BRICS

Basic Research in Computer Science

BRICS RS-94-38 Damgård & Knudsen: Enhancing the Strength of Conventional Cryptosystems

Enhancing the Strength of Conventional Cryptosystems

Ivan B. Damgård
Lars Ramkilde Knudsen

BRICS Report Series **RS-94-38**

ISSN 0909-0878 **November 1994**

**See back inner page for a list of recent publications in the BRICS
Report Series. Copies may be obtained by contacting:**

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK - 8000 Aarhus C
Denmark**

**Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@daimi.aau.dk**

Enhancing the Strength of Conventional Cryptosystems

Ivan B. Damgård (BRICS*, Aarhus University)
Lars Ramkilde Knudsen (Aarhus University)

September 7, 1994

Abstract

We look at various ways of enhancing the strength of conventional cryptosystems such as DES by building a new system which has longer keys and which uses the original system as a building block. We propose a new variant of two-key triple encryption which is not vulnerable to the meet in the middle attack by van Oorschot and Wiener. Under an appropriate assumption on the security of DES, we can prove that our system is at least as hard to break as single DES.

1 Introduction

Since its introduction in the late seventies, the American Data Encryption Standard (DES) has been the subject of intense debate and cryptanalysis. Like any other practical cryptosystem, DES can be broken by searching exhaustively for the key.

One natural direction of research is therefore to find attacks that will be faster than exhaustive search, measured in the number of necessary encryption operations. The most successful attack known of this kind is the linear attack by Matsui [2, 3]. This attack requires about 2^{43} known plaintext blocks. Although this is less than the expected 2^{55} encryptions required

*Basic Research in Computer Science, Centre of the Danish National Research Foundation for exhaustive key search, the attack is by no means more practical than exhaustive search. There are two reasons for this: first, one cannot in practice neglect the time needed to obtain the information about the plaintext; secondly, when doing exhaustive key search the enemy is free to invest as much in technology as he is capable of to make the search more efficient, in a known plaintext attack he is basically restricted to the technology of the legitimate owner of the key, and to the frequency with which the key is used. In virtually any practical application, a single DES key will be applied to much less than 2^{43} blocks, even in its entire life time. The difference between the two kinds of attacks is illustrated in a dramatic way by the results of Wiener [8] who shows by concrete design of a key search machine that if the enemy is willing to make a one million dollar investment, exhaustive key search for DES is certainly not infeasible.

As a result, we have a situation where DES has proved very resistant over a long period to cryptanalysis and therefore seems to be as secure as it can be in the sense that by far the most practical attack is a simple brute force search for the key. The only problem is that the key is too short given today's technology, and that therefore, depending on the value of the data you are protecting, plain DES may not be considered secure enough anymore.

What can be done about this problem? One obvious solution is to try to design a completely new algorithm. This can only be a very long term solution: a new algorithm has to be analysed over a long period before it can be considered secure; also the vast number of people who have invested in DES technology will not like the idea of their investments becoming worthless overnight. An alternative is to devise a new system with a longer key using DES as a building block. This way existing DES implementations can still be used.

We are in the situation, where we have a block cipher, that has proved to be very strong, the only problem being that the keys are too small and a simple brute-force attack has become possible. Thus, this section is motivated by the following general question: Given cryptosystem $\mathcal{X}$, which cannot in practice be broken faster than exhaustive key search, how can we build a new system $\mathcal{Y}$, such that

1. Keys in $\mathcal{Y}$ are significantly longer than keys in $\mathcal{X}$ (e.g. twice as long)

2. Given an appropriate assumption about the security of $\mathcal{X}$, $\mathcal{Y}$ is provably as hard to break as $\mathcal{X}$ under any natural attack (e.g. ciphertext only,

known plaintext, etc.).

3. It can be convincingly argued that $\mathcal{Y}$ can in fact not be broken faster than exhaustive key search, and is therefore in fact much stronger than $\mathcal{X}$.

