HOW TO MAKE DES-BASED SMARTCARDS FIT FOR THE 21-ST CENTURY

Cryptographic Techniques for Advanced Security Requirements

Stefan Lucks
Theoretische Informatik
*University of Mannheim, 68131 Mannheim, Germany**
lucks@th.informatik.uni-mannheim.de

Rüdiger Weis
convergence integrated media GmbH
Berlin, San Francisco, Amsterdam
ruedi@convergence.de

Abstract With its 56-bit key size, the data encryption standard (DES) seems to be at end of its useful lifetime. Also, the 64-bit DES block size is dangerously small for some applications. We discuss techniques such as triple DES and DESX to push up the key size, and we present DEAL to increase both block and key size. We propose DEAL^{KX}, a new variant of DEAL with an improved key schedule.

Keywords: DES cards, triple DES, DESX, DEAL, block cipher, advanced security

INTRODUCTION

The 'data encryption standard' (DES) is the most widely used block cipher in the world. For many – if not most – smartcards, a DES implementation is available. But soon, a DES successor AES ('advanced encryption standard') – the *'encryption standard for the 21-st century'* – is to be chosen. Will all the investments in the DES be lost, then?

With its tiny key space of 56 bit, ordinary single DES has long been known to be insecure, and most serious security applications use triple

*Supported by Deutsche Forschungsgemeinschaft (DFG) grant KR 1521/3-1.

DES instead. Triple DES usually comes in either of two variants: Two-key triple DES with a (formal) key size of $2 * 56 = 112$ bit, and three-key triple DES with $3 * 56 = 168$ key bits. For the AES, three key sizes are defined: 128, 192, and 256 bit.

Apart from being three times slower than single DES, triple DES still inherits the DES block size of 64 bits. Even for an ideal b-bit block cipher, security is at risk if more than $\sqrt{2^b} = 2^{b/2}$ bits are encrypted under the same key – the 'square-root bound'. Today, some practical applications can exceed the square-root bound for DES and triple DES with $b = 64$. Therefore, the AES block size was specified with $b = 128$.

In the current paper we discuss DES-based block ciphers with an improved key length, compared to single DES. We start with some remarks on the security of single DES in Section 1. We continue with Section 2 regarding double and triple DES. Section 3 treats DESX and frugal DESX to improve the DES key size. Both are about as efficient as single DES. In Section 4 we describe DEAL, a DES-based block cipher with 128 bit blocks. For the original key schedule of DEAL some security problems are described. To counter these problems, we propose two variants of DEAL in Section 5. We finally discuss why one may prefer DES-based encryption over the AES, consider smartcard specific implementation issues, and summarise our results.

1. SINGLE DES AND ITS SECURITY

With 56 bit keys, single DES is vulnerable to straightforward exhaustive key search attacks. On the average, such an attack needs only 2^{55} single DES encryptions. Are there other attacks for single DES? Note that good dedicated attacks on single DES might well endanger DES-based constructions such as the ones described in this paper, for which exhaustive key search is not feasible.

Dedicated attacks against single DES are known, which actually are faster than exhaustive key search. Such attacks are the differential and linear cryptanalysis of DES [3, 13] and the Davies' attack [2]. But all these known analytic attacks require more than 2^{40} known or chosen plaintexts to be encrypted under the same key. Like every 64-bit block cipher, single DES should not be used to encrypt more than 2^{32} blocks, anyway. Thus, while the above dedicated attacks constitute great theoretical advances, their practical relevance is quite low.

2. DOUBLE DES AND TRIPLE DES

Double DES takes two 56-bit keys L and M and computes the encryption function

$$\mathrm{DES}^2_{L,M}(P) = \mathrm{DES}_M(\mathrm{DES}_L(P)).$$

Similarly, triple DES takes three 56-bit keys L, M, and N, and computes

$$\mathrm{DES}^3_{L,M,N}(P) = \mathrm{DES}_N(\mathrm{DES}^{-1}_M(\mathrm{DES}_L(P))).$$

See also Figure 1.

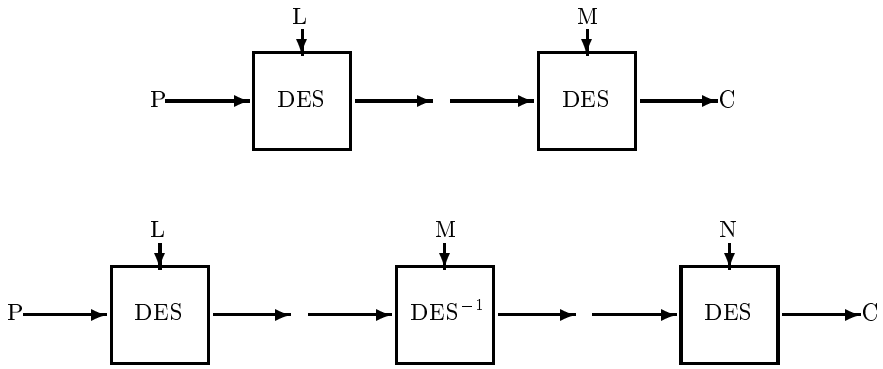

Figure 1 Double DES with two 56-bit keys L and M (top) and Triple DES with three 56-bit keys L, M, and N (bottom).

Double DES has long been known to be vulnerable to meet-in-the-middle (MITM) techniques, see Merkle and Hellman [14]. While the original MITM attack requires a huge amount of memory and is hardly feasible, Oorschot and Wiener [16, 17] developed improved MITM techniques with greatly reduced space requirements at the cost of a slightly increased computation time for the attacker.

