Elastic Block Ciphers: The Feistel Cipher Case

Debra L. Cook Moti Yung Angelos D. Keromytis
Department of Computer Science
Columbia University, New York, NY
{*dcook,moti,angelos*}@*cs.columbia.edu*

Technical Report May 19, 2004

Abstract

We discuss the elastic versions of block ciphers whose round function processes subsets of bits from the data block differently, such as occurs in a Feistel network and in MISTY1. We focus on how specific bits are selected to be swapped after each round when forming the elastic version, using an elastic version of MISTY1 and differential cryptanalysis to illustrate why this swap step must be carefully designed. We also discuss the benefit of adding initial and final key dependent permutations in all elastic block ciphers. The implementation of the elastic version of MISTY1 is analyzed from a performance perspective.

Keywords: Block Cipher Design, Elastic Block Cipher, Variable Length Block Cipher, Encryption Algorithm, MISTY1

1 Introduction

The concept of an elastic block ciphers was introduced in [5] and provides a method by which an existing block cipher can be modified to create a variable length block cipher accepting all block lengths up to twice its original block size. In the elastic version, bits beyond the normal block size are left out of the round function then XORed and swapped with bits output from the round function in order to become part of the input to the next round. The bits output from the round function involved in the XOR become the set left out in the next round. The number of rounds are increased such that the round function is applied to each bit position the same number of times as in the original block cipher. In general, care must be taken in selecting the bits to be swapped when the round function processes subsets of the bits differently. Initial and end of round whitening are also added when forming the elastic version if not already present and applied to the entire block.

We take a closer look at elastic block ciphers in terms of how bits are swapped into and out of positions acted on by the round function. Both the manner in which bit positions are swapped at the end of each round, and key dependent initial and final permutations are considered. The bit positions involved in the swap are of most interest in the elastic versions of block ciphers whose round function operates on a subset of the bits, such as occurs in a Feistel network and in MISTY1 [8]. We use differential cryptanalysis of elastic MISTY1 to illustrate why careful selection of which bits are omitted from each round is required. In addition, the implementation of the elastic version of MISTY1 is analyzed from a performance perspective. We selected to use MISTY1 for our analysis for several reasons. Aside from being an example of a block cipher with a Feistel-like structure whose round function processes subsets of bits differently, it is NESSIE's recommendation for a 64 bit block cipher [3] and the elastic version adds more overhead compared to the elastic version of AES [5] due to the lack of whitening in MISTY1.

The elastic version of AES [2] was analyzed in [5]. Since AES contains per round whitening, the end of round swap step was the determining factor in how the performance compares between AES and Elastic AES. Furthermore, the manner in which the bits are swapped is straightforward in that sequential bits are chosen with the starting position rotating within the first 128 bits. In contrast, the elastic version of MISTY1 requires adding initial whitening and end of round whitening, and due to the left and right halves of the 64 bit block being processed differently, the bits chosen for the swap alternate starting positions between the left and right halves. MISTY1 is also of interest in that it can be viewed as having two round functions, each used in alternate rounds. When creating the elastic version of MISTY1, two rounds from regular MISTY1 are viewed as a single round which is then augmented to contain end of round whitening and the swap step.

In addition to proper selection of bit positions for the swap step, we discuss the use of an initial key dependent permutation and mixing of bits in order to allow all bits to impact the output of the round function in the first round. This complicates differential cryptanalysis [4] by eliminating the existence of a first round differential that occurs with a probability of 1 in the elastic version of any block cipher. Likewise, appending such steps to the end of the cipher prevents a differential from ocurring with a probability of 1 when performing a differential attack that starts with the ciphertexts and uses differentials for decryption.

The remainder of this paper is organized as follows. Section 2 briefly reviews the method for constructing an elastic block cipher and the MISTY1 block cipher. Section 3 discusses general rules for selecting which bits to swap. Section 4 describes the elastic version of MISTY1 and illustrates the impact of careful selection by determing bounds on differential probabilities for Elastic MISTY1. Section 5 discusses the benefits of initial and final key dependent permutations and mixing of bits. Section 6 summarizes the performance of Elastic MISTY1 compared to regular MISTY1. Section 7 concludes the paper.

