Strength of Two Data Encryption Standard Implementations under Timing Attacks

Alejandro Hevia
Dept. de Ciencias de la Computación, U. Chile
and
Marcos Kiwi
Dept. de Ingeniería Matemática, U. Chile

We study the vulnerability of two implementations of the Data Encryption Standard (DES) cryptosystem under a timing attack. A timing attack is a method designed to break cryptographic systems that was recently proposed by Paul Kocher. It exploits the engineering aspects involved in the implementation of cryptosystems and might succeed even against cryptosystems that remain impervious to sophisticated cryptanalytic techniques. A timing attack is, essentially, a way of obtaining some user's private information by carefully measuring the time it takes the user to carry out cryptographic operations.

In this work we analyze two implementations of DES. We show that a timing attack yields the Hamming weight of the key used by both DES implementations. Moreover, the attack is computationally inexpensive. We also show that all the design characteristics of the target system, necessary to carry out the timing attack, can be inferred from timing measurements.

Categories and Subject Descriptors: E.3 [**Data**]: Data Encryption—*code breaking, standards*; C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems—*Smart-cards*

General Terms: Security

Additional Key Words and Phrases: Timing attack, data encryption standard, cryptography, cryptanalysis

---

A preliminary version of this paper appeared in the Proceedings of the 3-rd Latin American Symposium on Theoretical Informatics, Campinas, Brazil, pages 192–205, April 1998.
The research of A. Hevia was partially supported by FONDAP in Applied Mathematics 1997, and by FONDECYT No. 1960849.
The research of M. Kiwi was partially supported by FONDECYT No. 1960849, Fundación Andes, and FONDAP in Applied Mathematics 1997.
Name: Alejandro Hevia
Address: Dept. de Cs. de la Computación, Fac. de Cs. Físicas y Matemáticas, Av. Blanco Encalada 2120, Casilla 2777, Santiago, Chile, `ahevia@dcc.uchile.cl`.
Name: Marcos Kiwi
Address: Dept. de Ing. Matemática, Fac. de Cs. Físicas y Matemáticas, Casilla 170, Santiago 3, Chile, `mkiwi@dim.uchile.cl`.

1. INTRODUCTION

An ingenious new type of cryptanalytic attack was introduced by Kocher in [Kocher 1996]. This new attack is called *timing attack*. It exploits the fact that cryptosystems often take slightly different amounts of time on different inputs. Kocher gave several possible explanations for this behavior, among these: branching and conditional statements, RAM cache hits, processor instructions that run in non-fixed time, etc. Kocher's most significant contribution was to show that running time differentials can be exploited in order to find some of a target system's private information. Indeed, in [Kocher 1996] it is shown how to cryptanalyze a simple modular exponentiator. Modular exponentiation is a key operation in Diffie–Hellman's key exchange protocol [Diffie and Hellman 1976] and the RSA cryptosystem [Rivest et al. 1978]. A modular exponentiator is a procedure that on inputs $k, n \in \mathbb{N}$, $n \neq 0$, and $y \in \mathbb{Z}$ computes $(y^k \bmod n)$. In the cryptographic protocols mentioned above n is public and k is private. Kocher reports that if a passive eavesdropper can measure the time it takes a target system to compute $(y^k \bmod n)$ for several inputs y, then he can recover the secret exponent k. Moreover, the overall computational effort involved in the attack is proportional to the amount of work done by the victim. For concreteness sake and clarity of exposition we now describe the essence of Kocher's method for recovering the secret exponent of the fixed–exponent modular exponentiator shown in Fig. 1.

```
INPUT:   y ∈ ℤ
CODE:    z = 1
         Let k_l ··· k_0 be k in binary
         For i = l down to 0 do
              z = z^2 mod n
              If k_i = 1 then z = z · y mod n
OUTPUT:  z.
```

Fig. 1. Modular exponentiator.

The attack allows someone who knows $k_l \cdots k_t$ to recover k_{t-1}. (To obtain the entire exponent the attacker starts with $t = l+1$ and repeats the attack until $t = 1$.) The attacker first computes $l - t + 1$ iterations of the **for** loop. The next iteration requires the first unknown bit k_{t-1}. If the bit is set, then the operation $(z = z \cdot y \bmod n)$ is performed, otherwise it is skipped. Assume that each timing observation corresponds to an observation of a random variable $T = e + \sum_{i=0}^{l} T_{l-i}$ where T_{l-i} is the time required for the multiplication and squaring steps corresponding to the bit k_{l-i} and e is a random variable representing measurement error, loop overhead, etc. An attacker that correctly guesses k_{t-1} may factor out of T the effect of $T_l, \ldots, T_{t-1}$ and obtain an adjusted random variable of known variance (provided the times needed to perform modular multiplications are independent from each other and from the measurement error). Incorrect guesses will produce an adjusted random variable with a higher variance than the one expected. Computing the variance is easy provided the attacker collects enough timing measurements. The correct guess will be identified successfully if its adjusted values have the smaller variance.

In theory, timing attacks can yield some of a target system's private information. In practice, in order to successfully mount a timing attack on a remote cryptosystem a prohibitively large number of timing measurements may be required in order to compensate for the increased uncertainty caused by random network delays. Nevertheless, there are situations where we feel it is realistic to mount a timing attack. We now describe one of them. Challenge–response protocols are used to establish whether two entities involved in communication are indeed genuine entities and can thus be allowed to continue communication with each other. In these protocols one entity challenges the other with a random number on which a predetermined calculation must be performed, often including a secret key. In order to generate the correct result for the computation the other device must posses the correct secret key and therefore can be assumed to be authentic. Many smart cards, in particular dynamic password generators (tokens) and electronic wallet cards, implement challenge–response protocols (e.g. the message authentication code generated according to the ANSI X9.26 [Menezes et al. 1997, page 651] standard). It is expected that extensive use will be made of smart cards based in general purpose programmable integrated circuit chips. Thus, the specific functionality of each smart card will be achieved through programming. The security of these smart cards will be provided using tamper–proof technology and cryptographic techniques. The above described scenario is an ideal setting in which to carry out a timing attack. The widespread availability of a particular type of card will make it easy and inexpensive to determine the timing characteristics of the system on which to mount the attack. Later, the obtaining of precise timing measurements (e.g. by monitoring or altering a card reader or by gaining possession of a card) could be used to retrieve some of the secret information stored in the card by means of a timing attack. Thus, cards that implement challenge–response protocols where master keys are involved could give rise to a security problem. (See [Dhem et al. 1998] for a discussion of a practical implementation of a timing attack against an earlier version of the CASCADE smart card.)

New unanticipated strains of timing attacks might arise. Hence, timing attacks should be given some serious consideration. This work contributes, ultimately, in furthering our understanding of the strengths of this new cryptanalytic technique, the weaknesses it exploits, and the ways of eliminating the possibility of it becoming practical.

Kocher implemented the attack against the Diffie–Hellman key exchange protocol. He also observed that timing attacks could potentially be used against other cryptosystems, in particular against the *Data Encryption Standard* (DES). This claim is the motivation for this work.

2. SUMMARY OF RESULTS AND ORGANIZATION

We study the vulnerability of one of the most widely used cryptosystems in the world, DES, against a timing attack. The starting point of this work is the observation of Kocher [Kocher 1996] that in DES's key schedule generation process moving nonzero 28-bit C and D values using a conditional statement which tests whether a one–bit must be wrapped around could be a source of non–constant encryption running times. Hence, he conjectured that a timing attack against DES

could reveal the Hamming weight of the key.[1] We show that although Kocher's observation is incorrect (for the DES implementations that we analyzed), his conjecture is true. But, we do more.

In Sect. 3 we give a brief description of DES.

