Block Ciphers And Cryptanalysis

Fauzan Mirza

fauzan@dcs.rhbnc.ac.uk

Department of Mathematics
Royal Holloway University of London

Abstract

This report gives a basic introduction to block cipher design and analysis. The concepts and design principles of block ciphers are explained, particularly the class of block ciphers known as Feistel ciphers. Some modern block cipher cryptanalysis methods are demonstrated by applying them to variants of a weak Feistel cipher called Simplified TEA (STEA), which is based on the Tiny Encryption Algorithm (TEA).

1 Introduction

This report gives a basic introduction to block cipher design and analysis (intended to be understandable by anyone with some knowledge of discrete mathematics). We will examine the concept of block ciphers, how they obtain their security, and the principles involved in their design. A popular class of block ciphers, known as Feistel ciphers, will be described in detail. This report also describes and demonstrates some modern block cipher cryptanalysis methods by showing how they may be applied to variants of a specially-designed weak block cipher, which is loosely modelled on the Tiny Encryption Algorithm (TEA). The weaker variants of TEA are called Simplified TEA (STEA).

Section 2 is a basic introduction to block cipher design, summarising the types of attacks that a strong cipher should be able to resist, describing the concepts of confusion and diffusion, and explaining the principle of Feistel ciphers. In section 3, the TEA block cipher is described and its only known weakness is explained. In section 4, the STEA block cipher is introduced. Various methods of attacking STEA, including a very efficient known-plaintext attack and some general block cipher cryptanalysis methods are described in sections 5–10. Finally, acknowledgements are in section 11.

Preliminaries

This report assumes the following notation for some binary operations.

Exclusive-OR The operation of addition of n-tuples over the field $\mathbb{F}_2$ (also known as exclusive-or) is denoted by $x \oplus y$.

Integer Addition The operation of integer addition modulo 2^n is denoted by $x \boxplus y$ (where $x, y \in \mathbb{Z}_{2^n}$). The value of n should be clear from the context.

Integer Subtraction The operation of integer subtraction modulo 2^n is denoted by $x \boxminus y$ (where $x, y \in \mathbb{Z}_{2^n}$). The value of n should be clear from the context. Also, note that $x \boxminus y \equiv x \boxplus -y$.

Bitwise Shifts The logical left shift of x by y bits is denoted by $x \ll y$. The logical right shift of x by y bits is denoted by $x \gg y$.

Bitwise Rotations A left rotation of x by y bits is denoted by $x \lll y$. A right rotation of x by y bits is denoted by $x \ggg y$.

Hexadecimal numbers will be subscripted with 'H', e.g., $20_{\mathsf{H}} = 32$. The set of plaintext blocks is denoted by $\mathcal{P}$, the set of ciphertext blocks by $\mathcal{C}$ and the set of keys by $\mathcal{K}$. The message block length is m, where $|\mathcal{P}| = |\mathcal{C}| = 2^m$, and the key length is k, where $|\mathcal{K}| = 2^k$.

2 Block Cipher Design

A symmetric-key block cipher E is a function

$$E : \mathcal{P} \times \mathcal{K} \to \mathcal{C}. \tag{1}$$

Moreover, for every key $K \in \mathcal{K}$, we have functions

$$\begin{aligned} E_K &: \mathcal{P} \to \mathcal{C}, \\ D_K &: \mathcal{C} \to \mathcal{P}, \end{aligned}$$

and $D = E^{-1}$. In other words, the key determines the bijective mapping of plaintext to ciphertext; it also determines the inverse mapping of ciphertext to plaintext. If the key is fixed, and hence determines a specific mapping, the block cipher can be considered simply as a large lookup-table (substitution cipher). In particular, identical plaintext blocks encrypt to identical ciphertext blocks.

Substitution ciphers in which m is small (e.g., $m = 8$, implying 256 distinct message blocks) are insecure since if the attacker has knowledge of the statistical patterns in the plaintext, then they can attempt to recover the plaintext from the ciphertext by frequency analysis. Another problem is that if an attacker correctly guesses the plaintext that corresponds to some ciphertext then they can build a lookup-table of plaintext-ciphertext pairs corresponding to a particular key, known as a *codebook*. Codebooks can be used to decrypt ciphertext blocks without knowledge of the key.

Both problems are solved by increasing the block length, and thus increasing the number of possible messages. An attacker cannot exploit knowledge of plaintext patterns which are shorter than the block length. It also becomes impractical for an attacker to build plaintext-ciphertext codebooks. Typically the message block length m would be at least 64 bits.

We also require it to be infeasible for an attacker, with a known plaintext (P, C), to recover the key by exhaustive key-search (brute-force). That is, we do not want anyone to be able to determine the key by trying values of $K \in \mathcal{K}$ until they find the key K such that $C = E(P, K)$. By making the key length sufficiently large we can make brute-force attacks computationally infeasible (the cipher would then be *computationally secure*). The key length k should be at least 64 bits[1]

Having decided on the message block length and key length parameters, the designer must then develop the block cipher in such a way that it should be able to defeat or resist the following types of attacks, where an attacker would like to recover the key:

Ciphertext-only The attacker has knowledge of some ciphertext but not the plaintext nor the key. In this case, recovering the plaintext (without the key) may be a successful attack.

Known-plaintext The attacker has knowledge of some ciphertext and corresponding plaintext, encrypted under the unknown key

Chosen-plaintext The attacker has an opportunity to have any messages of their choosing to be encrypted under the unknown key.

[1]The Data Encryption Standard (DES) block cipher has a key length of 56 bits and was broken by brute-force in June 1997.

Adaptive chosen-plaintext The attacker has the unlimited capability to have messages of their choosing to be encrypted under the unknown key. We expect that an adaptive chosen plaintext attack would require less encryptions than a real chosen plaintext attack.

The designer follows *Kerckhoff's assumption*; that the attacker has full knowledge of the workings of the cipher system. The security of the cipher should rest entirely in the key.

2.1 Confusion And Diffusion

Except for small message block or key lengths, it is infeasible for the block cipher designer to explicitly specify a plaintext to ciphertext mapping for every possible key; this would be like specifying 2^k codebooks. It is far more practical to specify the block cipher as an equation or an algorithm. So for example, a simple block cipher with parameters $(m, k) = (64, 64)$ might be specified by

$$\begin{aligned} C &= P \oplus K, \\ P &= C \oplus K. \end{aligned} \tag{2}$$

The problem with the block cipher of equation 2 is that it is trivially broken with one known plaintext by

$$K = P \oplus C.$$

This block cipher is weak because it is purely linear and thus easily solvable. By using both linear *and* nonlinear operations we make the block cipher somewhat more difficult to manipulate by simple algebra. So for example, a slightly better idea would be the $(m, k) = (64, 128)$ block cipher,

$$\begin{aligned} C &= (P \oplus K_0) \boxplus K_1, \\ P &= (C \boxminus K_1) \oplus K_0, \end{aligned} \tag{3}$$

where K_0 and K_1 are *subkeys*, i.e., key-dependent variables. In this case, each subkey is half of the key K, and thus has length 64 bits.

To recover the key K an attacker would have to solve equation 3. As there are two unknowns, a unique solution would not be found with only one known plaintext. However, with two known plaintexts, (P, C) and (P', C'), we have equations

$$\begin{aligned} C &= (P \oplus K_0) \boxplus K_1, \\ C' &= (P' \oplus K_0) \boxplus K_1. \end{aligned}$$

The attacker could subtract (modulo 2^{32}) one ciphertext from another, obtaining

$$C \boxminus C' = (P \oplus K_0) \boxminus (P' \oplus K_0), \tag{4}$$

where the 64-bit subkey K_1 term has been eliminated. However, equation 4 cannot be manipulated further to an equation of the remaining subkey K_0 in terms of the plaintext and ciphertext blocks because the operations $\oplus$ and $\boxplus$ do not follow the distributive law. Furthermore, there is no simple algebraic relationship between addition modulo 2^m and exclusive-or of m-bit vectors since the groups $(\mathbb{Z}_{2^m}, \boxplus)$ and $(\mathbb{F}_2^m, \oplus)$ are not isomorphic [8].

The idea of mixing linear and nonlinear operations in order to obscure the relationship between the plaintext, ciphertext and key, is called *confusion* and is an important principle of cipher design [1, 8].

An equally important principle of block cipher design is that of *diffusion*, i.e., the idea that every bit of the ciphertext should depend on every bit of the plaintext *and* every bit of the key [1, 8]. This ensures that the statistics of the plaintext are dissipated within the ciphertext so that an attacker cannot predict the plaintext that corresponds to a particular ciphertext, even after observing a number of "similar" plaintexts and their corresponding ciphertexts.

A simple method of achieving confusion and diffusion in a block cipher is by repeatedly applying keyed substitutions and permutations to the message. The substitutions are used to introduce nonlinearity into the message (confusion) and the permutations are required to ensure that bits are affected by different substitutions on subsequent iterations (diffusion). A block cipher based on this principle is called an *iterated cipher*.

In practice, the designer develops a function ρ, known as the *round function*, which has inputs of a message block X and a subkey K, and outputs a partially-encrypted message block $Y = \rho(X, K)$. The round function is responsible for satisfying the basic confusion and diffusion requirements, hence we would want each bit of X and each bit of K to influence each bit of Y in a *nonlinear but invertible* manner. For example, the round function defined by

$$\rho(X, K) \quad \overset{\text{def}}{=} \quad (X \boxplus K) \lll 3 \tag{5}$$

fulfils the confusion requirement by mixing the nonlinear integer addition with the linear bitwise permutation, and the diffusion requirement by the bitwise permuatation. Note that this round function, by itself, is unsuitable as a block cipher because it is easily solvable[2].

By iterating the round function a fixed number of times we automatically obtain some security as a consequence of the fact that, after each iteration (or *round*), the output bits become more and more dependent on the input bits. We can use different subkeys in each iteration so that, for a 4-round block cipher, the encryption function becomes

$$C = E(P, K) = \rho(\rho(\rho(\rho(P, K_1), K_2), K_3), K_4).$$

Similarly, decryption would be acheived by

$$P = D(C, K) = \rho^{-1}(\rho^{-1}(\rho^{-1}(\rho^{-1}(C, K_4), K_3), K_2), K_1).$$

The subkeys would be derived from the key by the *key schedule* algorithm (which would also need to be designed carefully). Determining the number of rounds required for a suitable level of cryptographic security is not trivial. It depends on the design and security provided by the round function. Whilst choosing a large number of rounds may defeat most modern cryptanalytic attacks, it may also make the cipher undesirably slow for many practical purposes. Block cipher designers tend to fix the number of rounds after investigating the susceptibility of the cipher to various general attacks. A "conservative" number of rounds is usually chosen, which is a few rounds more than the minimum number of rounds required (i.e., the number of rounds which would cause the cipher to fall to some type of attack).