Triple DES is typically used in either of two flavours: three-key triple DES with three independent sub-keys L, M, and N, and two-key triple DES with $N = L$.

Two-key triple DES is vulnerable to Oorschot's and Wiener's MITM techniques [16] and can be broken with 2^{56} units of everything: memory, time and chosen plaintexts. An attack which requires 2^{56} plaintexts chosen by the attacker to be encrypted under the same key hardly seems practical. On the other hand, some trade-offs between memory, time and the number of chosen plaintexts are possible. While no practical attack for two-key triple DES is known, we would prefer to use three-key triple DES wherever possible.

Three-key triple DES can be attacked by the same standard MITM technique as double DES, but this attack requires 2^{112} steps of computation. Lucks [11] reduced this to about 2^{108} single DES encryptions. This is infeasible today and will remain so for decades. We conclude: The effective key size of three-key triple DES is sufficient to protect data for quite a long time.

But three-key triple DES also suffers from some drawbacks, such are:

1 Triple DES is slow – one triple DES encryption takes roughly the time of three single DES encryptions.

2 Three-key triple DES makes bad use of its key size, compared to an ideal 168-bit block cipher.[1]

3 Even if we assume that no practical attack against single DES exists, we do not know whether Lucks' attack is the best one against three-key triple DES.

4 The block size of three-key triple DES is only 64 bit, as in the case of single DES.

3. DESX AND FRUGAL DESX

In this section we describe the DES-based block cipher DESX. DESX is a 64-bit block cipher and uses a triple of two 64-bit sub-keys M_0 and M_1 and a 56-bit DES-key L as its key:

$$\mathrm{DESX}_{M_0,L,M_1}(P) = \mathrm{DES}_L(P \oplus M_0) \oplus M_1.$$

We also describe a variant DESX′ or 'frugal DESX', for which 120 key bits suffice by defining $M_0 = M_1$. In other words, frugal DESX takes a key pair (L, M) as its key, where the sub-key L is a 56-bit DES key and the sub-key M is a 64-bit 'whitening key':

$$\mathrm{DESX}'_{L,M}(P) = \mathrm{DES}_L(P \oplus M) \oplus M.$$

Both DESX and frugal DESX can be seen as a remedy for the drawbacks 1–3 of triple DES.

Due to the DES complementation property[2], both DESX and frugal DESX have pairs of equivalent keys, i.e. $\mathrm{DES}_{M_0,L,M_1} = \mathrm{DES}_{\overline{M_0},\overline{L},\overline{M_1}}$ for DESX and $\mathrm{DES}_{M,L,M} = \mathrm{DES}_{\overline{M},\overline{L},\overline{M}}$ for frugal DESX.

3.1. THE BLACK-BOX MODEL

Now, we describe the *'black-box model'* for block ciphers. While the internal structure of the DES is known, we argued above that the most

practical attack against single DES is exhaustive key search, i.e., does not exploit the internal structure of DES. In this context, it seems reasonable to view a block cipher such as DES as a black-box with unknown internals.

The block cipher F with α bit keys and β bit blocks, is assumed to consist of 2^α random permutations $F_K : \{0,1\}^\beta \longrightarrow \{0,1\}^\beta$ (for us, 'random' always implies 'chosen according to the uniform probability distribution'). 'F_K^{-1}' means decryption under K. The adversary has no access to the 'internal structure' of F, but for any $K \in \{0,1\}^\alpha$ and $b \in \{0,1\}^\beta$ she can ask a 'black box' for both $F_K(b)$ and $F_K^{-1}(b)$.

In the model, the black boxes are realised by an oracle. The adversary can ask the oracle for $F_K^{\pm 1}(X)$, the encryption or decryption of a text $X \in \{0,1\}^\beta$ under a key $K \in \{0,1\}^\alpha$. For the composed block cipher E, the adversary can also choose plaintexts

Given a query $(\pi, b) \in (\{\, E^{\pm 1} \,\} \cup \{\, F_k^{\pm 1} \mid k \in \{0,1\}^\alpha \,\}) \times \{0,1\}^\beta$, the oracle's reply is $\pi(b) \in \{0,1\}^\beta$. An adversary asking at most e queries $(E^{\pm 1}, b)$ and at most f queries $(F_k^{\pm 1}, b)$ is '(e,f)-*restricted*'. Usually, e is under the control of the security architect, and f is a measure for the computational resources needed for the attack.

The adversary A has to decide whether E is a random permutation $\pi : \{0,1\}^\beta \longrightarrow \{0,1\}^\beta$ or a permutation $E_{K^*}^* : \{0,1\}^\beta \longrightarrow \{0,1\}^\beta$, depending on F and a random key $K^* \in \{0,1\}^\gamma$ unknown to A. (Here, '$\{0,1\}^\gamma$' denotes the key space of E^*.) A's output is $b \in \{0,1\}$. A's *'advantage'* is the unsigned difference of two probabilities: Prob[A outputs 1 if $E = \pi$] and Prob[A outputs 1 if $E = E_{K^*}$].

The above describes a very weak adversary, a 'distinguishing adversary'. This is a theoretical concept – in practice an adversary wants to find something *useful* for her, not just distinguish the encryption used from a random permutation. But any practical attack, if successful, clearly would allow distinguishing, too. In other words, the non-existence of distinguishing attacks implies the security against practical attacks (assuming the number of plaintext/ciphertext pairs known to the adversary is well below the square-root bound).

