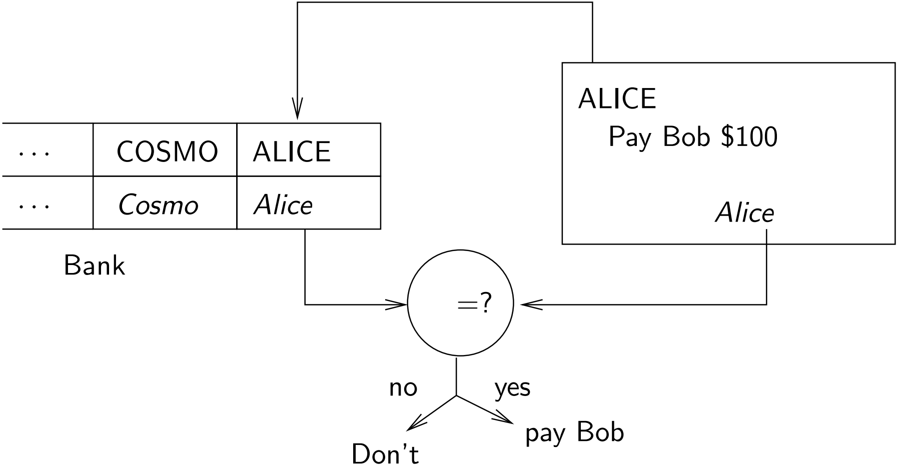
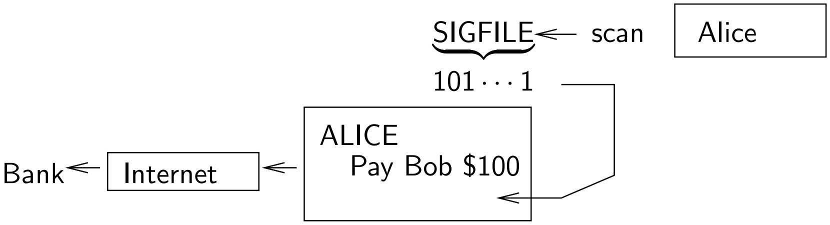
DIGITAL SIGNATURES

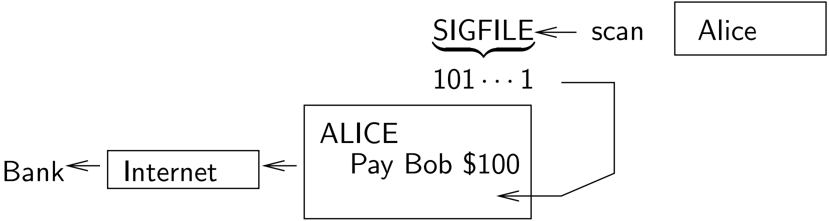
Signing by hand



Signing electronically



Signing electronically

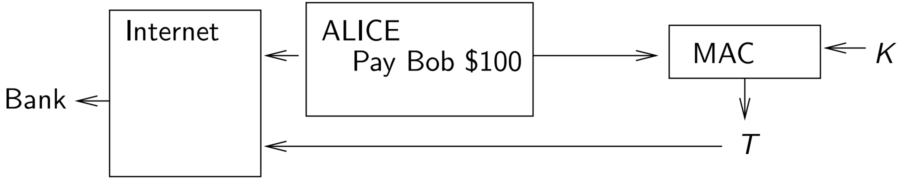


Problem: signature is easily copied

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

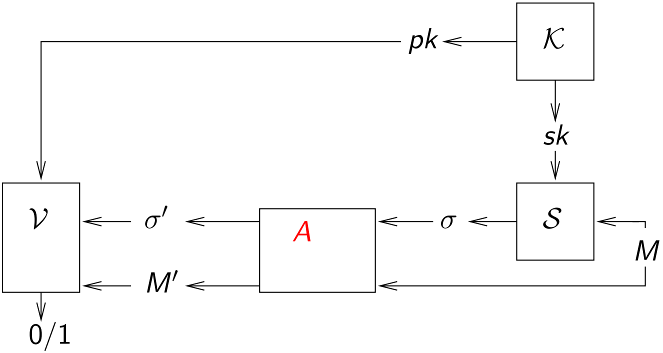


A digital signature will have additional attributes:

- 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.