The need for a separate decryption function can be avoided by designing the round function to be an involution (i.e., $\rho^{-1} = \rho$). To decrypt, the encryption function would be applied to the ciphertext but with the subkeys inverted (with respect to whatever operation is used to combine them with the data) and the order of the subkeys reversed. Effectively,

$$P = D(C, K) \quad = \quad \rho(\rho(\rho(\rho(C, K_4^{-1}), K_3^{-1}), K_2^{-1}), K_1^{-1}).$$

This may involve some additional work in computing the decryption subkeys from the encryption subkeys (as in the case of the IDEA cipher [8, 9]), but this would be handled by the key schedule

[2]The inverse of equation 5 is given by $\rho^{-1}(Y, K) \overset{\text{def}}{=} (Y \ggg 3) \boxminus K$.

algorithm and would be transparent to the encryption function. A cipher which has this property is called *E/D similar* [8].

One problem caused by this cipher structure is that it allows for the possibility of a one-round encryption of the message followed by its one-round decryption simply by choosing bad subkeys. In particular, a round may cancel the effect of its previous round if its subkey is the inverse of the previous round subkey (i.e., if $K_i = K_{i-1}^{-1}$ for round i). If this occurs then two rounds are wasted (they would have no effect on the message block) and so this weakness may lower the overall security of the block cipher. There are two ways of correcting this design. First, by designing a key schedule that guarantees that any round subkey is never the inverse of its previous round subkey. Second, by composing the round function of involutions (to maintain E/D similarity), but ensuring that the round function itself is not an involution. The second option is usually preferable over the first (the cipher should be reasonably strong, independent of the key schedule).

E/D similar ciphers are "symmetrical" in their design and we can use a more convienient notation to illustrate this point by giving the function ρ an index denoting the subkey which is used in that particular round. For example,

$$\begin{aligned} C = E(P,K) &= \rho_1(\rho_2(\rho_3(\rho_4(P)))), \\ P = D(C,K) &= \rho_4(\rho_3(\rho_2(\rho_1(C)))). \end{aligned}$$

If the round function is not an involution then the final round would be followed by an *output transformation* whose purpose is to cause the cipher to be symmetrical (and hence E/D similar).

A number of modern block ciphers use the same "general" round function as the DES cipher, because it has the useful property that the encryption function is E/D similar. These are known as *Feistel ciphers.*

2.2 Feistel Ciphers

The basic principle of Feistel ciphers is that the plaintext block is split into two halves and each half is used to encrypt the other half over a predetermined number of rounds. The result is the ciphertext block composed of the two encrypted half-blocks.

The round function consists of an involutary substitution and an involutary permutation operating on two blocks, each of length $\frac{m}{2}$. The substitution part of the round function, which we denote by σ, is defined by

$$\sigma_i(L,R) = (L \oplus F(R,K_i), R),$$

where F is the *cipher function* and K_i is a subkey. We can show that σ is an involution by showing that $\sigma^2(L,R) = (L,R)$, as follows:

$$\begin{aligned} \sigma_i^2(L,R) &= \sigma_i(L \oplus F(R,K_i), R) \\ &= (L \oplus F(R,K_i) \oplus F(R,K_i), R) \\ &= (L,R). \end{aligned}$$

This also shows that the cipher function does not need to be invertible. The permutation part of the round function, which we denote by π, is defined by

$$\pi(L,R) = (R,L),$$

which is a simple "swapping" of halves. This is obviously an involution. Our goal is to make an E/D similar cipher by alternately applying the substitution σ and permutation π. As both σ and π are involutions, this is easily acheived by ensuring that the cipher ends with the same function that is applied first (hence making the cipher appear "symmetrical"). The round function of a Feistel cipher is $\rho(L,R) = \pi(\sigma(L,R))$. This is not an involution, but the cipher is E/D similar if the round substitution σ is the first and last stage of the cipher:

$$\begin{aligned} C = E(P,K) &= \sigma_n(\rho_{n-1}(\cdots(\rho_1(P))\cdots)), \\ P = D(C,K) &= \sigma_1(\rho_2(\cdots(\rho_n(C))\cdots)). \end{aligned}$$

In practice, the round function ρ is applied n times and then the output transformation π is applied to undo the π permutation in the n-th round (so the last effective operation on the message block would be the substitution part σ_n, as required).

Another point that we have to address is how the decryption subkeys are derived from the encryption subkeys. It was mentioned that the decryption subkeys would be the inverse of the operation which is used to combine them with the data. In the function σ, the key-dependent output of the cipher function is combined with the message block using the exclusive-or operation. Hence the decryption subkeys are the same as the encryption subkeys, since the inverse of any element, with respect to $\oplus$, is itself.

The only difference between encryption and decryption is the order in which the subkeys are applied, hence this construction is E/D similar.

More specifically, the plaintext is divided into two blocks, $P = (L_0, R_0)$. Then, for an n-round cipher, the following round function is iterated (for $1 \le i \le n$):

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \oplus F(R_{i-1}, K_i), \end{aligned} \tag{6}$$

where F is the cipher function and the K_i are subkeys. The ciphertext is $C = (R_n, L_n)$. Note that the order of the message halves is reversed in the output; this is due to the output transformation (which undoes the swap of the n-th round).

In general we can use any group operation $\otimes$ in place of $\oplus$ in equation 6 so that the round function could be

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \otimes F(R_{i-1}, K_i). \end{aligned}$$

The round function required for decryption would be

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \otimes (F(R_{i-1}, K_i))^{-1}, \end{aligned}$$

where it would be necessary to compute the inverse (with respect to the operation $\otimes$) of the cipher function output. Consequently, the decryption round may not be the same as the encryption round, in which case the cipher will no longer be E/D similar. Examples of ciphers which use this modified Feistel structure are the TEA and STEA block ciphers, which use integer addition modulo 2^{32} (the operation $\boxplus$) instead of exclusive-or, and consequently both TEA and STEA require separate decryption functions.

The only parts of the block cipher which are not specified by the Feistel structure are the design of the cipher function and the key schedule algorithm which determines how the subkeys $(K_1, \ldots, K_n)$ are derived from the key K.

3 The TEA Block Cipher

The TEA (Tiny Encryption Algorithm) block cipher was presented at the Fast Software Encryption workshop in 1994 [15]. It is a 64-round Feistel cipher operating on 64-bit message blocks with a 128-bit key. It was designed for software implementation and all its operations are on 32-bit words and use arithmetic and logic operations rather than using (traditional) substitution and permutation tables in the cipher function.

As mentioned previously, the round function differs slightly from the usual specification in that integer addition modulo 2^{32} is used instead of exclusive-or as the combining operator:

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \boxplus F(R_{i-1}, K_i), \end{aligned} \tag{7}$$

There is no final unswap, but this is not a problem since the decryption function is separate and deals with this. The plaintext is $P = (L_0, R_0)$ and the ciphertext is $C = (L_{64}, R_{64})$. Each subkey K_i is an ordered 3-tuple of elements in $\mathbb{Z}_{2^{32}}$, of the form (T, U, V) where (T, U) are 64 bits of the secret key and V is a known 32-bit constant. The cipher function F is defined by

$$F(M, (T, U, V)) \overset{\text{def}}{=} ((M \ll 4) \boxplus T) \oplus ((M \gg 5) \boxplus U) \oplus (M \boxplus V). \tag{8}$$

To generate the subkeys, first the 128-bit key K is split into four 32-bit blocks $K = (W, X, Y, Z)$ then the subkeys K_i are determined by

$$\begin{aligned} \delta_i &= \delta \left\lfloor \frac{i+1}{2} \right\rfloor, \\ K_i &= \begin{cases} (W, X, \delta_i) & \text{if } i \text{ is odd} \\ (Y, Z, \delta_i) & \text{if } i \text{ is even} \end{cases} \qquad (\text{for } 1 \le i \le 64). \end{aligned}$$

For any key K the subkeys K_i are all distinct due to alternate subkeys having different multiples of δ by the key schedule. The key schedule constant $\delta = \mathtt{9E3779B9}_{\mathsf{H}}$ is derived from the golden number but its precise value has no cryptographic significance; its main purpose is to ensure that the subkeys are distinct [15].

The round function of TEA is considerably more complicated than the simple block ciphers in the previous sections of this report, which might raise the following questions:

1. Why are as many as 64 rounds used?
2. Why is the key schedule constant δ required?
3. Why is the message block shifted in the cipher function?
4. Are there any weaknesses in this system?

The last question can be answered immediately, but the answers to the first three questions should become apparent in the later sections when we analyse the STEA cipher.

3.1 Equivalent Keys

For any cipher system, a key K' is equivalent to key K if and only if

$$E_K(P) = E_{K'}(P). \tag{9}$$

As this is an equivalence relation, we can define equivalence classes in $\mathcal{K}$ and say that a pair of keys K and K' belong to the same equivalence class if equation 9 holds. For a well-designed cipher we would not want equivalent keys to exist (each key $K \in \mathcal{K}$ should specify a distinct mapping of plaintext to ciphertext) and so we would expect 2^k equivalence classes by equation 9.