In Sect. 4.1 we describe a timing attack against DES that assumes the attacker knows the target system's design characteristics. We first discuss experimental results that show that a computationally inexpensive timing attack against two implementations of DES could yield enough information to recover the Hamming weight of the DES key being used. Hence, assuming the DES keys are randomly chosen, an attacker can recover approximately 3.95 bits of key information. To the best of our knowledge, this is the first implementation of a timing attack against a symmetric cryptosystem. (Since the preliminary version of this work appeared two timing attacks against RC5 have been reported [Handschuh and Heys 1999].) In Sect. 4.1.1 we describe computational experiments that measure the threat implied by an actual implementation of a timing attack against DES.

Recovering 3.95 bits of a DES key is a modest improvement over brute force key search. But, recovering the Hamming weight of the key is, potentially, more threatening. In particular, an adversary can restrict attention to keys determined to have either a significantly low or high Hamming weight. Although such keys may be rare once the adversary determines that one such key is being used the ensuing key search may be significantly sped up. Thus, the adversary can balance the time to find such rare keys with the time needed for key recovery. In some systems, even the recovery of a single (although rare) key may be of serious concern.

In Sect. 4.1.2 we identify the sources of the dependencies between the encryption time and the key's Hamming weight in the implementations of DES that we studied. The most relevant are conditional statements.

In both DES implementations that we analyzed the encryption time T is roughly equal to a linear function of the key's Hamming weight X plus some normally distributed noise e. Since a DES key is a 56 bit long string and keys are chosen uniformly at random in the key space, we have that $X \sim Binom\,(56, 1/2)$.[2] Thus, for some α, β, and σ,

$$T = \alpha X + \beta + e \ , \qquad X \sim Binom\,(56, 1/2) \ , \qquad e \sim Norm\left(0, \sigma^2\right) \ .$$

In Sect. 4.2 we show that it is not necessary, in order to perform a timing attack against DES, to assume that the design characteristics of the target system are known. Indeed, we propose two statistical methods whereby a passive eavesdropper can infer from timing measurements all the target system's design information required to successfully mount a timing attack against DES. To the best of our knowledge, this is the first proof that it is possible to infer a target system's design characteristics through timing measurements.

We would like to stress that all of the timing attacks described in this work only require precise measurements of encryption times but no knowledge of the encrypted plaintexts or produced ciphertexts.

---

[1] Recall that the Hamming weight of a bitstring equals the number of its bits that are nonzero.

[2] Recall that the distribution $Binom\,(N,p)$ corresponds to the distribution of the sum of N independent identically distributed $\{0,1\}$-random variables with expectation p.

In Sect. 5 we propose a "blinding technique" that can be used to eliminate almost all of the execution time differentials in the analyzed DES implementations. This blinding technique makes both DES implementations that we study impervious to the sort of timing attack we describe in this work. Finally, we discuss under which conditions all, and not only the Hamming weight of, a DES key might be recovered through a timing attack.

2.1 Related Work

Modern cryptography advocates the design of cryptosystems based on sound mathematical principles. Thus, many of the cryptosystems designed over the last two decades can be proved to resist many sophisticated, mathematically based, cryptanalytic techniques (provided one is willing to accept some reasonable assumptions). Traditionally, the techniques used to attack such cryptosystems exploit the algorithmic design weaknesses of the cryptosystem. On the other hand, timing attacks take advantage of the decisions made when implementing the cryptosystems (specially those that produce non-fixed running times). But, timing attacks are not the only type of attacks that exploit the engineering aspects involved in the implementation of cryptosystems. Indeed, recently Boneh, Lipton, and DeMillo [Boneh et al. 1997] introduced the concept of *fault tolerant attacks*. These attacks take advantage of (possibly induced) hardware faults. Boneh et al. point out that their attacks show the danger that hardware faults pose to various cryptographic protocols. They conclude that even sophisticated cryptographic schemes sealed inside tamper–resistant devices might leak secret information.

A new strain of fault tolerant attacks, *differential fault analysis* (DFA), was proposed by Biham and Shamir [Biham and Shamir 1997]. Their attack is applicable to almost any secret key cryptosystem proposed so far in the open literature. DFA works under various fault models and uses cryptanalytic techniques to recover the secret information stored in tamper–resistant devices. In particular, Biham and Shamir show that under the same hardware fault model considered by Boneh et al., the full DES key can be extracted from a sealed tamper–resistant DES encryptor by analyzing between 40 and 200 ciphertexts generated from unknown but related plaintexts. Furthermore, in [Biham and Shamir 1997] techniques are developed to identify the keys of completely unknown ciphers sealed in tamper–resistant devices.

The new type of attacks described above have received widespread attention (see for example [English and Hamilton 1996; Markoff 1996]).

3. THE DATA ENCRYPTION STANDARD

DES is the most widely used cryptosystem in the world, specially among financial institutions. It was developed at IBM and adopted as a standard in 1977 [NBS 1977]. It has been reviewed every five years since its adoption.

DES has held up remarkably well against years of cryptanalysis. But, faster and cheaper processors allow, using current technology, to build a reasonably priced special purpose machine that can recover a DES key within hours [Stinson 1995, pp. 82–83]. For concreteness sake, we provide below a brief description of DES. For a detailed description see [NBS 1977]. More easily accessible descriptions of DES can be found in [Schneier 1996; Stinson 1995].

DES is a *symmetric* or *private–key* cryptosystem, i.e., a cryptosystem where the

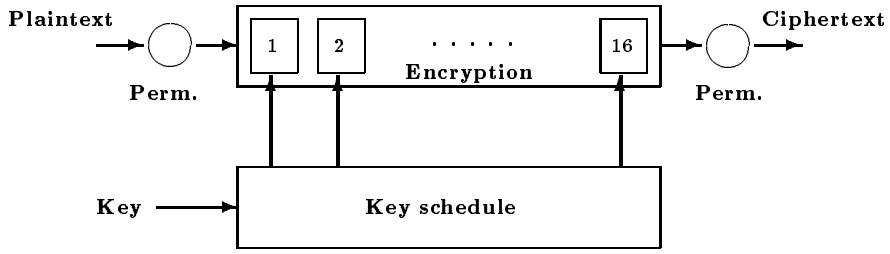

Fig. 2. DES encryption process.

parties that wish to use it must agree in advance on a common secret key which must be kept private. DES encrypts a message (plaintext) bitstring of length 64 using a bitstring key of length 56 and obtains a ciphertext bitstring of length 64. It has three main stages. In the first stage the bits of the plaintext are permuted according to a fixed initial permutation. In the second stage 16 iterations of a certain function are successively applied to the bitstring resulting from the first stage. In the final stage the inverse of the initial permutation is applied to the bitstring obtained in the second stage.

The strength of DES resides on the function that is iterated during the encryption process. We now give a brief description of this iteration process. The input to iteration i is the output bitstring of iteration $i-1$ and a 48 bit long string, K_i. Actually, each K_i is a permuted selection of bits from the DES key. The strings $K_1, \ldots, K_{16}$ comprise what is called the *key schedule*. During each iteration a 64 bit long output string is computed by applying a fixed rule to the two input strings. The encryption process is depicted in Fig. 2.

Decryption is done with the same encryption algorithm but using the key schedule in reverse order $K_{16}, \ldots, K_1$.

The best traditional cryptanalytic attacks known against DES are due to Biham and Shamir [Biham and Shamir 1991; Biham and Shamir 1993] and Matsui [Matsui 1994a; Matsui 1994b]. However, they are not considered a threat to DES in practical environments (see [Menezes et al. 1997, pp. 258–259]).

4. TIMING ATTACK OF DES

We now consider the problem of recovering the Hamming weight of the DES key of a target system by means of a timing attack. We first address the problem, in Sect. 4.1, assuming the attacker knows the design of the target system. We then show, in Sect. 4.2, that this assumption can be removed.

