A new use of a new crypto attack (Re: Shamir, RISKS-18.54)

Jean-Jacques Quisquater <Quisquater@dice.ucl.ac.be>
Thu, 24 Oct 1996 00:08:55 +0200 (MET DST)

Research announcement:

Short cut for exhaustive key search using fault analysis:
Applications to DES, MAC, keyed hash function, identification protocols, ...

Jean-Jacques Quisquater,
UCL Crypto Group, Louvain-la-Neuve, Belgium, jjq@dice.ucl.ac.be

23 Oct 1996

(ABSTRACT and DRAFT)

I confirm that the timing attack was well known to designers of smart cards for some time.
--- Jean-Jacques Quisquater, Dec. 20, 1995, sci.crypt

I'm a bit puzzled by the excitement over error analysis attacks -- they've been known for some time to cryptosystem implementors ...
--- Paul Kocher, Oct. 20, 1996, comp.security.misc

How to find a secret key faster than the exhaustive search without the help of the differential analysis.
--- This paper

During the last months very interesting programs, papers and announcements were released about the (cryptanalytic) use of transient faults in tamper resistant (or proof) devices by:
- some well-known anonymous authors (in the payTV context; SFS);
- Anderson and Kuhn (applications well fitted to the real world);
  [See Ross Anderson's comments in RISKS-18.52. PGN]
- Boneh, Demillo and Lipton: they specifically attack public key cryptosystems; their core attack bells an alert to the scientific community to publish faster;

- Biham and Shamir: they described how to obtain a secret key (e.g. for DES) using few ciphertexts.

This list is open, and we reserve some room in this paper for the - future unknown authors.

Here we describe a new use of such attacks in order to accelerate exhaustive keysearches in several contexts. We don't discuss if these attacks are feasible: our main goal is to enumerate all possible attacks and their cryptanalytic use against specific models. Knowing that will improve our trust about the current or future devices in order to obtain a reasonable level of security in a complete system.

Introduction

We suppose that the opponent is in possession of the secure device, able to know the (external) inputs and the outputs and to apply some physical constraints in order to trigger some transient errors at some random locations (RAM, registers, ...). We suppose that these errors do not interfere with the program used by the computations: these errors only modify some data at some stage of the computation. We do not here discuss the possibility of permanent errors: it will be explained in the full paper (incremental permanent errors with possible use of several secure devices, ...).

They are many protocols where the input message and the corresponding output message are accessible to everybody including the opponent if the device is physically in the hand of the user (maybe for some short period of time):

Let f be a public cryptographic function, computed by some secure device (smart card, secure black box, security hardware, ...), k a secret key, stored by the secure device, guessing its value is the goal of the opponent in this paper,

n the number of bits of the key k,
K is the set of all keys for f,
N the number of possible keys in K,
m an input message,
c an output message.

General protocol:

input: m
output: c = f(m, k)

Examples:

- encryption of m by any secret key algorithm (DES, IDEA, ...),
- decryption of m by any secret key algorithm (DES, IDEA, ...),
- computation of the hashing of m by the keyed hash function f (MAC, ...),
- m is a random number used by some identification protocol,
- ... .

Such protocols need very often some protections against possible abuses from (well-chosen by the opponent) messages m (see Biham-Shamir, Matsui, Vaudenay, ...) or not so random numbers. One necessary condition is to avoid the discovery of k by exhaustive key search: such a general search algorithm is now described.

Key search algorithm:

```
Given m, c
Enumerate all candidate keys i from K
  Compute f(m, i) = c_i
  If c_i = c then (output i and stop)
End of loop.
```

Mean work factor: N / 2.

Indeed, if we suppose that the key is unique for each pair (m, c) (it is not true for DES: they are sometimes collisions) then the number of computations of f is N/2 for the mean case and N for the worst case. The goal of this paper is to show how to improve such a complexity by the use of (randomly activated) transient faults in the secure device.

Model 1: Single fault in the secret key

- Working hypothesis:
the opponent is able to modify at a random location one bit of the key k (the new transient key is then k*) and to get the correct result of f(m, k*); after the reset of the secure device by the opponent, the internal secret key is again the correct one k. We suppose that the random modification is equidistributed on all n bits of the key k.

- Key search algorithm: given m,

1. Physical attack of the secure device:
   Obtain the n possible pairs (m, c_j) where c_j is equal to f(m, k_j);
   k_j is the modified key k with the jth bit being flipped;
   We need about n*log n "questions" to obtain the n different
   pairs (by the coupon collector paradox: in some way the dual of
   the birthday paradox), that is, if f is the DES for instance, about
   300 accesses to the secure device.

2. Enumerative key search on an external key search machine:

```
Given m, the c_j's
Enumerate (pseudo-) randomly all candidate keys i from K
  Compute f(m, i) = c_i
  If c_i = one of c_j's then (output i^*=i and stop)
End of loop.
```

3. Key correcting algorithm on an external computer:
   Given m, i^*, c,
   Enumerate the n values i coming from i^* with one flipped bit at
   every n possible positions
     Compute f(m, i) = c_i
     if c_i = c then (output i and stop) !the secret key is found!
   End of loop.

Mean work factor: (N / (2*n)) + n.

Indeed, the first step is very fast, the second step needs the mean work factor of the exhaustive key search divided by the number of bits of the key (if we suppose that the modified keys are randomly distributed in the key set K) and the last step needs indeed n computations of f.

For DES it means about 2^49 computations: let us recall that one FPGA device in one proposed implementation (see van Oorschot and Wiener) is able to do about 2^26 computations of DES, with key change, each second (using a pipelined implementation it is possible to compute a DES at each clock tick and we here suppose a very possible clock of 65 MHz). It means that one secret key will be recovered by such a small , accessible and inexpensive machine in 2^23 seconds, that is, less than 4 months. With p FPGAs working in parallel that time will be divided by p. The comparison operation in step 2 needs a modification of such a machine (a very easy step in software): it is a simple modification and thanks to the paper [Desmedt, Quisquater, EUROCRYPT '87] it is possible to implement it in the case of many c_j's without any large expense and using a very simple hash (non cryptographic) function.

Model 2: Multiple faults in the secret key

- Working hypothesis:
  The opponent is able to modify at random locations one or several bits of the key k (the new transient key is k*) and to get the correct result of f(m, k*); after the reset of the secure device, the internal secret key is again k. We suppose that the random modifications are equidistributed on all n bits of the key k. The main idea is that the opponent is able with a high probability to change randomly few bits of the secret key.

It is easy to adapt the key search algorithm in that context. The complexity of the attack is directly related to the number of modified keys. The step 3 is also easy: if the key change is too large in relation to the number of flipped bits, it is sometimes necessary to skip the search and to begin a new one.

Model 3: Attacking several secret keys in parallel using several secure devices

It is easy to see that the first secret key will be found by a number of computations equal to the number needed for the two first models divided by the number of secure devices used in parallel. It means a very fast discovery in case of a massive attack.

In the complete paper we will explain
- how to filter efficiently the noise (transient errors with useless output),
- how to combine such an attack with the one by Biham and Shamir,
- how to resist to these attacks without expensive computations by the secure device,
- how this attack is useful to know for public key cryptosystems.

Conclusion
We describe a new use of the attack by transient fault in a secure device: without any new protection and if this attack is feasible it means that a secret key will be obtained by about N / log (N) computations (or less!) instead of N/2 computations by the normal exhaustive keysearch. In that case this attack is really shortening your keys.