THE IMPORTANCE OF "GOOD" KEY SCHEDULING SCHEMES (HOW TO MAKE A SECURE DES* SCHEME WITH $\leq$ 48 BIT KEYS?)

Jean-Jacques Quisquater [a], Yvo Desmedt [b, 1] and Marc Davio [a, c]

[a] Philips Research Laboratory Brussels,
Avenue Van Becelaere, 2, B-1170 Brussels, Belgium;

[b] Katholieke Universiteit Leuven, Laboratorium ESAT,
Kardinaal Mercierlaan, 94, B-3030 Heverlee, Belgium;

[c] Université Catholique de Louvain, Bâtiment Maxwell,
Place du Levant, 3, B-1348 Louvain-la-Neuve, Belgium.

Abstract

In DES the key scheduling scheme uses mainly shift registers. By modifying this key scheduling, conventional cryptosystems can be designed which are, e.g., strong against exhaustive key search attacks (without increasing the key size), or have public key like properties. Other effects obtainable by modifying the key scheduling and their importance are discussed.

1. Introduction

In this paper we come up with several ideas which are in contradiction with the common points of view in cryptography. So in the first idea (see Section 2) we will propose to *reduce the key size of a cryptosystem to increase its security against exhaustive key search machines*. This idea sounds crazy, but can be realized for some cryptographic encryption algorithms (*e.g.* DES) if some very small modifications are used. In the second idea we will come up with a conventional cryptosystem which has public key like properties (see Section 3). In Section 4, we will give examples of *conventional* cryptosystems for which outsiders can prove the *existence* of a trapdoor in the scheme but they cannot use this information to find the trapdoor.

All previous ideas are realized by using new key scheduling schemes.

2. Enforcing cryptosystems against a key exhaustive search

DES [3] was criticized because the length of the key is only 56 bits. Several exhaustive key search machines were presented to break several modes of DES [5], [6], [7], [8]. and [9] Diffie and Hellman [7] proposed to use a larger key size to avoid exhaustive key search

[1] NFWO aangesteld navorser, sponsored by the National Science Foundation of Belgium.

H.C. Williams (Ed.): Advances in Cryptology - CRYPTO '85, LNCS 218, pp. 537-542, 1986.

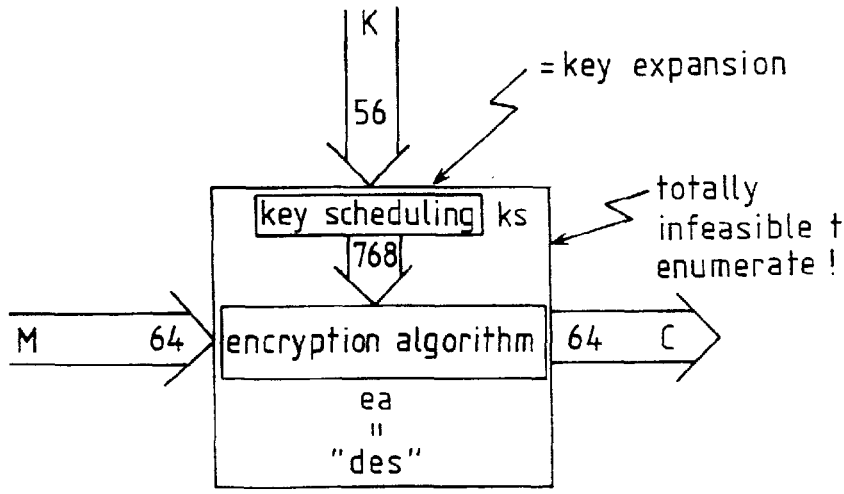

Figure 1: A schematic overview of DES.

attacks. Yasaki [14] cites Hellman: "But the point we're making here is that a small key guarantees insecurity". In this section we will explain that in theory an algorithm which uses a short key, can be made very strong against exhaustive key search machines. To realize this in practice some restraints are to be taken into consideration.