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

Security of a DS scheme

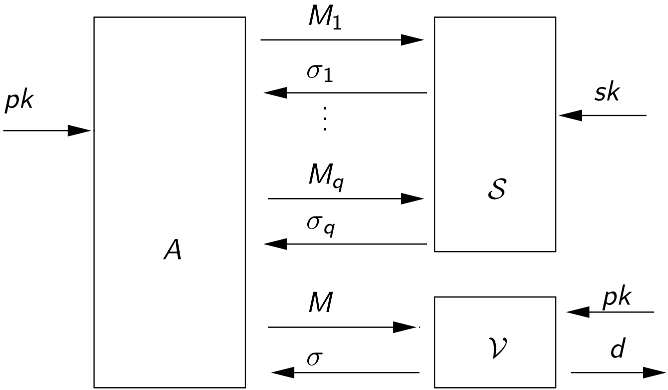
Possible adversary goals

- find *sk*
- Forge

Possible adversary abilities

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

uf-cma adversaries



A wins if

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

Security of a DS scheme

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 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 $(d = 1 \land M \notin S)$

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

The uf-cma advantage of A is

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

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.

RSA signatures

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 $\mathcal{S}_{N,d}(y)$:	**Alg** $\mathcal{V}_{N,e}(y, x)$:
$x \leftarrow y^d \bmod N$	**if** $x^e \equiv y \pmod{N}$ **then return** 1
return x	**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 $\forall y_1, y_2 \in \mathbb{Z}_N^*$

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

That is

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

where

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

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 $y_1, y_2 \in \mathbb{Z}_N^* - \{1\}$
$x_1 \leftarrow \textbf{Sign}(y_1)$; $x_2 \leftarrow \textbf{Sign}(y_2)$
return $(y_1 y_2 \bmod N, x_1 x_2 \bmod N)$

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 $H\colon \{0,1\}^* \to \mathbb{Z}_N^*$ 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 $\mathcal{S}_{N,d}(M)$:	**Alg** $\mathcal{V}_{N,e}(M, x)$:
$y \leftarrow H(M)$	$y \leftarrow H(M)$
$x \leftarrow y^d \bmod N$	**if** $x^e \equiv y \pmod N$ **then return** 1
return x	**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 $x_1 = H(M_1)^d \bmod N$ 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

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

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 $h\colon \{0,1\}^* \to \{0,1\}^{160}$ be a hash function (like SHA-1) and let $n = |N|_8 = 1024/8 = 128$.

Then

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

And

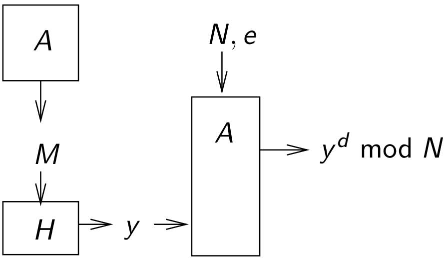
$$\mathcal{S}_{N,d}(M) = H_{PKCS}(M)^d \bmod N$$

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

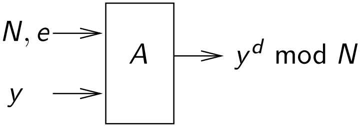
Does 1-wayness prevent forgery?

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

H_{PKCS} revisited

Recall

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

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.

Choice of H

A "better" choice of H might be something like

$$
\begin{aligned}
H(M) &= \text{first } n \text{ bytes of} \\
&\quad \mathrm{SHA1}(1 \,||\, M) \,||\, \mathrm{SHA1}(2 \,||\, M) \,||\, \cdots \,||\, \mathrm{SHA1}(11 \,||\, M)
\end{aligned}
$$

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

nnnnn

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

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

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

Security of ElGamal Signatures

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

Algorithm $\mathcal{S}_x(m)$
$k \stackrel{\$}{\leftarrow} \mathbf{Z}_{p-1}^*$
$r \leftarrow g^k \mod p$
$s \leftarrow (m - xr) \cdot k^{-1} \mod (p-1)$
return (r, s)

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

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

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.

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

Algorithm $\mathcal{V}_X(M, (r, s))$
$m \leftarrow H(M)$
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.

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

Algorithm $\mathcal{V}_X(M, (r, s))$
$m \leftarrow H(M)$
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 $a, n \mapsto a^n \bmod p$ 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.

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 $sk = x \xleftarrow{\$} \mathbf{Z}_q$

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

Algorithm $\mathcal{V}_X(M, (r, s))$
$m \leftarrow H(M)$
$w \leftarrow s^{-1} \mod q$
$u_1 \leftarrow mw \mod q$
$u_2 \leftarrow rw \mod q$
$v \leftarrow (g^{u_1} X^{u_2} \mod p) \mod q$
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 are known for variants.

Schnorr Signatures

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.

Schnorr Signatures

- 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 $sk = x \xleftarrow{\$} \mathbf{Z}_p$

Algorithm $\mathcal{S}_x(M)$
$r \xleftarrow{\$} \mathbf{Z}_p$
$R \leftarrow g^r$
$c \leftarrow H(R \| M)$
$a \leftarrow xc + r \bmod p$
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.