M_i$ . Then if  $\sigma^e \equiv y_i \pmod{N}$  and  $i = g$  we have  $\sigma^e \equiv y \pmod{N}$ , so

$$\text{Adv}_{\mathcal{DS}}^{\text{uf-cma}}(A) = q_H \cdot \text{Adv}_{\mathcal{K}_{rsa}}^{\text{owf}}(I).$$

# The case $q_s > 0$



# Replying to **Sign** queries

How can the Inverter  $I$  (not knowing  $d$ ) return the signature  $H(M)^d \bmod N$  in response to **Sign** query  $M$ ?

Trick: When  $M'_i$  is queried to  $H$ , Inverter will

- pick  $x_i \xleftarrow{\$} \mathbb{Z}_N^*$  and let  $y_i \leftarrow x_i^e \bmod N$
- Return  $H(M'_i) = y_i$

Then if there is a **Sign**( $M'_i$ ) query it can return  $x_i$  as the signature.

# Simplification

Assume that if  $A$

- Makes **Sign** query  $M$ , it has previously made  $H$ -query  $M$
- Outputs  $(M, \sigma)$  then it has previously made  $H$ -query  $M$  and not made **Sign** query  $M$

Can easily modify  $A$  to have these properties at the cost of increasing the number of  $H$ -queries to

$$q = q_s + q_H + 1.$$

Also assume  $A$  never repeats a  $H$ -query.

# Inverter for case $q_s > 0$

**Inverter**  $I(N, e, y)$ :

$g \xleftarrow{\$} \{1, \dots, q_H\}; j \leftarrow 0$

$(M, \sigma) \xleftarrow{\$} A^{\text{HSim}, \text{SignSim}}(N, e)$ :

**return**  $\sigma$

**subroutine**  $\text{SignSim}(M)$

$j \leftarrow \text{Ind}(M)$

**return**  $x_j$

**subroutine**  $\text{HSim}(M)$

$j \leftarrow j + 1; M_j \leftarrow M; \text{Ind}(M) \leftarrow j$

**if**  $j = g$  **then**  $H[M] \leftarrow y; x_j \leftarrow \perp$

**else**  $x_j \xleftarrow{\$} \mathbb{Z}_N^*; H[M] \leftarrow x_j^e \bmod N$

**return**  $H[M]$



Let  $i$  be such that  $A$  outputs  $(M, \sigma)$  with  $M = M_i$ . Then if  $i = g$

- $\sigma^e \equiv H(M_i) \equiv y \pmod{N}$  so inverter finds  $\sigma = y^d \pmod{N}$
- All  $A$ 's queries are correctly answered

Since  $i = g$  with probability  $1/q$  we have

$$\mathbf{Adv}_{\mathcal{K}_{rsa}}^{\text{owf}}(I) \geq \frac{1}{q} \cdot \mathbf{Adv}_{\mathcal{DS}}^{\text{uf-cma}}(A).$$

# Fundamental Lemma variant

**Lemma** [BR06] Let  $G_i, G_j$  be identical-until-bad games and  $A$  an adversary. Then for any  $y$

$$\Pr \left[ G_i^A \Rightarrow y \wedge G_i^A \text{ doesn't set bad} \right] = \Pr \left[ G_j^A \Rightarrow y \wedge G_j^A \text{ doesn't set bad} \right]$$

Games  $G_0, \boxed{G_1}$

**procedure Initialize**

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

$g \xleftarrow{\$} \{1, \dots, q_H\}; j \leftarrow 0; y \xleftarrow{\$} \mathbb{Z}_N^*$

**return**  $(N, e)$

**procedure Sign** $(M)$

$j \leftarrow \text{Ind}(M)$

**if**  $j = g$  **then**  $x_j \leftarrow y^d \bmod N$

**return**  $x_j$

**procedure**  $H(M)$

$j \leftarrow j + 1; M_j \leftarrow M; \text{Ind}(M) \leftarrow j$

**if**  $j = g$  **then**  $H[M] \leftarrow y; x_j \leftarrow \perp$

**else**  $x_j \xleftarrow{\$} \mathbb{Z}_N^*; H[M] \leftarrow x_j^e \bmod N$

**return**  $H[M]$

**procedure Finalize** $(M, \sigma)$

$j \leftarrow \text{Ind}(M)$

**if**  $j \neq g$  **then**  $\text{bad} \leftarrow \text{true}$

**return**  $(\sigma^e \equiv H[M] \pmod{N})$

Let  $\text{Bad}_i$  be the event that  $G_i$  sets bad. Then

$$\begin{aligned}\mathbf{Adv}_{\mathcal{K}_{rsa}}^{\text{owf}}(I) &\geq \Pr \left[ G_0^A \Rightarrow \text{true} \wedge \overline{\text{Bad}_0} \right] \\ &= \Pr \left[ G_1^A \Rightarrow \text{true} \wedge \overline{\text{Bad}_1} \right]\end{aligned}$$