Thus, for the TEA cipher we would expect 2^{128} equivalence classes. In actual fact, however, TEA has 2^{126} equivalence classes (there are 2^{126} distinct mappings of plaintext to ciphertext). This weakness manifests itself in the cipher function (generally the most sensitive part of a Feistel cipher). Note that for all values of $X, Y \in \mathbb{Z}_{2^{32}}$,

$$\begin{aligned} 2^{31} \boxplus 2^{31} &= 0, \\ X \boxplus 2^{31} &= X \oplus \texttt{80000000}_{\text{H}}. \end{aligned}$$

Hence,

$$X \boxplus Y \equiv (X \oplus \texttt{80000000}_{\text{H}}) \boxplus (Y \oplus \texttt{80000000}_{\text{H}}).$$

The cipher function (equation 8) can be be manipulated in a similar fashion to prove that

$$F(M, (T, U, V)) \equiv F(M, (T \oplus \texttt{80000000}_{\text{H}}, U \oplus \texttt{80000000}_{\text{H}}, V)).$$

This means that every 128-bit key $K = (W, X, Y, Z)$ has three equivalent keys of the form:

$$\begin{aligned} &(W \oplus \texttt{80000000}_{\text{H}}, X \oplus \texttt{80000000}_{\text{H}}, Y, Z), \\ &(W, X, Y \oplus \texttt{80000000}_{\text{H}}, Z \oplus \texttt{80000000}_{\text{H}}), \\ &(W \oplus \texttt{80000000}_{\text{H}}, X \oplus \texttt{80000000}_{\text{H}}, Y \oplus \texttt{80000000}_{\text{H}}, Z \oplus \texttt{80000000}_{\text{H}}). \end{aligned}$$

For example, encrypting plaintext $P = (\texttt{00000000 00000000}_{\text{H}})$ with any of the following keys produces the ciphertext $C = (\texttt{9327C497 31B08BBE}_{\text{H}})$:

$$\begin{aligned} &\texttt{00000000 80000000 00000000 00000000}_{\text{H}}, \\ &\texttt{80000000 00000000 00000000 00000000}_{\text{H}}, \\ &\texttt{80000000 00000000 80000000 80000000}_{\text{H}}, \\ &\texttt{00000000 80000000 80000000 80000000}_{\text{H}}. \end{aligned}$$

This weakness means that although TEA uses a 128-bit key, it provides at best the same security as a 126-bit key. In an exhaustive key-search, an attacker would only have to test a quarter of the total number of possible keys (because there is no need to try keys that are equivalent to those that have already been tested).

In spite of the equivalent keys, the reduction in the complexity of exhaustive key-search does not pose a threat to the security of TEA (although it does highlight a design flaw), since it is still computationally secure. However, the existence of equivalent keys means that TEA is unsuitable for use in a hash function that is based on block ciphers, such as those suggested in [8].

To eliminate the equivalent keys it would be necessary to redesign the TEA key schedule, which is precisely what the designers of TEA did [13]. Their improvements also defeat a “related-key” attack on TEA that requires 2^{23} chosen plaintexts encrypted under two related keys (the value of the second key depends on the value of the first key) [6].

4 The STEA Block Cipher

The STEA block cipher is a 64-round Feistel cipher operating on 64-bit message blocks with a 64-bit key. It was designed purely as a test block cipher for learning about modern cryptanalytic techniques and a number of variations exist for the same purpose (but designed to illustrate different attacks).

The STEA block cipher uses the round function as in equation 7 and the cipher function defined by

$$F(M, K_i) = M \oplus K_i, \tag{10}$$

where K_i is a 32-bit subkey. Considering the 64-bit key as two 32-bit blocks so that $K = (K_1, K_2)$, the subkeys are simply $K_i = K_1$ for i odd and $K_i = K_2$ for i even (where $1 \leq i \leq 64$). Thus the halves of the key are applied in alternate rounds.

To encrypt, the plaintext $P = (L_0, R_0)$ is input to the encryption function, which iterates the round function

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \boxplus (R_{i-1} \oplus K_i), \end{aligned} \tag{11}$$

for $1 \leq i \leq 64$. The output is the ciphertext $C = (L_{64}, R_{64})$. To decrypt, the ciphertext $C = (L_0, R_0)$ is input to the decryption function, which iterates

$$\begin{aligned} R_i &= L_{i-1}, \\ L_i &= R_{i-1} \boxminus (L_{i-1} \oplus K_i), \end{aligned}$$

for $1 \leq i \leq 64$. The plaintext is then $P = (L_{64}, R_{64})$. The subkeys would have to be applied in the reverse order, as usual, and so $K_i = K_2$ for i odd and $K_i = K_1$ for i even (where $1 \leq i \leq 64$).

To show the effectiveness of STEA, the following table lists some plaintexts (chosen to have low Hamming weights and distances from each other) and corresponding ciphertexts:

Key	Plaintext	Ciphertext
(00000000 00000000$_H$)	(00000000 00000000$_H$)	(00000000 00000000$_H$)
(00000000 00000000$_H$)	(00000000 00000001$_H$)	(61CA20BB 297A859D$_H$)
(00000000 00000000$_H$)	(00000001 00000001$_H$)	(297A859D 8B44A658$_H$)
(00000000 00000000$_H$)	(00000001 00000000$_H$)	(C7B064E2 61CA20BB$_H$)
(00000001 00000003$_H$)	(00000000 00000000$_H$)	(57112EA4 255E6231$_H$)
(00000001 00000003$_H$)	(00000000 00000001$_H$)	(F12AEA7D ED0EC712$_H$)
(00000001 00000003$_H$)	(00000001 00000001$_H$)	(C7B064E3 61CA20BB$_H$) $\star$
(00000001 00000003$_H$)	(00000001 00000000$_H$)	(C7B064E2 61CA20BA$_H$) $\star$
(55555555 55555555$_H$)	(22222222 22222222$_H$)	(0F3102B4 83B32497$_H$)

The two ciphertext blocks marked by $\star$ are remarkably similar, differing exactly in the places where their corresponding plaintexts differ. This indicates a serious problem caused by a lack of diffusion. There are other noticable patterns in the above table, but from the point of a ciphertext-only attack, there does not seem to be any resemblence between plaintexts and their corresponding ciphertexts (except for the case where the plaintext and key are both zero; the reason for this will be explained later).

STEA does not inherit the equivalent keys problem that TEA suffered from. However, it has a number of shortcomings of its own due its overly simplistic cipher function and key schedule.

5 Algebraic Solution

As explained in section 2, the round function need not be secure against algebraic solution for the cipher to be secure, provided that a sufficient number of rounds are used.

In 2-round STEA, the plaintext $P = (L_0, R_0)$ and the ciphertext $C = (L_2, R_2)$ are related by

$$\begin{aligned} L_2 &= L_0 \boxplus (R_0 \oplus K_1), \\ R_2 &= R_0 \boxplus (L_2 \oplus K_2). \end{aligned}$$

In each equation there is exactly one unknown term. Thus STEA needs a minimum of three rounds, otherwise we can recover the 64-bit key with only one known plaintext by

$$\begin{aligned} K_1 &= (L_2 \boxminus L_0) \oplus R_0, \\ K_2 &= (R_2 \boxminus R_0) \oplus L_0. \end{aligned}$$

Actually, any Feistel cipher would require a minimum of three rounds, since part of this problem is caused by the fact that each round of a Feistel cipher only encrypts half a message block. If the subkey can be recovered from the input and output of the cipher function (as in STEA) then the key can be recovered with only one known plaintext[3].

6 A Meet-In-The-Middle Attack

If we have a look at 3-round STEA with *independent subkeys*, i.e., no key schedule (so that the key length effectively becomes $k = 3 \times 32 = 96$), we find that the plaintext $P = (L_0, R_0)$ and ciphertext $C = (L_3, R_3)$ are related by

$$\begin{aligned} L_3 &= R_0 \boxplus ((L_0 \boxplus (R_0 \oplus K_1)) \oplus K_2), \\ R_3 &= L_0 \boxplus (R_0 \oplus K_1) \boxplus (L_3 \oplus K_3). \end{aligned} \tag{12}$$

In a known-plaintext attack, equation 12 shares four known terms (each half of the plaintext and ciphertext) and three unknown constants (the three subkeys). There is no way to recover any of the subkeys simply by manipulating these equations. An exhaustive key-search on this STEA variant would require at most 2^{96} encryptions and one known plaintext.

A *meet-in-the-middle attack* finds a useful tradeoff between memory and computation, for an efficient key-search. Equation 12 can be rearranged to

$$\begin{aligned} K_2 &= (L_3 \boxminus R_0) \oplus (L_0 \boxplus (R_0 \oplus K_1)), \\ K_3 &= L_3 \oplus (R_3 \boxminus L_0 \boxminus (R_0 \oplus K_1)). \end{aligned} \tag{13}$$

In a known-plaintext attack, equation 13 has subkeys K_2 and K_3 expressed in terms of four known terms and one unknown 32-bit constant (the subkey K_1). With one known plaintext and by trying all values of $K_1 \in \mathbb{F}_2^{32}$, we can find corresponding values of K_2 and K_3. In effect, we would have 2^{32} possible values for the 96-bit key. Another known plaintext block is then encrypted with each of these 2^{32} possible keys; if the output of a trial encryption matches the actual ciphertext then with high probability we have recovered the key[4]. Thus, to recover the 3-round STEA 96-bit key in a meet-in-the-middle attack requires at most 2^{33} encryptions and two known plaintexts. A meet-in-the-middle attack on 4-round STEA with independent subkeys ($k = 128$) requires at most 2^{65} encryptions and two known plaintexts. With the STEA key schedule ($k = 64$) the attack on four rounds is less efficient than exhaustive key-search.

A meet-in-the-middle attack could also be applied to the block cipher of equation 3. Note that

$$K_1 = C \boxminus (P \oplus K_0).$$

Now we can recover the key using two known plaintexts in the following way. For each known plaintext pair, try every possible value of $K_0 \in \mathbb{F}_2^{64}$ and obtain two possible values of K_1. If the two possible

[3]The first published attack on reduced-round DES was an attack on two rounds, based on this method of attacking 2-round Feistel ciphers [5].

[4]The probability that a key might be found which satisfies equation 13 for the two known plaintexts but is not the correct key is 2^{-64}. The correct key can be uniquely identified by testing against a third known plaintext.

values of K_1 match then we have found the correct key. This means we have to try at most 2^{65} encryptions to recover the key, compared with 2^{128} that would be required for an exhaustive key-search.

There are many variations in the implementation of a meet-in-the-middle attack. A naive attempt to double the effective key-length of a block cipher might be to encrypt the plaintext with key K_1 then decrypt the result with a different key K_2, so that

$$C = D(E(P, K_1), K_2).$$

An exhaustive key-search would require up to 2^{2k} trials for the key (K_1, K_2) and one known plaintext. However, a meet-in-the-middle attack on this construction can recover the key much faster. For $k \leq m$, the attack can be attempted as follows. Let (P, C) and (P', C') be two known plaintexts. We use a lookup-table T and denote the i-th entry in T by $T(i)$. Complete the lookup-table by $T(E(P, K_1)) = K_1$, for every key $K_1 \in \mathbb{F}_2^k$. Now, for every key K_2, let $K_1 = T(E(C, K_2))$ and compute a trial encryption of P'; if $C' = D(E(P', K_1), K_2)$ then we have found the correct $2k$-bit key (K_1, K_2). This meet-in-the-middle attack recovers the key with up to 2^{k+1} trials, memory for 2^k messages, and two known plaintexts. If $k > m$ then the lookup-table structure needs to be slightly different since each $T(i)$ may have more than one value.