To explain the basic idea let us consider DES as an example. The time needed by an exhaustive key search machine to find the key is in worst cases proportional to:

$$\frac{\max(t_{ks}, t_{ea}) \times (\text{size of the key space})}{(\text{number of used processors})}$$

where, for DES, the size of the key space is 2^{56}. As mentioned in Fig. 1: the time t_{ks}, respectively t_{ea}, is the time for executing the key scheduling, respectively the encryption algorithm without the key scheduling. Remark that for many hardware modules DES we have $t_{ea} \gg t_{ks}$. To avoid exhaustive key search attacks, several ideas can be used. The most known is to increase the key space (*i.e.* key size). Another is to slow down the algorithm (*e.g.* using more rounds in DES). This solution however reduces the practical use of the algorithm. The solution we present here, is to increase t_{ks}, such that $t_{ks} \gg t_{ea}$. In Fig. 2, a DES–like algorithm is presented which is 16 times harder to break than DES with an exhaustive key search machine! The used key K is 56 bits long and α is some fixed public known 64 bit pattern. Remark that if *the key is held constant this DES–like algorithm has the same encryption and decryption speed as DES*! Hereto the so called "subkeys" are stored, and were calculated beforehand. Essential for its security is that the DES algorithms in the new key scheduling of the DES–like algorithm (see Fig. 2) are chained. It is evidently necessary that no trapdoor in DES would allow to shortcut this chaining. From now on we will assume that DES doesn't have the discussed trapdoor.

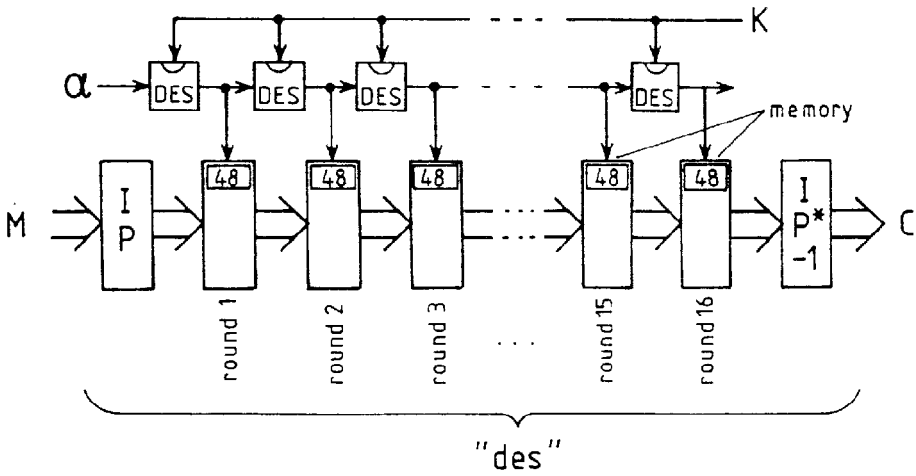

Figure 2: A DES* which is 16 times harder to break with an exhaustive key search machine than DES.

It is now trivial to propose other DES-like algorithms which use a similar key scheduling. *E.g.* one could use 16 times DES for the calculation of the next subkey, instead of one. This would slow down the key scheduling (and the exhaustive key search) with a total factor of 256. Evidently if last discussed scheme is used with a key K of only 48 bits (the other 8 bits can be all zero), the security remains the same as for DES, from the point of view of exhaustive key search attacks. In theory one can increase the key scheduling time as much as we want, however in practice the users modify the key. Then the frequency of the modification of the key determines what an acceptable increase of the execution time of the key scheduling is. However one cannot reduce the key size too far. Indeed if the key is too short, one can precalculate *ones and for all* the subkeys for all possible keys and store them. In this case exhaustive key search machines use the precalculated subkeys instead of calculating the key scheduling. A similar improved technique is valid for a chosen text attack. Remark that the key scheduling scheme is in fact a key expansion, which is an important concept in modern cryptosystems [11].

Several other schemes can be used as key expansion instead of DES. *E.g.* one can use a modified RSA [13]. The key K of 56 bits is enlarged with zero's to an input for the RSA algorithm of 768 bits, e and n as in RSA are public, however here n is a prime number of 769 bits. The 768 bit output of this RSA scheme is used as the 16 subkeys of 48 bits.

The analysis done on more rounds in DES by the authors [4], is no longer valid for the DES-like algorithm, as a consequence of the harder cryptosystems. In other words the ideas presented here can also be used to make a cryptosystem harder against other attacks than the exhaustive key search.