3.2. FORMAL TREATMENT AND RESULTS

Even and Mansour [6] described a block cipher using 'a single pseudo-random permutation' without a secret key. (One could think of using, say, single DES under a fixed non-secret 56-bit key.) Their main result was that the advantage of an (e,f)-restricted adversary does not exceed $ef/2^\beta$ (asymptotically). By describing and analysing an appropriate attack, Daemen [5] demonstrated this bound to be tight.

The work of Kilian and Rogaway [7] on 'FX' (where F is a block cipher such as the DES, with α-bit keys and β-bit blocks) can be seen as a generalisation of Even's and Mansour's work. They proved:

Theorem 1 *Let A be an (e, f)-restricted adversary attacking FX. A's advantage is at most $ef/2^{\alpha+\beta-1}$.*

(Even's and Mansour's scheme is the special case with $\alpha = 0$). By generalising Daemen's attack Kilian and Rogaway also demonstrated the tightness of their bound. They described an attack on FX to recover the secret FX key. The attack treats F as a black-box and on the average asks for about $2^{\alpha+\beta+1}/e$ F-encryptions.

Theorem 1 bounds the advantage of a black-box attacker from below. The attack found by Kilian and Rogaway is an upper bound extremely close to the lower bound.

If we use DES for the block cipher template F, we get DESX. As mentioned by Kilian and Rogaway, DESX has been invented by Rivest and is used by RSA inc. since about 1985. Note that DESX, as described by Kilian and Rogaway, utilises 184 independent key bits: a 64-bit pre-whitening key M_0, a 64-bit post-whitening key M_1, and a DES key L. The original variant of DESX from RSA inc. just derives the sub-key M_1 from M_0 and L [19]. The advantage of the original variant is that no obvious pairs of equivalent keys seem to exist. But it is unclear how to prove something similar to Theorem 1 for the original variant, which we ignore in this paper, writing 'DESX' for the Kilian/Rogaway variant.

What about the security of frugal FX? Surprisingly, it is just as secure as FX itself. In a side-note, Kilian and Rogaway point out that Theorem 1 also holds for frugal FX (which they denoted 'FX′'). This is in spite of an FX key being β bit longer than a key for frugal FX. In other words: **DESX with its 184-bit key is as secure as frugal DESX with its 120-bit key!** This is an important observation which seems to have been overlooked by many engineers implementing DESX before. A formal treatment of FX′ and a proof of security are given in the appendix.

What does Theorem 1 (or Theorem A1 in the appendix) tell us about the security of DESX (or frugal DESX)? If we allow $e \leq 2^{32}$ blocks to be encrypted under the same key – more than this is dangerous anyway – the adversary has to do the equivalent of more than 2^{85} single DES encryptions for a key-search attack. A paper written in 1995 [4] estimates that the key-length of a cryptosystem should be at least 75 bits 'to provide adequate commercial security' against well-funded adversaries (such as government agencies). Due to Moore's law, DESX and frugal DESX provide protection until at least 2010. If we further restrict the number

e of blocks encrypted under the same key, we improve the security. (In Section 5, we describe an application of frugal DESX with e being tiny.)

4. DEAL

In [9], Knudsen proposes the 128-bit cipher DEAL with a block size of 128 bits. It uses DES in the round function and accepts three different key sizes, namely 128, 192, and 256 bits. For the first two sizes, Knudsen recommends to use six rounds, for for 256 bit keys eight rounds are recommended. Depending on the key size, the three variants of DEAL are denoted DEAL-128, DEAL-192, and DEAL-256. DEAL has been a candidate for the NIST AES standard, but was not selected as one of the five finalists. In this paper, we often write 'DEAL' for DEAL-128. Parts of this section have been taken from [12].

4.1. A DESCRIPTION OF DEAL

4.1.1 The DEAL Core. A 128-bit plaintext is split up into two halves $(x_0, y_0) \in (\{0,1\}^{64})^2$. Two consecutive rounds j and $j+1$ of DEAL take the 128-bit block $(x_{j-1}, y_{j-1}) \in (\{0,1\}^{64})^2$ and the two round keys R_j and R_{j+1} as the input to compute the output block $(x_{j+1}, y_{j+1}) \in (\{0,1\}^{64})^2$ by

$$\begin{array}{lcllcl} x_j & := & x_{j-1}, & & y_j & := & y_{j-1} \oplus E_{R_j}(x_{j-1}), \\ x_{j+1} & := & x_j \oplus E_{R_{j+1}}(y_j), & \text{and} & y_{j+1} & := & y_j, \end{array}$$

where $\oplus$ describes the bit-wise XOR-operation for 64-bit strings and j is odd. By E, we denote the DES encryption function. Figure 2 describes the full six rounds of DEAL.

Thus, DEAL (=DEAL-128) needs 6 round keys $R_1, \ldots R_6$. Internally, every round key is used as a DES key, ignoring the 'parity bits' and hence consists of 56 bits. DEAL uses a 'key schedule' to generate the round keys from the given 128-bit master key.

4.1.2 The DEAL Key Schedule. The key schedule of DEAL takes two keys K_1, K_2, each of 64 bit, and returns 6 round keys R_1, $\ldots$, R_6 of 56 bits each. The round keys are generated using DES encryption under a fixed DES-key R_* (which is, in hexadecimal notation, $R_* =$ `0123456789abcdef`). $(\underline{1})$, $\ldots$, $(\underline{4})$ are four different constant 64-bit strings, where none is `000...0`. (For details, how the constants $(\underline{i})$ are defined, we refer the reader to [9]. See also our definition of similar constants in Section 5.3.)