The meet-in-the-middle attacks are useful only when we can decompose the encryption process into more than one equation to solve, each equation has different unknown terms, and the total length of the unknown terms in each equation is less than the key length k.

If more than three rounds are used in a Feistel cipher then the meet-in-the-middle attack becomes impractical, since the complexity of the attack increases exponentially with the length of each subkey for each additional round.

7 A Divide-And-Conquer Attack

There is a very effective known-plaintext attack on STEA that can recover the entire 64-bit key with only 32 encryptions. This attack is based on the *divide-and-conquer* principle whereby the cipher structure is divided into parts and the smaller parts are attacked separately. So far we have reviewed STEA only from a general algebraic point. We will now take a closer look at the confusion and diffusion process and reveal a major weakness in STEA.

We denote the i-th bit of A by $A[i]$, where the least-significant bit (LSB) of A is $A[0]$. The algorithm for computing $S = A \boxplus B$ involves the equations

$$\begin{aligned} C[0] &= 0, \\ S[i] &= A[i] \oplus B[i] \oplus C[i], \\ C[i+1] &= A[i]B[i] \oplus A[i]C[i] \oplus B[i]C[i]. \end{aligned} \tag{14}$$

where S represents the *sum* and C is the *carry*. First note that $S[0] = A[0] \oplus B[0]$; the LSB is linear over $\mathbb{F}_2$. Secondly, the only nonlinearity from addition is due to the carry, which only propagates upwards (diffusion in only one direction).

The STEA round function for the least-significant bit of the message block is linear:

$$\begin{aligned} L_i[0] &= R_{i-1}[0] \\ R_i[0] &= L_{i-1}[0] \oplus R_{i-1}[0] \oplus K_i[0], \end{aligned}$$

Thus we can recover two key bits (the least-significant bits of K_1 and K_2) using known plaintext by finding the linear expressions relating the LSBs of the plaintext, ciphertext and key. Denoting the plaintext $P = (L_0, R_0)$ and ciphertext $C = (L_{64}, R_{64})$, we get the following equations after 64 rounds:

$$\begin{aligned} K_1[0] &= L_{64}[0] \oplus R_{64}[0] \oplus L_0[0], \\ K_2[0] &= L_0[0] \oplus R_0[0] \oplus R_{64}[0]. \end{aligned} \tag{15}$$

The terms on the right-hand side are known, and hence we recover the values of $K_1[0]$ and $K_2[0]$.

Since each message bit is independent of all other message bits, except for the interference of the nonlinear carry, we can express any key bit in terms of known message bits by

$$\begin{aligned} K_1[i] &= L_{64}[i] \oplus R_{64}[i] \oplus L_0[i] \oplus C_1[i], \\ K_2[i] &= L_0[i] \oplus R_0[i] \oplus R_{64}[i] \oplus C_2[i], \end{aligned}$$

where i is the bit position, and C_1 and C_2 are nonlinear functions of the message and key bits, due to carry.

Now consider the next bit, bit 1, which is affected by the carry from bit 0. If we eliminate the carry from bit 1 then we are left with the linear part corresponding to equation 15, which can be solved to recover two more key bits. We continue in this manner until the entire key is recovered. The easiest way to eliminate the carry is to calculate its value for the appropriate bit and then to exclusive-or it with the ciphertext bit. To calculate the carry for the i-th bit, encrypt the $(i-1)$ least-significant bits of the plaintext, using the known $(i-1)$ key bits. The value of the bit in the i-th position is the carry (which we exclusive-or against the actual ciphertext bit).

Thus, we can rapidly recover the 64-bit key using only one known plaintext and 32 encryptions. In general, for message block length m, this attack will recover the m-bit key with only $\frac{m}{2}$ encryptions, independent of the number of rounds.

This attack cannot be applied to TEA because the shifts in the cipher function provide much better diffusion. As each round passes, each bit of the message block becomes more dependent on other message and key bits. Each ciphertext bit of TEA is a complex nonlinear function of a number of plaintext and key bits; there is no simple way to solve the equations. The lack of diffusion in STEA meant that a linear relation could be found between message and key bits in the same position; and this, of course, was its Archilles heel.

8 Linear Cryptanalysis

The *linear cryptanalysis* method is one of the most recent techniques of analysing block ciphers. It is a known-plaintext (statistical) attack, developed by Mitsuru Matsui, used to break the DES cipher using 50 workstations and 2^{43} known plaintexts [10, 11].

Linear cryptanalysis approximates the nonlinear part of a cipher to a linear equation (so that the linear cipher will give the same results as the nonlinear cipher most of the time, but will also give some incorrect results). Since the linear approximation only holds probabilistically, we need a lot of known plaintexts to be able to make use of the approximation, particularly if the probability of the approximation is extremely close to half (in this case the approximation provides only a slight advantage over randomly guessing the value of a key bit).

There are two stages to applying linear cryptanalysis to a block cipher: (1) finding suitable linear approximations of the cipher, and (2) applying the known-plaintext attack algorithm.

An overview of how suitable linear approximations for a Feistel cipher may be found is as follows:

Step 1 Find linear equations which are good approximations to the nonlinear part of the cipher function. Take note of the probability with which the linear approximation holds.

Step 2 Extend the linear approximations to the round function, and thus formulate a linear equation for each approximation.

Step 3 Construct a linear approximation of the block cipher by compounding linear equations for the round function, making sure all intermediate unknown message terms are cancelled out. The probability of this cipher linear approximation can be calculated from the probabilities of the round approximations.

Step 4 Calculate the number of known plaintexts required for the known plaintext attack. This depends on the probability of the cipher approximation as well as the required degree of success.

To illustrate what is meant by a linear approximation, we will examine the properties of the three-input nonlinear boolean function F, defined by

$$F(A,B,C) \quad = \quad ABC \oplus BC \oplus B \oplus C. \tag{16}$$

The following table shows the output of F for all possible inputs:

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Note that this nonlinear function is biased: the output has more ones than zeros. It is unwise to use biased functions as the nonlinear component of a block cipher (or indeed any cipher system). The bias would definitely leak some information about the plaintext and key in the ciphertext; and this may lead to a chosen- or known-plaintext attack [1].

The *Walsh transform* is used to measure the linearity of a boolean function. Specifically, for a nonlinear function $F : \mathbb{F}_2^n \to \mathbb{F}_2$, calculate

$$S_F(\alpha) \quad = \quad \sum_{x\in\mathbb{F}_2^n} (-1)^{x\cdot\alpha} F(x), \tag{17}$$

where $x \cdot \alpha$ denotes the inner-product of vectors x and α. The vector α is the n-tuple of input bits which are being tested for a correlation to the output bit. The result of equation 17 is an integer in the range -2^{n-1} to 2^{n-1}, and will be even only if the nonlinear function is *0-1 balanced* (the output has the same number of ones and zeros). In particular, the result is less than zero if the correlation holds with probability greater than $\frac{1}{2}$, zero if there is no correlation, and greater than zero if the correlation holds with probability less than $\frac{1}{2}$.

When analysing the DES S-boxes, Matsui generalised the Walsh transform to a function $N_F : \mathbb{F}_2^n \to \mathbb{F}_2^m$, which measures linearity between subsets of the inputs and the outputs of the nonlinear function (the Walsh transform is the case when the nonlinear function has a one-bit output) [10].

If the Walsh transform is computed on equation 16 over all possible values of α, it shows that there are seven distinct linear approximations (linear equations whose output correlates to the output of the nonlinear function), as follows:

$$\text{Prob=5/8} \quad : \quad C,$$

$$\begin{aligned} \text{Prob}=5/8 &\ :\ B, \\ \text{Prob}=7/8 &\ :\ B \oplus C, \\ \text{Prob}=5/8 &\ :\ A \oplus 1, \\ \text{Prob}=5/8 &\ :\ A \oplus C, \\ \text{Prob}=5/8 &\ :\ A \oplus B, \\ \text{Prob}=5/8 &\ :\ A \oplus B \oplus C \oplus 1. \end{aligned}$$

These can be verified by comparing their output against the output of the nonlinear equation:

A	B	C	F	$B \oplus C$	$A \oplus 1$	$A \oplus C$	$A \oplus B$	$A \oplus B \oplus C \oplus 1$
0	**0**	**0**	**0**	**0**	1	**0**	**0**	1
0	0	**1**	**1**	**1**	**1**	**1**	0	0
0	**1**	0	**1**	**1**	**1**	0	**1**	0
0	**1**	**1**	**1**	0	**1**	**1**	**1**	**1**
1	**0**	**0**	**0**	**0**	**0**	1	1	**0**
1	0	**1**	**1**	**1**	0	0	**1**	**1**
1	**1**	0	**1**	**1**	0	**1**	0	**1**
1	1	1	**0**	**0**	**0**	**0**	**0**	**0**
3	5	5	**8**	7	5	5	5	5

Clearly the best approximation to the nonlinear expression $ABC \oplus BC \oplus B \oplus C$ is the linear expression $B \oplus C$, which gives matching output for 7 of the 8 possible inputs.

We will now demonstrate linear cryptanalysis of a simple 4-round Feistel cipher according to the overview given earlier. Let the cipher function be defined by

$$F(M, (Y, Z)) \quad = \quad MY \oplus MYZ \oplus Z,$$

where Y and Z are halves of the round subkey. Thus the encryption process is

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \oplus R_{i-1}Y_i \oplus R_{i-1}Y_iZ_i \oplus Z_i, \end{aligned} \tag{18}$$

for $1 \leq r \leq 4$. The plaintext $P = (L_0, R_0)$ and ciphertext $C = (R_4, L_4)$. This can be attacked by a divide-and-conquer approach since each ciphertext bit depends on one plaintext bit and each message bit is encrypted independently (there is no diffusion). Applying a bitwise permutation after the cipher function would repair the lack of diffusion, but would not defend against linear cryptanalysis in this case.