4.1 Timing Characteristics of Two Implementations of DES

We studied the timing characteristics of two implementations of DES. The first one was obtained from the RSAEuro cryptographic toolkit [Kapp 1996], henceforth referred to as RSA–DES. The other implementation of DES that we looked at was one due to Louko [Louko 1992], henceforth referred to as L–DES. We studied both implementations on a 120-MHz Pentium™ computer running MSDOS™. The advantage of working on an MSDOS™ environment is that it is a single process operating system. This facilitates carrying out timing measurements since there are

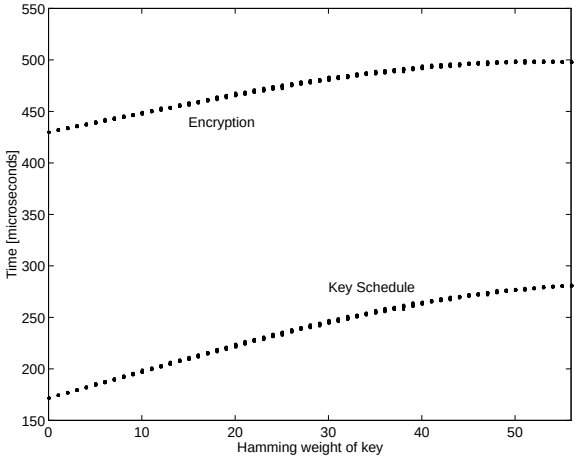

Fig. 3. RSA-DES.

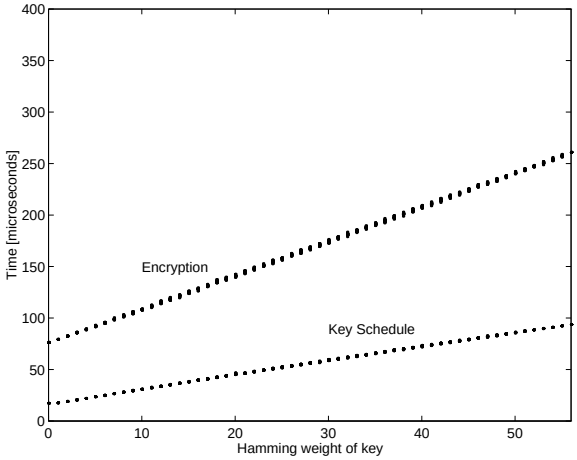

Fig. 4. L-DES.

no other interfering processes running and there are less operating system maintenance tasks being performed. We measured time in microseconds (μs).

In our first experiment we fixed the input message to be the bitstring of length 64 all of whose bits are set to 0. For each $i \in \{0, \ldots, 56\}$ we randomly chose 32 keys of Hamming weight i. For each selected key we encrypted the message a total of 16 times. During each encryption we measured the time it took to generate the key schedule and the total time it took to encrypt the message. The plots, for each of the implementations that we looked at, of the average (for each key) encryption and key schedule generation times are shown in Fig. 3 and Fig. 4.

Only obvious outliers were eliminated. In fact the only outliers that we noticed appeared at fixed intervals of 2^{16} clock ticks. These outliers were caused by system maintenance tasks.

A randomly chosen DES key has a Hamming weight between 23 and 33 with probability approximately 0.86. Thus, the most relevant data points shown in Fig. 3 and Fig. 4 are those close to the middle of the plots.

For various keys chosen at random we performed 2^{16} time measurements (for each key) of the encryption and key schedule generation times. After discarding obvious outliers we graphed the empirical frequency distributions of the collected data. The empirical distributions we observed were roughly symmetric and concentrated in a few contiguous values (usually three or four). This concentration of values is due to the fact that we were only able to perform time measurements with an accuracy of $0.8381\mu s$ and that time differentials among encryptions performed under the same key were rarely larger than $3.0\mu s$. (For an explanation of how to measure time with this precision on an MSDOS™ environment see Appendix A). The above suggests, as one would expect, that the variations on the running time observed when the same process is executed many times over the same input are due to the effect of normally distributed random noise.

For different values of $i \in \{8, \ldots, 48\}$ we randomly chose 2^8 keys of Hamming weight i. After throwing away outliers we graphed the empirical frequency distributions of the collected data. The empirical frequencies observed looked like normal distributions with small deviations (typically $1.2\mu s$ for L–DES and $1.8\mu s$ for RSA–DES). We conclude that the variations on the encryption and key schedule generations times observed among keys of same Hamming weight are mostly due to the *total number* of bits of the key that are set and not by the *position* where these set bits occur. Thus, the effect of *which* bits are set among keys of same Hamming weight is negligible.

We repeated all the experiments described so far but instead of leaving the input message fixed we chose a new randomly selected message at the start of each encryption process. All the results reported above remained (essentially) unchanged. There was only a negligible increase in the measured deviations.

Assuming that the attacker knows the design of the target system, he can build on his own a table of the average encryption time versus the Hamming weight of the key. The clear monotonically increasing relation between the encryption time and the Hamming weight of the key elicited by our experiments is a significant implementation flaw. It allows an attacker to determine the Hamming weight of the DES key. Indeed, the attacker has to obtain a few encryption time measurements and look in the table he has built to determine the key's Hamming weight from which such time measurements could have come. Thus, the attacker can recover $H(\mathrm{wt}(K)) \approx 3.95$ bits of key information (H denotes the binary entropy function).

REMARK 1. *A precise estimation of the Hamming weight of the DES key can be achieved by means of a timing attack if two situations hold. First, accurate time measurements can be obtained. Second, the variations in the encryption and key–schedule generation time produced by different keys with identical Hamming weight is small compared to the time variations produced by keys with one more or one less set bit. We have noticed that the latter situation approximately holds. An exact estimation of Hamming weight of the DES key can be achieved if the attacker can accurately perform time measurements of several encryptions of the same plaintext. But, this requires a more powerful attacker, one that should be capable of fixing the*

INPUT: $M \in \{0,1\}^{64}$, $C \in \{0,1\}^{64}$, $\tau \in \mathbb{R}$.
/* Where τ is the time it takes DES to generate ciphertext C from message M */
CODE: **For** $i = 0$ **up to** 56,
Let l **be such that** $|\{\, j \,:\, |T_j - \tau| < |T_l - \tau| \,\}| = i$.[3]
Let $\mathcal{K}_l = \{\, K \in \{0,1\}^{56} \,:\, \mathsf{wt}(K) = l \,\}$.
Randomly choose m **in** $\{\, 0, \ldots, |\mathcal{K}_l| - 1 \,\}$.
For $j = 0$ **up to** $|\mathcal{K}_l| - 1$,
Let K **be the** $(m + j) \bmod |\mathcal{K}_l|$ **lexicographically first elem. of** $\mathcal{K}_l$.
If (DES encryption of M under key K yields C) **then** return(K).

Fig. 5. Key recovery procedure based on a timing attack that reveals the Hamming weight of the key.

input message fed into the encryption process.

More remarkable than the established monotonically increasing relation between the encryption times and the Hamming weight of the key is the linear dependency that exists between the two measured quantities. The correlation factors for the data shown in Fig. 3 and Fig. 4 are 0.9760 and 0.9999 respectively. The sharp linear dependency between encryption times and Hamming weight allows an attacker to infer the target system's information which is required to carry out the attack described above. This topic is discussed in the next section.

4.1.1 *Experimental Results.* In this section we describe a computational experiment that shows the expected reduction in the size of the key space search that would be achieved by the implementation of the timing attack described in the previous section.

