# A Universal Encryption Standard

[Published in H. Heys and C. Adams, Eds., *Selected Areas in Cryptography*, vol. 1758 of *Lecture Notes in Computer Science*, pp. 1–12, Springer-Verlag, 2000.]

Helena Handschuh and Serge Vaudenay

[1] Gemplus – ENST
`handschuh@gemplus.com`
[2] École Normale Supérieure – CNRS
`Serge.Vaudenay@dmi.ens.fr`

**Abstract.** DES and triple-DES are two well-known and popular encryption algorithms, but they both have the same drawback: their block size is limited to 64 bits. While the cryptographic community is working hard to select and evaluate candidates and finalists for the AES (Advanced Encryption Standard) contest launched by NIST in 1997, it might be of interest to propose a secure and simple double block-length encryption algorithm. More than in terms of key length and block size, our Universal Encryption Standard is a new construction that remains totally compliant with DES and triple-DES specifications as well as with AES requirements.

## 1 Introduction

For many years, DES [9] has been used as a worldwide encryption standard. But as technology improved for specialized key-search machines [26, 8], its 56-bit key size became too short, and a replacement was needed. 2-key triple-DES has since become the traditional block cipher used both by the cryptographic community as well as industry. However, there is a second drawback to DES which is also the case for triple-DES: its 64-bit block size. Therefore NIST launched a contest to select and evaluate candidates for a new encryption standard, the AES, in late 1997 [1]. The basic requirements for this new algorithm were that it be at least as secure and fast as triple-DES, but that its block size be of 128 bits instead of 64, and that its key size take possible values of 128, 192 and 256 bits.

Meanwhile, people are still using DES and triple-DES, and may want to start developping applications where these two as well as the new AES may independently be used as the encryption components. In order to be compliant with DES and triple-DES, we propose a new construction which is based on these building blocks, but which can take AES specifications as a requirement for its key and block sizes. Therefore, when AES is finally selected, it will come as a natural plug-in replacement of the actual structure whithout anybody being forced to change input and output interfaces.

We notice that double block-length encryption primitives based on DES already exist: as an example, take DEAL, which uses DES as the round function in a traditional 6-round Feistel scheme [16]. One can also think of multiple modes with two blocks, where DES is the underlying cipher [10], but except for two-key triple DES in outer CBC mode which is vulnerable to dictionary and matching ciphertext attacks, none of these constructions are backward compliant with DES and triple-DES, nore do they make use of the full strength of a 128-bit block size (the second half of the plaintext never influences the first half of the ciphertext). Furthermore, multiple modes are either insecure [3–6] or require confidentiality or integrity protected initial values [25, 11]. We are also aware of the attacks by Lucks on 3-key triple DES [18] and DEAL [19].

The rest of the paper is organized as follows: section 2 presents our new encryption standard. Sections 3 and 4 provide details on collision attacks when some of the components of our UES are cut out. Section 5 provides additional security arguments on our construction and evaluates its strength based on the FX construction. Finally, we argue why we believe our construction is sound.

## 2 A Universal Encryption Standard

In this section we give the specifications of our new double block-length encryption algorithm. It basically runs two triple-DES encryptions in parallel and exchanges some of the bits of both halves inbetween each of the three encryption layers. Note that Outerbridge proposed a similar idea [21]. We investigated several related constructions and decided to add pre and post-whitening with extra keys, as well as an additional layer where bits of the left and the right half of the scheme are swapped under control of the extended secret key. Justification for these final choices will be given throughout this paper. The key schedule is considered to be the same as DEAL's.