where last line is due to Fundamental Lemma variant. But the events " $G_1^A \Rightarrow \text{true}$ " and " $\overline{\text{Bad}_1}$ " are independent so

$$\begin{aligned}&= \Pr \left[ G_1^A \Rightarrow \text{true} \right] \cdot \Pr \left[ \overline{\text{Bad}_1} \right] \\ &= \mathbf{Adv}_{DS}^{\text{uf-cma}}(A) \cdot \frac{1}{q}\end{aligned}$$

# Security of FDH in RO model

**Theorem:** [BR96] Let  $\mathcal{K}_{rsa}$  be a RSA generator and  $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$  the associated FDH RO-model signature scheme. Let  $A$  be a uf-cma adversary making  $q_s$  signing queries and  $q_H$  queries to the RO  $H$  and having running time at most  $t$ . Then there is an inverter  $I$  such that

$$\mathbf{Adv}_{\mathcal{DS}}^{\text{uf-cma}}(A) \leq (q_s + q_H + 1) \cdot \mathbf{Adv}_{\mathcal{K}_{rsa}}^{\text{owf}}(I).$$

Furthermore the running time of  $I$  is that of  $A$  plus the time for  $\mathcal{O}(q_s + q_H + 1)$  computations of the RSA function.

# Choosing a modulus size

Say we want 80-bits of security, meaning a time  $t$  attacker should have advantage at most  $t \cdot 2^{-80}$ :

- For inverting RSA, this is provided by a 1024 bit modulus assuming NFS is the best attack.
- But according to the BR96-reduction, FDH could be less secure than RSA by a factor of  $q_S + q_H + 1$ , so that a bigger modulus would be needed for 80-bit security.

## Choosing a modulus size

Say we want 80-bits of security, meaning a time  $t$  attacker should have advantage at most  $t \cdot 2^{-80}$ . The following shows modulus size  $k$  and cost  $c$  of one exponentiation, with  $q_H = 2^{60}$  and  $q_s = 2^{45}$  in the FDH case:

Task	$k$	$c$
Inverting RSA	1024	1
Breaking FDH as per [BR96] reduction	3700	47

This (for simplicity) neglects the running time difference between  $A, I$ .

This motivates getting tighter reductions for FDH, or alternative schemes with tighter reductions.

**Theorem:** [Co00] Let  $\mathcal{K}_{rsa}$  be a RSA generator and  $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$  the associated FDH RO-model signature scheme. Let  $A$  be a uf-cma adversary making  $q_s$  signing queries and  $q_H$  queries to the RO  $H$  and having running time at most  $t$ . Then there is an inverter  $I$  such that

$$\mathbf{Adv}_{\mathcal{DS}}^{\text{uf-cma}}(A) \leq \mathcal{O}(q_s) \cdot \mathbf{Adv}_{\mathcal{K}_{rsa}}^{\text{owf}}(I).$$

Furthermore the running time of  $I$  is that of  $A$  plus the time for  $\mathcal{O}(q_s + q_H + 1)$  computations of the RSA function.



## Choosing a modulus size

Say we want 80-bits of security, meaning a time  $t$  attacker should have advantage at most  $t \cdot 2^{-80}$ . The following shows modulus size  $k$  and cost  $c$  of one exponentiation, with  $q_H = 2^{60}$  and  $q_S = 2^{45}$  in the FDH case:

Task	$k$	$c$
Inverting RSA	1024	1
Breaking FDH as per [BR96] reduction	3700	47
Breaking FDH as per [Co00] reduction	2800	21

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

algorithm $\mathcal{S}_{N,d}^{h,g_1,g_2}(M)$ $r \xleftarrow{\$} \{0,1\}^{160}$ $w \leftarrow h(M \parallel r)$ $r^* \leftarrow g_1(w) \oplus r$ $y \leftarrow 0 \parallel w \parallel r^* \parallel g_2(w)$ return $y^d \bmod N$	algorithm $\mathcal{V}_{N,e}^{h,g_1,g_2}(M, x)$ $y \leftarrow x^e \bmod N$ $b \parallel w \parallel r^* \parallel P \leftarrow y$ $r \leftarrow r^* \oplus g_1(w)$ if $(g_2(w) \neq P)$ then return 0 if $(b = 1)$ then return 0 if $(h(M \parallel r) \neq w)$ then return 0 return 1
---	---

Here  $h, g_1: \{0,1\}^* \rightarrow \{0,1\}^{160}$  and  $g_2: \{0,1\}^* \rightarrow \{0,1\}^{k-321}$  are random oracles where  $k = |N|$ .