2 Background

2.1 Elastic Block Cipher Algorithm

We review the algorithm from [5] for modifying the encryption and decryption functions of existing block ciphers to accept blocks of size b to $2b - 1$, where b is the block size of the original block cipher. The algorithm was designed such that it neither modifies the round function of the block cipher nor changes the number of rounds applied to each bit, but rather creates a method by which bits beyond the supported block size can be interleaved with bits in the supported block size. Additional key material beyond that generated by the block cipher's key schedule is required due to end of round whitening applied to all bits and optional key dependent permutations. The exact key schedule for the elastic version of the cipher will depend on the block cipher. Options from [5] include include modifying the cipher's original key schedule to provide the extra key bits, using an existing stream cipher as the key schedule or a combination of the cipher's key schedule and a stream cipher.

Figure 1 from [5] illustrates the general structure of the elastic block cipher using AES as the original cipher. The following notation and terms will be used in the description of the elastic block cipher.

Notation:

- G denotes any existing block cipher that is structured as a sequence of rounds.
- r denotes the number of rounds in G.
- b denotes the block length of the input to G in bits.
- P denotes a single block of plaintext.
- C denotes a single block of ciphertext.

- y is an integer in the range $[0, b-1]$.
- G' denotes the modified G with $b+y$ bit input for any valid value of y. G' will be referred to as the elastic version of G.
- G'_{b+y} denotes G' for a specific value of y.
- r' denotes the number of rounds in G'.
- k denotes a key.
- rk denotes a set of round keys resulting from the key expansion.
- G_k and G_{rk} will refer to G with the round keys resulting from expanding key k, and to G with the round keys rk, respectively.

Terminology:

- A bit (position) input to a block cipher is called *active* in a round if the bit is input to the round function. For example, in DES [1] $\frac{1}{2}$ of the bits are active in each round, while in AES all bits are active in each round.
- The round function will refer to one entire round of G. For example, if G is a Feistel network, the round function of G will be viewed as consisting of one entire round of the Feistel network as opposed to just the function used within the Feistel network.

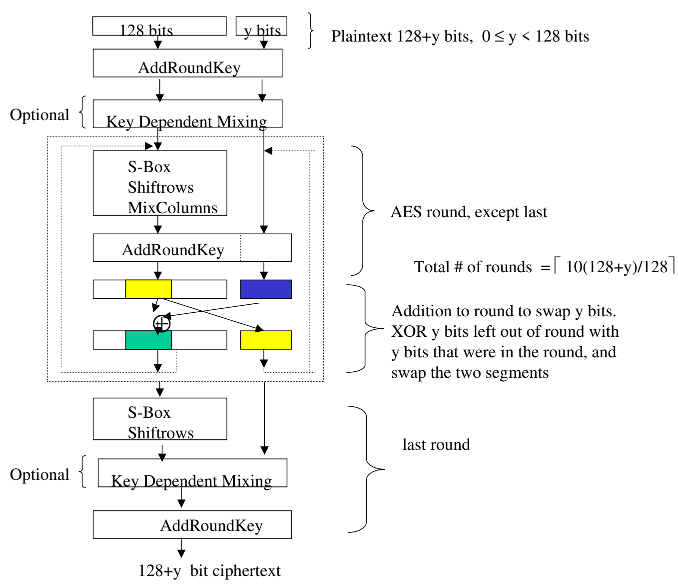

Figure 1: **Elastic Version of AES**

Given G and a plaintext P of length $b+y$ bits, make the following modifications to G's encryption function to create the encryption function of G':

1. Set the number of rounds, r', such that each of the $b+y$ bits is input to and active in the same number of rounds in G' as each of the b bits is in G. $r' = \lceil (b+y)(r)/b \rceil$.

2. XOR all $b+y$ bits with key material as the first step. If G includes whitening as the first step prior to the first round, the step is modified to include $b+y$ bits. If G does not have an initial whitening step, this step is added to G'.