Assume that for every $i \in \{\, 0, \ldots, 56 \,\}$ we have a $T_i \in \mathbb{R}$ corresponding to the expected time it takes the target DES implementation to encrypt a message with a key of Hamming weight i. Furthermore, assume that $T_i < T_{i+1}$ (as supported by our experimental observations). Consider the procedure of Fig. 5 for recovering the DES encryption key through a timing attack that exploits the facts reported in Sect. 4.1. Note that it is possible to experimentally determine the expected number of keys that this procedure would try *without* having to actually execute it. Indeed, if the DES encryption of plaintext M under key K generates the ciphertext C in τ-μs, then the expected size of the key space searched by the given procedure is,

$$\left|\{\, K' \in \{0,1\}^{56} \,:\, |T_{\mathsf{wt}(K')} - \tau| < |T_{\mathsf{wt}(K)} - \tau| \,\}\right| + \frac{1}{2}\left|\{\, K' \in \{0,1\}^{56} \,:\, T_{\mathsf{wt}(K')} = T_{\mathsf{wt}(K)} \,\}\right| .$$

For both DES implementations we studied we randomly chose DES message/key pairs, measured the encryption time, and computed the expected number of keys that the procedure of Fig. 5 would have tried before finding the correct encryption key. From our discussion of Sect. 4.1 it follows that the best that one can hope for is to have to try half of the keys whose Hamming weight equals that of the correct encryption key. This corresponds to 3.24 percent of all the key space, since if $p_k = \binom{56}{k}\frac{1}{2^{56}}$ then $0.0324 \approx 2^{-\sum_{k=0}^{56} p_k \log_2 p_k}/2$. We found that for RSA–DES 5.30

[3] In the (unlikely) event that l is not uniquely defined, perturb τ by a value uniformly chosen in the interval $[-\phi, \phi]$, where ϕ is tiny compared to the precision of the timing measurements.

k	14	16	18	20	22	24	26	28
p_k	0.008	0.058	0.295	1.090	2.973	6.044	9.224	10.615
RSA	0.004	0.131	0.387	1.155	2.221	4.274	6.943	9.865
Louko	0.004	0.029	0.425	0.646	1.654	3.362	5.366	6.472

k	30	32	34	36	38	40	42	44
p_k	9.224	6.044	2.973	1.090	0.295	0.058	0.008	0.001
RSA	8.744	6.054	2.088	1.098	0.459	0.074	0.004	0.001
Louko	5.337	3.768	1.770	0.571	0.172	0.043	0.004	0.001

Table 1. Results of computational experiment.

percent of the key space would have been searched, in average, before finding the correct encryption key. For L–DES, the percentage goes down to 3.84 percent.

Table 1 shows in more detail some of the data collected in our experiments. Columns are labeled according to the weight of the DES key. We denote the weight of a key by k. The second row represents the percentage of the total key space corresponding to DES keys of Hamming weight k (with a precision of 0.0005). We denote this value by p_k. For each DES key of weight k we estimated ($16000 \cdot p_k$ times) the expected percentage of the key space that would have been searched before finding the encryption key. Each of theses estimates was based on $16000 \cdot p_k$ measurements in order to insure that at least 16 measurements were considered for every estimate associated to nonzero p_k's. The last two rows of Table 1 show, for each DES implementation and some key weights, the average of the values obtained.

Recovering 3.95 bits of a DES key gives a modest improvement in the time needed to recover the key. But, Table 1 implies that a timing attack that reveals the Hamming weight of the key is potentially more threatening. In particular, an adversary can restrict attention to keys determined to have either a significantly low or high Hamming weight. The adversary can do this by performing timing measurements until one is found to be either significantly low or high. Once the adversary detects such a rare key the subsequent key search can be much less than the usual amount. Thus, the adversary can balance the time to find such rare keys with the time needed for key recovery. In some systems the recovery of even a single key may cause total disruption and/or forward vulnerability.

4.1.2 *Sources of the dependency between DES encryption time and key's Hamming weight.* The key schedule generation in L–DES is carried out by a procedure called `des_set_key`. This procedure computes the resulting key schedule bitstring by performing a bitwise OR with some pre-computed constants. For each bit of the key, such bitwise or's are computed if and only if the key bit is set. For that purpose, it uses a piece of code of the following form: **If** (*condit*) **then** $instr1; \ldots; instr32;$ **else** *instr*. The number of times *condit* is true turns out to be exactly the Hamming weight of the DES key. This is the main source of running time differentials in L–DES.

In RSA–DES's key schedule generation code there is also a procedure that contains two conditional statements. These conditional statements are used in the computation of the subkeys. More precisely, they implement a fixed permutation

$PC2$ of some bits of the key. Their code is of the following form: **If** (*condit*) **then** *instr*. The total number of times *condit* is true is equal to the sum of the Hamming weight of all subkeys. Thus, the number of times *instr* is executed is directly proportional to the Hamming weight of the DES key.

As mentioned in Sect. 2, Kocher [Kocher 1996] conjectured that in DES's key schedule the rotation of nonzero bits using conditional statements could give rise to running time differentials. In the implementations of DES we analyzed we found no evidence to support this conjecture.

Finally, note that it is clear from Fig. 4 that in L-DES there is a source of non–fixed running times which does not depend on the key schedule generation process. This is evidenced by the non–constant distance between the two curves shown in Fig. 4. The source of these time differentials is not due to conditional statements. We were not able to identify the cause of this dependency nor able to exploit it in order to recover all of the DES key.

4.2 Derivation of the Timing Characteristics of the Target System

As discussed in Sect. 4.1, in both DES implementations that we studied the encryption time was roughly equal to a linear function of the key's Hamming weight plus some normally distributed random noise. In this section we exploit this fact in order to derive all the necessary information needed to perform a timing attack that reveals the Hamming weight of the target system's DES key.

First we need to introduce some notation. Assume we have m measurements on the time it takes the target system to perform a DES encryption. The time measurements might correspond to encryptions performed under different DES keys. For $i \in \{1, \ldots, k\}$, denote by K_i the i-th key that is used by the target system during the period that timing measurements are performed. We make the (realistic) assumption that $K_1, \ldots, K_k$ are chosen at random in $\{0,1\}^{56}$ and independent of each other. Let $X^{(i)}$ denote the Hamming weight of key K_i. Thus, the distribution of $X^{(i)}$ is a $Binom\,(56, 1/2)$. Since we are assuming that the K_i's are chosen independently we have that $X^{(1)}, \ldots, X^{(k)}$ are independent random variables. Note that successive time measurements can correspond to encryptions of the message under the same key. For $i \in \{1, \ldots, k\}$, let $\tau_i \in \{1, \ldots, m\}$ be the index of the last measurement corresponding to an encryption performed with key K_i. For convenience's sake, let $\tau_0 = 0$. Hence, $0 = \tau_0 < \tau_1 < \ldots < \tau_{k-1} < \tau_k = m$. Denote by I_i the set of indices that correspond to time measurements under key K_i, i.e. for $i \in \{1, \ldots, k\}$ let $I_i \stackrel{\text{def}}{=} \{ n \in \mathbb{N} : \tau_{i-1} < n \leq \tau_i \}$. For $i \in \{1, \ldots, k\}$ and $j \in I_i$ let $T_j^{(i)}$ be the random variable representing the time it takes the target system to perform the j-th encryption of the message with key K_i. Finally, for $j \in I_i$ let $e_j^{(i)}$ be a random variable representing the effect of random noise on the j-th encryption with key K_i. Thus, the $e_j^{(i)}$'s represent measurement inaccuracies and the target system's running time fluctuations.

We now have all the notation necessary to formally state the problem we want to address. Indeed, the linear dependency between the encryption time and the Hamming weight of the key in both DES implementations that we studied implies

that there exists α, β, and σ, such that for all $i \in \{1, \ldots, k\}$ and $j \in I_i$

$$T_j^{(i)} = \alpha X^{(i)} + \beta + e_j^{(i)}, \quad X^{(i)} \sim Binom(56, 1/2), \quad e_j^{(i)} \sim Norm(0, \sigma^2) \quad (1)$$

Our problem is to infer from timing measurements the parameters α, β, and σ for which (1) holds. We address two variations of this problem. In Sect. 4.2.1 we show how to deal with the case where the τ_i's are known. In Sect. 4.2.2 we show how to handle the case where the τ_i's are unknown. The former case is the most realistic one. Indeed, a standard cryptanalytic assumption is that the attacker knows the key management procedure of the target system.