The DEAL round keys are generated from K_1, K_2 as follows:

$$R_1 := E_{R_*}(K_1)$$

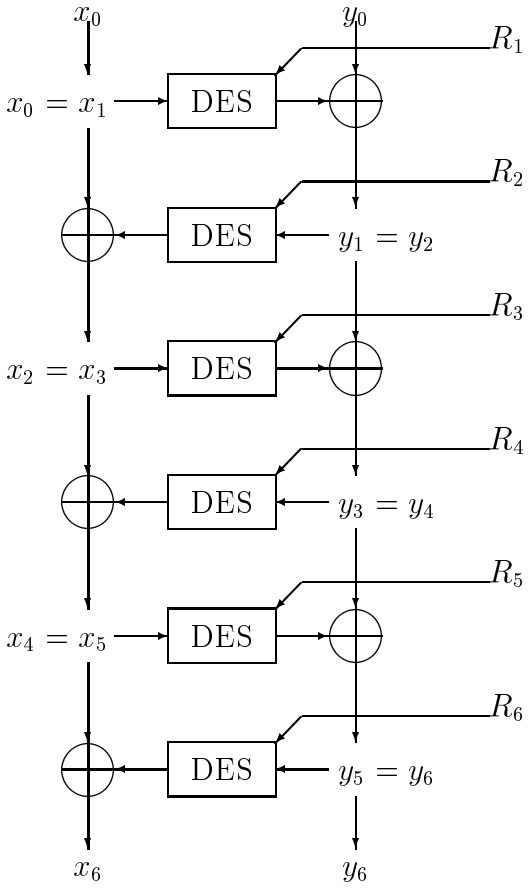

Figure 2 The DEAL core

$$
\begin{aligned}
R_2 &:= E_{R_*}(K_2 \oplus R_1) \\
R_3 &:= E_{R_*}(K_1 \oplus R_2 \oplus (\underline{1})) \\
R_4 &:= E_{R_*}(K_2 \oplus R_3 \oplus (\underline{2})) \\
R_5 &:= E_{R_*}(K_1 \oplus R_4 \oplus (\underline{3})) \\
R_6 &:= E_{R_*}(K_2 \oplus R_5 \oplus (\underline{4}))
\end{aligned}
$$

The parity bits of the 64-bit values R_i are ignored when R_i is used as a DES-key (i.e., a DEAL round key), but are relevant for computing R_{i+1}.

4.2. THE SECURITY OF DEAL

MITM techniques can be used to recover the DEAL round keys. This was stressed by Knudsen himself. The six-round version of DEAL is vulnerable to a meet-in-the-middle attack requiring roughly $(2^{56})^3 = 2^{168}$ encryptions. For the eight-round version, the attack needs roughly $(2^{56})^4 = 2^{224}$ encryptions.

Thus the theoretical key size of DEAL is approximately no more than 168 for the six-round version and 224 for the eight-round version. This bounds the theoretical key size of DEAL-192 (six rounds) and DEAL-256 (eight rounds). Due to their memory requirements, these meet-in-the-middle techniques are quite unrealistic, though trade-off techniques to save storage space at the cost of increased running time are known [17].

Knudsen describes a chosen plaintext attack to recover the round keys of the six-round version of DEAL. His attack requires about 2^{121} DES-encryptions using roughly 2^{70} chosen plaintexts. Due to the huge amount of chosen plaintexts, this attack appears to be quite impractical. But this results indicates that theoretically the effective key size of DEAL cannot exceed 2^{122}.

So far, so good! But several authors [12, 20, 10] expressed concerns about the DEAL key schedule:

- The first round key does not depend on all bits of the master key (K_1, K_2) [12, 20].
- By generating 64-bit values and cutting off the parity bits, sets of equivalent or near-equivalent keys are generated [10].

While these weaknesses do not directly enable practical attacks, they can nevertheless cause serious security problems in practice.

If we fix K_1, the remaining DEAL cipher is not just insecure because of its 64-bit key K_2: The first round key R_1 only depends on K_1, and thus the first round becomes a known transformation. Now, the attacker only has to attack 5 rounds of DEAL, which is significantly more easy than attacking a 6-round variant under the same key size. In other words, it is more useful for the attacker to know K_1 than to know K_2.

Now, as a concrete example, let a 80-bit key be chosen at random. (Today, 80-bit of key entropy should be just enough to frustrate even well-funded adversaries.) The 80-bit key is somehow padded to make a 128-bit key, used as a master key for DEAL, e.g. 48 bits of the 128-bit master key (K_1, K_2) are constantly zero. An application programmer may reasonably expect that the security of such a scheme does not greatly depend on which bits are set to zero. But exactly this happens in the case of DEAL: If 48 bit of K_2 are set to zero, the remaining cipher is

as secure as could be expected from a variant of DEAL with only 80-bit keys. If 48-bit of K_1 are set to zero, the cipher is significantly weaker – essentially, 2^{16} attacks on five-round DEAL with 64-bit keys are to be mounted.

We conclude: DEAL constitutes an excellent way way of getting a DES-based block cipher with 128-bit blocks, but the key schedule of DEAL is somewhat worrisome.

5. THE DEAL$^{\mathrm{KX}}$-FAMILY

In this section, we propose new variants of DEAL (that is, variants with the same core but using a different key schedule) to fix the problems described above. We propose DEAL$^{\mathrm{KX}}$, a new family of DEAL variants, and specifically we propose DEAL$^{\mathrm{KX}}$-120 for the use of 120-bit keys and DEAL$^{\mathrm{KX}}$-128 for 128-bit keys.

Recall that frugal DESX provides excellent security if the number e of plaintext/ciphertext pairs is tiny. Our variants of DEAL use frugal DESX (=DESX′) in the key schedule. This approach has been inspired by Knudsen's 'strong key schedules' [8].