Step 1. After one round of encryption, the unknown message block is (L_1, R_1), where L_1 is known plaintext and $R_1 = L_0 \oplus R_0Y_1 \oplus R_0Y_1Z_1 \oplus Z_1$. In particular, R_1 is a four-input nonlinear function of the form $A \oplus B \oplus BCD \oplus CD$. In fact, this nonlinear function is 0-1 balanced. The Walsh transform reveals eight linear approximations:

$$\begin{aligned} \text{Prob}=10/16 &\ :\ A \oplus 1, \\ \text{Prob}=10/16 &\ :\ A \oplus D, \\ \text{Prob}=10/16 &\ :\ A \oplus C, \\ \text{Prob}=10/16 &\ :\ A \oplus C \oplus D \oplus 1, \\ \text{Prob}=14/16 &\ :\ A \oplus B, \\ \text{Prob}=10/16 &\ :\ A \oplus B \oplus D, \\ \text{Prob}=10/16 &\ :\ A \oplus B \oplus C, \\ \text{Prob}=10/16 &\ :\ A \oplus B \oplus C \oplus D \oplus 1. \end{aligned}$$

Thus, the best linear approximation is simply

$$A \oplus B \oplus BCD \oplus CD \quad \approx \quad A \oplus B,$$

which holds with probability $14/16 = 0.875$.

Step 2. By matching the terms of the round function (equation 18) to the expression $A \oplus B \oplus BCD \oplus CD$ and substituting in the suggested linear approximation $A \oplus B$, we obtain the round linear approximation for the ℓ-th output bit,

$$\begin{aligned} L_i[\ell] &= R_{i-1}[\ell], \\ R_i[\ell] &= L_{i-1}[\ell] \oplus Z_i[\ell], \end{aligned}$$

This one-round linear equation matches the output of the actual nonlinear round function with a probability of 0.875 (i.e., the same as the linear approximation).

It is important to note that the probability of this linear approximation applies only to one bit, not the entire block. For this reason the position of the bits in a linear approximation are usually also recorded in the equation. This is necessary if different nonlinear functions are computed on sublocks, as in the DES S-boxes, or if bitwise permutations are used in the cipher. We will omit the notation, since the following analysis only applies to bits in the same position within the block.

Step 3. If we had more than one linear equation for the round, this step would involve adding one-round linear equations so that the output message bits of one equation become the input message bits of some later round, so that they will be cancelled out eventually. The resulting equation would represent the Feistel cipher (holding probabilistically, of course) and express a linear relation between some plaintext, key and ciphertext bits. As we only have one linear equation this step simply involves expanding the Feistel cipher in terms of the one-round equation. Once again, the plaintext is $P = (L_0, R_0)$ and the ciphertext is $C = (R_4, L_4)$. By compounding one round linear approximations we obtain:

$$L_0 \oplus R_0 \oplus L_4 \oplus R_4 \quad = \quad Z_1 \oplus Z_2 \oplus Z_3 \oplus Z_4. \tag{19}$$

The probability that this equation holds for a random known plaintext depends on the probabilities of each linear approximation from which it is composed. The cipher linear approximation will hold if and only if either all of the one-round linear approximations hold, or if none of them hold. In this case the probability of equation 19 is calculated by

$$\begin{aligned} (0.875)(0.875) + (1 - 0.875)(1 - 0.875) &= 0.781 \\ \rightarrow (0.781)(0.875) + (1 - 0.781)(1 - 0.875) &= 0.711 \\ \rightarrow (0.711)(0.875) + (1 - 0.711)(1 - 0.875) &= 0.658. \end{aligned}$$

The method of obtaining the probability of the cipher linear approximation from the probabilities of the one-round linear approximations in this manner is due to the *piling-up lemma* described by Matsui [10]. Equivalently, the probability of the cipher linear approximation may be calculated by

$$\frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \left(p_i - \frac{1}{2} \right),$$

which evaluates the probability that the exclusive-or of n independent random variables will equal zero. The piling-up lemma cannot be used blindly; it fails to give an accurate probability estimate on some block ciphers, particularly if keyed permutations or data-dependent permutations are used.

Step 4. The success of an attack depends on the probability of the cipher linear approximation and also on the number of known plaintexts. Matsui estimates the success rate of the algorithm and the number of plaintexts required to acheive a particular degree of success. The success rate is based on a maximum likelihood method and is derived by approximating binary distribution with normal distribution [10]. For a 97.7% success rate, the equation $N = |p - \frac{1}{2}|^{-2}$ is suggested. Thus, for a success rate of 97.7%, we would require $|0.658 - 0.500|^{-2} = 40$ known plaintexts. If the number of rounds is increased from four to eight then the probability for the cipher linear approximation is 0.550 and around 400 known plaintexts are required for a 97.7% success rate.

To perform the known-plaintext attack on this 4-round Feistel cipher, obtain $N = 40$ known plaintexts (encrypted under an unknown key). Let $X = 0$ then, for each known plaintext, evaluate the one-bit value of the left-hand side (LHS) of the cipher linear approximation (equation 19) and increment the counter X by one if the LHS has value zero. If $X > N/2$ then guess that the value of the right-hand side of equation 19 is zero, otherwise guess that it is one. This gives us the one-bit solution to $Z_1 \oplus Z_2 \oplus Z_3 \oplus Z_4$, which would be used to fix the value of a key bit in an exhaustive key-search (thus halving the complexity of exhaustive key-search).

This is the basic known-plaintext attack; there are improvements over this, which attempt recover a larger portion of the key at a time and also require less known plaintext [11].

The nonlinear component in STEA (and TEA) is the integer addition operation ($\boxplus$). The nonlinearity in integer addition is due to the carry function, as shown in equation 14. The function $S[i] = (A \boxplus B)[i]$ (i.e., the algebraic normal form of $S[i]$ where $S = A \boxplus B$) becomes more complex as i increases, because of the fact that carry propagates upwards. The following table shows the operation of the carry function:

$A[i]$	$B[i]$	$C[i]$	$C[i+1]$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The Walsh transform reveals four linear approximations to the carry function, each holding with probability $6/8 = 0.750$:

$$\begin{aligned} \text{Prob=6/8} &: C[i], \\ \text{Prob=6/8} &: B[i], \\ \text{Prob=6/8} &: A[i], \\ \text{Prob=6/8} &: A[i] \oplus B[i] \oplus C[i] \oplus 1. \end{aligned}$$

This gives us two useful linear approximations for one carry operation, $C[i] = A[i-1]$ and $C[i] = B[i-1]$, both holding with probability 0.750. The corresponding one-round linear approximations are

$$R_i[\ell] = L_{i-1}[\ell - 1], \quad (20)$$

$$R_i[\ell] = R_{i-1}[\ell - 1] \oplus K_i[\ell - 1], \quad (21)$$

To find the linear approximation for the 64-round cipher, these equations must be extended to a suitable linear path from the plaintext to the ciphertext. The optimal construction for a linear path in

STEA would be to use equation 20 for all the odd rounds and equation 21 for all the even rounds. The result is the STEA linear approximation

$$L_{64}[32] \oplus R_{64}[32] \quad = \quad K_1[31],$$

which is invalid since bit 32 does not exist for 32-bit blocks. Linear cryptanalysis is ineffective on the STEA cipher, because there is no linear approximation which spans 64 rounds. In general, the best linear approximation in n-round STEA is given by

$$L_n[n/2] \oplus R_n[n/2] \quad = \quad K_1[(n/2) - 1].$$

Hence, for 16-round STEA, we have the cipher linear approximation

$$L_{16}[8] \oplus R_{16}[8] \quad = \quad K_1[7].$$

which holds with probability $\frac{1}{2} + 2^{-17}$. To recover one key bit (bit 7 of K_1) of 16-round STEA using linear cryptanalysis would require around 2^{34} known plaintexts.

9 Weak Key Schedule And Weak Cipher Function

In this section, we show the importance of having a key schedule that generates distinct round subkeys. First we will follow on from the analysis in section 5 and describe a curious property of the STEA cipher.

If the least-significant bits of the subkeys and plaintext halves are all equal, then these bits will "filter through" to the ciphertext. For example,

$$\begin{aligned} P &= (\texttt{xxxxxyyy xxyyyyyy}_{\mathsf{H}}), \\ K &= (\texttt{xxxxyyyy xxxxxxyy}_{\mathsf{H}}), \\ C &= (\texttt{xxxxxyyy xxxxxxyy}_{\mathsf{H}}), \end{aligned}$$

where 'x' is any digit, and 'y' is either the bit-sequence of all ones (F_H) or zero (0_H). This should be obvious in the case where y=0, since addition and exclusive-or also results in 0, and no carry.

Also, if K_1 equals the right-half of the plaintext and K_2 equals the left-half of the plaintext, then the plaintext is not encrypted. So if an attacker has a large number of ciphertext blocks, and one block looks suspiciously like plaintext, then it could well be the key.

This problem is caused by the output of the cipher function (equation 10) being zero for all rounds. The output of the cipher function is zero if and only if the input message block is equal to the subkey. Since the message block cannot be encrypted if the cipher function output is zero it is crucial that the next subkey that affects the message block be different to ensure that the cipher function output is non-zero (and cause the message block to be partially-encrypted, as required).

In an IBM research report, Edna Grossman and Bryant Tuckerman described a number of attacks on a weak Feistel cipher called NDS (New Data Seal), which appears to be a simplified version of the Lucifer block cipher [1, 4]. The NDS cipher had a weak key schedule and a weak cipher function. The key schedule was practically non-existant; all the round subkeys were the same, which effectively meant that in each round the entire key is input in the same form (i.e., unmodified between rounds). The cipher function was weak for two reasons: it was usually possible to derive key bits from knowledge of an input block and (encrypted) output block of the cipher function; furthermore, it would be possible to predict the output of the cipher function for certain inputs, regardless of the key. Consequently, NDS appeared highly susceptible to an attack which, in its most advanced form, could break

NDS with 556 chosen plaintexts. However, the principle of their attack is fairly general and would be applicable to any Feistel cipher that does not use distinct round subkeys and whose cipher function may allow recovery of key bits from known input and output blocks. An interesting point about this attack is that its effectiveness does not depend on the number of rounds.

The basic principle of their attack hinges on the fact that the encryption process of an n-round Feistel cipher with no distinct round subkeys can be simplified to $E(P, K) = \pi(\rho^n(P, K))$. In the following explanation, we will omit the output transformation (π) for simplicity, although the analysis is still applicable to Feistel ciphers with the final swap. It is fairly easy to extend the proofs to take account of the final swap.