4.2.1 *Known $\tau_i's$.* We propose two alternative statistical methods for deducing the parameters α, β, and σ for which (1) holds. One method is based on maximum likelihood estimators and the other one on asymptotically unbiased estimators. Since the following discussion heavily relies on standard concepts and results from probability and statistics we refer the reader unfamiliar with these subjects to [Feller 1966; Ross 1988; Zacks 1971] for background material and terminology.

MAXIMUM LIKELIHOOD ESTIMATORS: Let $X = (X^{(i)})_{i=1}^k$, $T^{(i)} = (T_j^{(i)})_{j \in I_i}$, and $T = (T^{(i)})_{i=1}^k$. Thus, $X, T^{(1)}, \ldots, T^{(k)}$, and T denote random variables. Furthermore, let $x = (x_i)_{i=1}^k$, $t^{(i)} = (t_j^{(i)})_{j \in I_i}$, and $t = (t^{(i)})_{i=1}^k$ be the actual values taken by X, $T^{(i)}$, and T respectively.

Let $f_T(t; \alpha, \beta, \sigma)$ be the marginal distribution of T given α, β, and σ. For a fixed collection of time measurements t the values of α, β, and σ that maximize $f_T(t; \alpha, \beta, \sigma)$ are the maximum likelihood estimators we are looking for. The maximum likelihood estimators are the values most likely to have produced the observed time measurements. They can also be regarded as the values minimizing the loss function $-\log f_T(t; \alpha, \beta, \sigma)$. This explains why maximum likelihood estimators are thought to be good predictors. Thus, in order to determine good estimators for α, β, and σ we first compute $f_T(t; \alpha, \beta, \sigma)$.

PROPOSITION 1. *The marginal distribution of T given α, β, and σ is*

$$f_T(t; \alpha, \beta, \sigma) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{m}{2}} \prod_{i=1}^{k} \mathbb{E}_{X^{(i)}} \left[e^{-\frac{1}{2\sigma^2} \sum_{j \in I_i} (t_j^{(i)} - (\alpha X^{(i)} + \beta))^2} \right] .$$

PROOF. Let $f_{X,T}(\cdot; \alpha, \beta, \sigma)$, $f_{T/X=x}(\cdot; \alpha, \beta, \sigma)$ and $f_X(\cdot; \alpha, \beta, \sigma)$ denote the joint density function of X and T, the density function of T given $X = x$, and the probability distribution of X respectively. For convenience's sake, we henceforth omit α, β, and σ from the expressions for $f_{X,T}$, $f_{T/X=x}$, and f_X.

Observe that the independence of the $X^{(i)}$'s and $e_j^{(i)}$'s imply that the $T^{(i)}$'s are independent. Thus, the joint density function of X and T given α, β, and σ is

$$f_{X,T}(x, t) = f_X(x) \cdot f_{T/X=x}(t) = \prod_{i=1}^{k} f_{X^{(i)}}(x_i) \cdot f_{T^{(i)}/X^{(i)}=x_i}(t^{(i)}) ,$$

where the last equality follows since the $X^{(i)}$'s are independent and the $T^{(i)}$'s are independent.

From (1) we get that $T_j^{(i)}$ given $X^{(i)} = x_i$ distributes like a $Norm\left(\alpha x_i + \beta, \sigma^2\right)$. Moreover, for fixed i, the $T_j^{(i)}$'s are independent random variables. Hence,

$$f_{T^{(i)}/X^{(i)}=x_i}(t^{(i)}) = \prod_{j \in I_i} \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{1}{2}} e^{-\frac{1}{2\sigma^2}(t_j^{(i)} - (\alpha x_i + \beta))^2}$$
$$= \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{|I_i|}{2}} e^{-\frac{1}{2\sigma^2}\sum_{j \in I_i}(t_j^{(i)} - (\alpha x_i + \beta))^2} .$$

Since $X^{(i)} \sim Binom\,(56, 1/2)$, we know that $f_{X^{(i)}}(x_i) = \frac{1}{2^{56}}\binom{56}{x_i}$. Thus,

$$f_{X,T}(x,t) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{m}{2}} \prod_{i=1}^{k} \frac{1}{2^{56}}\binom{56}{x_i} e^{-\frac{1}{2\sigma^2}\sum_{j \in I_i}(t_j^{(i)} - (\alpha x_i + \beta))^2} .$$

The marginal distribution of T given α, β, and σ equals the sum, over all values taken by x, of $f_{X,T}(x,t)$. Hence,

$$f_T(t\,;\,\alpha,\beta,\sigma) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{m}{2}} \prod_{i=1}^{k} \sum_{x_i=0}^{56} \frac{1}{2^{56}}\binom{56}{x_i} e^{-\frac{1}{2\sigma^2}\sum_{j \in I_i}(t_j^{(i)} - (\alpha x_i + \beta))^2} .$$

The conclusion follows directly from the previous equality and the fact that $X^{(i)} \sim Binom\,(56, 1/2)$. □

For a given t the values of α, β, and σ that maximize the right hand side of the expression in Proposition 1 are the maximum likelihood estimators sought. As is often the case when dealing with maximum likelihood estimators it is difficult to solve explicitly for them. (See [Zacks 1971, Ch. 5, §2] for a discussion of computational routines that can be used to calculate maximum likelihood estimators.)

The advantage of the above described approach for determining the parameters relevant for carrying out the timing attack is that it uses all the available timing measurements. But, it does not allow us to determine how many measurements are sufficient in order to obtain accurate estimations of the parameters sought. The alternative approach described below solves this problem.

ASYMPTOTIC ESTIMATORS: Our goal is to find good estimators $\widehat{\alpha}$, $\widehat{\beta}$, and $\widehat{\sigma}$ for α, β, and σ. Moreover, we are interested in determining the asymptotic (on the number of timing measurements) behavior of such estimators. In particular, their asymptotic distributions, their limiting values, and their rate of convergence.

We will now derive good predictors for α, β, and σ. We start with a key observation. Since the expectation and variance of a $Binom\,(56, 1/2)$ are 28 and 14 respectively, taking the expectation and variance in (1) yields that for all $i \in \{1, \ldots, k\}$ and $j \in I_i$

$$\mu_T \stackrel{\text{def}}{=} \mathbb{E}\left[T_j^{(i)}\right] = 28 \cdot \alpha + \beta , \qquad \sigma_T^2 \stackrel{\text{def}}{=} \mathbb{V}\left[T_j^{(i)}\right] = 14 \cdot \alpha^2 + \sigma^2 . \quad (2)$$

Hence, if we knew μ_T, σ_T^2, and σ^2 we could solve for α and β in (2). This suggests that if we can find good estimators for μ_T, σ_T^2, and σ^2, then we can derive good estimators for α and β. We now provide candidates for $\widehat{\mu_T}$, $\widehat{\sigma_T^2}$, and $\widehat{\sigma^2}$, the estimators for μ_T, σ_T^2, and σ^2 respectively. But, we first need to introduce additional

notation. Let

$$\overline{T}^{(i)} \stackrel{\text{def}}{=} (\sum_{j \in I_i} T_j^{(i)})/|I_i| \ , \qquad \overline{T} \stackrel{\text{def}}{=} (\sum_{i=1}^{k} \overline{T}^{(i)})/k \ , \qquad \overline{e}^{(i)} \stackrel{\text{def}}{=} (\sum_{j \in I_i} e_j^{(i)})/|I_i| \ .$$

Define

$$\widehat{\mu_T} \stackrel{\text{def}}{=} \overline{T} \ , \quad \widehat{\sigma_T^2} \stackrel{\text{def}}{=} \frac{1}{k}\sum_{i=1}^{k} \frac{1}{|I_i|} \sum_{j \in I_i} (T_j^{(i)} - \overline{T})^2 \ , \quad \widehat{\sigma^2} \stackrel{\text{def}}{=} \frac{1}{k}\sum_{i=1}^{k} \frac{1}{|I_i|} \sum_{j \in I_i} (T_j^{(i)} - \overline{T}^{(i)})^2 \ .$$

Solving for α and β in (2) yields that the two natural candidates for $\widehat{\alpha}$ and $\widehat{\beta}$, the estimators for α and β, are

$$\widehat{\alpha} \stackrel{\text{def}}{=} \frac{1}{\sqrt{14}} (\widehat{\sigma_T^2} - \widehat{\sigma^2})^{1/2} \ , \qquad \widehat{\beta} \stackrel{\text{def}}{=} \widehat{\mu_T} - 28 \cdot \widehat{\alpha} \ .$$

We now prove that $\widehat{\alpha}$ is well defined.