### 2.1 Notations

We use the following notations for our scheme as well as for the attacks presented in the next sections (all operations are on bitstrings):

$a|b$ : concatenation of $a$ and $b$
$a \oplus b$ : bitwise "exclusive or" of $a$ and $b$
$a \wedge b$ : bitwise "and" of $a$ and $b$
$\overline{a}$ : bitwise 1-complement of $a$
$\texttt{001110100111}_{\mathrm{b}}$ : bitstring in binary notation
$\texttt{3a7}_{\mathrm{x}}$ : bitstring in hexadecimal notation with implicit length (multiple of four)

In addition we let $\mathrm{DES}_k(x)$ denote the DES encryption of a 64-bit block $x$ by using a 56-bit key $k$, and we let $3\mathrm{DES}_{k_1,k_2}(x)$ denote the 2-key triple-DES encryption of $x$ in EDE mode (Encryption followed by Decryption followed by Encryption) , *i.e.*

$$3\mathrm{DES}_{k_1,k_2}(x) = \mathrm{DES}_{k_1}\left(\mathrm{DES}_{k_2}^{-1}\left(\mathrm{DES}_{k_1}(x)\right)\right).$$

### 2.2 Basic building blocks

We already mentioned that we use parallel 3DES as well as a kind of keyed swap. In order to further formalize our proposal, let us define the following three basic building blocks which refer to operations on 128-bit strings. For convenience, we split a 128-bit string $x$ into two 64-bit halves $x_h$ and $x_l$.

1. **Keyed Translation.** Let $k = k_h|k_l$ be a 128-bit string. We define
$$T_k(x) = x \oplus k.$$
2. **Keyed Swap.** Let $k$ be a 64-bit string. We define
$$S_k(x) = (x_h \oplus u)|(x_l \oplus u)$$
where $u = (x_h \oplus x_l) \wedge k$. This actually consists of exchanging the bits which are masked by $k$ in the two halves.
3. **Parallel Encryption.** Let $k = k_h|k_l$ be two concatenated keys for two keyed algorithms $C$ and $C'$. We define
$$P_{k,C,C'}(x) = C_{k_h}(x_h)|C'_{k_l}(x_l).$$

Our algorithm is a combination of three rounds of products of these transformations with additional operations before the first and after the last encryption layer.

### 2.3 Our new DES and 3DES-compliant construction

Having defined the above components, let $m = \texttt{00000000ffffffff}_{\mathrm{x}}$, and let $k' = k_1|k_2|k_3|k_4$ and $m' = m_1|m_2|m_3|m_4$ be respectively two 256-bit extended keys derived from $k$ by the key schedule.

**Definition 1.**

$$\mathrm{UES}^*_k = P_{k_1|k_3,\mathrm{DES},\mathrm{DES}} \circ S_m \circ P_{k_2|k_4,\mathrm{DES}^{-1},\mathrm{DES}^{-1}} \circ S_m \circ P_{k_1|k_3,\mathrm{DES},\mathrm{DES}}$$

See figure 1. Then the precise formula to encrypt a plaintext under key $k$ using UES reads as follows:

**Definition 2.**

$$\mathrm{UES}_k = S_{m_4} \circ T_{m_3|m_3} \circ \mathrm{UES}^*_k \circ T_{m_2|m_2} \circ S_{m_1}$$

See figure 2. This algorithm has two interesting properties. Namely if we set $m' = 0$ and $k' = k$, we have

*Property 1.*

$$\mathrm{UES}_{k_1|k_2|k_1|k_2}(x_l|x_l) = \mathrm{UES}^*_{k_1|k_2|k_1|k_2}(x_l|x_l) = 3\mathrm{DES}_{k_1,k_2}(x_l)|3\mathrm{DES}_{k_1,k_2}(x_l)$$

