Financial Cryptography ‘ 2001

19-22 February 2001
Grand Cayman Islands - BWI

Monotone Signatures

Joint work with David Naccache and Christophe Tymen
(Gemplus, France)

David Pointcheval
Département d ’Informatique
ENS - CNRS

David.Pointcheval@ens.fr   http://www.di.ens.fr/users/pointche

---

Overview

- Introduction
- Monotone Signatures
- Attackers
  - Immediate Attacks
  - Delayed Attacks
- Optimized Solution
- Conclusion

Cryptography

Cryptography proposes many solutions for

- Confidentiality
- Authentication
- Integrity
- …

but often based on some secret data

Corruption

However, no secret can be guaranteed for any time

- Corruption
- Kidnapping

to force the authority to publish the secret data in the newspaper

E-cash

We can easily prevent duplication of coins while checking double/multiple spending

However, we are aware of the problem caused by the so-called

Bank-Robbery Attack

$\Rightarrow$ protections have been found, but they are very costly

ID Cards

Previous protections (against Bank-Robbery Attacks) require an on-line context, which is not suitable to any situation such as ID-cards, Driving License, etc

Another possibility: threshold signature but one cannot prevent a massive corruption of k share-holders

Achievement

A Signature Scheme such that,
after a corruption, one updates
the verification process
in such a way that

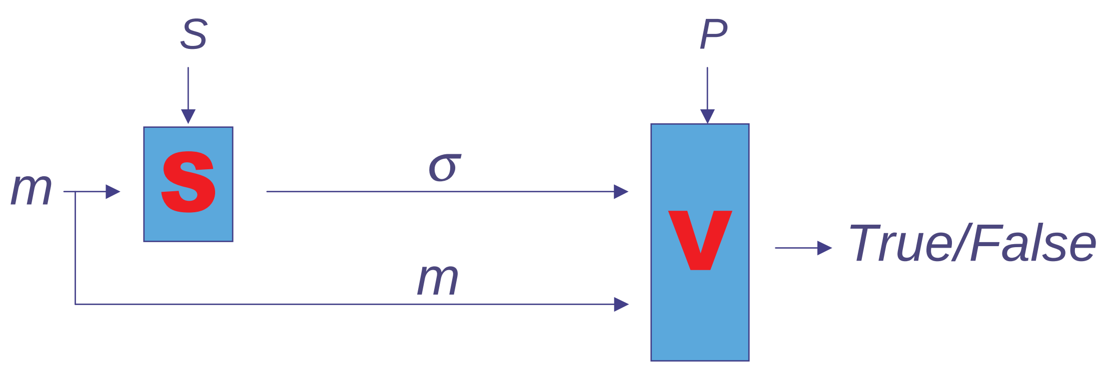
only "really" valid signatures
are accepted

However, at the time of the corruption,
the adversary "thinks"
he holds the secret key

Signatures

Signing Algorithm **S**

Verification Algorithm **V**



Security: it is impossible to produce
a new valid pair (m,σ)

Monotone Predicates

The Verification Algorithm checks a predicate: $\mathbf{P}(m,\sigma) = \mathbf{V}_P(m,\sigma)$

Predicates $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n$ are said to be ***monotone*** if for any input x

$$\mathbf{P}_n(x) \Rightarrow \mathbf{P}_{n\text{-}1}(x) \Rightarrow \ldots \Rightarrow \mathbf{P}_2(x) \Rightarrow \mathbf{P}_1(x)$$

- $\mathbf{P}_1(x)$ = x is an integer
- $\mathbf{P}_2(x)$ = x is even
- $\mathbf{P}_3(x)$ = x is zero

Monotone Signature

- A Key Generation Algorithm
  $\mathbf{G}(1^k,1^n) \to (S_1,\ldots,S_n;P_1,\ldots,P_n)$
- A Signing Algorithm
  $\mathbf{S}_{S_1,\ldots,S_n}(m) \to \sigma$
- A list of n ***Monotone*** Verifying Algorithms
  $\mathbf{V}^i_{P_1,\ldots,P_i}(m,\sigma) \to$ *True/False*
  for $i=1,\ldots,n$

Properties

As for any Signature Scheme:

- Completeness:
  $\sigma = \mathbf{S}_{S_1,\ldots,S_n}(m) \Rightarrow \mathbf{V}^n_{P_1,\ldots,P_n}(m,\sigma) = True$
- Soundness: (No Existential Forgery)
  for any adversary A, the probability of
  $(m,\sigma) \leftarrow A(S_1,\ldots,S_{i-1},P_1,\ldots,P_i)$:
  $\mathbf{V}^i_{P_1,\ldots,P_i}(m,\sigma) = True$
  is negligible

Indistinguishability

Missing public keys must not change the distribution:

For any $i \leq n$, there exists a simulator $\mathcal{S}$ such that the distributions, for any m

- $\mathcal{S}_{S_1,\ldots,S_i}(m)$
- $\mathbf{S}_{S_1,\ldots,S_n}(m)$

are indistinguishable for someone who does not know the $S_{i+1},\ldots,S_n$

Attacks

As usual, one can consider

- no-message attacks:

  the adversary just knows the verification algorithm (*i.e.* the public key)

- known-message attacks:

  she knows some message-signature pairs

- (adaptively) chosen-message attacks:

  she has access to a signature oracle

Corruption

But we have to consider the corruption: the adversary

- gets some secret keys $S_1,\ldots,S_j$
- checks their validity w.r.t. $P_1,\ldots,P_j$

