Probing Attacks on Tamper-Resistant Devices

Helena Handschuh[1], Pascal Paillier[1], and Jacques Stern[2]

[1] Gemplus/ENST, France
[2] Ecole Normale Supérieure, France

Abstract. This paper describes a new type of attack on tamper-resistant cryptographic hardware. We show that by locally observing the value of a few RAM or adress bus bits (possibly a single one) during the execution of a cryptographic algorithm, typically by the mean of a probe (needle), an attacker could easily recover information on the secret key being used; our attacks apply to public-key cryptosystems such as RSA or El Gamal, as well as to secret-key encryption schemes including DES and RC5.

1 Introduction

In recent years, many researchers have started investigating the security of tamper-resistant devices such as smart-cards. Along many other cryptanalytic attacks on cryptographic algorithms, new attacks have been suggested. These attacks usually assume the existance of some kind of side-channel and retrieve secret information on the process being executed aboard the device [1].

We distinguish between two types of side-channel attacks, namely passive or intrusive attacks. Typical examples of the first kind are timing attacks [10, 7] and power attacks [11], in which the execution time or the power consumption are monitored while secrets are being handled by the device. Other examples are side-channel attacks described by Schneier *et al.* which include (for instance) carry bit analysis [17].

On the other hand, some authors described agressive scenarios which consist in influencing or perturbating the behavior of the device in order to infer the secret. These attacks include Boneh *et al.*'s induction of transient faults during RSA computations [3] or even cutting wires and forcing given bit values, such as in Differential Fault Analysis [2, 14] of DES.

In this paper we consider a new kind of passive attacks, which appears to be more powerful than the previous ones. No statistical analysis is needed in most cases. We suppose that the attacker simply has access to a probe station, which (for the non-specialist) is a kind of needle that allows to monitor the value of a single bit during the execution of some cryptographic algorithm. Such devices are, of course, not self-sufficient. In practice, the attacker must first prepare the surface of the chip in specific way and overcome a long list of technical problems such as line width, protective layers, the lack of information allowing to match a physical chip location with a given gate, security detectors, and other purely physical phenomena such as the needle's electrical bading or pure signal synchronization problems. More, depending on whether it is known to which register the bit belongs, probing may present a wide range of hardness degrees. Part of the analysis consists of guessing which bit is being recorded and once this is done, infering the secret key or private exponent becomes easy.

The paper is organised as follows. The next two sections investigate probing attacks on RSA and DSA-like cryptosystems. Section 4 and 5 focus on applying specific probing-based cryptanalysis on secret key encryption schemes such as DES and RC5.

2 Probing Attacks in Public-key Cryptography

Most public-key cryptosystems require modular exponentiations. Unless specifically adressed by a dedicated hardware design, the modular exponentiation is usually available aboard a cryptographic device as a software implementation. Although many variants exist, most real-life devices implement the well-known Square-and-Multiply algorithm in its nominal version.

This section is intended to introduce a new (probing-based) cryptanalytic attack that completely recovers the exponent of a typical Square-and-Multiply implementation, thus providing a tool for breaking RSA, El Gamal, DSA, Schnorr-type signature schemes, and so forth. We first introduce and comment our adversarial model.

2.1 Our Attack Model

We will denote by SM-1 the standard Square-and-Multiply algorithm that outputs $m^d \bmod n$, given a base m, an exponent d and a modulus n. During the modular exponentiation, d is scanned bit by bit from left to right and modular multiplication or squarings are successively applied to an accumulator A depending on the current bit exponent. Denoting by $|d|$ the bitlength of d, we recall the procedure on Fig. 1.

In the sections that follow, we will be considering an attack scenario in which the adversary is given access to some bits of the accumulator

```
Initialization
        $N \leftarrow n$, $M \leftarrow m$
        $A \leftarrow 1$, $i \leftarrow |d|$

Scanning Loop
        While $(1 \leq i)$
        {
          $A \leftarrow A \cdot A \bmod N$
          If $(d[i] == 1)$,    $A \leftarrow A \cdot M \bmod N$
          $i \leftarrow i - 1$
        }

Output  $A = m^d \bmod n$
```

Fig. 1. Standard Square-and-Multiply Procedure (SM-1)

throughout the exponentiation and attempts to recover information about the exponent d. Before going further, we state that any attack in the above model could be sophisticated or generalized in various ways to support known variants of SM-1, such as right-to-left or multiple bit scanning.