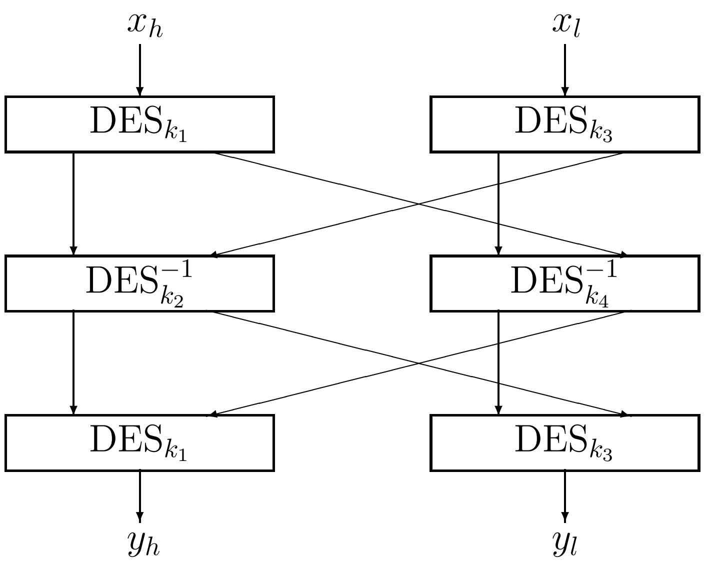

**Fig. 1.** UES*: Double-block length parallel triple DES

and

*Property 2.*

$$\mathrm{UES}_{k_1|k_1|k_1|k_1}(x_l|x_l) = \mathrm{UES}^*_{k_1|k_1|k_1|k_1}(x_l|x_l) = \mathrm{DES}_{k_1}(x_l)|\mathrm{DES}_{k_1}(x_l).$$

In addition it operates on 128-bit block messages. This makes the algorithm compatible with the forthcoming AES, and usable in DES or triple-DES mode. Finally, if we set $m = 0$, we can even run two full DES or 3DES encryptions in parallel, which doubles the encryption speed (two blocks are encrypted applying UES$^*$ only once).

Note that this scheme enables to construct double block-length encryption algorithms no matter what the underlying cipher is. For simplicity throughout this paper we will consider DES, but any other secure 64-bit block cipher could do the job. We will also focus on generic attacks that do not exploit the internal structure of the component encryption algorithm. Specific attacks such as differential [7] or linear cryptanalysis [20], truncated or higher order differentials [15] do not apply in this context as at least three layers of basic encryption are applied. We also believe that the best way to attack the scheme by a generic method is to try to create inner collisions.

## 2.4 The key-schedule

In Table 1 below, we summarize in which different modes UES may be used.

| Mode<br>Key size | DES<br>56 | 3DES<br>112 | AES<br>128/192/256 |
|---|---|---|---|
| Block size<br>64 bits | $k' = k\|k\|k\|k$<br>$m' = 0,\ m = 0$ | $k' = k\|k$<br>$m' = 0,\ m = 0$ | - |
| Block size<br>128 bits | $k' = k\|k\|k\|k$<br>$m' = 0, x_h = x_l$ | $k' = k\|k$<br>$m' = 0, x_h = x_l$ | $k' = k_1\|k_2\|k_3\|k_4$<br>$m' = m_1\|m_2\|m_3\|m_4$ |

**Table 1.** Key-schedule for DES, 3DES and AES modes

The four subkeys and the four submasks used in AES-mode are derived from the user key using DEAL's key-schedule (for a 256-bit key). The user key is first divided into $s$ subkeys of 64 bits each for $s = 2, 3, 4$.Then expand these $s$ keys to 8 keys by repetition and exor the keys with a new constant for every repetition. Encrypt the expanded list of keys using DES in CBC mode with a fixed key $K$ = `0123456789abcdef`$_\mathrm{x}$ and with the initial value set to zero. In order to partially allow on the fly key generation, start by deriving $m_1$ and $m_2$, next derive the four DES keys forming $k'$, and finally derive $m_3$ and $m_4$.

We are aware of Kelsey and Schneier's [13] key-schedule cryptanalysis of DEAL. It turns out UES may have a very small class of equivalent keys in the 192-bit key case, because of the use of 56-bit keys for the inner DES blocks,

