Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks

joint work with Pierre-Alain Fouque

Asiacrypt '01
Gold Coast - Australia
December 2001

David Pointcheval
Département d'Informatique
ENS - CNRS

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

Overview

- Distributed cryptography
- Chosen-ciphertext attacks
- Naor-Yung construction
- Our construction
- Conclusion

Distributed cryptography

In classical cryptography,
only one server for signing or decrypting

- one people has all the power

$\Rightarrow$ just one machine to attack

- to get all the secret
- to disable the service

In distributed cryptography,
power is distributed among several servers

Threshold cryptography

The crucial operation is distributed among
n servers such that k are required in

- the signature process
- the decryption process

The power is distributed

But also, several machines to attack

- k to get the whole secret
- n-k+1 to disable the service

if $n \geq 2k-1 \Rightarrow k$ servers to attack

Adversaries

We consider t-adversaries,

which corrupt up to t servers ($n \geq 2t+1$):

- Static: choose them at the beginning
- Adaptive: choose them dynamically

- Passive: get the t secret parts
- Active: take the entire control of them

Threshold cryptosystems

Key generation: public key k_p,
distributed private keys k_{s_i} ($i = 1, \ldots, n$)
and possibly verification keys k_{v_i}

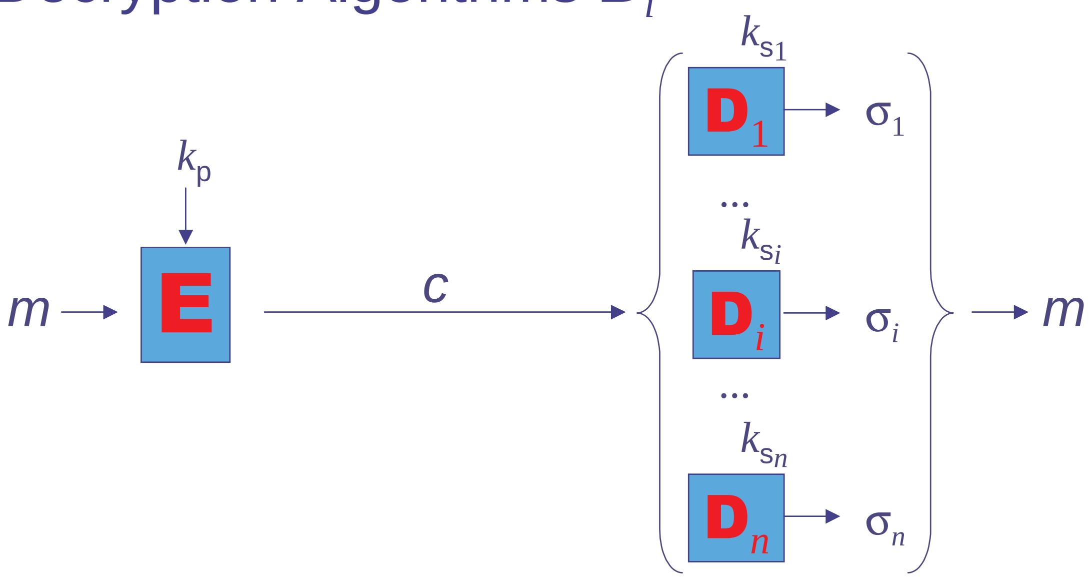
Encryption: $\mathbf{E}(k_p,m) \rightarrow$ ciphertext c

Decryption: $\mathbf{D}_i(k_{s_i},c) \rightarrow$ decryption share σ_i
maybe with some interactions

Combination: with k correct decryption shares,
and the verification keys, one recovers m

Distributed cryptosystems

- Encryption Algorithm **E**
- Decryption Algorithms $\mathbf{D}_i$



Encryption: security notions

- Security (impossibility to):
  - one-wayness: recover the whole plaintext
  - semantic security: learn any information
- Attacks:
  - chosen-plaintext: with the public-key only
  - chosen-ciphertext (adaptively):
    access to a decryption oracle

Chosen-ciphertext attacks

In distributed systems,
the adversary gets more information:

for a given ciphertext (chosen or not),
the adversary sees all the decryption shares,
the plaintext, and all the communications

Chosen-ciphertext attacks:

the adversary gets t secret keys,
and can run all the decryption algorithms
on any ciphertext of her choice