To be more precise, our model assumes that some "monitoring oracle" provides the adversary with the value of certain accumulator bits suppposedly updated *at each execution* of the internal loop of SM-1. This means that bit-values are collected by the attacker just after the accumulator was squared or squared-then-multiplied. Therefore, we implicitely consider that the monitoring oracle is capable of synchronizing perfectly its observations with the actual execution of SM-1 aboard the device.

2.2 Probing Attack of SM-1

In this section, we show how to infer d by probing a single computation of the form $m^d \bmod n$ for given m and n when it is known that the exponentiation is done by SM-1. We will first consider the case when a few bits (possibly a single one) are probed at known positions $J \subset \{1, \cdots, |A|\}$ in the accumulator. We denote by $A(J)$ the set of bits appearing at positions belonging to J in A. Section 2.3 extends the attack to the case when J is unknown.

Let d_i be the integer formed by the i leading bits of d, for $i = 1, \cdots, |d|$. Clearly, after the i-th step of SM-1 was executed, the accumulator contains the value $A^i = m^{d_i} \bmod n$. Meanwhile, the monitoring oracle has provided the attacker with the sequence

$$T_i = \left(A^1(J), A^2(J), \cdots, A^i(J)\right) \ . \tag{1}$$

Now if δ is a guess for d_i, the attacker can easily simulate SM-1 given (m, n, δ) and collect the bits at positions J in the simulated accumulator A', *i.e.* the sequence

$$T'_i(\delta) = \left(A'^1(J), A'^2(J), \cdots, A'^i(J)\right) \ . \tag{2}$$

It is now clear that δ is a correct guess only if $T_i = T'_i(\delta)$. Then, the procedure can be iterated at step $i+1$ by relying on the surviving guesses at step i. This attack strategy is summarized in Fig. 2.

```
$\Delta \leftarrow \{0\}$
For $(i = 1, \cdots, |d|)$
{
    (1)  $\Delta_0 \leftarrow \{2\delta, 2\delta + 1 \mid \delta \in \Delta\}$
    (2)  $\Delta \leftarrow \{\delta \mid \delta \in \Delta_0$ and $T_i = T'_i(\delta)\}$
}

return $\Delta$
```

Fig. 2. The Adversary's Strategy

Since $T_i = T'_i(d_i)$ for all i, it is clear that when the attack ends Δ contains at least $d = d_{|d|}$. The point here resides in detecting whether the number of guesses to be checked is likely to explode or not while carrying out the attack. Although it seems hard to answer this question in the general case, one can still adress it by the mean of a heuristic reasoning. Let us first consider stage (2) during which wrong guesses are eliminated from Δ_0. At step i, each element δ of Δ_0 entering (2) fulfills $T_{i-1} = T'_{i-1}(\delta_{i-1})$ where $\delta_{i-1} = \lfloor \delta/2 \rfloor$. Furthermore, $\delta \in \Delta_0$ passes the test if (and only if) the equality $A^i(J) = A'^i(J)$ holds. This happens with probability :

$$p(\delta) = P[A^i(J) = A'^i(J) \mid A^{i-1}(J) = A'^{i-1}(J)] \ .$$

By heuristically assuming that bits $A'^i(J)$ and $A^i(J)$ are decorrelated[1] for any wrong guess, we get that $p(\delta)$ is nearly $\varepsilon = 2^{-|J|}$. Obviously, the

[1] in theory, these bits are somehow correlated, but the modular multiplication offers such excellent diffusion properties that our assumption remains highly realistic.

correct exponent yields $p(d_i) = 1$. Since stage (1) just doubles the number of elements in Δ, one has, denoting by u_i the average number of surviving guess after step i, that :

$$u_i = \sum_{\delta \in \Delta_0} p(\delta) = p(d_i) + \sum_{\delta \in \Delta_0 \backslash d_i} p(\delta) = 1 + \varepsilon(2u_{i-1} - 1) \ . \qquad (3)$$