We obtain a known plaintext (P, C), where $C = \rho^n(P, K)$, and predict the result of a one-round encryption of P; we denote this by P' so that $P' = \rho(P, K)$. As we are dealing with a Feistel cipher, we know that $P = (L_0, R_0)$ and $P' = (L_1, R_1)$, where only the right half of P' is unknown and has to be predicted. Predicting the result of a one-round encryption of NDS was easy due to its weak cipher function. In fact, the NDS cipher function is considerably weaker than the STEA cipher function (it is impossible to predict the output of the STEA cipher function if the round subkey is unknown). If it is not possible to predict the output of the cipher function and the message block length is small enough (e.g., $m = 64$) then we could try each possible value of R_1 until we find the right value; the attack would require at most $2^{m/2}$ chosen-plaintext encryptions.

We obtain the chosen-plaintext encryption of P' to get $C' = \rho^n(P', K)$. Notice that C' is the $(n+1)$-round encryption of P, i.e., $C' = \rho(C, K)$:

$$\begin{aligned} C' &= \rho^n(P', K) \\ &= \rho^n(\rho(P, K), K) \\ &= \rho^{n+1}(P, K) \\ &= \rho(\rho^n(P, K), K) \\ &= \rho(C, K). \end{aligned}$$

In particular, for a correct guess of $P' = \rho(P, K)$, the right half of C should equal the left half of C' since $C = (L_n, R_n)$ and $C' = (L_{n+1}, R_{n+1})$. Once we have found the right value of $P' = \rho(P, K)$ then we can attempt to recover the key K from the equations:

$$\begin{aligned} L_0 \oplus R_1 &= F(R_0, K), \\ L_n \oplus R_{n+1} &= F(R_n, K). \end{aligned}$$

Now the problem is to recover the key given the input and output of the cipher function. As we have two such equations, we can use one equation to determine the key and then verify the key that we recovered by testing it in the second equation. If the cipher function is weak then recovering the key given the cipher function input and output should be relatively simple (as in the case of NDS or STEA).

A variant of this attack can be applied to STEA with a higher computational cost. STEA does not use the same subkey in every round, but it does use the same subkey in alternate rounds. Let K_1 and K_2 be the 32-bit halves of the 64-bit key K. We can express the repetitive nature of STEA by letting R denote two rounds:

$$R(M, K) = \rho(\rho(M, K_1), K_2).$$

Hence 64-round STEA can be represented as 32 successive applications of the R function. In particular, the key does not change between applications of R so that $E(P, K) = R^{32}(P, K)$. We now have to be able to predict the output of R for a chosen plaintext. The STEA cipher function does not allow us to do this; we must know the value of the subkey to predict the output of one round. The only way to deal with this situation is to obtain enough well-distributed known plaintexts that we have a reasonable chance of finding a pair of plaintexts (P, P') such that $P' = R(P, K)$. By the *birthday paradox*, this would require around 2^{32} known plaintexts. Since we cannot identify a correct pair (P, P'), we must attempt the following analysis on every possible pair until we recover the key. Therefore, the worst-case time complexity of this attack would be $O(2^{64})$. This is comparable to the complexity of an exhaustive key-search.

For each pair of known plaintexts (P, P') we solve the following equations

$$\begin{aligned} K_1 &= R_0 \oplus (L_2 \boxminus L_0), \\ K_2 &= L_2 \oplus (R_2 \boxminus R_0), \\ K_1 &= R_{64} \oplus (L_{66} \boxminus L_{64}), \\ K_2 &= L_{66} \oplus (R_{66} \boxminus R_{64}), \end{aligned}$$

where $P = (L_0, R_0)$, $P' = (L_2, R_2)$, $C = (L_{64}, R_{64})$ and $C' = (L_{66}, R_{66})$. If the two values of K_1 match and the two values of K_2 match then we have found the correct key. Once again, this attack will work regardless of the number of rounds and may still be effective if the cipher function was slightly more complex (e.g., if the output of the cipher function was subjected to a bitwise rotation, which would render the divide-and-conquer attack ineffective).

A similar attack can be attempted on a version of TEA with fixed subkey constants δ_i (e.g., $\delta_i = \delta$, for all i); this version of TEA would have two distinct subkeys (like STEA). The attack on this TEA variant was described by Roger Fleming [3]. It has a complexity of $O(2^{98})$ and requires around 2^{32} known plaintexts. Here the complexity of attack is better than that of exhaustive key-search (which was not the case with STEA). An improvement of the attack, which reduces the complexity to $O(2^{64})$ at the cost of requiring an additional 2^{32} chosen plaintexts, was outlined by David Wagner.

10 Differential Cryptanalysis

In section 2 it was shown that the difficulty of solving nonlinear equations can be useful for designing block ciphers. For example, let us return to the block cipher of equation 3. The furthest we could get by simple manipulation of this cipher was eliminating a subkey by taking the difference of the ciphertexts, arriving at equation 4. Now notice that if the cipher was linear, taking the "difference" of a pair of ciphertexts would have cancelled out both subkeys, leaving the equation:

$$C \oplus C' = P \oplus P'.$$

This simply tells us that the difference between the ciphertexts is the same as the difference between their plaintexts; and as this holds for all messages, taking the difference does not lead to any information about the key.

For a nonlinear cipher, the difference between ciphertexts is *not the same* as the difference between their plaintexts for all possible plaintexts. Furthermore, for a specific difference in the plaintexts, the difference in the corresponding ciphertexts may depend on the key and hence reveal information about the key. This idea is the principle behind *differential cryptanalysis*, a general block cipher analysis method developed by Eli Biham and Adi Shamir [2]. A successful application of differential cryptanalysis leads to a chosen-plaintext attack.

We denote the difference between a pair of messages (M, M') by $\Delta M = M \otimes (M')^{-1}$, where $\otimes$ is an appropriate operation for defining "difference". The exclusive-or operation, by its nature, is the most versatile choice for defining difference. However, it may be preferable or necessary to use another group operation for this purpose; for example, when applying differential cryptanalysis to an iterated cipher, it is preferable to use the difference operation which will cause the cipher to be a "Markov cipher" [8, 9].

The example block cipher of equation 3 is not an iterated block cipher, but it is suitable for demonstrating the basic ideas of differential cryptanalysis. We assume the message block length $m = 32$ and the key length $k = 64$ (i.e., half the block lengths of the original description).

Letting difference be defined by subtraction modulo 2^{32}, the following equation gives the appropriate ciphertext difference:

$$\begin{aligned} \Delta C &= C \boxminus C' \\ &= (P \oplus K_0) \boxminus (P' \oplus K_0). \end{aligned}$$

Hence, an attacker who knows the values P, P' and ΔC may be able to recover K_0 if they can efficiently solve equations of the form

$$C = (A \oplus X) \boxminus (B \oplus X),$$

where A, B, C are known constants. So let us examine this type of equation in more detail, to determine precisely how much information we may be able to recover. First we express the equation in a simpler form:

$$\begin{aligned} C &= (A \oplus X) \boxminus (B \oplus X) \\ &= (A \oplus X) \boxminus (A \oplus X \oplus D) \\ &= Y \boxminus (Y \oplus D), \end{aligned}$$

where $D = A \oplus B$ and $Y = A \oplus X$. The goal is to recover Y. The subtraction modulo 2^{32} can be replaced by addition modulo 2^{32} with a little more manipulation:

$$\begin{aligned} C &= Y \boxminus (Y \oplus D) \\ &= Y \boxplus -(Y \oplus D) \\ &= Y \boxplus (Y \oplus D \oplus \mathtt{FFFFFFFF_H}) \boxplus 1, \\ E &= Y \boxplus (Y \oplus F), \end{aligned}$$

where $F = D \oplus \mathtt{FFFFFFFF_H}$ (the bitwise complement of D) and $E = C \boxminus 1$. Once again, Y is unknown (and its value would lead to the subkey), and E and F are known. In a chosen-plaintext attack, the attacker would be able to fix values of F by careful selection of plaintexts A and B. If $F = 0$ then we would have $E = Y \boxplus Y = Y \lll 1$, effectively revealing 31 of the 32 bits of Y. To set $F = 0$, the *exclusive-or difference* between plaintexts must be ($\mathtt{FFFFFFFF_H}$); it does not matter what the actual plaintext values are.

Consider a chosen-plaintext attack on this block cipher. The attacker chooses a random plaintext P, then calculates a second plaintext $P' = P \oplus \Delta P$ (so that the exclusive-or difference between plaintexts is ΔP). Both chosen plaintexts are encrypted under the unknown key. The least-significant 31 bits of subkey K_0 are then recovered from the values of P and $\Delta C = C \boxminus C'$ by

$$K_0 = ((\Delta C \boxminus 1) \gg 1) \oplus P.$$

The most-significant bit of the recovered K_0 will be '0'. The subkey K_1 is then evaluated, using either one of the two chosen plaintexts, by $K_1 = C \boxminus (P \oplus K_0)$. Hence, the 64-bit key (K_0, K_1) is easily recovered with one known plaintext and one adaptive chosen plaintext (or two chosen plaintexts)[5].

[5]If the most-significant bit of the original subkey K_0 was equal to '1' (and hence incorrect in the recovered K_0) then the recovered K_1 will also be incorrect in the most-significant bit. The result is that the 64-bit key (K_0, K_1) recovered by this method may not be equal to the original key, but it will certainly be an equivalent key.

In this example, the operation for defining difference was different for the plaintext (exclusive-or) and ciphertext (subtraction modulo 2^{32}). The approach to analysing iterated ciphers requires that the difference operation be the same for input and output. In the remainder of this report, we will define the difference operation by exclusive-or.

Now we will attempt differential cryptanalysis on the block cipher defined by

$$C = (P \boxplus K_0) \oplus K_1.$$

This cipher differs from the block cipher of equation 3 in that the addition and exclusive-or operations have swapped places. The difference operation is defined by exclusive-or, and so we have the following equations for the plaintext and ciphertext differences:

$$\begin{aligned} \Delta P &= P \oplus P', \\ \Delta C &= C \oplus C' \\ &= (P \boxplus K_0) \oplus (P' \boxplus K_0). \end{aligned}$$

The subkey K_1 is eliminated from the ciphertext difference ΔC. We need to choose values of P and P', with a specific difference ΔP, that will reveal some information about K_0 by the difference ΔC.