PROPOSITION 2. $\widehat{\sigma_T^2} - \widehat{\sigma^2} = \frac{1}{k}\sum_{i=1}^{k} (\overline{T}^{(i)} - \overline{T})^2 \geq 0$.

PROOF. Just note that

$$\widehat{\sigma_T^2} = \frac{1}{k}\sum_{i=1}^{k} \frac{1}{|I_i|} \sum_{j \in I_i} (T_j^{(i)} - \overline{T}^{(i)} + \overline{T}^{(i)} - \overline{T})^2 = \widehat{\sigma^2} + \frac{1}{k}\sum_{i=1}^{k} (\overline{T}^{(i)} - \overline{T})^2.$$

□

We henceforth denote a chi–square distribution with l degrees of freedom by χ_l^2.

PROPOSITION 3. *If $|I_1| = \ldots = |I_k| = n$, then the distribution of $\widehat{\sigma_T^2} - \widehat{\sigma^2}$ is (approximately) $\frac{1}{k}(14 \cdot \alpha^2 + \frac{1}{n}\sigma^2)\chi_{k-1}^2$.*

PROOF. Since $T_j^{(i)} = \alpha X^{(i)} + \beta + e_j^{(i)}$, we have that $\overline{T}^{(i)} = \alpha X^{(i)} + \beta + \overline{e}^{(i)}$. Since $\overline{e}^{(i)}$ is the average of n independent $Norm\left(0, \sigma^2\right)$ random variables, $\overline{e}^{(i)} \sim Norm\left(0, \sigma^2/n\right)$. In addition, the de Moivre–Laplace Theorem [Hazewinkel 1988, pp. 397] states that the $Binom\,(m, p)$ distribution can be expressed in terms of the standard normal distribution. Moreover, if $m \to \infty$, then such an expression is exact, and if $mp(1-p) \geq 10$, then the expression provides a good approximation of the Binomial distribution [Ross 1988, pp. 170–171]. Thus, since $X^{(i)} \sim Binom\,(56, 1/2)$, the distribution of $X^{(i)}$ is well approximated by a $Norm\,(28, 14)$. Hence, since $X^{(i)}$ is independent of $\overline{e}^{(i)}$ and the sum of independent normal distributions is a normal distribution, it follows that $\overline{T}^{(i)}$ is approximately distributed as a $Norm\left(\mu_T, 14\alpha^2 + \frac{\sigma^2}{n}\right)$. The desired conclusion follows from a classical statistics result [Hogg and Tanis 1997, Theorem 5.3.4] and Proposition 2. □

PROPOSITION 4. *If $|I_1| = \ldots = |I_k| = n$ and σ^2/n is negligible, then $\sqrt{k}(\widehat{\alpha}^2 - \alpha^2)$ converges (in distribution)[4] to a $Norm\left(0, 3\alpha^4\right)$ plus some small constant error term when $k \to \infty$.*

[4] Recall that when $X_1, X_2, \ldots, X$ are random variables on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$ it is

PROOF. First, note that if we neglect σ^2/n then Proposition 2 and our definition of $\widehat{\alpha}$ imply that the distribution of $\widehat{\alpha}^2 = \frac{1}{14}(\widehat{\sigma_T^2} - \widehat{\sigma^2})$ is (approximately) a $\frac{\alpha^2}{k}\chi^2_{k-1}$. Second, recall that the sum of the squares of l independent identically distributed normal random variables with zero mean and variance equal to 1 is distributed according to a χ^2_l. Equivalently, the sum of l independently distributed χ^2_1 random variables is distributed according to a χ^2_l. Hence, since the expectation and variance of a χ^2_1 random variable are 1 and 3 respectively, the Central Limit Theorem implies that $\sqrt{k-1}(\frac{1}{k-1}\chi^2_{k-1} - 1)$, converges (in distribution) to a $Norm\,(0,3)$.

Putting the two observations together shows that $\sqrt{k-1}(\widehat{\alpha}^2 - \alpha^2)$ converges (in distribution) to a $Norm\left(0, 3\alpha^4\right)$ plus some small constant term when $k \to \infty$. The stated result follows immediately. □

THEOREM 1. *If $|I_1| = \ldots = |I_k| = n$, σ^2/n is negligible and k is sufficiently large, then the distribution of $\widehat{\alpha}$ is (approximately) a $Norm\left(\alpha, \frac{3}{4k}\alpha^2\right)$.*

PROOF. The Law of Large Numbers implies that $\widehat{\sigma_T^2}$ and $\widehat{\sigma^2}$ converge (almost surely)[5] to σ_T^2 and σ^2 respectively. Hence, by continuity, $\widehat{\alpha} = \frac{1}{\sqrt{14}}\left(\widehat{\sigma_T^2} - \widehat{\sigma^2}\right)^{1/2}$ converges (almost surely) to $\alpha = \frac{1}{\sqrt{14}}\left(\sigma_T^2 - \sigma^2\right)^{1/2}$ when $k \to \infty$. This fact and Proposition 4 yield that if $k \to \infty$, then $\sqrt{k}(\widehat{\alpha} - \alpha) = \frac{\sqrt{k}(\widehat{\alpha}^2 - \alpha^2)}{\widehat{\alpha} + \alpha}$ converges (in distribution) to a $Norm\left(0, \frac{3}{4}\alpha^2\right)$ plus some small error term. The desired conclusion follows immediately. □

REMARK 2. *Theorem 1 provides an* approximation *to the distribution of $\widehat{\alpha}$. The approximation error arises from three sources. The first one is the use of the de Moivre–Laplace Theorem to express a $Binom\,(56, 1/2)$ in terms of a $Norm\,(28, 14)$. The second one is due to the use of the Central Limit Theorem to approximate the distribution of an estimator by its limit distribution. The final source of error is due to the use of the Law of Large Numbers to approximate an estimator by its asymptotic value. These three sources of approximation error can be bounded through the de Moivre–Laplace Theorem, Berry–Essen's inequality[Hazewinkel 1988, pp. 369], and Chebyshev's inequality[Ross 1988, pp. 337] respectively. A bound on the accumulated approximation error shows that Theorem 1 is fairly accurate.*

COROLLARY 1. *If $|I_1| = \ldots = |I_k| = n$, σ^2/n is negligible and k is sufficiently large, then*

$$\mathbb{P}\left[\,|\widehat{\alpha} - \alpha| \geq \epsilon|\alpha|\,\right]\,,\ \mathbb{P}\left[\,|\widehat{\beta} - \beta| \geq \epsilon|\beta|\right] \ \leq\ \frac{1}{k\epsilon^2}O(1)\ .$$

PROOF. The bound concerning $\widehat{\alpha}$ follows from Theorem 1 and Chebyshev's inequality[Ross 1988, pp. 337]. In order to prove the other bound recall that $\widehat{\beta} =$

said that X_n converges in distribution to X as $n \to \infty$, if $\mathbb{P}\left[\,X_n \leq x\,\right] \to \mathbb{P}\left[\,X \leq x\,\right]$ as $n \to \infty$ for all points x at which $F_X(x) = \mathbb{P}\left[\,X \leq x\,\right]$ is continuous.