Consequently, we get :

$$u_i = \frac{1 - \varepsilon - \varepsilon(2\varepsilon)^i)}{1 - 2\varepsilon} \leq \frac{1 - \varepsilon}{1 - 2\varepsilon} \ ,$$

when $2\varepsilon < 1$ and $u_i = 1 + \frac{i}{2}$ if $\varepsilon = 1/2$. This yields :

$$E(|\Delta|) \leq \frac{1 - \varepsilon}{1 - 2\varepsilon}$$

throughout the attack. If J has a single element, that is if a single bit is observed by the monitoring oracle, then $\varepsilon = \frac{1}{2}$ and

$$E(|\Delta|) \leq 1 + \frac{|d|}{2} \ ,$$

which proves that the attack has a low heuristic complexity.

2.3 Random Hit Attacks

We now address the case when the attacker does not actually know the position of the monitored bits. One could of course address the problem by executing the above strategy with all possible positions simultaneously (there are $|A|^{|J|}$ such possibilities). Instead, we provide a much faster strategy derived from the previously discussed one. The idea exploited here is that the adversary can guess the position of these bits while performing the attack. Again, at step i, δ is a correct guess for d_i if (and only if) T_i coincides with some "simulated trace" that the attacker produces, for instance the complete accumulator history

$$T'_i = \left(A'^1, A'^2, \cdots, A'^i\right) \ .$$

Let us define $T'_i(j; \delta)$ as the sequence $\left(A'^1(j), \cdots, A'^i(j)\right)$. From now on, if $T_i \not\subset T'_i(j; \delta)$ holds for some $j \in \{1, \cdots, |A|\}$ then either δ is a wrong guess for d_i or $j \notin J$. This motivates the attack strategy depicted in Fig. 3.

In this setting, one can show that :

$$E(|\Delta|) \leq \frac{1 - \varepsilon}{1 - 2\varepsilon} + |A|(2\varepsilon)^{|d|} \quad \text{with} \quad \varepsilon = 2^{-|J|} \ , \qquad (4)$$

which means that the attack would succeed again.

```
$J \leftarrow \{1, \cdots, |A|\}$
$\forall j \in J \quad \Delta_j \leftarrow \{0\}$
For ($i = 1, \cdots, |d|$)
{
  For ($j \in J$)
  {
    $\Delta_j \leftarrow \{2\delta, 2\delta + 1 \mid \delta \in \Delta_j\}$
    $\Delta_j \leftarrow \{\delta \mid \delta \in \Delta_j$ and $T_i \subset T_i'(\delta)\}$
  }
  If $\Delta_j = \emptyset$ then $J \leftarrow J - j$
}

return $\Delta = \cup_J \Delta_j$
```

Fig. 3. Random Hit Attack

2.4 Discrete-Log based Signatures

Several Discrete Log (DL)-based cryptosystems have been proposed in the literature. In virtually all of them (El Gamal [5], Schnorr [23], DSA [4]), a fixed known base g has to be raised to a random power k modulo some known prime p. The security of the cryptosystem relies on the randomness and the *secrecy* of the exponent k, which plays in this context a role similar to the one of a secret key.

From the above two attacks, it turns out that probing a single bit during a single signature generation suffices to recover the random exponent k. The device's secret key can thus be easily infered from the knowledge of k and the output signature.

3 Probing Attacks on DES

Following Biham and Shamir's work [2] on Differential Fault Analysis of Secret Key Cryptosystems, as well as Schneier et al.'s ideas on side-channel attacks [17], we take a closer look at probing secret key algorithms, which may be considered as yet another side-channel where information leak. As we saw before, probing is considered as a passive attack, whereas cutting wires would be an agressive attack.

The goal of this section is therefore to show that even a passive attacker may retrieve the secret key of a DES implementation given one single bit of information at each round.

3.1 The Information Leakage Model

The attack we subsequently present works in the following context. Suppose an attacker uses an electronic station to locally observe the value of a given bit during the execution of DES. We require that the attacker have sufficient knowledge of the device to be able to recognize two specific registers which are the R and L data registers on which the round function of an iterated Feistel [6] cipher applies. In the case of DES, any bit of one or the other register is enough to attack the first and the last round subkeys.