5.1. DEAL$^{\mathrm{KX}}$-120

Let a 120-bit key (L, M) be given, where L is a 56-bit value and M is a 64-bit value. Let l be the first bit of the 56-bit DES key L, i.e. bit no. 1. (As specified in the standard [15], a DES key consists of 64 bits, numbered from 1 to 64. Eight of these, namely bit 8, bit 16, ..., bit 64 are used as parity bits.) The key schedule of DEAL$^{\mathrm{KX}}$-120 is straightforward: Define 12 distinct constants $\underline{(1,0)}$, $\underline{(1,1)}$, ..., $\underline{(6,0)}$, and $\underline{(6,1)}$ and generate the round keys R_i with $1 \le i \le 6$ as follows:

$$R_i := \mathrm{DES}_L(\underline{(i,l)} \oplus M) \oplus M. \tag{1}$$

(In Section 5.3, some details are given how to define the constants.) What about the security of DEAL$^{\mathrm{KX}}$-120 and its new key schedule? First note that for a given key (L, M), only six plaintext/ciphertext pairs are produced, where the plaintexts are constants $\underline{(1,l)}$, ..., $\underline{(6,l)}$ and the ciphertexts are the round keys R_1, ..., R_6. Assume the adversary's workload to be limited to the equivalent of 2^{110} single DES encryptions, i.e., $f \le 110$. Theorem 1 (or rather Theorem A1, proven in the appendix) shows that the the chance to distinguish the six round keys R_1, ..., R_6 from random values is negligible. This makes DEAL$^{\mathrm{KX}}$ for $f \le 110$ as secure as DEAL with independent round keys.

5.2. DEAL^{KX}-128

DEAL^{KX}-120 has the drawback of a nonstandard 120-bit key size. In the future, many block cipher applications will require the use of 128-bit keys for interface-compatibility with the AES or other standards. Given a 128-bit key, we could simply drop eight of the key bits and use DEAL^{KX}-120. But up to eight bits of valuable entropy might get lost. If, e.g., the 128 key bits consists of 16 lower-case letters 'a'–'z', the entropy of the full 128-bit key is $\log_2(26) * 16 \approx 75$ bit, but the entropy of the 120-bit key is slightly more than 70 bit.

As a remedy, we propose a dedicated key schedule for 128-bit keys (L, M, n), where L is a 56-bit DES key, M is a 64-bit value, and the 16-th key byte is denoted by $n \in \{0,1\}^8$. Similarly to the previous section, we consider a set of $6 * 2 * 256 = 3072$ distinct constants $(\underline{i,l,n})$, i and l as above, and $n \in \{0,1\}^8$ is the sixteenth key byte. Generate the round keys $R_1, \ldots, R_6$ as follows:

$$R_i := \mathrm{DES}_L((\underline{i,l,n}) \oplus M) \oplus M.$$

We stress that our reason for proposing DEAL^{KX}-128 was not to improve the effective key size, compared to DEAL^{KX}-120. The main point of DEAL^{KX}-128 is to protect possible low-entropy keys of 128-bit size from further loss of entropy.

5.3. DETAILS AND REMARKS

How do we define the constants $(\underline{i,l})$ for DEAL^{KX}-120 and the constants $(\underline{i,l,n})$ for DEAL^{KX}-128? Let $z = (0,0,0,0,0,0,0,0)$ be the all-zero byte. To make DEAL^{KX}-128 upward compatible to DEAL^{KX}-120, we set $(\underline{i,l}) = (\underline{i,l,z})$.

From a point of security, it suffices that any two constants $(\underline{i,l,n})$ and $(\underline{i',l',n'})$ with $i \neq i'$ or $l \neq l'$ or $n \neq n'$ are different. Thus we may define the constants such that these can be generated efficiently on a smartcard. Certainly, we would not want to store a huge number of fixed 'random' constants in the program code. Any constant $(\underline{i,l,n})$ is a 64-bit string, regarded as an 8-tuple $(X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7)$ of bytes. Byte X_0 stores the bits $0, \ldots, 7$, byte X_1 the bits $8, \ldots, 15$, and so on.

Within this paragraph, we use hex notation for byte constants. We set $X_2 = \cdots = X_6 = 00$. The first byte X_0 is used as a 'round counter': for $i = 1$ we set $X_0 = 01$, for $i = 2$ we set $X_0 = 02$, ..., and $X_0 = 06$ for $i = 6$. The second byte is used for l: if $l = 0$ then $X_1 = 00$ else $X_1 = 01$. The last byte X_7 is identical to the key byte n. As an example, $(\underline{3,1,\mathrm{BF}}) = (03, 01, 00, 00, 00, 00, 00, \mathrm{BF})$.

Why did we treat the key-bit l of L in such a special way? Recall that due to the DES complementation property, the DESX′ key space consists of 2^{119} pairs of equivalent keys. We want to avoid that DEAL^{KX} inherits this weakness from DESX′. Equation (1) makes sure that if we encrypt under two different but equivalent DESX′ keys, we encrypt different constants – and thus produce different round keys.

What are the advantages of the new key schedule proposal for DEAL, compared to previous ones? As described above, the original DEAL key schedule suffered from some security problems. Like the original DEAL key schedule for 128 bit keys, the DEAL^{KX} key schedules require six single DES encryptions, all under the same DES key. In other words, both offer essentially the same performance.

Originally, DEAL was defined for key sizes of 128, 192 and 256 bit. All three key schedules for the different key sizes follow the same design principle. On the other hand, it is difficult to extend the design principle behind the DEAL^{KX} key schedules for larger key sizes. In this sense, the DEAL^{KX} key schedules are less flexible than the original one.