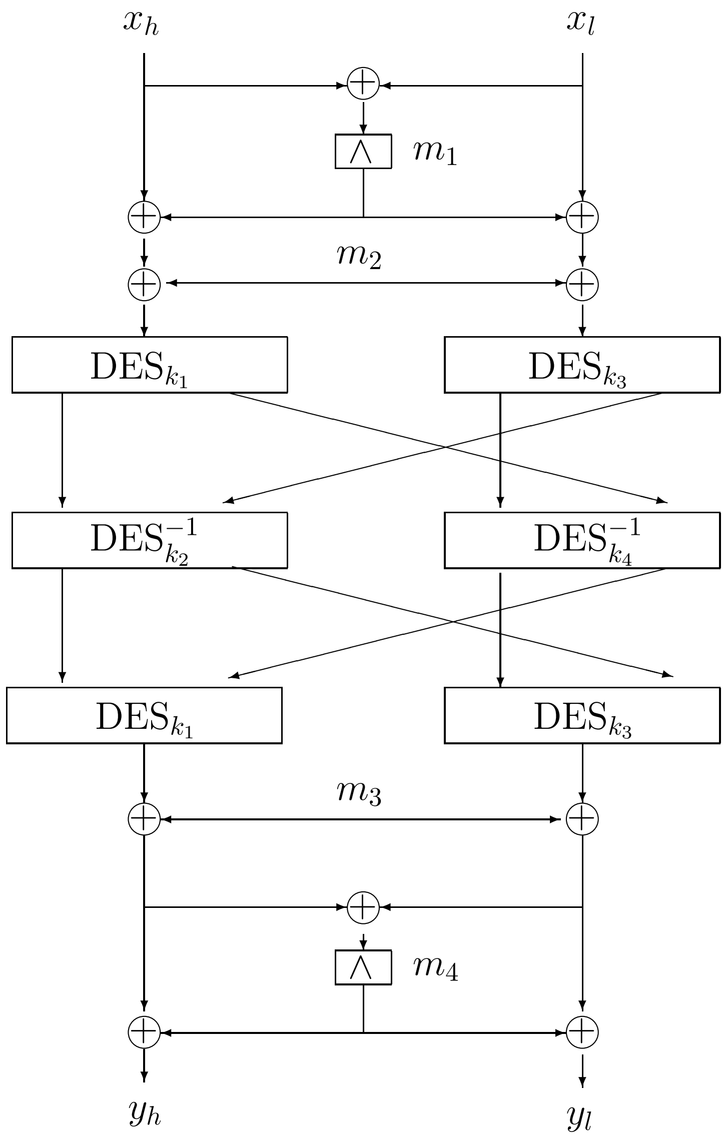

**Fig. 2.** Encryption with UES.

whereas 64 bit subkeys are generated by the key-schedule. We also worked out a similar related-key attack with John Kelsey, which recovers the keys in complexity $2^{64}$ using $2^{33}$ related keys. However, these attacks apply in a very limited number of practical settings. Developpers should still make sure an attacker is not allowed to choose the keys in such a way.

## 3 Collision Attacks on Parallel DES

In this section, we consider the variant of UES previously defined as:

$$\mathrm{UES}^*_k = P_{k_1|k_3,\mathrm{DES},\mathrm{DES}} \circ S_m \circ P_{k_2|k_4,\mathrm{DES}^{-1},\mathrm{DES}^{-1}} \circ S_m \circ P_{k_1|k_3,\mathrm{DES},\mathrm{DES}}$$

We will show that this straightforward way of doubling the block size is not secure because a collision attack can be mounted against it (this phenomenon has been independently observed by Knudsen [17]). This is due to the fact that the construction is not a multipermutation. In other words, it may very well happen that if half of the input bits have a fixed value, half of the output bits also have, which would not be the case if the multipermutation property had been satisfied [22]. However, our intention is to prove that we can nevertheless use the structure if the input and output bits to this variant are unknown to the attacker. Therefore we begin by showing where the problem comes from, and justify our additional layers of swapping and masking in the final version of UES.

### 3.1 Public intermediate swapping

We first show how to break UES$^*$ by recovering the secret key with about $2^{34}$ chosen plaintexts, $2^{59}$ DES operations, and a memory of 16GB. The attack consists in the following steps.

**Step 1.** First fix $x_h = a$ to a constant and try many $x_l = u_i$ values for $i = 1, \ldots, n$. Request $y_i|z_i = \mathrm{UES}^*(a|u_i)$.