3.2 Attacking DES

DES [13] is a 16-round Feistel scheme which can be described as follows :

Let m be a 64-bit message divided into two 32-bit halves : the left half m_L and the right half m_R ; let c be the corresponding ciphertext and k_i the i-th round 48-bit subkey. Finally let IP and IP^{-1} denote the initial and final permutations and F the round function of DES. The algorithm is briefly depicted on Fig. 4.

```
$(L_0|R_0) = IP(m_L|m_R)$
for $i = 1$ to 16 :
$L_i = R_{i-1}$;
$R_i = F(R_{i-1}, K_i) \oplus L_{i-1}$;
end for
$c = IP^{-1}(R_{16}|L_{16})$
```

Fig. 4. Brief Description of DES

In this attack we ignore the initial and final permutations as these are public anyway. Suppose the plaintext is simply $(L_0|R_0)$ and the ciphertext $c = (L_{16}|R_{16})$. We shall now explain how to recover 6 bits of the last round subkey. Assuming that the probe station enables us to record the value of bit number b of register L at each round. The F-function in DES is such that the output bits can be related to a given S-box having a 6-bit input. We refer the reader to [13] for more information on the structure of the F-function. As $L_{16} = R_{15}$, for any ciphertext we can select the 6 bits entering the F-function that produce bit b as an output. These six bits are exored with six bits of the secret subkey k_{16} before entering a specific S-box. Therefore, for each possible value of the 6-bit secret key entering the S-box, the attacker can compute the expected output bit b^* of the F-function. Notice that she also has access to the real value of this bit as she knows bit b from the probe of L_{15} as well as the corresponding bit of the ciphertext R_{16}. Therefore, with one ciphertext, the attacker can eliminate those key guesses where the expected value b^* and the real value of bit b of the output of the round-function are not the same. On average, half of the key candidates will survive. Thus 6 bits of the key are recovered with 6 different ciphertexts.

The same attack can be carried out on six bits of the secret key of the first round. DES happens to be designed such that these 6 bits are different from the 6 bits recovered from the last round subkey. As a matter of fact, the initial permutation on the secret key results in no two consecutive bits entering the same S-box among the first round and the last round subkey. Therefore a total of 12 bits can be recovered by this attack. The remaining 44 bits of the key can be found by exhaustive search.

3.3 Discussion

Note that the attack works on both registers R and L. As a matter of fact, if the attacker probes register R, we can apply exactly the same attack as before because the iterated Feistel structure guarantees that $R_{14} = L_{15}$. Thus we can still compute the input and expected output of the round function and compare the latter to the real value derived from R_{14} as well as from the ciphertext.

This attack uses the same principle as the one described by Biham and Shamir, but does not require the attacker to be able to cut wires or induce faults. In our setting, the prober simply observes the value of a given bit throughout the execution of the block cipher. The complexity is very low : only a handfull plaintext/ciphertext pairs are needed, and the number of offline encryptions is 2^{44}.

Finally, we note that the attack cannot be carried out on two-key triple-DES or triple modes of encryption using DES as a building block. This comes from the fact that only 12 bits (6 from the first encryption component and 6 from the last encryption component) of one of the secret keys can be recovered, therefore the overall exhaustive search on the remaining key bits still amounts to 2^{100} offline encryptions. Additionnally, no output bit of the round-function depends on the input bit at the same bit position due to the bit permutation after the S-box layer. Thus the intermediate values of bit b are of no use to the attacker if she cannot get any other information.

4 Probing RC5

As another example, let us consider RC5. Other iterated algorithms such as RC6 are equally vulnerable to probing. RC5 has been extensively studied from a regular cryptanalysis point of view. See for example [9, 8, 18].

4.1 Description of RC5

RC5 is an iterative secret-key block cipher designed by R. Rivest [15]. It has variable parameters such as the key size, the block size and the number of rounds. A particular (instanced) RC5 algorithm is denoted by RC5-$w/r/b$ where w is the word size (a block is made of two words), r is the number of rounds and b the number of secret key bytes. Our attack works for every choice of these parameters.