But, if one has to meet such extreme security requirements that one needs more than, say, 110 bits of key size, one can (and perhaps should) abandon DES-based encryption and move to the AES.

Lucks [12] described a modification of the original DEAL key schedule, to remedy some of its weaknesses. Lucks' modification offers the same degree of flexibility as the original key schedule and thus allows key sizes of 128, 192 and 256 bit. But the modification is slightly slower than the original key schedule and the DEAL^{KX} one, requiring seven single DES encryption operations for 128-bit keys, instead of six.

6. DISCUSSION

6.1. DES-BASED ENCRYPTION OR AES?

As mentioned in the introduction, the DES successor AES will soon be chosen. One member of this conference's program committee criticised our approach: *'With AES, the next generation of cards will implement the new algorithms and the life time of cards is typically only a few years.'* So why did we consider DES-based ciphers at all?

First, this paper has been written to explain that there is no need for hecticness when moving from DES-based to AES-based encryption. DES-based encryption offers excellent security for many years.

Second, though we believe that many smartcards will support the AES soon, we expect DES-based encryption to coexist with the AES for more or less unlimited time. Some reasons for this expectation are:

1 Even after the AES is chosen, DES-based legacy standards will persist. (In his talk at the submitters' session at the third AES Candidate Conference, April 2000, Ross Anderson provided an example for such a legacy standard: A DES-based speed limiter system standardised in the European Union long ago.)

2 By the end of 1999, triple DES has been re-affirmed as an official standard *64-bit* block cipher [15]. The coexistence of triple DES and the AES has been signalised before [18].

3 Using a 128-bit block cipher is not always preferable to using a 64-bit one. If the application typically deals with small plaintexts, the bandwidth penalty for a 128-bit block cipher can be high.[3]

Note that the bandwidth issue is particularly important for smartcards, where the communication bandwidth can be a major performance obstacle (see e.g. [1, 21]). And if a smartcard is supposed to support both 64-bit block encryption *and* 128-bit block encryption, it suffices to implement DES-based encryption schemes, such as triple DES and DEAL. If one insists on using the AES for 128-bit block encryption, one may have to additionally implement triple DES on the same smartcard to support 64-bit block encryption, too.

6.2. IMPLEMENTATION CONSIDERATIONS

Consider an implementation of either cipher using a subroutine for single DES. On a typical smartcard without built-in DES hardware, the subroutine's code size can be expected to greatly exceed the additional code to implement either DESX, DESX′, triple DES, or DEAL$^{\mathrm{KX}}$. So we argue: in practice the **code sizes** of these ciphers are about the same.

What about the **throughput** of our ciphers? We simply count the calls to the subroutine, neglecting the time for the XOR operations. DESX and DESX′ need one subroutine call to encrypt a 64-bit block, triple DES needs three calls, and DEAL needs six calls to encrypt a 128-bit block. In other words, triple DES and DEAL achieve the same throughput, while DESX and DESX′ are three times faster. [4]

In the case of DEAL$^{\mathrm{KX}}$, we run a **key generation** procedure before we start encrypting. It calls the DES subroutine six times, i.e., takes as much time as encrypting one 128-bit block. Neither DESX, nor DESX′, nor triple DES need an expensive key generation.

Memory, and especially **RAM**, can be precious on a smartcard. For DESX, DESX′ and triple DES we just need to store the data block (64 bit) and the key (120–184 bit) in RAM, together 23–31 byte. DEAL needs storage space for the data block (128 bit) and the six round keys

$R_1, \ldots, R_6$ of $6 * 56 = 336$ bit, i.e. 464 bit or 58 byte. If we also store the master key (120–128 bit), we need another 15 or 16 byte of RAM.

On low-end smartcards, this may be too much for our given application. (Note that we cannot expect to use all the RAM on the smartcard just for the sole purpose of encrypting one plaintext block.) Fortunately, the DEAL$^{\mathrm{KX}}$ key schedule allows **key generation 'on the fly'**. I.e., we store the master key, generate the round key R_i when we need it, and we overwrite R_i when we need the next round key. Now less than 40 byte of storage space suffice: for the data block (128 bit), for one round key (56 bit) and for the master key (120–128 bit). This slows down encryption by a factor of 2, i.e., to encrypt one 128-bit block, we have to call the DES subroutine 12 times. The DEAL$^{\mathrm{KX}}$ key schedule is better suited for key generation 'on the fly' than the original key schedule, especially if our smartcard has to support both DEAL encryption and DEAL decryption.

6.3. SUMMARY

This paper describes triple DES and DESX, DES-based block ciphers with an improved key size. Triple DES is widely believed to be secure, but no formal justification of this belief is known. The security of DESX can be justified by a rigorous formal treatment, assuming no unknown weakness of single DES itself exits. A frugal variant of DESX, also called DESX$'$, is as secure as DESX and makes better use of the key space.

This paper also describes DEAL, a DES-based block cipher which, apart from improving the key size, also doubles the DES block size. Because of problems with the original DEAL key schedule, new variants of DEAL are proposed, namely DEAL$^{\mathrm{KX}}$-120 and DEAL$^{\mathrm{KX}}$-128.

ACKNOWLEDGEMENT

By an early discussion on block ciphers with non-standard (i.e. 120 bit) key sizes, Peter Honeyman motivated the authors to define DEAL$^{\mathrm{KX}}$-128 in addition to DEAL$^{\mathrm{KX}}$-120. Richard Outerbridge inspired a couple of clarifications for the description of DEAL$^{\mathrm{KX}}$. The anonymous referees made many useful suggestions how to improve this paper.