[5] Recall that when $X_1, X_2, \ldots, X$ are random variables on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$ it is said that X_n converges almost surely to X as $n \to \infty$, if $\{\,\omega \in \Omega : X_n(\omega) \to X(\omega),\ \text{as}\ n \to \infty\,\}$ is an event whose probability is 1.

$\widehat{\mu_T} - 28 \cdot \widehat{\alpha}$ and $\beta = \mu_T - 28 \cdot \alpha$, thus

$$\begin{aligned}
\mathbb{P}\left[\, |\widehat{\beta} - \beta| \geq \epsilon|\beta| \,\right] &= \mathbb{P}\left[\, |\widehat{\mu_T} - \mu_T - 28(\widehat{\alpha} - \alpha)| \geq \epsilon|\mu_T - 28 \cdot \alpha| \,\right] \\
&\leq \mathbb{P}\left[\, |\widehat{\mu_T} - \mu_T| \geq \epsilon|\mu_T| \,\right] + \mathbb{P}\left[\, |\widehat{\alpha} - \alpha| \geq \epsilon|\alpha| \,\right] \\
&\leq \frac{1}{\epsilon^2}\left(\frac{1}{\mu_T^2}\mathbb{V}\left[\,\widehat{\mu_T}\,\right] + \frac{1}{\alpha^2}\mathbb{V}\left[\,\widehat{\alpha}\,\right]\right) \quad ,
\end{aligned}$$

where the last inequality is a consequence of applying Chebyshev's inequality twice. Note that from Theorem 1 we have that $\mathbb{V}\left[\,\widehat{\alpha}\,\right] = \frac{3}{4k}\alpha^2$. Moreover, $\mathbb{V}\left[\,\widehat{\mu_T}\,\right] = \frac{1}{k}\mathbb{V}\left[\,\overline{T}^{(i)}\,\right] = \frac{1}{k}(14 \cdot \alpha^2 + \sigma^2/n)$. The result follows. □

Corollary 1 tells us that with probability at least $1 - \delta$, it suffices to take n time measurements for each of $\frac{1}{\delta}O(\frac{1}{\epsilon^2})$ different keys to approximate α and β to within a multiplicative factor of $(1 \pm \epsilon)$.

4.2.2 *Unknown $\tau_i's$.* The assumption that the τ_i's are known made in the previous section is not strictly necessary since an attacker may alternate between performing several timing measurements over a short period of time and resting for an appropriately long period of time. Hence, the problem of deducing the target system's design characteristics reduces to the case in which the τ_i's are known provided that the keys are not changed too often and the attacker's resting period is longer than a key's lifetime. (Changing keys too often creates a key management problem for the cryptosystem's user. Thus, it is reasonable to assume that a key's lifetime is not excessively short.)

We now discuss another approach for handling the case of unknown τ_i's under the assumption that the attacker has access to several identical copies of the target system, e.g., several copies of a smart card supporting a DES based challenge–response protocol. Lets make the reasonable assumption that the target system's keys are independently generated. In this case the attacker may perform, over a short period of time, several timing measurements for each copy of the target system. If the key's are not changed too often the attacker can deduce the target system's relevant timing characteristics as in Sect. 4.2.1. Indeed, the attacker can assume that all the timing measurements arising from the same copy of the system come from encryptions performed under the same key. Since keys corresponding to different copies of the target system are independently generated and the copies of the system are identical, the problem of deducing the target system's design characteristics reduces to the case in which the τ_i's are known.

Tests of statistical hypothesis give rise to another alternative for handling the case of unknown τ_i's. Indeed, consider the situation in which an attacker determines m timing measurements $t_1, \ldots, t_m$ arising from random variables satisfying (1). Assume keys are not changed too often, i.e., at least $n \ll m$ timing measurements come from encryptions performed under the same key. Thus, for each j such that $n \leq j \leq m - n$ the attacker can perform a test of equality of two normal distributions[Hogg and Tanis 1997, pp. 372–385] on the samples of $t_{j-n+1}, \ldots, t_j$ and $t_{j+1}, \ldots, t_{j+n}$. The significance level of such tests allows the attacker to determine the measurements around where a change of key occurs. Discarding the measurements around where the attacker suspects a change of key occurs yields a sequence

of timing measurements from which the target system's design characteristics can be deduced as in the case of known τ_i's.

5. FINAL COMMENTS

In [Kocher 1996] a "blinding technique" similar to that used for blind signatures [Chaum 1983] is proposed in order to prevent a timing attack against a modular exponentiator. For both implementations of DES we studied, blinding techniques can be adapted to produce (almost) fixed running time for the key schedule generation processes. Indeed, let K be the DES key of Hamming weight $\mathsf{wt}(K)$ whose key schedule we want to generate. Let K' be a bitstring of length 56 generated as follows: randomly choose $\lfloor \frac{\mathsf{wt}(K)}{2} \rfloor$ (respectively $\lceil \frac{56-\mathsf{wt}(K)}{2} \rceil$) of the bits of K which are set to 1 (respectively 0) and set the corresponding bits of K' to 0 (respectively 1). Denote the bitwise xor of K and K' by $K \oplus K'$. Note that $\mathsf{wt}(K') = \mathsf{wt}(K \oplus K') = 28$ when the Hamming weight of K is even, and $\mathsf{wt}(K') = 28$ and $\mathsf{wt}(K \oplus K') = 29$ when the Hamming weight of K is odd. Modify the key schedule generation processes so key schedules for keys K' and $K \oplus K'$ are generated. Note that the work required for this is independent of the Hamming weight of K. Hence, no sources of non–fixed running time are introduced during this step. Let $K'_1, \ldots, K'_{16}$ and $K_1, \ldots, K_{16}$ be the key schedules obtained. Recall that K_i (respectively K'_i) is a permuted selection of bits from the key $K \oplus K'$ (respectively K'). Thus, the key schedule of K is $K_1 \oplus K'_1, \ldots, K_{16} \oplus K'_{16}$. Figure 6 plots the encryption times of RSA–DES as previously explained. Note the very clear reduction in time differentials. The reduction is achieved at the expense of increasing the encryption time by a factor of approximately 1.6. Unfortunately, this blinding technique still leaks the parity of the weight of the original DES key, i.e., 1 bit of information. (A careful look at Fig. 6 confirms this fact). This fact can be fixed using the idea developed above. Indeed, for a given DES key K one can generate three DES keys K_1, K_2, and K_3, two of them with Hamming weight 28 and one with Hamming weight 27. Of the three keys one will be spurious meaning that its key schedule should be generated and the results discarded. The xor of the key schedules generated by the two non–spurious keys will give rise to the key schedule sought. When the Hamming weight of the original DES key is even (respectively odd) the spurious key will be the one of Hamming weight 27 (respectively one of the keys of Hamming weight 28).

We have seen that the main source of non–fixed running times were caused by the key schedule generation procedure. In many fast software implementations key setup is an operation which is separated from encryption. This would thwart a timing attack if encryption time is constant. But, in several systems it is impractical to precompute the key schedule. For example, in smart cards pre–computations are undesirable due to memory constraints.