3. (Optional) Add a simple key dependent mixing step that permutes or mixes the bits in a manner that any individual bit is not guaranteed to be in the rightmost y bits with a probability of 1. This will be referred to as the mixing step and it is viewed as the identity function if it is omitted. Similarly, a final key dependent mixing step may be added.

4. Input the leftmost b bits output from the mixing step into the round function.

5. If the round function includes XORing with key material at the end of the round and/or as a final step in the algorithm, the whitening should be performed on all $b+y$ bits. If G does not contain end-of-round whitening and/or whitening as the last step in the algorithm, add these whitening steps and apply them to all $y+b$ bits.

6. Alternate which y bits are left out of the round by XORing the y bits left out of the previous round with y bits from the round's output then swap the result with the y bits left out of the previous round. Specifically:

   (a) Let Y denote the y bits that were left out of the round.

   (b) Let X denote some subset of y bits from the round's output of b bits. A different set of X bits (in terms of position) is selected in each round. How to best select X depends on the specific block cipher.

   (c) Set $Z = X \oplus Y$.

   (d) Swap Z and Y to form the input to the next round.

   This step will be referred to as "swapping" or the "swap step" and may be added to the last round if we require that all rounds be identical. However, having the swap in the last round does not imply additional security.

The result, G', is a permutation on $b+y$ bits. Its inverse, the decryption function, consists of the same steps with the round keys applied in the reverse order and the round function replaced by its inverse, if it is not its own inverse.

2.2 MISTY1

The following summarizes the general structure of MISTY1 which is shown in Figure 2. The cipher is a Feistel network with the addition of a function applied to each half of the data at the start of the odd numbered rounds, thus the round function differs between even and odd numbered rounds. Refer to [7] and [8] for the definitions of the individual components. While the number of rounds is not fixed, eight rounds are recommended [3]. The block size is 64 bits.
Notation:

- L_i and R_i denote the left and right halves of output, respectively, of the i^{th} round after the halves are switched with $i = 0$ denoting the input to round 1.

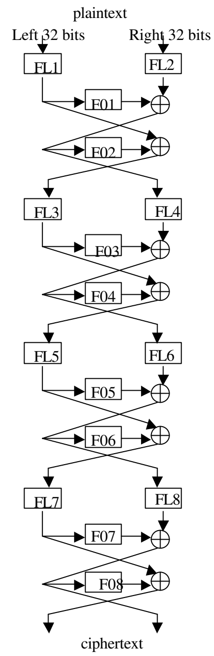

Figure 2: **MISTY1**

- $FL(x, KL)$ and $F0(x, K0, KI)$ are functions taking bit string x and key material $KL, K0, KI$.
- FL_i and $F0_i$ denote the i^{th} occurrence of FL and $F0$, respectively.
- KL, KI and $K0$ denote subkeys from the expanded key material, with a subscript of i denoting the i^{th} component.

The output of odd numbered rounds is defined by:

$R_i = FL_i(L_{i-1}, KL_i)$
$L_i = FL_{i+1}(R_{i-1}, KL_{i+1}) \oplus F0_i(R_i, K0_i, KI_i)$

The output of even numbered rounds is defined by:

$R_i = L_{i-1}$
$L_i = R_{i-1} \oplus F0_i(R_i, K0_i, KI_i)$

3 Bit Swapping

Since the general algorithm for converting block ciphers into elastic block ciphers was first described, only elastic AES has been described and analyzed [5]. In the round function of AES each step is applied to every bit input to the round function. Specifically, all of the 128 bits input to the round function have the SubBytes, Shiftrows and MixColumns operations applied. In ciphers where all input bits are equally impacted by the round function in each round, the swap step may be implementable by selecting a consecutive sequence of bits from the round function's output to swap with the y bits left out of the round, with the starting position rotating amongst the byte positions in the leftmost b bits.