RC5 works as follows : the secret key is first extended into a table of $2r+2$ secret w-bit words S_i. We will assume that RC5's key schedule is one-way and focus on recovering the extended secret key table and not the secret key itself. The detailed description of the key schedule can be found in [15]. By letting (L_0,R_0) denote the left and right halves of the plaintext, the encryption algorithm is depicted on Fig. 5.

```
$L_1 = L_0 + S_0$
$R_1 = R_0 + S_1$
for $i = 2$ to $2r+1$ do
$L_i = R_{i-1}$
$R_i = ((L_{i-1} \oplus R_{i-1}) \lll R_{i-1}) + S_i$
```

Fig. 5. Brief Description of RC5

The ciphertext is (L_{2r+1},R_{2r+1}). The transformation performed for a given i value is called a half-round : there are $2r+2$ half rounds. Each half-round involves exactly one sub-key S_i. All additions are mod 2^w and the rotations are mod w. As usual, $\oplus$ denotes a bitwise exclusive or.

4.2 Probing Attack on RC5

For our purposes, we suppose that the attacker once again has access to all intermediate values of some bit b of either register L or register R, which is the case when she can probe one of these two in an iterated hardware implementation of the algorithm or a specific RAM buffer. We start by describing an attack where the adversary probes some register R. Our technique uses some of the ideas presented in [7] to derive the subkeys one by one in reverse order starting with S_{2r+1}.

Step 1.

First, collect a few multiples of w plaintext/ciphertext pairs and sort them by the value of the $\log(w)$ least significant bits of the left half of the ciphertext L_{2r+1}. There should be at least a few texts available in each such 'category'. Then consider the texts which belong to category $(w-b)[w]$ (*i.e.* the value of the $\log(w)$ least significant bits of L_{2r+1} is $(w-b)[w]$). We probe the value of bit b of register $R_{2r-1} = L_{2r}$. Since we know the value of bit b of $R_{2r} = L_{2r+1}$ as well as the value of the last rotation from the left ciphertext half, we can compute the least significant bit of register L_{2r} just before the last subkey addition. Therefore the least significant bit of the last subkey can be found.

Step 2.

Next, consider "category" $(w-b+1)[w]$. After the last rotation, bit b of L_{2r} will be in first position. Applying the same method as in step 1, we can derive the first bit of the last round subkey (taking into account the carry bit created by the addition of the least significant bit, which is by now already known to the attacker and so on for the remaining $(w-2)$ bits of the secret key. They are derived one by one using different ciphertexts, from the low order end to the high order end of the key.

Step 3.

After recovering the last subkey, decrypt one half-round, sort the ciphertexts according to the new value of the $\log(w)$ least significant bits of L_{2r} and derive S_{2r}. Derive all subsequent subkeys up to the very first four.

Step 4. The last four subkeys can be found by cryptanalysing a two-round block cipher, which is straightforward. This concludes successfully the probing attack on RC5.

4.3 Discussion

The attack works in a similar way when we probe register L directly. So the knowledge of a single intermediate bit at each round enables to derive the complete extended secret key and thus to further recover the initial secret key. On the average a few multiples of w known plaintext/ciphertext pairs are needed, in order to be sure that at each round at least one text corresponding to a given rotation value is available. If this should not be the case, the attacker can still query some more pairs of texts while she is working backwards towards the first rounds of the cipher. The complexity of this attack is actually very low and requires less than the exhaustive search of a single 32-bit subkey. Depending on the case, either S_0, S_2 or S_1, S_3 have to be determined otherwise than by the above attack. They can either be recovered by the key schedule, or by guessing a few bits of S_2 or S_3 at a time, and checking for consistency on the corresponding bits of S_0 or S_1.

5 Conclusion

We have shown that probing attacks are a powerful tool to derive information on secret keys in embedded hardware. The interesting feature of these attacks resides in that they are not desctructive, as many previously suggested attacks are. In essence, probing does not require the cutting of wires or inducing faults or even stressing the device to make it behave abnormally, for we just observe (spying) a single bit during execution. We have shown that public key algorithms using exponentiation or the discrete logarithm such as RSA or DSA, as well as secret key algorithms such as DES or RC5 would be vulnerable to such powerful attacks.