Overall both DES implementations we studied are fairly resistant to a timing attack. This leads us to the question of whether a timing attack can find all of the DES key and not only its Hamming weight. Although we did not succeed in tuning the timing attack technique in order to recover all the bits of a DES key, we identified in L–DES a source of non-fixed running time that is not due to the key generation process. Indeed, the difference in the slopes of the curves plotted in Fig. 4 shows that the encryption time, not counting the key generation process, depends on the key used. This fact is a weakness that could (potentially) be exploited in order to

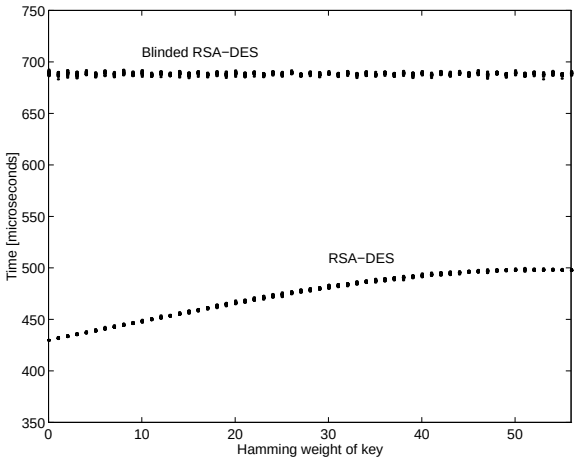

Fig. 6. RSA–DES and modified RSA–DES encryption times.

recover all of the DES key. It opens the possibility that the time it takes to encrypt a message M with a key K is a non–linear function of both M and K, e.g., it is a monotonically increasing function in the Hamming weight of $M \oplus K$. This would allow a timing attack to recover a DES key by carefully choosing the messages to be encrypted. We were not able to identify clear sources of non–linear dependencies between time differentials and the inputs to the DES encryption process in either of the DES implementations that we studied. Nevertheless, we feel that the partial information leaked by both implementations of DES that we analyzed suggests that care must be taken in the implementation of DES, otherwise, all of the key could be compromised through a timing attack.

ACKNOWLEDGMENTS

We are grateful to Shang–Hua Teng for calling to our attention the work of Kocher. We thank Raul Gouet, Luis Mateu, Alejandro Murua, and Jaime San Martin for helpful discussions. We also thank Paul Kocher for advise on how to measure running times accurately on an MSDOS™ environment. Finally, we thank an anonymous referee for pointing out that the blinding technique of Sect. 5 was leaking information about the parity of the Hamming weight of the key.

APPENDIX

A. APPENDIX

Standard C routines allow to measure time events in an MSDOS™ environment with an accuracy of $54.9254\mu s$ [Heidenstrom 1995]. In order to measure with a time precision of $0.8381\mu s$ on a Pentium™ computer running MSDOS™ we followed Kocher's advice [Kocher 1997]. He suggested reading the value of a high–precision timer by accessing port 64. Whenever this timer overflows to 65536 it generates one interrupt. Interrupts occur once every $54925.4\mu s$. Hence, one can measure time intervals with a precision of $54925.4\mu s/65536 \approx 0.8381\mu s$. It is also a good idea to run from a RAM disk. For more information on how to perform accurate time

measurements on the PC family under DOS the reader is referred to [Heidenstrom 1995]. Performing the timing measurements in a Windows or Unix environment is clearly a bad idea since both of them are multi–process operating systems.

REFERENCES

BIHAM, E. AND SHAMIR, A. 1991. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology 4*, 3–72.

BIHAM, E. AND SHAMIR, A. 1993. Differential cryptanalysis of the full 16–round DES. In E. F. BRICKELL Ed., *Advances in Cryptology — CRYPTO'92*, Number 740 in Lecture Notes in Computer Science (Santa Barbara, California, 1993), pp. 494–502. Springer–Verlag.

BIHAM, E. AND SHAMIR, A. 1997. Differential fault analysis of secret key cryptosystems. Technical Report CS0910, Technion, Computer Science Department.

BONEH, D., DEMILLO, R. A., AND LIPTON, R. J. 1997. On the importance of checking cryptographic protocols for faults. In *Advances in Cryptology — EUROCRYPT'97*, Lecture Notes in Computer Science (1997), pp. 37–51. Springer–Verlag.

CHAUM, D. 1983. Blind signatures for untraceable payments. In D. CHAUM, R. L. RIVEST, AND A. T. SHERMAN Eds., *Advances in Cryptology — CRYPTO'82* (Santa Barbara, California, 1983), pp. 199–203. Plenum Press.

DHEM, J.-F., KOEUNE, F., LEROUX, P.-A., MESTRÉ, P., QUISQUATER, J.-J., AND WILLEMS, J.-L. 1998. A practical implementation of the timing attack. In J.-J. QUISQUATER AND B. SCHNEIER Eds., *Proc. CARDIS 1998, Smart Card Research and Advanced Applications*, LNCS (1998). Springer–Verlag.

DIFFIE, W. AND HELLMAN, M. E. 1976. New directions in cryptography. *IEEE Transactions on Information Theory IT–22*, 6 (Nov), 644–654.

ENGLISH, E. AND HAMILTON, S. 1996. Network security under siege. The timing attack. *Computer 30*, 5 (March).

FELLER, W. 1966. *An introduction to probability theory and its applications*, Volume I & II. John Wiley & Sons, Inc. second printing.

HANDSCHUH, H. AND HEYS, H. 1999. A timing attack on RC5. In S. TAVARES AND H. MEIJER Eds., *Proc. SAC'98, Selected Areas in Cryptography*, Number 1556 in LNCS (1999), pp. 306–318. Springer–Verlag.

HAZEWINKEL, M. Ed. 1988. *Encyclopaedia of Mathematics*, Volume 1. Kluwer Academic Press. An updated and annotated translation of the Soviet 'Mathematical Encyclopaedia'.

HEIDENSTROM, K. 1995. FAQ/application notes: Timing on the PC family under DOS. Ver. 19951220, Rel. 3, (`ftp://garbo.uwasa.fi/pc/programming/pctim003.zip`).

HOGG, R. AND TANIS, E. 1997. *Probability and statistical inference* (fifth ed.). Prentice Hall.

KAPP, J. S. A. 1996. RSAEuro: A cryptographic toolkit. Ver. 1.04 Internet Rel. Distrib.

KOCHER, P. 1996. Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems. In N. KOBLITZ Ed., *Advances in Cryptology — CRYPTO'96*, Number 1109 in Lecture Notes in Computer Science (Santa Barbara, California, 1996), pp. 104–113. Springer–Verlag.

KOCHER, P. 1997. Private communication.

LOUKO, A. 1992. DES package. Helsinki University of Technology Computing Centre. Ver. 2.1, (`ftp://kampi.hut.fi`).

MARKOFF, J. 1996. Potential flaw seen in cash card security. New York Times.

MATSUI, M. 1994a. The first experimental cryptanalysis of the data encryption standard. In Y. G. DESMEDT Ed., *Advances in Cryptology — CRYPTO'94*, Number 839 in Lecture Notes in Computer Science (Santa Barbara, California, 1994), pp. 1–11. Springer–Verlag.

MATSUI, M. 1994b. Linear cryptanalysis method for DES cipher. In T. HELLESETH Ed., *Advances in Cryptology — EUROCRYPT'93*, Number 765 in Lecture Notes in Computer Science (Lofthus, Norway, 1994), pp. 386–897. Springer–Verlag.

Menezes, A. J., van Oorschot, P. C., and Vanstone, S. A. 1997. *Handbook of Applied Cryptography* (first ed.). CRC Press.

N. B. of Standards 1977. Data encryption standard (DES). FIPS Publication 46.

Rivest, R. L., Shamir, A., and Adleman, L. 1978. A method for obtaining digital signatures and public–key cryptosystems. *Communications of the ACM 21*, 120–126.

Ross, S. 1988. *A first course in probability* (third ed.). Macmillan Publishing Company.

Schneier, B. 1996. *Applied Cryptography: protocols, algorithms and source code in C* (second ed.). John Wiley & Sons, Inc.

Stinson, D. R. 1995. *Cryptography, Theory and Practice* (first ed.). CRC Press.

Zacks, S. 1971. *The theory of statistical inference.* John Wiley & Sons, Inc.