Here we look at how to implement the swap step in ciphers where input bits to the round function are treated differently. Regardless of where the swap step is added, we note that the bit positions chosen for the swap step should vary amongst the leftmost b bits in order to avoid some bit positions from being involved in multiple swaps while others are involved in no swaps and to maintain some sense of ”equality” amongst bit positions. We first consider the case of a block cipher with a Feistel network as the underlying structure, *e.g.* as found in DES. We use the term half to refer $\frac{b}{2}$ of the input bits to the round function as opposed to half of the entire $b+y$ bit data block in the elastic version. In a basic Feistel network, only half of the input bits to the round are fed into the function, with the output being XORed with the bits left out of the function. If the swap step is added after every round, the bits swapped out would have to be swapped back into the same half they were in when swapped out. If the starting bit position within the b bits for the swap varies and if $y > \frac{b}{2}$ then some bits will end up in the wrong half compared to where they would be in the original cipher. This will result in some bits participating in the left half or right half more than required and not participating in the other half the required amount of times.

Since the requirement is that each bit participates in the round function and be acted upon in the same manner at least the same number of times as in the regular version of the block cipher, we decide where to add the swap step by determining the series of steps through which each bit has each component of the round function applied to it then add the swap step and whitening at this point. In the case of a Feistel network, the swap step and whitening is added after every two rounds as shown in Figure 3. When the swap occurs after every two rounds, each of the leftmost b bits participates in the actions applied to each half once prior to potentially being swapped out. When computing the number of rounds for the elastic version, at most one additional round will be needed when inserting the swap after each pair of rounds compared to when inserting it after every single round. For example, when $y = 1$, one additional round is required if the swap was added after single rounds; whereas, one additional pair of rounds in now required.

In general, if parts of the bits input to the round function are treated differently it may be feasible to define a round to consist of all the steps in the regular cipher until a bit receives each part of the treatment and add the swapping and end-of-round whitening at this point rather than determine how to select bits appropriately for the swap step in order to maintain each bit being processed by all components of the round function. We emphasize that each cipher must be treated individually. It may be the case that the bit positions for the swap step can be selected in a manner that allows the swap step to be added after each original round to minimize the rounds needed for the elastic version.

4 Elastic MISTY1

4.1 Overview

We first describe how the swap step and end of round whitening are added to create the elastic version of MISTY1. MISTY1 does not use the same round function in every round, instead alternating between round functions by including FL only in odd number rounds and $F0$ in every round. We use the term original round to refer to the round function as defined in the regular version of MISTY1. By defining a round to consist of a pair of original rounds, we will view a round as consisting of the application of FL followed by two applications of $F0$. In the elastic version of a cipher, every bit position must pass through the round function at least the same number of times as a bit in the regular version of the cipher. Regardless of whether MISTY1's round function is viewed according to its original definition or as we define it, the same number of $F0$ and FL applications are added. Furthermore, we require that each bit position pass through the FL and $F0$ functions in the same order as in regular MISTY1, meaning a bit output from FL applied to the right