Classical cryptosystem: $n = k = 1$ and $t = 0$

Distributed computation vs. distributed decryption

- One "can" distribute the evaluation of any function on secret inputs
- One can efficiently distribute the inversion of classical primitives (RSA, El Gamal, etc)
- But most of efficient chosen-ciphertext secure cryptosystems (generic conversions):
  - invert the basic primitive $\Rightarrow$ alleged plaintext
  - check some redundancy (with hashing)

$\Rightarrow$ the adversary learns the alleged plaintext

Publicly verifiable validity

A nice solution:

- the validity of the ciphertext can be checked first, and better, in a public way
- the decryption process would be:
  - each server checks the validity of the ciphertext
  - if it is valid, builds the decryption share

Since this last step can be done efficiently, with no interaction, for several primitives, one gets an efficient decryption process

The Naor-Yung paradigm

Naor and Yung (‘90): on any IND-CPA (**K**,**E**,**D**)

(**K’**,**E’**,**D’**) is defined as follows:

- **K’** runs twice **K**, to get two pairs of keys
  $\mathbf{K'}(1^k) \rightarrow (k^1_{\mathrm{s}}, k^1_{\mathrm{p}})$ and $(k^2_{\mathrm{s}}, k^2_{\mathrm{p}})$
- **E’** encrypts twice the message m,
  $c_1 = \mathbf{E}(k^1_{\mathrm{p}},m)$ and $c_2 = \mathbf{E}(k^2_{\mathrm{p}},m)$
  provides a proof p of “$\mathbf{D}(k^1_{\mathrm{s}},c_1) = \mathbf{D}(k^2_{\mathrm{s}},c_2)$”
- **D’** checks the proof, and decrypts the ciphertexts:

$$\mathbf{D'}((k^1_{\mathrm{s}}, k^2_{\mathrm{s}}),(c_1 ,c_2,p)) = m = \mathbf{D}(k^1_{\mathrm{s}},c_1) = \mathbf{D}(k^2_{\mathrm{s}},c_2)$$

The Naor-Yung proof

In the common random string model,
p can be a NIZK of membership

Decryption simulator: knows $k^2{}_s$ (for ex.)
$\Rightarrow$ perfect simulation unless wrong proof

Reduction: use of ZK simulator

- the adversary outputs m_0 and m_1
- one gets $c_1 = \mathbf{E}(k^1{}_s, m_b)$ from the challenger
- one computes $c_2 = \mathbf{E}(k^2{}_s, m_d)$ for a random d
- one simulates a proof p on c_1 and c_2

$\Rightarrow$ (c_1, c_2, p) is the challenge ciphertext

The Naor-Yung result

With probability 1/2, the simulator builds
a wrong proof p on c_1 and c_2

ZK says

- valid proofs do not leak any information
- nothing about simulated (wrong) proofs

$\Rightarrow$ the simulated wrong proof may help
the adversary to forge a wrong proof

$\Rightarrow$ incorrect decryption simulation

Hence, non-adaptive chosen-ciphertext attacks
(*a.k.a.* lunchtime attacks)

The Random Oracle Model

In the random oracle model:

- efficient NIZK proofs of membership
- easy and perfect simulations
- **simulation soundness:**
  any simulated proof (correct or wrong) does not help to forge a wrong proof

$\Rightarrow$ correct decryption simulation

Hence the adaptive chosen-ciphertext attacks

Our construction

Exactly the same as the Naor-Yung,
but in the random oracle model
$\Rightarrow$ simulation soundness of the NIZK proofs

Reduction: use of ZK simulator and ROM

- the adversary outputs m_0 and m_1
- one gets $c_1 = \mathbf{E}(k^1{}_s, m_b)$ from the challenger
- one computes $c_2 = \mathbf{E}(k^2{}_s, m_d)$ for a random d
- one simulates a proof p on c_1 and c_2,
  defining the random oracle at some point

simulation soundness $\Rightarrow$ does not help the adversary

Conclusion

Cryptosystems

1. easily based on any IND-CPA scheme
2. efficient: just twice as slow
3. the validity of the ciphertext
   can be checked publicly

The IND-CPA scheme can be distributed
$\Rightarrow$ the construction provides
a distributed IND-CCA cryptosystem

E.g. El Gamal (DDH), Paillier (HR)