Possible answers to this question have already appeared in the literature. The most well known example is known as two-key triple encryption, where we encipher under one key, decipher under a second key, and finally encipher under the first key. Van Oorschot and Wiener [7] have shown, refining an attack of Merkle and Hellman [5], that this construction is not optimal: under a known plaintext attack, it can be broken significantly faster than exhaustive key search.

We propose a new variant of two-key triple encryption, which has all the properties we require above.

2 Multiple encryption

In this section, we look at methods for enhancing cryptosystems based on the idea of encrypting plaintext blocks more than once. Following the notation of the introduction, we let $\mathcal{X}$ be the original system, and we let E_K resp. D_K denote encryption resp. decryption in $\mathcal{X}$ under key K. We assume that the key space of $\mathcal{X}$ consists of all k-bit strings, and that the block length of $\mathcal{X}$ is m. In a *cascade of ciphers* it is assumed that the keys of the component ciphers are independent. The following result was proved by Maurer and Massey.

Theorem 2.1 (The importance of being first [4].) *A cascade of ciphers is at least as hard to break as the first cipher.*

By restricting ourselves to the most powerful attack, the chosen plaintext attack, we can prove the following more general result.

Theorem 2.2 (The importance of being there.) *A cascade of ciphers is at least as hard to break under any attack as any of the component ciphers in the cascade under a chosen plaintext attack.*

Proof: Assume that we have an algorithm A, which on input the encryptions of n known or chosen plaintexts or on input just n ciphertexts, breaks a cascade of N_c ciphers, $\mathcal{Y}$. We will use A to break any of the component ciphers in a chosen plaintext attack. Assume that $\mathcal{X}$ is the i'th cipher of the N_c ciphers in the cascade and that we can get encryptions of any chosen plaintext. Choose $N_c - 1$ keys at random for the ciphers exclusive $\mathcal{X}$. Whenever A asks for the encryption of a chosen or known plaintext P, we multiple encrypt P using the first $i-1$ keys, yielding PP. In a ciphertext only attack we choose a plaintext P ourselves. Then we get the encryption CC of PP in the chosen plaintext setting from $\mathcal{X}$. Now use the remaining $N_c - i$ keys to multiple encrypt CC, yielding C, which we input to A. Since by assumption, A breaks the cascade, it will output the N_c keys, amongst which we will get a candidate for the secret key of $\mathcal{X}$. We have proved that if we can break the cascade, we can break any of the component ciphers under a chosen plaintext attack. Thus, if a component cipher $\mathcal{X}$ is secure against a chosen plaintext attack, then a cascade of ciphers containing $\mathcal{X}$ is as secure against any attack. $\square$

A special case of a cascade of ciphers is when the component ciphers are equal, also called multiple encryption. In the following we consider different forms of multiple encryption.

2.1 Double Encryption

The simplest idea one could think of would be to encrypt twice using two keys K_1, K_2, i.e. let the ciphertext corresponding to P be $C = E_{K_2}(E_{K_1}(P))$. It is clear (and well-known), however, that no matter how K_1, K_2 are generated, there is a simple meet-in-the middle attack that breaks this system under a known plaintext attack using 2^k encryptions and 2^k blocks of memory, i.e. the same time complexity as key search in the original system. Even though the memory requirements may be unrealistic, it is clear that this is not a satisfactory improvement over $\mathcal{X}$.

2.2 Triple Encryption

Triple encryption with three independent keys K_1, K_2, and K_3, where the ciphertext corresponding to P is $C = E_{K_3}(E_{K_2}(E_{K_1}(P)))$, is also not a satisfactory solution for a similar reason as for double encryption. A simple meet-in-the-middle attack will break this in time about 2^{2k} encryptions and space 2^k blocks of memory. Thus we do not get full return for our effort in tripling the key length - as stated in demand 3 in the introduction, we would like attacks to take time close to 2^{3k}, if the key length is $3k$. In addition to this, if $\mathcal{X} = DES$, then a simple triple encryption would preserve the complementation property, and preserve the existence of weak keys.