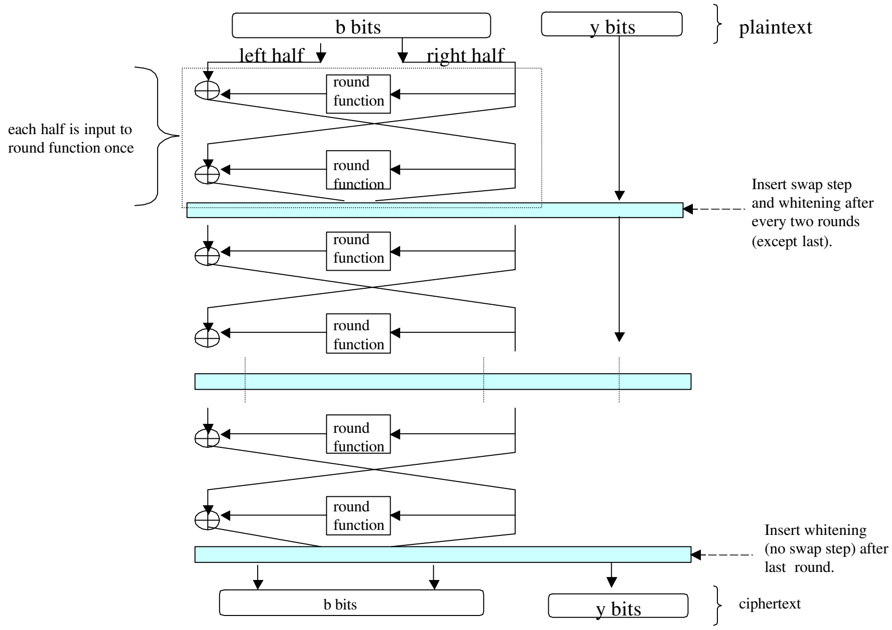

Figure 3: **Insertion of Swap Step and Whitening in Feistel Network**

half of the round function input is XORed with the output of $F0$ to form the input to the second occurence of $F0$. It is difficult to insure this if the elastic version is defined to position the swapping of bits at the end of each original round. A bit swapped out of one half of the round function's output may be swapped back into the half where it would normally not be in regular MISTY1, thus resulting in some bits serving as the left half of input to the round function more than the right half or vice versa.

By viewing a round as two original rounds and positioning the swap step after each such round, every bit ends up in the left half and right half of input to the round function at least the same number of times as occurs in regular MISTY1. Therefore, elastic MISTY1 is created by adding the swap step and whitening after every two original rounds. The round function in the elastic version is the function shown in Figure 4. Notice that we do not modify MISTY1's round function, but merely redefine where a round begins and ends. The bit positions from a round's output that are involved in the swap step will vary across each round to avoid some bit positions from being swapped every round while others are never involved in the swap. We choose to alternate the starting position for the swap between the left and right halves of the round's output and, within each halve, rotate the starting position one byte each time. The bit positions swapped are chosen sequentially from the entire 64 bit round output from the starting position. Since the round function is two original rounds, regardless of where a bit was positioned when swapped out and where it is when it is swapped back in, the bit will end up in the left and right halves of the original round function the same number of times as in regular MISTY1. Furthermore, since the swap XORs the bits being swapped out with those being swapped in, the bits swapped out continue to influence the next round in the same positions they would have influence had they not been swapped out.

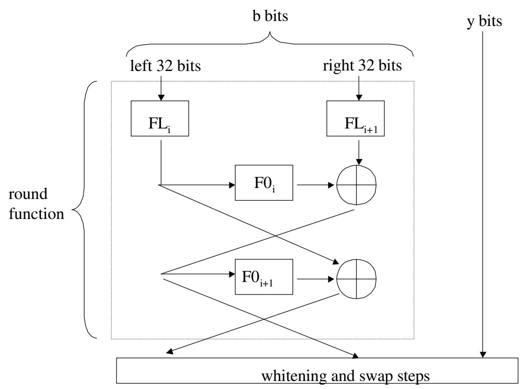

Figure 4: **Round Function for Elastic Misty**

4.2 Differential Cryptanalysis

In order to illustrate the need to carefully select how bits are swapped in and out of the data segment input to the round function, we determine upper bounds on the probabilities of a differential occurring in the elastic version of MISTY1. We conclude it is best if the first swap impacts the left half of the 64 bit input to the round function in order to minimize the upper bound.

Notation:

- FL and F0 refer to the functions within a round of MISTY1 as shown in Figure 2.
- b refers to the number of bits, 64, in regular MISTY1.
- y refers to the number of bits beyond 64 in the data block. $0 \leq y < 64$.
- A single round will be defined as the application of FL to each half of the leftmost 64 bits of data and two applications of F0. The swap step will occur after the second application of F0 as indicated in Figure 5.
- Round input and output refers to the inputs and outputs of MISTY1's round function unless otherwise stated.
- Δ indicates the difference between two bit strings.
- ΔL and ΔR refer to the Δ for the left and right halves of input to the round function, respectively.
- $\Delta outL$ and $\Delta outR$ refer to the Δ for the left and right halves of output from the round function prior to the swap.
- ΔY refers to the Δ for the y bits left out of a round.
- An i after any Δ indicates the i^{th} round.