**Step 2.** For any collision $y_i = y_j$, guess that this comes from collisions on the two inputs of the second and third internal DES higher permutations (these are called "good collisions"). The expected number of good collisions is $n^2.2^{-65}$, and the expected number of "natural" (bad) collisions is the same. Try all possible $k_3$ until there is a collision on both

$$\mathrm{DES}_{k_3}(x_i) \wedge m = \mathrm{DES}_{k_3}(x_j) \wedge m$$

and

$$\mathrm{DES}^{-1}_{k_3}(z_i) \wedge m = \mathrm{DES}^{-1}_{k_3}(z_j) \wedge m.$$

Note that a single (good) collision will always suggest the good $k_3$ value and an expected number of $2^{-8}$ random ones, and a bad collision will suggest

$2^{-8}$ random values on average. It is thus likely that we get the $k_3$ key once a $k_3$ value is suggested twice (namely with a confidence of $2^{72} : 1$). We thus need only two good collisions. This requires $n \approx 2^{33}$.

**Step 3.** Perform a similar attack on $k_1$.

**Step 4.** Recover $k_2$, then $k_4$ by exhaustive search.

This shows that this algorithm is just a little more secure than DES, and far less secure than triple-DES. We add that Bart Preneel pointed out to us that in Step 1, a collision on the other half of the ciphertext occurs with the same probability, therefore we get an extra condition satisfied by $k_3$ as well as $k_1$ from the same number of chosen plaintexts. This slightly decreases the number of required chosen plaintexts.

As a matter of fact, the previous attack holds whenever $m$ is any other public value. Namely let $w$ denote the Hamming weight of $m$. Without loss of generality, let us assume that $w \leq 32$ (otherwise, let us consider the lower $\mathrm{DES}^{-1}$ operation). In the attack above, the number of expected good collisions is $n^2.2^{-2w-1}$, and the number of bad collisions is $n^2.2^{-65}$. The attack thus still holds but with a complexity of $n \approx 2^{w+1}$. In general, the complexity is thus $n \approx 2^{33-|32-w|}$. Actually, the complexity is the highest for UES*, because $m$ has a balanced Hamming weight. (Note that this analysis does not hold if $m = 0$ or $m = \overline{0^{\ell=64}}$ for which we have two triple-DES in parallel, and no possible collision.)

## 3.2 Keyed intermediate swapping

At first sight, one might think that introducing a keyed inner swap significantly increases the complexity of the attack. However this is not the case. Let us show why.

If $m$ is a part of the key, we cannot exhaustively search for $k_3$ because we do not know $m$. In the worst case ($w = 32$) we have 2 good collisions and 2 bad ones. Let us assume we have guesses which are the good ones (this may lead to an overhead factor of 16). For each possible $k_3$ we can look on which bits $\mathrm{DES}_{k_3}(x_i)$ and $\mathrm{DES}_{k_3}(x_j)$ collide, as well as $\mathrm{DES}_{k_3}^{-1}(z_i)$ and $\mathrm{DES}_{k_3}^{-1}(z_j)$. For the good $k_3$ we will find $w + \frac{64-w}{4}$ bits on average with a standard deviation of $\sqrt{(64-w)\frac{3}{16}}$. For a random $k_3$ we will find $16 \pm 2\sqrt{3}$ bits (which means that 16 is the average and $2\sqrt{3}$ the standard deviation). In order to simplify the analysis, let us consider the worst case where $w = 32$. So for the right key $k_3$ and for a good collision, the number of colliding bits is $40 \pm \sqrt{6}$.

Now for each possible $k_3$ count the collisions that have say more than $t = 30$ bits as they will much more likely result from the right key value rather than from a random key value. Then the key guess associated to the most such collisions will be the right one with high probability.

Let

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt. \tag{1}$$

Then as a matter of fact, the average number $C_t$ of collisions on more than $t$ bits is

$$C_t = n^2.2^{-65}(1-\varphi) \pm n.2^{-32.5}\sqrt{\varphi(1-\varphi)} \quad (2)$$

where $\varphi = \varphi_1 = \varphi\left(\frac{t-16}{2\sqrt{3}}\right) \approx 1 - 10^{-4.6}$ for $t = 30$ in case of a random key value, and $\varphi = \varphi_2 = \varphi\left(\frac{t-40}{\sqrt{6}}\right) \approx 2^{-15.8}$ for the correct key guess.

So for $n = 2^{34}$ the average number of such collisions is $2^{-5.6}$ for a wrong guess, whereas it is about 4 for the right key.