Notes

1. Breaking an ideal block cipher with a 168-bit key would require 2^{167} full encryptions on the average.

2. Let $\overline{a}$ denote the bit-wise complement of $a \in \{0,1\}^*$. By the 'DES complementation property' we describe the fact that for all keys $\kappa \in \{0,1\}^{64}$ and all plaintexts $\pi \in \{0,1\}^{56}$ the equation $\mathrm{DES}_\kappa(\pi) = \overline{\mathrm{DES}_{\overline{\kappa}}(\overline{\pi})}$ holds.

3. We stress that a 64-bit block cipher should only be used if the application designer can guarantee the number of plaintexts encrypted under the same key to be well below 2^{32}. Otherwise, one better accepts the performance penalty and uses a 128-bit block cipher, such as DEAL or the AES.

4. Both DEAL and triple DES can be optimised slightly. A DES encryption starts with applying a fixed permutation on the plaintext bits and finishes with applying the inverse permeation on the ciphertext bits. Hence, 'inside' triple DES and DEAL, these permutations may be left out.

References

[1] B. Bakker, R. Weis, S. Lucks *How to Ring a Swan – Adding Tamper Resistant Authentication to Linux IPSec*, SANE2000 – 2nd International System Administration and Networking Conference, Maastricht (2000).

[2] E. Biham, A. Biryukov, *An improvement of Davies' attack on DES*, J. Cryptology, Vol. 10 (1997), 195–205.

[3] E. Biham, A. Shamir, *Differential cryptanalysis of the data encryption standard*, Springer (1993).

[4] M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson, M. Wiener, *Minimal key lengths for symmetric ciphers to provide adequate commercial security.*
Online: `http://theory.lcs.mit.edu/~rivest/publications.html`

[5] J. Daemen, *Limitations of the Even-Mansour construction*, Asiacrypt '91, Springer LNCS 739, 495–498.

[6] S. Even, Y. Mansour, *A construction of a cipher from a single pseudorandom permutation*, Asiacrypt '91, Springer LNCS 739, 210–224.

[7] J. Kilian, P. Rogaway, *How to protect DES against exhaustive key search*, Crypto '96, Springer LNCS 1109, 252–267. Full version online: `http://wwwcsif.cs.ucdavis.edu/~rogaway/papers/list.html`

[8] L. Knudsen, *Practically secure Feistel Ciphers*, Fast Software Encryption 93, Springer LNCS 809, 211–221.

[9] L. Knudsen: 'DEAL – a 128-bit Block Cipher', February 21, 1998, revised May 15, 1998: `http://www.ii.uib.no/~larsr/aes.html`.

[10] J. Kelsey, B. Schneier, *Key-Schedule Cryptanalysis of DEAL*, SAC '99.

[11] S. Lucks, *Attacking triple encryption*, Fast Software Encryption 98, Springer LNCS 1372, 239–257.

[12] S. Lucks, *On the security of the 128-bit block cipher DEAL*, Fast Software Encryption 99.

[13] Mitsuru Matsui, *Linear cryptanalysis method for DES cipher*, Eurocrypt '93, Springer LNCS 765, 386–397.

[14] R.C. Merkle, M.E. Hellman, *On the security of multiple encryption*, Communications of the ACM, Vol. 24, No. 7 (1981).

[15] National Institure of Standards and Technology, *Data Encryption Standard (DES)*, FIPS PUB 46-3, Federal Information Processing Standards Publication, Reaffirmed 1999 October 25, US Department of Commerce.

[16] P.C. van Oorschot, M.J. Wiener, *A known-plaintext attack on two-key triple encryption*, Eurocrypt '90, Springer LNCS 473, 318–325.

[17] P.C. van Oorschot, M.J. Wiener, *Improving implementable meet-in-the-middle attacks by orders of magnitude*, Crypto '96, Springer LNCS 1109, 229–236.

[18] M. Smid, E. Roback, J. Foti, 'AES Workshop. To Discuss the AES Evaluation Criteria and Submission Requirements.' National Institute of Standards and Technology, April 15, 1997.
Online: `http://csrc.nist.gov/encryption/aes/pre-round1/earlyaes.htm`

[19] R. Outerbridge, private communication.

[20] S. Vaudenay, 'On Comparing the Security of Block Ciphers', manuscript, 1998.

[21] R. Weis, W. Effelsberg, S. Lucks, 'Remotely Keyed Encryption with Java Cards: A Secure and Efficient Method to Encrypt Multimedia Streams', IEEE International Conference on Multimedia and Expo (2000).

Appendix: The Security of Frugal FX

The cipher FX′ is defined by

$$\mathrm{FX}'_{L,M}(P) = F_L(P \oplus M) \oplus M.$$

Here, we provide a sketch of proof for the security of FX′. By 'A', we denote an adversary. We define two experiments:

Experiment R The oracle realises $2^\alpha+1$ independently chosen random permutations $F_k, E : \{0,1\}^\beta \longrightarrow \{0,1\}^\beta$, with $k \in \{0,1\}^\alpha$.