Acknowledgements

We are very grateful to David Naccache for motivating this research and would also like to thank the numerous people who contributed to our investigations.

References

1. R. Anderson, M. Kuhn. Low Cost Attacks on Tamper-Resistant Devices. In *Security Protocol Workshop'97, LNCS 1361*, pp. 125-136. Springer-Verlag.1997.

2. E. Biham, A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *Advances in Cryptology - Crypto'97, LNCS 1294*, pages 513-525. Springer-Verlag, 1997.
3. D. Boneh, R. DeMillo and R. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. n *Advances in Cryptology - Eurocrypt'97, LNCS 1233*, pages 37-51. Springer-Verlag, 1997.
4. FIPS PUB 186, February 1, 1993, *Digital Signature Standard.*
5. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *IEEE Transactions on Information Theory*, volume IT–31, no. 4, pages 469–472, July 1985.
6. H. Feistel. Cryptography and computer privacy. In *Scientific american*, 1973.
7. H. Handschuh and H. Heys. A Timing Attack on RC5. In *SAC'98 - Workshop on Selected Areas in Cryptography, LNCS 1556*, pages 306-320. Springer-Verlag, 1999.
8. B. S. Kaliski and Y. L. Yin. On Differential and Linear Cryptanalysis of the RC5 Encryption Algorithm. In *Advances in Cryptology - Crypto'95, LNCS 963*, pages 171-184. Springer-Verlag, 1995.
9. L. R. Knudsen and W. Meier. Improved Differential Attacks on RC5. In *Advances in Cryptology - Crypto'96, LNCS.* Springer-Verlag, 1996.
10. Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology - Crypto'96, LNCS.* Springer-Verlag, 1996.
11. P. Kocher, J. Jaffe, B. Jun. Introduction to Differential Power Analysis and Related Attacks. Available from *http://www.cryptography.com/dpa/technical/.*
12. M. Matsui. Linear cryptanalysis method for DES Cipher. In *Advances in Cryptology - EUROCRYPT'93, LNCS 765.* Springer-Verlag, 1994.
13. U.S. National Bureau of Standards. *Data Encryption Standard*, Federal Information Processing Standard Publication 46-2, 1977.
14. P. Paillier. Evaluating Differential Fault Analysis of Unknown Cryptosystems. In *Public Key Cryptography - PKC'99, LNCS 1560.* Springer-Verlag, 1999.
15. R. L. Rivest. The RC5 Encryption Algorithm. In *Fast Software Encryption - Second International Workshop, Leuven, Belgium, LNCS 1008*, pages 86-96, Springer-Verlag, 1995.
16. R. L. Rivest, A. Shamir, L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystem. In *Communications of the ACM*, vol. 21, 1978.
17. B. Schneier et al. Side-Channel Attacks. To appear In *Cardis'98 - LNCS.* Springer-Verlag, 1998.
18. A. A. Selçuk. New results in linear cryptanalysis of RC5. In *Fast Software Encryption 5 - LNCS 1372.* pages 1-16, Springer-Verlag, 1998. Springer-Verlag, 1998.
19. J. Kilian, P. Rogaway, "How to protect DES against exhaustive key search, *CRYPTO'96, LNCS 1109,* Springer-Verlag, 1996, pp. 252-267.

20. E. Biham & A. Shamir, *The next stage of differential fault analysis : How to break completely unknown cryptosystems*, Preprint, 1996.

21. R. Anderson, *Robustness principles for public-key protocols*, LNCS, Advances in Cryptology, Proceedings of Crypto'95, Springer-Verlag, pp. 236–247, 1995.

22. R. Anderson & S. Vaudenay, *Minding your p's and q's*, LNCS, Advances in Cryptology, Proceedings of Asiacrypt'96, Springer-Verlag, pp. 26–35, 1996.

23. C. Schnorr, *Efficient Identification and Signatures for Smart-Cards*, Advances in Cryptology: Eurocrypt'89 (G. Brassard ed.), LNCS 435, Springer-Verlag, pp. 239-252, 1990.