Assumptions:

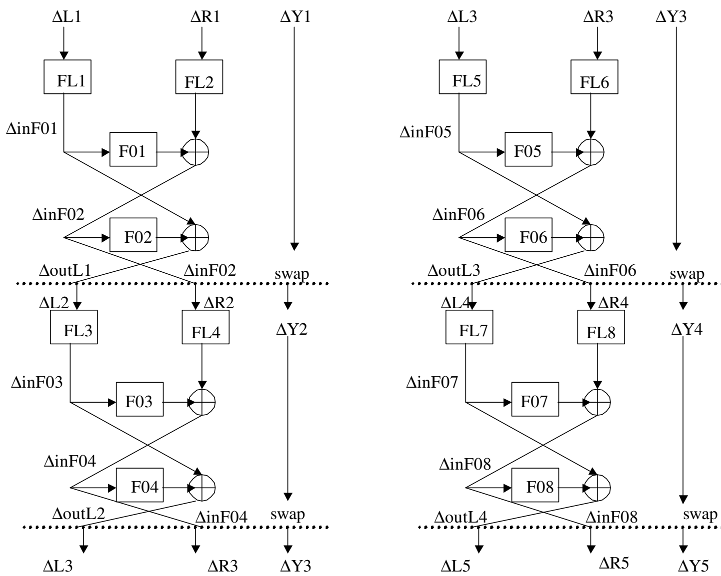

Figure 5: **Differentials for Elastic Misty**

- We assume the probabilities for per round differentials in the regular version of MISTY1 are independent of other rounds. Specifically, the probability $F0$ produces a specific differential in round $i+1$ is independent of the differential produced by $F0$ in round i.

Facts:

- Fact 1: The probability a specific differential occurs as the output of $F0$ is $\leq 2^{-14}$. [6]
- Fact 2: FL does not influence the probability of a differential. This follows directly from the definition of FL in [8].
- Fact 3: A non-zero differential in one half of the round's input impacts both halves of the round's output. This follows directly from the definition of MISTY1.
- Fact 4: If $\Delta Li = 0$ and $\Delta Ri \neq 0$, the probability of a differential occurring in round i is $\leq 2^{-14}$ because there is a non-zero delta input only to the second $F0$ in the round.
- Fact 5: If $\Delta Li \neq 0$ and $\Delta Ri = 0$, the probability of a differential occurring in round i is $\leq 2^{-28}$ because there is a non-zero delta input to both $F0's$ in the round.
- Fact 6: The probability of a differential occurring in MISTY1 with 8 original rounds is $\leq 2^{-56}$. [7]

Claim 1: In the elastic version of MISTY1, once a non-zero delta occurs as input to a round, the delta for the $b+y$ bits cannot become 0.

Proof: The following is independent of which bits from the leftmost 64 bits are involved in the swap step.

Case 1: Suppose for round i that $\Delta Li \parallel \Delta Ri \neq 0$ and $\Delta Yi = 0$, then all output bits of the round are impacted by ΔRi. This will lead to a non-zero input to round $i+1$. There will be a non-zero delta output of round $i+2$. If by chance the swap step involving $\Delta Y(i+1)$ and the output of round $i+2$ creates $\Delta L(i+3) \parallel \Delta R(i+3) = 0$, then $\Delta Y(i+3)$ must be non-zero and $\Delta L(i+4) \parallel \Delta R(i+4) \neq 0$, $\Delta Y(i+4) = 0$. Otherwise, $\Delta L(i+3) \parallel \Delta R(i+3) \neq 0$.

Case 2: Suppose for round i that $\Delta Yi \neq 0$ and $\Delta Li \parallel \Delta Ri = 0$. Then $\Delta Y(i+1) = 0$ and $\Delta L(i+i) \parallel \Delta R(i+1) \neq 0$. Then case 1 applies for round $i+1$.

Case 3: $\Delta Yi \neq 0$ and $\Delta Li \parallel \Delta Ri \neq 0$. If the swap step involving ΔYi and the output of round i creates $\Delta L(i+1) \parallel \Delta R(i+1) = 0$ then $\Delta Y(i+1)$ must be non-zero and case 2 applies starting at round $i+1$ Otherwise, $\Delta L(i+1) \parallel \Delta R(i+1) \neq 0$ and either case 1 or case 3 applies.