Experiment X The oracle realises 2^α independently chosen random permutations F_k with $k \in \{0,1\}^\alpha$, randomly chooses the key (L, M) in $\{0,1\}^{\alpha+\beta}$, and defines the permutation E by

$$E(P) = F_L(P \oplus M) \oplus M. \tag{A.1}$$

After making her queries, A responds one bit $A(\cdot) \in \{0,1\}$. Experiment R corresponds to F being a random permutation, experiment X corresponds to F being a permutation of the FX′-type. By $\mathrm{pr}_{\mathrm{R}}(A)$, and $\mathrm{pr}_{\mathrm{X}}(A)$, we denote the probabilities that A responds a 1 under the conditions of experiments R and experiment X:

$$\mathrm{pr}_{\mathrm{R}}(A) = \mathrm{Prob}[A(R) = 1] \quad \text{and} \quad \mathrm{pr}_{\mathrm{X}}(A) = \mathrm{Prob}[A(X) = 1].$$

Then the advantage adv_A is $\mathrm{adv}_A = |\mathrm{pr}_{\mathrm{R}}(A) - \mathrm{pr}_{\mathrm{X}}(A)|$.

Kilian and Rogaway [7] mention FX′ and point out that it is as secure as FX itself. Actually, their proof regarding FX works for FX′, as well. But for the sake of self-containment, and since we suggest to use FX′ for key set-up purposes, we prove the following on our own:

Theorem A1 *If A is (e,f)-restricted,* $\mathrm{adv}_A \leq 2ef/2^{\alpha+\beta}$.

It is easy to simulate experiment R: Simulate $2^{\alpha}+1$ random permutations. A's chance to respond 1 is $\mathrm{pr}_{\mathrm{R}}(A)$. For experiment X we simulate 2^{α} random permutations F_k. Given the key $(L,M) \in \{0,1\}^{\alpha+\beta}$, the treatment of oracle queries (E,P) and $E^{-1}(C)$ is described below. A responds 1 with the probability $\mathrm{pr}_{\mathrm{X}}(A)$.

(E,P): $X := P \oplus M$
$Y := F_L(X)$
(∗ oracle query (F_L, X) ∗)
$C := Y \oplus M$

(E^{-1},C): $Y := C \oplus M$
$X := F_L^{-1}(Y)$
(∗ oracle query (F_L^{-1}, Y) ∗)
$P := X \oplus M$

Now we define a third experiment by running experiment R and then producing an output value bad $\in \{0,1\}$, invisible for A:

Experiment X* Do the following:

1 Run experiment R.
2 Choose a random key $(L,M) \in \{0,1\}^{\alpha+\beta}$.
3 Run the badness-test described in Figure A.1 and output bad.

By 'Prob[bad = 1]' we denote the probability that running experiment X* results in the simulator to output 'bad = 1'.

Claim 1 $\mathrm{adv}_A \leq \mathrm{Prob}[\,\mathrm{bad} = 1\,]$.

Proof. For A, the experiments R and X* are the same, since A does not know the value of bad. Given a key (L,M), compare the simulation of experiment X and experiment X*. Except when Condition (A.1) is violated, a query $(E^{\pm 1}, b)$ and its response define a relationship

$$E(P) = C \text{ with } X = P \oplus M,\ Y = C \oplus M,\ \text{and } F_L(X) = Y. \qquad \text{(A.2)}$$

```
bad := 0;
for all P ∈ def(E) do C := E(P);  X := P ⊕ M;  Y := C ⊕ M;
                      if X ∈ def(E) then if F_K(X) ≠ Y then bad := 1;
                                                end; (* if *)
                                         else  if Y ∈ def(F_K^{-1})
                                                 then bad := 1;
                                                 else define(F_K(X) = Y);
                                                end; (* if *)
                      end; (* if *)
end; (* for *)
```

Figure A.1 Badness Test for experiment X*

When simulating experiment X, relationship (A.2) is always valid – if necessary, additional oracle queries $(F_L^{\pm 1}, b')$ are asked. Experiment X* does not make such an additional oracle query, but it can by chance behave just as if it did. For every pair (P, C) where $E(P) = C$ is defined, we may compute the corresponding values $X = P \oplus M$ and $Y = C \oplus M$, and verify relationship (A.2). We distinguish:

1 $X \in \mathrm{def}(F_L)$: relationship (A.2) is violated if $X \oplus M \neq Y$.

2 $X \in \mathrm{free}(F_L)$: relationship (A.2) requires $F(X) = Y$. If $F_L^{-1}(Y)$ is defined before, then $F_L^{-1}(Y) = X' \neq X$, and (A.2) is violated.

The above describes what the badness-test does. I.e., if relationship (A.2) is violated, experiment X* outputs 'bad = 1'.

Thus, if 'bad = 0', A can't find out that her oracle is defined by experiment X* instead of experiment X. Anyway, X* and R are the same for A. Hence $\mathrm{adv}_A = |\mathrm{pr}_{\mathrm{X}}(A) - \mathrm{pr}_{\mathrm{R}}(A)| \leq \mathrm{Prob}[\mathrm{bad} = 1]$. □

Claim 2 $\mathrm{Prob}[\mathrm{bad} = 1] \leq 2ef/2^{\alpha+\beta}$.

Sketch of proof. Consider all ef pairs of oracle queries $(E^{\pm 1}, b)$, $(F_K^{\pm 1}, c)$. Every such pair defines values P, C, K, X, Y with $E(X) = Y$ and $F_K(X) = Y$. For every key $(L, M) \in \{0,1\}^{\alpha+\beta}$, relationship (A.2) can only be violated if $K = L$ and ($X = P \oplus M$ or $Y = C \oplus M$). Hence for every pair of queries, there are at most two keys (L, M) which induce a violation of relationship (A.2). With ef pairs of queries, $\mathrm{Prob}[\mathrm{bad} = 1] \leq 2ef/2^{\alpha+\beta}$.

□

The Claims 1 and 2 prove Theorem A1.

Remark. Due to the DES complementation property, DESX' (and DESX) effectively loose one bit of their formal key length. So we count the DES key size as $\alpha = 55$, while its block size is $\beta = 64$, as usual.