It is clear, however, that no matter how the three keys in triple encryption are generated, the meet-in-the-middle attack mentioned is still possible, and so the time complexity of the best attack against *any* triple encryption variant is no larger than 2^{2k}. It therefore seems reasonable to try to generate the three keys from two independent $\mathcal{X}$-keys K_1, K_2, since triple encryption will not provide security equivalent to more than 2 keys anyway.

2.3 Two-key Triple Encryption

One variant of this idea is well-known as two-key triple encryption, proposed by W. Tuchmann [6]: we let the ciphertext corresponding to P be $E_{K_1}(D_{K_2}(E_{K_1}(P)))$. Compatibility with a single encryption can be obtained by setting $K_1 = K_2$. As one can see, this uses a particular, very simple way of generating the three keys from K_1, K_2. For two-key triple encryption there is a result similar to Theorem 2.2.

Theorem 2.3 *Under a chosen plaintext/ciphertext attack two-key triple encryption is at least as hard to break as the underlying cipher.*

Proof: Assume that we have an algorithm B, which on input n chosen plaintexts, breaks a two-key triple encryption scheme, $\mathcal{Z}$, where $\mathcal{W}$ is the underlying cipher. Choose one key $K_{1,3}$ at random. Whenever B asks for the encryption of plaintext P, we encrypt P using the key $K_{1,3}$, yielding PP. Then we get the decryption CC of PP in the chosen ciphertext setting from $\mathcal{W}$. Now encrypt CC using again the key $K_{1,3}$ yielding C, which is input to B. Since by assumption B breaks the two-key triple scheme, it will output a candidate for the key in the second round, i.e. for the secret key of $\mathcal{W}$. $\square$

Even though this result establishes some connection between the security of two-key triple encryption with the underlying cipher, it holds only for a chosen plaintext/ciphertext attack and still does not meet our second demand.

For the two-key triple encryption scheme, each of K_1 and K_2 only influences particular parts of the encryption process. Because of this, variants of the meet-in-the-middle attack are possible that are even faster than exhaustive search for K_1, K_2. In [5] Merkle and Hellman describes an attack on two-key triple DES encryption requiring 2^{56} chosen plaintext-ciphertext pairs and a running time of 2^{56} encryptions using 2^{56} words of memory. This attack was refined in [7] into a known plaintext attack on the DES, which on input n plaintext-ciphertext pairs finds the secret key in time $2^{120}/n$ using n words of memory. The attacks can be applied to any block cipher.

Since the attacks exploit that the keys used in the first and third encryption are equal, an initial attempt to thwart the attacks could be to let the third key be dependent on both the first and second key. Define encryption by $E_{K_1}(D_{K_2}(E_{K_3}(P)))$, where $K_3 = E_{K_1}(K_2) \oplus K_2$. Compatibility with a single encryption can still be obtained by setting $K_2 = D_{K_1}(0)$, in that way $K_2 = K_3$. By the security of the DM-scheme, the Davies-Meyer hashing scheme, knowing K_1 (or K_2) does not give immediate knowledge about K_3 and the scheme seems invulnerable to the attacks by Merkle and Hellman. However, we found no proof that this scheme is at least as secure as a single encryption.

We therefore propose what we believe to be stronger methods for generating the keys. Our main idea is to generate them *pseudorandomly* from 2 $\mathcal{X}$ keys using a generator based on the security of $\mathcal{X}$. In this way, an enemy trying to break $\mathcal{Y}$ either has to treat the 3 keys as if they were really random which means he has to break $\mathcal{X}$, according to Theorem 2.1; or he has to use the dependency between the keys - this mean breaking the generator which was also based on $\mathcal{X}$! Thus, even though we have thwarted attacks like Merkle-Hellman and van Oorschot-Wiener by having a strong interdependency between the keys, we can still, if $\mathcal{X}$ is secure enough, get a connection between security of $\mathcal{X}$ and $\mathcal{Y}$.