## Choosing a modulus size

Say we want 80-bits of security, meaning a time  $t$  attacker should have advantage at most  $t \cdot 2^{-80}$ . The following shows modulus size  $k$  and cost  $c$  of one exponentiation, with  $q_H = 2^{60}$  and  $q_S = 2^{45}$  in the FDH and PSS cases:

Task	$k$	$c$
Inverting RSA	1024	1
Breaking FDH as per [BR96] reduction	3700	47
Breaking FDH as per [Co00] reduction	2800	21
Breaking PSS as per [BR96] reduction	1024	1

## Choosing a modulus size

There are no attacks showing that FDH is less secure than RSA, meaning there are no attacks indicating FDH with a 1024 bit modulus has less than 80 bits of security. But to get the provable guarantees we must use larger moduli as shown, or use PSS.

- RSA PKCS#1 v2.1.
- IEEE P1363a
- ANSI X9.31
- RFC 3447
- ISO/IEC 9796-2
- CRYPTREC
- NESSIE

# 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  $sk = x \xleftarrow{\$} \mathbf{Z}_{p-1}$

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

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

$r \leftarrow g^k \pmod p$

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

**return**  $(r, s)$

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

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

**then return** 0

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

**then return** 1

**else return** 0

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

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

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

# Security of ElGamal Signatures

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

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

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

$r \leftarrow g^k \bmod p$

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

**return**  $(r, s)$

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

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

**then 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  $r$  ...?

# Forgery of ElGamal Signatures

Adversary has better luck if it picks  $m$  itself:

**Adversary**  $A(X)$

$r \leftarrow gX \pmod{p}$ ;  $s \leftarrow (-r) \pmod{p-1}$ ;  $m \leftarrow s$

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

Then:

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

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  $sk = x \xleftarrow{\$} \mathbf{Z}_{p-1}$

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

**Algorithm**  $S_x(M)$

$m \leftarrow H(M)$

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

$r \leftarrow g^k \bmod p$

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

**return**  $(r, s)$

**Algorithm**  $V_X(M, (r, s))$

$m \leftarrow H(M)$

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

**then return** 0

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

**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  $sk = x \xleftarrow{\$} \mathbf{Z}_{p-1}$

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

**Algorithm**  $\mathcal{S}_x(M)$

$m \leftarrow H(M)$

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

$r \leftarrow g^k \bmod p$

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

**return**  $(r, s)$

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

$m \leftarrow H(M)$

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

**then return** 0

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

**then return** 1

**else return** 0

Requirements on  $H$ :

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

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  $a, n \mapsto a^n \bmod p$  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.

- 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  $q$
- $H: \{0, 1\}^* \rightarrow \mathbf{Z}_q$  a hash function
- Signer keys:  $pk = X = g^x \in \mathbf{Z}_p^*$  and  $sk = x \xleftarrow{\$} \mathbf{Z}_q$

**Algorithm**  $S_x(M)$ 

```

 $m \leftarrow H(M)$ 
 $k \xleftarrow{\$} \mathbf{Z}_q^*$ 
 $r \leftarrow (g^k \bmod p) \bmod q$ 
 $s \leftarrow (m + xr) \cdot k^{-1} \bmod q$ 
return  $(r, s)$ 

```

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

```

 $m \leftarrow H(M)$ 
 $w \leftarrow s^{-1} \bmod q$ 
 $u_1 \leftarrow mw \bmod q$ 
 $u_2 \leftarrow rw \bmod q$ 
 $v \leftarrow (g^{u_1} X^{u_2} \bmod p) \bmod q$ 
if  $(v = r)$  then return 1
else return 0

```

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

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, even if  $H$  is a RO. Proofs are 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.

- Let  $G = \langle g \rangle$  be a cyclic group of prime order  $p$
- $H: \{0, 1\}^* \rightarrow \mathbf{Z}_p$  a hash function
- Signer keys:  $pk = X = g^x \in G$  and  $sk = x \stackrel{\$}{\leftarrow} \mathbf{Z}_p$

## Algorithm $\mathcal{S}_x(M)$

$r \stackrel{\$}{\leftarrow} \mathbf{Z}_p$

$R \leftarrow g^r$

$c \leftarrow H(R\|M)$

$a \leftarrow xc + r \pmod p$

**return**  $(R, a)$

## Algorithm $\mathcal{V}_X(M, (R, a))$

if  $R \notin G$  then return 0

$c \leftarrow H(R\|M)$

if  $g^a = RX^c$  then return 1

else return 0

# 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.