As a result of this enhaced collision attack, we chose not to use any keyed inner swapping as it unnecessarily complicates the design (also more key materiel is needed) without significantly increasing its security. Instead of this, we chose to add the features described hereafter.

## 4 Introducing Pre- and Post-Whitening

The scheme of the previous section is compliant with DES and 3DES, but is not secure enough against key recovery attacks. So the next most straightforward idea is to protect against the exhaustive key search (once a collision is found) by adding whitening keys before and after the current structure. This considerably increases the work factor and derives from a principle discussed in the construction of DESX [14].

Let us define this new variant of UES by:

**Definition 3.**

$$\text{UES}_k^{**} = T_{m_3|m_3} \circ \text{UES}_k^{*} \circ T_{m_2|m_2}$$

The complexity of exhaustive key search increases to about $2^{56+64} = 2^{120}$ offline encryptions given one single collision due to the DESX phenomenon (other trade-offs can be achieved if more than one collision is available). However, this variant can still be distinguished from a random permutation by the previous attack because collisions are far more likely in this setting (they occur twice more often). Note that a collision on one half of the output of UES$^*$ leads to a collision on the same half of the ciphertext because the value $m_3$ is kept constant. Therefore with the same complexity as the collision attack, this second variant may be distinguished from a random permutation, which is not a desirable feature.

## 5 On the Importance of Pre- and Post-Swapping

Having solved the key recovery problem, we still face the distinguisher problem. Therefore the next and last step towards our final UES is to add yet another layer of swapping, but this time under control of the secret key. We will first show how the addition of $m_4$ increases the complexity of the collision attack, and next which additional workfactor is introduced by $m_1$.

### 5.1 Keyed swap of the output

Adding a keyed post-swapping, our current structure becomes:

**Definition 4.**

$$\mathrm{UES}_k^{***} = S_{m4} \circ \mathrm{UES}^{**} = S_{m4} \circ T_{m3|m_3} \circ \mathrm{UES}^{*} \circ T_{m_2|m_2}$$

In order to build the same attack as the distinguisher of the previous section (key recovery is hopeless by now), we need to create collisions on one half of the ouput. However, this time these collisions are much harder to spot, as we do not know which output bits correspond say to the left half of the output of the butterfly structure.

Nevertheless, we can still use a property of $S_{m_4}$. Namely, if $x_h = x'_h$, we let $\Delta = S_{m_4}(x) \oplus S_{m_4}(x')$ and it holds that:

1. $\Delta_h \wedge \Delta_l = 0$
2. $(\Delta_h)_i \leq (m_4)_i$
3. $(\Delta_l)_i \leq (\overline{m_4})_i$

for any bit $i = 1, \ldots, 64$.

*Proof.* Let $y_h = (S_{m_4}(x))_h$, $y_l = (S_{m_4}(x))_l$, $y'_h = (S_{m_4}(x'))_h$, $y'_l = (S_{m_4}(x'))_l$. Then we have the following results:

1. When $x_h = x'_h$, the following relations hold: $y_h \oplus y'_h = (x_l \wedge m_4) \oplus (x'_l \wedge m_4) = (x_l \oplus x'_l) \wedge m_4$
   $y_l \oplus y'_l = (x_l \wedge m_4) \oplus (x'_l \wedge m_4) \oplus x_l \oplus x'_l = (x_l \oplus x'_l) \wedge \overline{m_4}$
   Thus
   $\Delta_h \wedge \Delta_l = ((x_l \oplus x'_l) \wedge m_4) \wedge ((x_l \oplus x'_l) \wedge \overline{m_4}) = 0$
2. $\Delta_h = (x_l \oplus x'_l) \wedge m_4$ so when $(m_4)_i = 0$, $(\Delta_h)_i = 0$ and when $(m_4)_i = 1$, $(\Delta_h)_i = 0$ or 1. The result follows.
3. The third point is symmetric to the second.

□

Given the above properties, the attack consists in the following steps.

**Step 1.** First fix $x_h = a$ to a constant and try many $x_l = u_i$ values for $i = 1, \ldots, n$. Request $y_i = \mathrm{UES}^{***}(a|u_i)$.