To start, note that if $\Delta P = 0$ then $\Delta C = 0$; we have no useful information. Also note that this can be generalised to the case that a sequence of '0' least-significant bits in the input difference will cause the same least-significant bits to be '0' in the output difference, e.g., if $\Delta P = \texttt{xxxx0000}_\mathsf{H}$ then $\Delta C = \texttt{xxxx0000}_\mathsf{H}$. This is because the nonlinear carry only propagates upwards and, once again, we will exploit this fact.

The *Hamming weight* of an integer is the number of ones in its binary representation. We now look at what happens if the least-significant bit of the input difference is non-zero. Note that since lower bits of input difference zero do not affect higher output difference bits, the analysis holds for any input difference where precisely one bit is set (i.e., any input difference of Hamming weight 1). If the input difference $\Delta P = 1$ then we would have

$$\Delta C = (P \boxplus K_0) \oplus ((P \oplus 1) \boxplus K_0).$$

From this we see that the least-significant bit of K_0 is added to each of '0' and '1' and then the exclusive-or difference is taken; the value of the least-significant bit of P has no effect on this calculation. The LSB of ΔC will have value '1' since it is linear. If the LSB of $K_0 = 0$ then no carry will be generated by either of the additions and so none of the higher bits of the output difference will be affected (they will have value zero). In this case, $\Delta P = \Delta C$. If the LSB of $K_1 = 1$ then the calculation of $(1 \boxplus 0) \oplus (1 \boxplus 1)$ will generate a carry, which will affect higher bits. We expect that, in this case, $\Delta P \neq \Delta C$. So we can identify the value of the least-significant bit of K_0 by using an input difference of 1 and checking if the output difference is equal to the input difference. If the output difference is the same as the input difference then assume that the LSB of K_0 is '0' otherwise assume it is '1'.

We can recover an arbitrary bit of K_0 by the same technique; use an input difference of Hamming weight 1 (set the bit in the i-th position) and check the Hamming weight of the output difference; if it has Hamming weight 1 then assume that the i-th bit of K_0 is '0' otherwise assume it is '1'. Proceeding like this, we can recover the $(m - 1)$ least-significant bits of K_0. We cannot recover the value of the most-significant bit of K_0 in this analysis since we lose the carry anyway. However, this loss is compensated by the fact that we automatically recover this bit when calculating the value of K_1 from a known plaintext (we will either recover the original key or an equivalent key). Hence, for an m-bit message block size (and $2m$-bit key), this simple block cipher cipher is broken by differential cryptanalysis with one known plaintext and m chosen plaintexts.

It is possible (but much more complicated) to extend this attack to the block cipher of equation 3 defining input difference and output difference by exclusive-or. We would have output difference

$$\Delta C = ((P \oplus K_0) \boxplus K_1) \oplus ((P' \oplus K_0) \boxplus K_1).$$

Imagine that we supply an input difference of Hamming weight 1, with the i-th bit set. The complication arises because we cannot guarantee that the input bits to the addition that are lower than the i-th bit have value zero, because some of them may have been inverted by the exclusive-or with K_0. In particular, if the $(i-1)$-th bit of K_0 is '1' and the $(i-1)$-th bit of K_1 is also '1' then the subsequent carry that would be generated by their addition would disturb the higher bits, and hence the analysis. This problem can be dealt with by some elaborate analysis, but there is little point in doing so since the result cannot be generalised (which would be the principal reason for using exclusive-or for defining input and output difference in this case).

There are two stages to applying differential cryptanalysis to an iterated cipher (cf. linear cryptanalysis): (1) finding suitable differentials of the cipher, and (2) applying the chosen-plaintext attack algorithm.

We will now apply differential cryptanalysis to a 6-round block cipher with message block length $m = 8$ and key length $k = 48$. Let the nonlinear and invertible function $F : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ be defined by

$$F(x) = ((x \oplus \texttt{5C}_\texttt{H}) \boxplus \texttt{89}_\texttt{H}) \ggg 6.$$

We know that for a linear function the difference between any pair of inputs will be equal to the difference between their corresponding outputs. As the function F is nonlinear, the output difference will not equal the input difference for all possible inputs. In fact, for many specific input differences there are a number of possible output differences that could occur. For example, let the difference of inputs X and X' be denoted by $\Delta X = X \oplus X'$ and the difference of their corresponding outputs Y and Y' be $\Delta Y = F(X) \oplus F(X')$. We find that there are 7 possible output differences that may occur if the input difference to F is $\texttt{01}_\texttt{H}$, as follows (the differences are given in binary for ease of interpretation):

- There are 64 pairs (X, X') such that $\Delta X = 00000001$ and $\Delta Y = 00001100$.
- There are 32 pairs (X, X') such that $\Delta X = 00000001$ and $\Delta Y = 00011100$.
- There are 16 pairs (X, X') such that $\Delta X = 00000001$ and $\Delta Y = 00111100$.
- There are 8 pairs (X, X') such that $\Delta X = 00000001$ and $\Delta Y = 01111100$.
- There are 4 pairs (X, X') such that $\Delta X = 00000001$ and $\Delta Y = 11111100$.
- There are 2 pairs (X, X') such that $\Delta X = 00000001$ and $\Delta Y = 11111101$.
- There are 2 pairs (X, X') such that $\Delta X = 00000001$ and $\Delta Y = 11111111$.

A pair of input and output differences to a function form a *characteristic* of the function. Thus, the function F has a characteristic $(\texttt{01}_\texttt{H}, \texttt{1C}_\texttt{H})$ that holds with probability $32/128 = 0.250$. There are two characteristics of F that hold with probability 1.000:

- There are 128 pairs (X, X') such that $\Delta X = 00000000$ and $\Delta Y = 00000000$.
- There are 128 pairs (X, X') such that $\Delta X = 10000000$ and $\Delta Y = 00000010$.

Of these, only the latter characteristic is useful and it is due to the equivalence of $X \boxplus 2^{n-1}$ over $\mathbb{Z}_{2^n}$ and $X \oplus 2^{n-1}$ over $\mathbb{F}_2^n$.

The block cipher iterates the round function defined by

$$T_i \quad = \quad F(T_{i-1} \oplus K_i), \qquad \text{(for } 1 \leq i \leq 6\text{)},$$

where T_0 is the plaintext and T_6 is the ciphertext (this is not a Feistel cipher). The subkeys $(K_1, \ldots, K_6)$ are independent, and since each subkey is of length 8 bits, the key length k is 48 bits.

When applying differential cryptanalysis to iterated ciphers, the attacker analyses the propagation of differences between plaintexts through the block cipher and attempts to find a series of differences between rounds which hold with high probability. This is done by compounding one-round characteristics so that the output difference to one round is the input difference to the next round. The chain of differences for i rounds of a cipher forms an i*-round characteristic*. For an n-round block cipher, the attacker hopes to find an n-round characteristic that holds with high probability. The characteristic that has the optimal probability is called the best characteristic.

Note that for inputs (X, X') and corresponding outputs (Y, Y') the one-round input difference is $\Delta X = X \oplus X'$ and the output difference is $\Delta Y = Y \oplus Y'$. Also,

$$\begin{aligned} \Delta Y &= F(X \oplus K) \oplus F(X' \oplus K) \\ &= F(X \oplus K) \oplus F(X \oplus K \oplus \Delta X) \\ &= F(Z) \oplus F(Z \oplus \Delta X), \end{aligned}$$

where $Z = X \oplus K$ is unknown. This shows that we are not interested in the actual value of the inputs; only in their difference. It also puts the earlier analysis of the function F into context of the cipher, since the subkey is "absorbed" into the input of the function F and we need not concern ourselves with its value for the analysis. We are implicitly assuming that the value of the plaintext and subkeys will not significantly affect the propagation of differences through the cipher (this is the *hypothesis of stochastic equivalence* [8, 9]).

We know that the best one-round characteristic is $(80_\mathsf{H}, 02_\mathsf{H})$, which holds with probability 1.000. We shall use this as a starting point to explore the possible differences that may be reached after 6 rounds. We will choose to follow the characteristic that has the highest immediate probability. This may not necessarily lead to the best 6-round characteristic; we would have to follow each possible path of difference (including the low-probability differences) or else use some heuristic technique to find the best characteristic.

(1)
- There are 128 pairs (X, X') such that $\Delta X = 10000000$ and $\Delta Y = 00000010$.

(2)
- There are 64 pairs (X, X') such that $\Delta X = 00000010$ and $\Delta Y = 00001000$.
- There are 32 pairs (X, X') such that $\Delta X = 00000010$ and $\Delta Y = 00011000$.
- There are 16 pairs (X, X') such that $\Delta X = 00000010$ and $\Delta Y = 00111000$.
- There are 8 pairs (X, X') such that $\Delta X = 00000010$ and $\Delta Y = 01111000$.
- There are 4 pairs (X, X') such that $\Delta X = 00000010$ and $\Delta Y = 11111000$.
- There are 2 pairs (X, X') such that $\Delta X = 00000010$ and $\Delta Y = 11111001$.
- There are 2 pairs (X, X') such that $\Delta X = 00000010$ and $\Delta Y = 11111011$.

(3)
- There are 16 pairs (X, X') such that $\Delta X = 00001000$ and $\Delta Y = 00100000$.
- There are 56 pairs (X, X') such that $\Delta X = 00001000$ and $\Delta Y = 01100000$.
- There are 28 pairs (X, X') such that $\Delta X = 00001000$ and $\Delta Y = 11100000$.
- There are 14 pairs (X, X') such that $\Delta X = 00001000$ and $\Delta Y = 11100001$.

- There are 14 pairs (X, X') such that $\Delta X = 00001000$ and $\Delta Y = 11100011$.

(4)
- There are 18 pairs (X, X') such that $\Delta X = 01100000$ and $\Delta Y = 10000000$.
- There are 92 pairs (X, X') such that $\Delta X = 01100000$ and $\Delta Y = 10000001$.
- There are 18 pairs (X, X') such that $\Delta X = 01100000$ and $\Delta Y = 10000010$.

(5)
- There are 64 pairs (X, X') such that $\Delta X = 10000001$ and $\Delta Y = 00001110$.
- There are 32 pairs (X, X') such that $\Delta X = 10000001$ and $\Delta Y = 00011110$.
- There are 16 pairs (X, X') such that $\Delta X = 10000001$ and $\Delta Y = 00111110$.
- There are 8 pairs (X, X') such that $\Delta X = 10000001$ and $\Delta Y = 01111110$.
- There are 2 pairs (X, X') such that $\Delta X = 10000001$ and $\Delta Y = 11111101$.
- There are 4 pairs (X, X') such that $\Delta X = 10000001$ and $\Delta Y = 11111110$.
- There are 2 pairs (X, X') such that $\Delta X = 10000001$ and $\Delta Y = 11111111$.