3 Multiple encryption with minimum key

3.1 General Description of $\mathcal{Y}$

Let a block cipher $\mathcal{X}$ be given, as described above. The key length of $\mathcal{X}$ is denoted by k. By $E_K(P)$, we denote $\mathcal{X}$-encryption under K of block P, while $D_K(C)$ denotes decryption of C. We then define a new block cipher $\mathcal{Y}$ using a function G:

$$G(K_1, K_2) = (X_1, X_2, X_3)$$

which maps 2 $\mathcal{X}$-keys to 3 $\mathcal{X}$-keys. We display later a concrete example of a possible G-function. This is constructed from a few $\mathcal{X}$-encryptions. Keys in $\mathcal{Y}$ will consist of pairs (K_1, K_2) of $\mathcal{X}$-keys. Encryption in $\mathcal{Y}$ is defined by

$$E_{K_1,K_2}(P) = E_{X_3}(E_{X_2}(E_{X_1}(P))),$$

where $(X_1, X_2, X_3) = G(K_1, K_2)$. Decryption is clearly possible by decrypting using the X_i's in reverse order.

3.2 Relation to the security of $\mathcal{X}$

We would like to be reasonably sure that we have taken real advantage of the strength of $\mathcal{X}$ when designing $\mathcal{Y}$. One way of stating this is to say that $\mathcal{Y}$ is at least as hard to break as $\mathcal{X}$. By Theorem 2.1, this would be trivially true if the three keys used in $\mathcal{Y}$ were statistically independent. This is of course not the case, since the X_i's are generated from only 2 keys. But if the generating function G has a pseudorandom property as stated below, then the X_i's are "as good as random" and we can still prove the result we want.

Definition 3.1 *Consider the following experiment: an enemy B is presented with three k-bit blocks X_1, X_2, X_3. He then tries to guess which of two cases has occurred:*

1. *The X_i's are chosen independently at random.*
2. *The X_i's are equal to $G(K_1, K_2)$, for randomly chosen K_1, K_2.*

Let p_1 be the probability that B guesses 1 given that case 1 occurs, and p_2 the probability that B guesses 1 given that case 2 occurs. The generator function G is said to be pseudorandom*, if for any B spending time equal to T encryption operations,*

$$|p_1 - p_2| \leq \frac{T}{V},$$

where V is the total number of possible values of the pair (K_1, K_2).

The intuition behind this is that B could always spend his time simply trying random pairs of keys, seeing if they could be a possible value of K_1, K_2, and guessing that he is in case 2 if he finds a solution. If case 2 really occurs, he finds the right value with probability at most T/V (we assume here that he would need at least one encryption to test a pair). In case 1 there is most likely no solution. Thus the definition says that if G is pseudorandom, there is no better method for B than this naive attack. Definition 3.1 is inspired by the complexity theoretic definition of a strong pseudorandom generator introduced by Blum and Micali [1].

In the rest of this subsection we consider attacks against $\mathcal{X}$ and $\mathcal{Y}$ in a fixed scenario with a given plaintext distribution and a given form of attack, such as known plaintext, chosen plaintext, etc. We do not specify these things further, because the reasoning below will work for any such scenario. The time unit will be encryptions operations in system $\mathcal{X}$.

The next theorem shows the promised connection between security of $\mathcal{X}$ and $\mathcal{Y}$, i.e. in a given amount of time, an attack cannot do much better against $\mathcal{Y}$ than what is possible against $\mathcal{X}$.

Theorem 3.1 *Let p be the success probability of the best attack against $\mathcal{X}$ running in time T. Assume now that an attacker A against our new system $\mathcal{Y}$ runs in time T and has success probability $p + \epsilon$. If the function G used to construct Y is pseudorandom, then*

$$\epsilon \leq \frac{T}{V}$$

Proof: Let $\mathcal{Y}_0$ be the same system as $\mathcal{Y}$, but with independent keys X_i. By Theorem 2.1, using A against $\mathcal{Y}_0$ leads to an attack against $\mathcal{X}$ with the same success probability. Hence by assumption, A's success probability against $\mathcal{Y}_0$ will be at most p. But then we can use A to make an algorithm B that fits Definition 3.1: Given X_1, X_2, X_3, B uses these as keys in the triple encryption system and simulate A's attack. If A is successful, B will guess that the X_i's are generated from K_1, K_2, if not, B will guess that they are independent. Since in one case A will be attacking $\mathcal{Y}$, and in the other case $\mathcal{Y}_0$, it is clear that for this B, we have by Definition 3.1