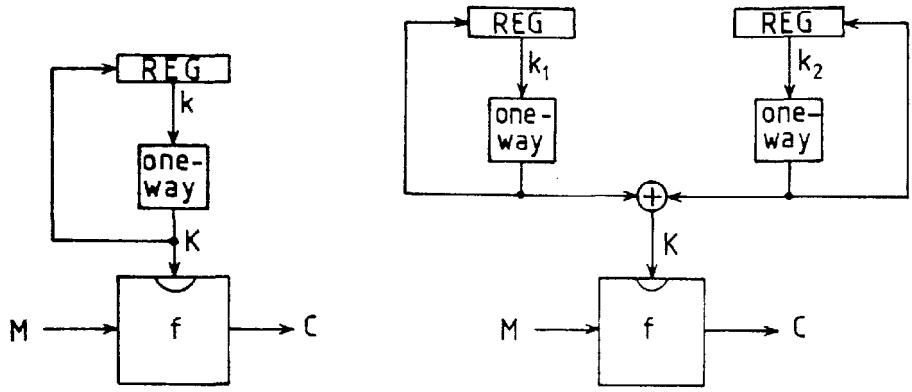

Figure 3: Using a one–way key scheduling scheme in feedback.

3. Using feedback and one–way functions

At the end of the key scheduling scheme in the original DES, the original key reappears in the shift registers C and D. This can be considered as some feedback. Indeed in DES no extra register is necessary to store the used key, if the output of the key scheduling scheme is feedbacked. In the schemes we proposed in this paper, this extra register is necessary, in order to continue to use the same key. This is however not necessary! We can use a feedback at the end of the key scheduling. If sender and receiver remain synchronized and no transmitted bits are lost, the used keys are modified, but both use the correct key. Let us now discuss two cases a little more. In the first one, the key scheduling scheme itself uses a secret master key to calculate the actual "session" key. Remark that the new session key has not to be transmitted! In the second case no extra master key is used. The key scheduling scheme uses only public known information (as in DES). In this case no real advantage seems to be obtained with this feedback. We will now explain that this impression is wrong.

Suppose last discussed feedback key scheduling scheme is used. What happens if the used function ks (see Fig. 3, a) in the feedback is one–way? In that case the conventional cryptosystem acts partially as a public key system. We will explain this by an example. Suppose that a cryptosystem is located in some physical unsafe area. The security of the key is actually tamper free, but this can change at any time (e.g. a bank located in an unstable political regime). One wants to design a cryptosystem, such that if the key of the sender is stolen, the cryptanalyst cannot understand the sent messages. A first possibility is to use a public key system. Indeed the public key (of the receiver), which is used to protect the privacy of the messages, cannot be used to decipher. The second possibility is to use the one–way key scheduling scheme with feedback. Even if the key is stolen, it cannot help to decipher previous messages! Similar examples can easily be found for other areas, in space applications for instance. Applications of the same idea can be used to protect keys in non–tamper free areas, e.g. in chip cards. Indeed chip cards are more

secure than magnetic cards, nevertheless not necessary completely tamper free. Related to this example, a similar scheme was presented by Beker [1].

Several schemes can be used for this one–way function. DES is a good candidate for that. Some caution is necessary because only the so–called leader part of the feedback scheme is useful [10]: The cyclic part of the scheme is not very secure.

Another idea is given at Fig. 3, b. It has the same properties as the one we discussed for the scheme of Fig. 3, a. Additionally the future is protected even if the key K is found by any practical method different from physically stealing the keys k_1 and k_2.

Intead of one–way functions one can also use hard pseudo–random functions in the sense of Blum and Micali [2].

4. Trapdoors in key scheduling schemes

As we yet discussed in Section 2 trapdoors in the key scheduling are possible. To obtain the improvements, chaining DES must be free of a shortcut solution. In this section more trapdoors in key scheduling schemes will be discussed.

We discussed at the end of Section 2 the use of a modified RSA scheme for key scheduling. We used however there a prime number n, instead of the product of two primes. Suppose however that the user of the cryptosystem verifies if n is indeed prime, and suppose it isn't. He knows for sure that it can be that the one who designed the cryptosystem has deliberately chosen n as a product of primes kept secret by the designer. Using the Chinese remainder theorem this allows the designer to speed up RSA [12] and so the key scheduling and his exhaustive key search machine. Remark that *the user of the cryptosystem can indeed verify the possibility of a trapdoor but cannot use this knowledge*! Evidently if n is large enough, it can be the product of several primes, giving more advantage at the designer. We propose to use the expression "*trapdoor algorithms*" for this kind of algorithms, *i.e.* for the algorithms where the computation complexity depends on the knowledge of some information.