(6)
- There are 16 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 00001000$.
- There are 8 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 01001000$.
- There are 16 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 01101000$.
- There are 32 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 01111000$.
- There are 4 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 11001000$.
- There are 2 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 11001001$.
- There are 2 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 11001011$.
- There are 8 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 11101000$.
- There are 4 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 11101001$.
- There are 4 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 11101011$.
- There are 16 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 11111000$.
- There are 8 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 11111001$.
- There are 8 pairs (X, X') such that $\Delta X = 00001110$ and $\Delta Y = 11111011$.

Thus our 6-round characteristic is

$$(\texttt{80}_\mathsf{H}, \texttt{02}_\mathsf{H}, \texttt{08}_\mathsf{H}, \texttt{60}_\mathsf{H}, \texttt{81}_\mathsf{H}, \texttt{0E}_\mathsf{H}, \texttt{78}_\mathsf{H}),$$

holding with probability

$$\frac{128}{128} \times \frac{64}{128} \times \frac{56}{128} \times \frac{92}{128} \times \frac{64}{128} \times \frac{32}{128} \quad \approx \quad \frac{1}{51}.$$

This would imply that for every 51 random and uniformly distributed pairs of chosen plaintexts with difference $\Delta P = \texttt{80}_\mathsf{H}$ we expect about 1 pair of corresponding ciphertexts to have a difference of $\texttt{78}_\mathsf{H}$. However, it is not quite as simple as this, since the subkeys are fixed (not random) and may cause some difference paths found by the above method to be impossible to follow in practice no matter how many chosen plaintexts we have.

Also, there may be more than one n-round characteristic with the same plaintext difference and ciphertext difference. For example, another 6-round characteristic of the above cipher is

$$(\texttt{80}_\mathsf{H}, \texttt{02}_\mathsf{H}, \texttt{18}_\mathsf{H}, \texttt{A3}_\mathsf{H}, \texttt{86}_\mathsf{H}, \texttt{0A}_\mathsf{H}, \texttt{78}_\mathsf{H}),$$

which holds with probability $\approx \frac{1}{1561}$. There is no way to tell which of these two 6-round characteristics was used to arrive at a ciphertext difference of $\texttt{78}_\mathsf{H}$ from a plaintext difference of $\texttt{80}_\mathsf{H}$. As there may be many ways of arriving at a particular output difference, the probability of obtaining a pair of plaintexts that cause the output difference of a characteristic may be higher than the probability of the characteristic. The attacker is not usually concerned with the message differences after each round, only in the last difference of a characteristic. An *i-round differential* is a pair $(\Delta X, \Delta Y)$ specifying the input difference and output difference after i rounds [8, 9]. The probability of a specific differential is the sum of probabilities of all characteristics that have the same input and final output difference as the differential.

A *right pair*, with respect to an i-round differential $(\Delta X, \Delta Y)$, is a pair of plaintexts such that their difference is equal to ΔX and their difference after i rounds of encryption is equal to ΔY. A *wrong pair* is a pair of plaintexts that is not a right pair.

To execute the differential cryptanalysis chosen-plaintext attack on an n-round block cipher:

1. Find an $(n-1)$-round differential $(\Delta X, \Delta Y)$ that holds with high probability (preferably the best $(n-1)$-round differential).

2. Choose a random plaintext P and compute $P' = P \oplus \Delta X$. Encrypt both plaintexts so that $C = E(P, K)$ and $C' = E(P', K)$, where K is the unknown key.

3. Recover all possible values of the subkey K_n that may cause the one-round outputs of C and C' from inputs with difference ΔY. A basic method of identifying possible subkeys is to deduce the subkeys K that can tranform round input Y into output C, for every possible input Y, and then compare the one-round encryption of $(Y \oplus \Delta Y)$ with each deduced subkey against ciphertext C'. Any subkey K which passes this test is a possible candidate for the actual last subkey, and we increment a counter corresponding to K by 1.

4. Repeat the test on another chosen plaintext, until one value of the subkey has been counted significantly more than the others. This subkey (or small set of subkeys) is assumed to be the actual subkey used in the n-th round.

If we know some n-round differential that includes the $(n-1)$-round differential, then we may be able to discard some pairs of plaintexts on the basis that their ciphertext differences would not follow from the $(n-1)$-round difference.

We can apply differential cryptanalysis to an 8-round variant of STEA. The variant applies a bitwise left rotation to the message block after the addition in the round function. The message block is rotated by 1 bit on odd rounds, and 9 bits on even rounds, so that the round function is described by

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= (L_{i-1} \boxplus (R_{i-1} \oplus K_i)) \lll S_i, \end{aligned}$$

where $K_i = K_1$ and $S_i = 1$ for i odd, and $K_i = K_2$ and $S_i = 9$ for i even. The 64-bit plaintext is (L_0, R_0) and the 64-bit ciphertext is (L_8, R_8). The bitwise rotation provides better diffusion in this STEA variant and would defeat the divide-and-conquer attack (but not linear cryptanalysis). This STEA has a 6-round characteristic described by

$$\begin{aligned} & & \texttt{E0000000 C0000000}_\mathsf{H}, \\ \text{Prob=0.250} &: & \texttt{C0000000 40000000}_\mathsf{H}, \\ \text{Prob=0.500} &: & \texttt{40000000 00000000}_\mathsf{H}, \\ \text{Prob=0.500} &: & \texttt{00000000 80000000}_\mathsf{H}, \end{aligned}$$

$$\begin{aligned} \text{Prob=1.000} &: \quad \texttt{80000000 00000100}_\mathrm{H}, \\ \text{Prob=0.500} &: \quad \texttt{00000100 00000201}_\mathrm{H}, \\ \text{Prob=0.125} &: \quad \texttt{00000201 00060200}_\mathrm{H}, \end{aligned}$$

which holds with probability 1/256. We expect about one plaintext pair of 256 pairs to have difference $\texttt{00000201 00060200}_\mathrm{H}$ after 6 rounds of encryption. As we are dealing with an 8-round cipher, we cannot identify right pairs with this characteristic. We could attempt the chosen-plaintext attack if we knew the ciphertexts after 7 rounds of encryption, and we can recover these by decrypting the ciphertext by only one round with every possible subkey K_8. Thus, for each ciphertext pair $C = (L_8, R_8)$ and $C' = (L'_8, R'_8)$, we have 2^{32} possible values of (L_7, R_7) and (L'_7, R'_7). We can test whether this is likely to be a right pair by taking the difference of the left halves of these ciphertexts and comparing against the expected difference:

$$L_7 \oplus L'_7 \quad = \quad \texttt{00060200}_\mathrm{H}.$$

If this condition is not satisfied, then we discard this pair and try again on another pair of random chosen plaintexts. Otherwise, we increment a counter for K_8 and proceed with the attack assuming that the pair is a right pair and deducing possible values of K_7 (if any) for which

$$L_6 \oplus L'_6 \quad = \quad \texttt{00000201}_\mathrm{H}.$$

This would require an additional 2^{32} trial decryptions for every pair of ciphertexts that passed the test from round 7. Thus, the complexity of testing one pair of chosen plaintexts is about 2^{33} and we would expect to recover the values of the last two round subkeys (and hence the key).

The differential cryptanalysis method was discovered before linear cryptanalysis, so there has been considerable more research on the security of block ciphers with respect to differential cryptanalysis. There are a number of improvements and evolutions of the basic idea, such as the concepts of truncated and higher-order differentials (which have been used to attack the SAFER and reduced-round IDEA block ciphers) [7].

11 Acknowledgements

Thanks to Sean Murphy, Fred Piper, Peter Wild and Chris Mitchell for their advice and for providing me with some of the literature references. Thanks also to David Wagner for some interesting discussions.

References

[1] Henry Beker and Fred Piper, *Cipher Systems: The Protection of Communications*, Northwood Books, 1982.

[2] Eli Biham and Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1992.

[3] Roger Fleming, “An attack on a weakened version of TEA”. Posted on `sci.crypt`, 22 Oct 1996.

[4] Edna K. Grossman and Bryant Tuckerman, “Analysis of a Feistel-like cipher weakened by having no rotating key”. IBM Research Report RC6375, Yorktown Heights, NY, 1977.

[5] M. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, and P. Schweitzer, "Results of an initial attempt to cryptanalyze the NBS Data Encryption Standard". Technical Report SEL 76-042, Department of Electrical Engineering, Stanford University, 1976.

[6] John Kelsey, Bruce Schneier, and David Wagner, "Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA". In *Information and Communications Security—Proceedings of ICICS 1997*, Lecture Notes in Computer Science 1334, Springer-Verlag, 1997.

[7] Lars R. Knudsen, "Truncated and Higher Order Differentials". In *Fast Software Encryption—Proceedings of the Second International Workshop*, Lecture Notes in Computer Science 1008, Springer-Verlag, 1994.

[8] Xuejia Lai, *On the Design and Security of Block Ciphers*, ETH Series in Information Processing 1, Hartung-Gorre Verlag, 1992.

[9] Xuejia Lai, James L. Massey, and Sean Murphy, "Markov Ciphers and Differential Cryptanalysis". In *Advances in Cryptology—Proceedings of EUROCRYPT '92*, Lecture Notes in Computer Science 547, Springer-Verlag, 1992.

[10] Mitsuru Matsui, "Linear Cryptanalysis Method for DES Cipher". In *Advances in Cryptology—Proceedings of EUROCRYPT '93*, Lecture Notes in Computer Science 765, Springer-Verlag, 1993.

[11] Mitsuru Matsui, "The First Experimental Cryptanalysis of the Data Encryption Standard". In *Advances in Cryptology—Proceedings of CRYPTO '94*, Lecture Notes in Computer Science 839, Springer-Verlag, 1994.

[12] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[13] Roger M. Needham and David J. Wheeler, "Tea extensions". October 1997.

[14] Rainer A. Rueppel, *Analysis and Design of Stream Ciphers*, Springer-Verlag, 1986.

[15] David J. Wheeler and Roger M. Needham, "TEA, a Tiny Encryption Algorithm". In *Fast Software Encryption—Proceedings of the Second International Workshop*, Lecture Notes in Computer Science 1008, Springer-Verlag, 1994.