- immediate attacks:

  the adversary forges signatures before the update to $\mathbf{V}^{j+1}_{P_1,\ldots,P_{j+1}}$ (thus without P_{j+1})

- delayed attacks:

  the adversary waits for the new verification algorithm (with P_{j+1}) before starting to forge

Immediate Attacks

- $\mathcal{A}$ runs the Key Generation Algorithm
  $\mathbf{G}(1^k,1^n) \rightarrow (S_1,\ldots,S_n;P_1,\ldots,P_n)$
- $\mathcal{A}$ publishes a partial public key $(P_1,\ldots,P_i)$
- $\mathcal{A}$ produces signatures $\mathbf{S}_{S_1,\ldots,S_n}(m) \rightarrow \sigma$
- Corruption: the adversary gets $(S_1,\ldots,S_j)$
- Forgeries: the adversary forges new signatures
- $\mathcal{A}$ publishes new public keys $(P_{i+1},\ldots)$

Random-looking Redundancy

To prevent immediate attacks, one can simply use

- subliminal channel (low bandwidth)
- secret-redundancy

From a signature scheme (**G**,**S**,**V**), one signs a redundant message

$\mu = m \,\|\, r$, where r “looks” random

but $r_i = f_i\,(m,r_1,\ldots,r_{i-1})$ for some i

Symmetric Monotone Signatures

The published verification key is just the public key of the basic scheme

After corruption (and thus publication of the signing key), one publishes some redundancy criteria

$\Rightarrow$ immediate forgeries will be spotted

Further corruptions (under immediate attacks) will be prevented until some secret redundancy remains.

Delayed Attacks

- $\mathcal{A}$ runs the Key Generation Algorithm $\mathbf{G}(1^k,1^n) \rightarrow (S_1,\ldots,S_n;P_1,\ldots,P_n)$
- $\mathcal{A}$ publishes a partial public key $(P_1,\ldots,P_i)$
- $\mathcal{A}$ produces signatures $\mathbf{S}_{S_1,\ldots,S_n}(m) \rightarrow \sigma$
- Corruption: the adversary gets $(S_1,\ldots,S_j)$
- $\mathcal{A}$ publishes new public keys $(P_{i+1},\ldots)$
- Forgeries: the adversary forges new signatures

Concatenation of Signatures

To prevent delayed attacks, one can concatenate mixture of signatures and random strings:

$$\mathbf{S}_{S_1,\ldots,S_n}(m) = \mathbf{S}_{S_1}(m) \,||\, \mathbf{S}_{S_2}(m) \,||\, R_3 \,||\, \mathbf{S}_{S_4}(m) \,||\, \ldots \,||\, R_n$$

But then, the distributions, for any key S_i, and any message m,

$$\mathbf{S}_{S_i}(m) \text{ and } R \leftarrow \{0,1\}^l$$

must be indistinguishable

Example: Schnorr's Signature

$\mathbf{G} = \langle g \rangle$ of prime order q

x : **secret** key $y=g^x$: **public** key

Signature of the message m :

from a random $k \in \mathbf{Z}_q$ get $r=g^k$

then $e=h(m,r)$ and $s = k-xe \bmod q$

$\sigma = (e,s)$

Verification of (m,σ) : test whether $e=h(m, g^s y^e)$

Actually $\mathbf{S}(m) = (e,s) \in_R \mathbf{Z}_q \times \mathbf{Z}_q$

$\Rightarrow$ indistinguishable from a random pair

Don't use (r,s) as output signature!

Properties

- At least n Schnorr's signatures to prevent up to n corruptions
- And about n random values as well

Therefore:

- Cost: n times the basic computational time
  - n exponentiation per signature
  - $2i$ exponentiations per verification
- Length: $2n$ times the basic length
  $\Rightarrow 2n \times 320$ bits $= 80\ n$ Bytes

Okamoto-Schnorr Signature

Extending the Okamoto's variant:
$\mathbf{G} = \langle g \rangle$ of order q and $g_1,\ldots,g_n \in \mathbf{G}$

- $(x_1,\ldots,x_n)$: **secret** key
- $y=g_1^{x_1}\ldots g_n^{x_n}$: **public** key

- Signature of m:
  - $t_1,\ldots,t_n$ and then $r=g_1^{t_1}\ldots g_n^{t_n}$
  - get $e=h(m,r)$
  - $s_i = t_i - x_i e \bmod q$
- Verification:

$$e=h(m, g_1^{s_1}\ldots g_n^{s_n} y^e)$$

Degrees of Freedom

$$e = h(m, g_1^{s_1} \ldots g_n^{s_n} y^e)$$

Without any relation between the g_i's, one has no freedom about the s_i's, since e is provided once the t_i's are fixed

With some relations, one can hide secret redundancy into some s_i's.

The more relations are known, the more of s_i's can be chosen:

$$s_i = f_i(m||r)$$

Properties

- At least k relations must exist to prevent up to k corruptions
- And about k independent values as well

Therefore:

- Cost:
  - k exponentiation per signature
  - $2k$ exponentiations per verification
- Length: only $2k+1$ elements in $\mathbf{Z}_q$

  $\Rightarrow (2k+1) \times 160$ bits $\approx 40\ k$ Bytes

Conclusion

Monotone Signatures propose new features

- Resistance against many corruptions,
- Prevention of the immediate attacks:
  - Symmetric Monotone Signatures
    which are almost as efficient
    as the basic signature scheme
- Prevention of the delayed attacks:
  - Concatenation of Signatures
  - Signatures with various degrees of freedom can improve efficiency