A trapdoor can also be build in the feedback one–way key scheduling system. Indeed instead of using a one–way function, a trapdoor one–way function can be used. This allow the designer to reverse the feedback and decrypt previous messages, while this is impossible (hard) for outsiders. Remark that the cryptosystem remains a conventional cryptosystem!

Such trapdoors (which are useless for outsiders if they only know the existence and location of the trapdoor) can be used, *e.g.* to *reduce the misuse* of an authentication system *after* it has been tampered.

5. Conclusions

In contradiction with the common ideas *the key length is* not only *the thing* to protect against exhaustive key search machines. Cryptosystems were proposed acting partially similar as public key systems. Combining ideas of public key and conventional schemes, we proposed trapdoors in conventional systems. The trapdoors are detectable, but useless for outsiders.

A good key scheduling scheme is important. Very hard cryptosystems can be build, starting from simple ones, iterating them and using a hard key expansion (scheduling) scheme.

REFERENCES

[1] H. Beker and M. Walker, "Key management for secure electronic funds transfer in a retail environment," *Advances in Cryptology*, Proceedings of Crypto '84, Santa Barbara, August 1984 (Lecture Notes in Computer Science, Springer–Verlag, Berlin, 1985), pp. 401 – 410.

[2] M. Blum and S. Micali "How to generate cryptographically strong sequences of pseudo–random bits," *SIAM J. Comput.*, Vol. 13, No. 4, Nov. 1984, pp. 850 –864.

[3] "Data Encryption Standard," *FIPS* (NBS Federal Information Processing Standards Publ.), no. 46, January 1977.

[4] Y. Desmedt, J.–J. Quisquater and M. Davio, "Dependence of output on input in DES: Small avalanche characteristics," *Advances in Cryptology*, Proceedings of Crypto '84, Santa Barbara, August 1984 (Lecture Notes in Computer Science, Springer–Verlag, Berlin, 1985), pp. 359 – 376.

[5] Y. Desmedt, "Unconditionally secure authentication schemes and practical and theoretical consequences," presented at Crypto '85, Santa Barbara, August, 1985, to appear in the proceedings: *Advances in Cryptology* (Lecture Notes in Computer Science, Springer–Verlag, Berlin, 1986).

[6] Y. Desmedt, F. Hoornaert and J.–J. Quisquater, *paper in preparation.*

[7] W. Diffie and M. E. Hellman, "Exhaustive cryptanalysis of the NBS Data Encryption Standard," *Computer*, vol. 10, no. 6, pp. 74 – 84. June 1977.

[8] M. E. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Trans. Inform. Theory*, vol. 26, no. 4, pp. 401 – 406, July 1980.

[9] F. Hoornaert, J. Goubert, and Y. Desmedt, "Efficient hardware implementations of the DES," *Advances in Cryptology*, Proceedings of Crypto '84, Santa Barbara, August 1984 (Lecture Notes in Computer Science. Springer–Verlag, Berlin, 1985), pp. 147 – 173.

[10] B. S. Kaliski, R. L. Rivest and A. T. Sherman "Is DES a pure cipher? (Results of more cycling experiments on DES)," presented at Crypto '85, Santa Barbara, August, 1985, to appear in the proceedings: *Advances in Cryptology*, (Lecture Notes in Computer Science. Springer–Verlag, Berlin. 1986).

[11] A. Konheim, "*Cryptography: A Primer*," John Wiley. Toronto, 1981.

[12] J.–J. Quisquater and C. Couvreur, "Fast decipherment for RSA public-key cryptosystem," *Electronics Letters*, vol. 18, 14th October 1982, pp. 905 – 907.

[13] R. L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21, pp. 294 – 299, April 1978.

[14] E. K. Yasaki, "Encryption algorithm : key size is the thing," *Datamation*, vol. 22, no. 5, pp. 164 – 166. March 1976.