**Step 2.** For all $(i, j)$ pairs, let $y_i \oplus y_j = z|t$. If we have $z \wedge t = 0$, guess that we have a collision as in the above attack. Guess that $m_4$ is an intermediate mask between $z$ and $t$. (If $z_i = \mathtt{1}$, then $(m_4)_i = \mathtt{1}$, and if $t_i = \mathtt{1}$, then $(m_4)_i = \mathtt{0}$.) Thus the expected number of good events (the signal) is $n^2.2^{-2w-1}$ and the number of bad events (the noise) is $\frac{n^2}{2}.\left(\frac{3}{4}\right)^{64} \approx n^2.2^{-27.5}$. One good event suggests 2/3 of the bits of $m_4$ on average. One bad event also suggests 2/3 of the bits of $m_4$ on average, but in a random way so that we can check for consistency. It is thus likely that we recover the right mask $m_4$ within four good events (mask bits suggested by bad events will happen to be inconsistent with the good ones with high probability).

**Step 3.** From $m_4$ and the above collisions, apply the same attack as before to distinguish UES$^{***}$ from a random permutation.

Since we need four good events to occur, in the worst case ($w = 32$) we need $n \approx 2^{33.5}$, and considering all $(i, j)$ pairs in Step 2 leads to a complexity of $2^{67}$. (These are however very simple tests, so this complexity can actually be compared to an exhaustive search for DES.) We can expect to get $\frac{n^2}{2}\left(\frac{3}{4}\right)^{64} \approx 2^{39.4}$ bad events on average. Each event suggests a pattern for $m_4$ with determined bits (0 or 1), and undetermined ones. A pair of events may thus be consistent with probability $\left(\frac{7}{9}\right)^{64} \approx 2^{-23.2}$. We can thus expect to find $2^{77.8}.2^{-23.2} = 2^{54.6}$ consistent pairs of bad events. More generally speaking, we are looking for multiple events in which each pair is consistent with a unique mask $m_4$. This is the same problem as seeking $k$-cliques in the consistency graph of the bad events. Any $k$-clique will be consistent with probability $\left(\frac{2^{k+1}-1}{3^k}\right)^{64}$. And there are exactly $\frac{n^k}{k!}$ such cliques. Therefore, when $k$ gets larger ($k \geq 11$), no $k$-clique in the consistency graph will survive this filtering process. The complexity of this algorithm is subject to combinatorial optimizations, and we believe that the bottleneck complexity will actually come from the exhaustive search of DES keys.

## 5.2 Keyed swap of the intput

Taking into account what we just saw, our final construction must make it as hard as possible for the attacker to find the required $2^{34}$ different values entering say the right half of the structure. Therefore all we have to do is make it hard to find 34 bits entering the right half (else the attacker tries all the values of these 34 bits and keeps the rest constant which leads to the above result.

Adding a final extra layer of keyed swapping on input to the structure will lead to this result. The attacker now has to guess 34 bits that enter one half in order to subsequently attack $m_4$ and then be able to distinguish UES from a random permutation. The extra work factor is thus $\binom{128}{34}/\binom{64}{34}$ which is $2^{43}$. Then the total complexity is $2^{43+35} = 2^{78}$.

We believe this security level is acceptable.

## 5.3 Other alternatives

If the security level is still a concern to some people, one might also consider replacing the keyed outer swapping by a keyed permutation at the bit level. This will add a bit more complexity again. The attacker will have yet a harder time finding which 35 bits enter say the left part of the structure. However, bit permutations are very costly in terms of speed, therefore this alternative shall only be considered if execution time is not that much an issue.

We also considered byte permutations, but these are far too trivial to attack. The overhead complexity is only about $2^{12}$.

Our final construction is therefore UES with keyed outer swapping and whitening ”à la DESX”.

## 6 Conclusion

We have investigated several variants of a double-block length encryption scheme based on DES which is compliant with DES, 3DES as well as with AES specifications. This may be useful for applications where DES or 3DES are still in use, but where people start to think about double block length and key sizes. Once the final AES is chosen and becomes a standard, it can be plugged into applications in place of our scheme with ease. Among several variants, we selected the best one in terms of security and simplicity, and showed that there is no practical attack that can endanger our scheme. Key recovery does not seem possible and in order to distinguish this new cipher from a random permutation, the workload is basically very high.