$$\epsilon \leq |p_1 - p_2| \leq \frac{T}{V}$$

□

As an example of what the statement of the theorem means, consider an ideal case, where the best an attack against $\mathcal{X}$ can do, is to spend its time choosing random keys and test whether they fit with the information available. The success probability for time T would then be $T/2^k$ assuming a key can be tested in 1 encryption. Then the above theorem says that if G is pseudorandom, the success probability of any attack against $\mathcal{Y}$ running in time T can be at most $T/2^k + T/2^{2k}$. This is larger than the original success probability against $\mathcal{X}$ by a factor of only $1 + 2^{-k}$.

3.3 A Concrete Two-key Triple Encryption Construction

We propose here a new construction for triple encryption, called **TEMK** for **T**riple **E**ncryption with **M**inimum **K**ey. In this construction X_1, X_2, X_3 are all used as keys for encryption. We define this construction of $G(K_1, K_2) = (X_1, X_2, X_3)$ by:

$$\begin{aligned} X_1 &= E_{K1}(D_{K2}(E_{K1}(IV_1))) \\ X_2 &= E_{K1}(D_{K2}(E_{K1}(IV_2))) \\ X_3 &= E_{K1}(D_{K2}(E_{K1}(IV_3))) \end{aligned}$$

where IV_i are three initial values, e.g. $IV_i = C + i$, where C is a constant. It is seen that two-key triple encryption is used.

Here, the reader may ask a (very legitimate) question: why are we using ordinary two-key triple DES here, when we have just spent half a paper arguing that it does not provide good enough security? The answer is that we are using two-key triple DES in a special situation where we can guarantee that for any particular pair of keys, the enemy will get at most a known plaintext attack with three known plaintexts. This follows from the fact that the three constants IV_1, IV_2, IV_3 are universally fixed, such that the pair of keys K_1, K_2 will never be applied to anything else than the IV_i's. The best known attack against two-key triple DES with three known plaintexts is the one by van Oorschot and Wiener [7], which has the complexity $2^{120}/3 \simeq 1.3 \times 2^{118}$. Since in our case the keys are only 112 bits, we conjecture that this G is pseudorandom with the value $V = 2^{112}$. The most natural attack

Scheme	Key size	# KS	# EN	Total	W.k.	C.p.
TEMK-DES	112	5	9	19	No	No
DES	56	1	0	2	Yes	Yes
Two-key triple-DES	112	2	0	4	Yes	Yes
Three-key triple-DES	168	3	0	6	Yes	Yes

Table 1: Comparison of the proposed schemes and the existing ones, all used with DES

against pseudorandomness of G seems to be to guess either K_1 or K_2 and try to find the other value faster than exhaustive search.

The key scheduling in the above construction is slower than for the two-key triple encryption. In most software applications of the DES the key scheduling takes about twice the time of a single encryption. Using this estimate the key scheduling in the triple encryption scheme above takes time about 19 DES-encryptions. For comparison the key schedules for two-key triple DES and triple DES with three independent keys take 4 and 6 encryptions, respectively. In encryption with our new construction the key schedule should be performed once and the three round keys stored. In that way encryption with TEMK-DES is as fast as for other triple encryption schemes with fixed keys.

We conjecture that for the above construction, the fastest attack is a simple meet in the middle attack, which will be of time complexity at least 2^{2k}. In particular, we conjecture that because of the strong interdependency between the X_i's, attacks like the ones from [5, 7] will not be possible. Finally we note that the absence of weak keys are guaranteed, since the three round keys are never equal and the complementation property does not hold. In Table 1 we give a schematic overview of the differences between our proposed scheme and the existing ones. KS and EN are the numbers of DES key schedules and DES encryptions respectively, needed in the key schedule of the triple encryption scheme. 'Total' is the total number of encryptions in the new key schedule using the above estimate. Finally we state if weak keys exist and if the complementation property holds.

