Malicious Cryptography on “Secure” Devices

Mirosław Kutyłowski

Wrocław University of Technology
Institute of Mathematics and Computer Science

EFPE 2008

Vulnerability of Software

Infection possibilities

1. it is easy to hide malicious code in big systems
2. inspecting what software is really doing in a rigorous way is impossible in nontrivial cases
   $\Leftarrow$ basic mathematical facts about *halting problem* ...

Secure signature devices

- it must be checkable what a device is really doing
- any security relevant change must be evident to the user

Is EU Directive ignoring the mathematical facts known already for decades?

Advantages of Hardware Solutions

secure signature creation devices

Idea

1. perform sensitive operations (storing key, signing) on a dedicated unit

Advantages of Hardware Solutions

secure signature creation devices

Idea

1. perform sensitive operations (storing key, signing) on a dedicated unit
2. implement only those functionalities that are absolutely necessary – a simple system is easier to check

Advantages of Hardware Solutions

secure signature creation devices

Idea

1. perform sensitive operations (storing key, signing) on a dedicated unit
2. implement only those functionalities that are absolutely necessary – a simple system is easier to check
3. implement in hardware where a change is impossible

Advantages of Hardware Solutions

secure signature creation devices

Idea

1. perform sensitive operations (storing key, signing) on a dedicated unit
2. implement only those functionalities that are absolutely necessary – a simple system is easier to check
3. implement in hardware where a change is impossible
4. implement in hardware with an extra physical protection

Solution

a dedicated signing chip

Misunderstandings

Common problems

1. expanding legal definition of secure device on PC (Polish problem)

   saying that a software on PC can satisfy the requirements is a lie
2. increasing functionality of a signing chip – new applications ...
3. **how do you know that a chip tested is the same as the chip you get?**

Fault cryptanalysis
idea

Mechanism of a fault attack

1. a chip might be tamper-proof, but some kind of faults are inevitable (piece of uranium on top of a chip, particles changing state of registers)
2. a computation performed with a (random) fault and correctly on the same input
3. a difference between the correct output and the faulty output may show the secret key used
4. classical attack of this type: on RSA with Chinese Remainder Theorem implementation

Fault attack countermeasures

Solutions

1. check the signature on chip before outputting it
2. yet some information can be leaked (approximate number of ones in the key)

Fault attack countermeasures

Solutions

1. check the signature on chip before outputting it
2. yet some information can be leaked (approximate number of ones in the key)
3. checking on-the-fly

Fault attack countermeasures

Solutions

1. check the signature on chip before outputting it
2. yet some information can be leaked (approximate number of ones in the key)
3. checking on-the-fly

Problems

1. slowing down signature creation
2. increasing cost
3. faults in checks?

Techniques of hidden information transfer

kleptographic channel

Properties

1. no extra information is sent outside
2. output information according to protocol description, no audit can find irregularities,
3. only knowledge of secret key (not included in a chip) enables retrieval of the encoded information.

Properties

1. no extra information is sent outside
2. output information according to protocol description, no audit can find irregularities,
3. only knowledge of secret key (not included in a chip) enables retrieval of the encoded information.

Implementation

1. on protocol level
2. in SSL
3. ... anywhere using random parameters

Technical idea

kleptographic channel- example

1. a protocol uses g^k, where g is a generator of a group with hard Discrete Logarithm problem, k chosen at random,

Technical idea

kleptographic channel- example

1. a protocol uses g^k, where g is a generator of a group with hard Discrete Logarithm problem, k chosen at random,
2. $Y = g^x$ is a public key to be used by infected code, x – private key

Technical idea

kleptographic channel- example

1. a protocol uses g^k, where g is a generator of a group with hard Discrete Logarithm problem, k chosen at random,
2. $Y = g^x$ is a public key to be used by infected code, x – private key
3. infected chip computes k so that SHA-1(Y^k) betrays secret information

Technical idea

kleptographic channel- example

1. a protocol uses g^k, where g is a generator of a group with hard Discrete Logarithm problem, k chosen at random,
2. $Y = g^x$ is a public key to be used by infected code, x – private key
3. infected chip computes k so that SHA-1(Y^k) betrays secret information
4. adversary monitors transmission z, computes

$$U := z^x, \quad \mathrm{SHA\text{-}1}(U)$$

Technical idea

kleptographic channel- example

1. a protocol uses g^k, where g is a generator of a group with hard Discrete Logarithm problem, k chosen at random,
2. $Y = g^x$ is a public key to be used by infected code, x – private key
3. infected chip computes k so that SHA-1(Y^k) betrays secret information
4. adversary monitors transmission z, computes

$$U := z^x, \quad \text{SHA-1}(U)$$

Check: for $z = g^k$ from transmission:

$$U = z^x = (g^k)^x = (g^x)^k = Y^k$$

General Situation
corollaries

1. no reliable solution so far,
2. electronic devices may make more trouble than help in case of sensitive appications (voting, signing)

General Situation

corollaries

1. no reliable solution so far,
2. electronic devices may make more trouble than help in case of sensitive appications (voting, signing)

What to do?

Trusted Platforms? What technology of testing without giving a backdoor to a secret key?

Unpredictability versus Randomness

verifiable "randomness"

Idea

1. in many situation we do not require random parameters
2. **we require parameters that cannot be guessed** by malicious Mallet trying to break the scheme
3. afterward the secret parameters can be revealed for many protocols.

Unpredictability versus Randomness

verifiable “randomness”

Idea

1. in many situation we do not require random parameters
2. **we require parameters that cannot be guessed** by malicious Mallet trying to break the scheme
3. afterward the secret parameters can be revealed for many protocols.
   but not for DSA!

Unpredictability instead of Randomness

DH case

Original DH key exchange

1. Alice chooses k_1 at random, computes $z_1 := g^{k_1}$
2. Bob chooses k_2 at random, computes $z_2 := g^{k_2}$
3. Alice and Bob exchange z_1 and z_2,
4. Alice and Bob compute shared key K:

$$K := z_1^{k_2} \quad \text{or by} \quad K := z_2^{k_1}$$

Danger

$z_i^x = Y^{k_i}$ for $Y = g^x$ can encode the next exponent
– full kleptographic attack possible

Patch to DH
CANS'2006, Gołebiewski, Kutyłowski, Zagórski

Modified DH key exchange

1. Alice and Bob agree upon parameter a in clear (a might be the current time)
2. Alice computes $k_1 := \mathrm{hash}(\mathrm{RSA}_{Alice}(a))$, computes $z_1 := g^{k_1}$
3. Bob computes $k_1 := \mathrm{hash}(\mathrm{RSA}_{Alice}(a))$, computes $z_2 := g^{k_2}$
4. Alice and Bob exchange z_1 and z_2,
5. Alice and Bob compute shared key K:

$$K := z_1^{k_2} \quad \text{or by} \quad K := z_2^{k_1}$$

6. using channel encrypted with K, Alice and Bob reveal themselves signatures of a
7. Alice and Bob check k_1 and k_2 used

Derandomization

key defense idea

unexpectedness instead of randomness

1. in many cases we do not need random strings, we need string that cannot be guessed by third parties
2. deterministic signatures cannot be predicted by third parties

Derandomization
unsolved problems

Discrete Log Signatures

revealing the random exponent used reveals the signing key

Problem

we do not know any technique that would secure DL signatures against kleptography

Secure signature devices

kleptography issues

Corollaries

1. unless new algorithms developed, DL schemes should not be declared suitable for secure signature creation devices
2. deterministic schemes seem to be more suitable

Thanks for your attention