## References

1. http://www.nist.gov/aes
2. ANSI draft X9.52, *"Triple Data Encryption Algorithm Modes of Operation," Revision 6.0,* 1996.
3. E. Biham, "On modes of operation," *Fast Software Encryption'93, LNCS 809,* Springer-Verlag, 1994, pp. 116–120.
4. E. Biham, "Cryptanalysis of multiple modes of operation," *ASIACRYPT'94, LNCS 917,* Springer-Verlag, 1994, pp. 278–292.
5. E. Biham, "Cryptanalysis of triple-modes of operation," *Technion Technical Report CS0885,* 1996.
6. E. Biham, L. R. Knudsen, "Cryptanalysis of the ANSI X9.52 CBCM mode," *EUROCRYPT'98, LNCS 1403,* Springer-Verlag, 1998, pp. 100–111.
7. E. Biham, A. Shamir, *"Differential Cryptanalysis of the Data Encryption Standard,"* Springer-Verlag, 1993.
8. The Electronic Frontier Foundation, *"Cracking DES. Secrets of Encryption Research, Wiretap Politics & Chip Design,"* O'Reilly, May 1998.
9. FIPS 46, *"Data Encryption Standard,"* US Department of Commerce, National Bureau of Standards, 1977 (revised as FIPS 46–1:1988; FIPS 46–2:1993).
10. FIPS 81, *"DES Modes of Operation,"* US Department of Commerce, National Bureau of Standards, 1980.
11. H. Handschuh, B. Preneel, "On the security of double and 2-key triple modes of operation", *Fast Software Encryption'99,* Springer-Verlag, 1999, pp. 215–230.
12. B. S. Kaliski, M. J. B. Robshaw, "Multiple encryption: Weighing security and performance," *Dr. Dobb's Journal*, January 1996, pp. 123–127.
13. J. Kelsey, B. Schneier, "Key-Schedule Cryptanalysis of DEAL," in these proceedings.
14. J. Kilian, P. Rogaway, "How to protect DES against exhaustive key search, *CRYPTO'96, LNCS 1109,* Springer-Verlag, 1996, pp. 252–267.
15. L. R. Knudsen, *"Block Ciphers – Analysis, Design and Applications,"* PhD thesis, Aarhus University, Denmark, 1994.
16. L. R. Knudsen, *"DEAL: a 128-bit block cipher,"* AES submission, 1998.
17. L. Knudsen, "On Expanding the Block Length of DES," unpublished manuscript, nov. 98.
18. S. Lucks, "Attacking triple encryption," *Fast Software Encryption'98, LNCS 1372,* Springer-Verlag, 1998, pp. 239–253.

19. S. Lucks, “On the security of the 128-bit block cipher DEAL,” preprint, 1998.
20. M. Matsui, “Linear cryptanalysis method for DES cipher,” *EUROCRYPT’93, LNCS 765,* Springer-Verlag, 1993, pp. 386–397.
21. B. Schneier, *Applied Cryptography*, Wiley & Sons, 1995, pp. 364.
22. S. Vaudenay, ”On the need for multipermutations: Cryptanalysis of MD4 and SAFER,” *Fast Software Encryption 2, LNCS 1008,* Springer-Verlag, 1995, pp. 286–297.
23. P. C. van Oorschot, M. J. Wiener, “A known-plaintext attack on two-key triple encryption,” *EUROCRYPT’90, LNCS 473,* 1990, pp. 318–325.
24. P. C. van Oorschot, M. J. Wiener, “Improving implementable meet-in-the-middle attacks by orders of magnitude,” *CRYPTO’96, LNCS 1109*, 1996, pp. 229–236.
25. D. Wagner, “Cryptanalysis of some recently-proposed multiple modes of operation,” *Fast Software Encryption’98, LNCS 1372,* Springer-Verlag, 1998, pp. 254–269.
26. 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 and reprinted in W. Stallings, *Practical Cryptography for Data Internetworks,* IEEE Computer Society Press, 1996, pp. 31–79.