3.4 Extensions

In the preceding sections we focused on triple encryption schemes. It is clear that our ideas can be extended to quadruple, quintuple, ..., n-fold schemes. Let $\mathcal{X}$ be a component cipher with key size k. In general a $2i$-fold encryption scheme based on $\mathcal{X}$ is vulnerable to a meet in a middle attack using 2^{ik} words of memory taking time about 2^{ik}. Similarly, a $(2i+1)$-fold encryption scheme based on $\mathcal{X}$ is vulnerable to a meet in a middle attack using 2^{ik} words of memory taking time about $2^{(i+1)k}$. Therefore, one does not get the security of the full key length. It is obvious that by generating the $2i$ $(2i+1)$ keys pseudorandomly, defined in a similar manner as Definition 3.1, from i $(i+1)$ keys one can prove a similar result as Theorem 3.1.

References

[1] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, pages 856–864, 1984.

[2] M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology - Proc. Eurocrypt'93, LNCS 765*, pages 386–397. Springer Verlag, 1993.

[3] M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In Y. G. Desmedt, editor, *Advances in Cryptology - Proc. Crypto'94, LNCS 839*, pages 1–11. Springer Verlag, 1994.

[4] U. Maurer and J.L. Massey. Cascade ciphers: The importance of being first. *Journal of Cryptology*, 6(1):55–61, 1993.

[5] R. Merkle and M. Hellman. On the security of multiple encryption. *Communications of the ACM*, 24(7):465–467, 1981.

[6] W. Tuchman. Hellman presents no shortcut solutions to DES. *IEEE Spectrum*, 16(7):40–41, July 1979.

[7] P.C. van Oorschot and M.J. Wiener. A known-plaintext attack on two-key triple encryption. In I.B. Damgård, editor, *Advances in Cryptology - Proc. Eurocrypt'90, LNCS 473*, pages 318–325. Springer Verlag, 1990.

[8] M.J. Wiener. Efficient DES key search. Technical Report TR-244, School of Computer Science, Carleton University, Ottawa, Canada, May 1994. Presented at the Rump Session of Crypto'93.

Recent Publications in the BRICS Report Series

RS-94-38 **Ivan B. Damgård and Lars Ramkilde Knudsen.** ***Enhancing the Strength of Conventional Cryptosystems*****. November 1994. 12 pp.**

RS-94-37 **Jaap van Oosten.** ***Fibrations and Calculi of Fractions*****. November 1994. 21 pp.**

RS-94-36 **Alexander A. Razborov.** ***On provably disjoint* NP*-pairs*****. November 1994. 27 pp.**

RS-94-35 **Gerth Stølting Brodal.** ***Partially Persistent Data Structures of Bounded Degree with Constant Update Time*****. November 1994. 24 pp.**

RS-94-34 **Henrik Reif Andersen, Colin Stirling, and Glynn Winskel.** ***A Compositional Proof System for the Modal μ-Calculus*****. October 1994. 18 pp. Appears in: Proceedings of LICS '94, IEEE Computer Society Press.**

RS-94-33 **Vladimiro Sassone.** ***Strong Concatenable Processes: An Approach to the Category of Petri Net Computations*****. October 1994. 40 pp.**

RS-94-32 **Alexander Aiken, Dexter Kozen, and Ed Wimmers.** ***Decidability of Systems of Set Constraints with Negative Constraints*****. October 1994. 33 pp.**

RS-94-31 **Noam Nisan and Amnon Ta-Shma.** ***Symmetric Logspace is Closed Under Complement*****. September 1994. 8 pp.**

RS-94-30 **Thore Husfeldt.** ***Fully Dynamic Transitive Closure in Plane Dags with one Source and one Sink*****. September 1994. 26 pp.**

RS-94-29 **Ronald Cramer and Ivan Damgård.** ***Secure Signature Schemes Based on Interactive Protocols*****. September 1994. 24 pp.**

RS-94-28 **Oded Goldreich.** ***Probabilistic Proof Systems*****. September 1994. 19 pp.**