In every case, the delta for the $b+y$ bits cannot become 0 and the claim holds.

As explained previously, the swap step is designed such that y consecutive bits from the leftmost 64 bits are swapped with Yi, wrapping around to the leftmost bit as needed, and the starting position alternates between the left and right halves of the input to the round function. We are not determining whether it is possible for the y bits swapped in to result in a delta of 0 in either the left or right half of input to the round function, but instead determine bounds assuming this could occur. Recall that the probability a specific differential occurs in round 1 is $\leq 2^{-14}$ per Fact 4.

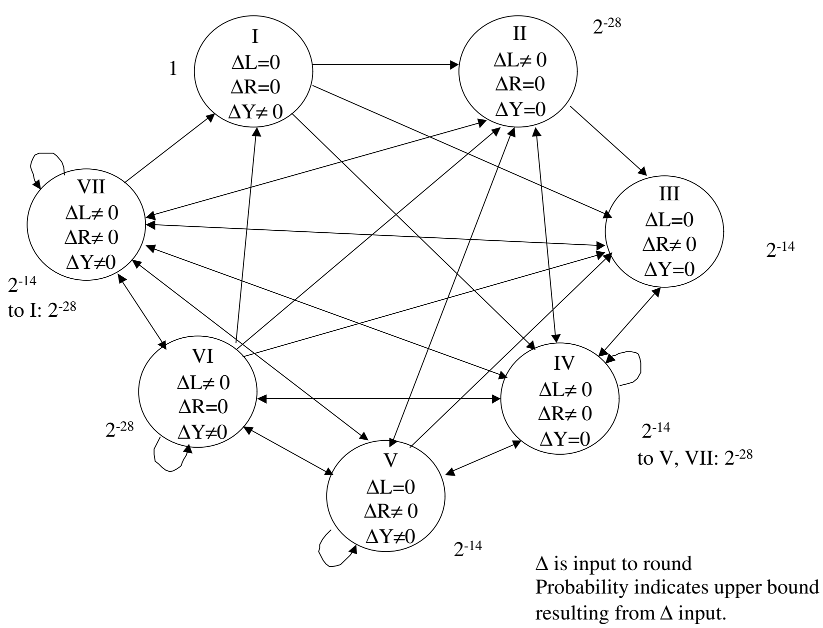

Figure 6: **Differential States**

The possible transitions between the states shown in Figure 6 are indicated in Table 1. In Figure 6, the value next to each state is the upper bound on the probability a specific differential occurs regardless of the next state. In two cases, the bound is lower if a specific next state occurs and is listed separately along with the corresponding next state(s). For example, State VII has an upper bound of 2^{-14} except when the next state is State I. The specific differential that must occur to move from State VII to State I occurs with probability $\leq 2^{-28}$.

The state transitions assume all outcomes are possible in theory. Specifically, in cases where $\Delta outL \neq 0$ and $\Delta outR \neq 0$, we make no assumptions on the bits swapped and assume it is possible that the y bits

From State	To State
I	II,III,IV
II	III,IV,V,VII
III	IV,VII
IV	II,III,IV,V,VI,VI
V	II,III,IV,V,VI,VII
VI	all
VII	all

Table 1: **Differential State Transitions**

swapped from the round output can produce $\Delta Y = 0$ (except in the case where $y = 63$ since then either all of $\Delta outL$ or all of $\Delta outR$ becomes part of ΔY which we omit indicating since it only serves to reduce the bounds for one specific value of y). We note that as y approaches 63, it is unlikely that the majority of bits in $\Delta outL \parallel \Delta outR$ are 0, as required to obtain $\Delta Y = 0$ entering the next round. As y increases, it is also unlikely that a non-zero ΔY will result in $\Delta L = 0$ and/or $\Delta R = 0$ in the input to the next round.

State I is the only state for which a differential holds through the round with probability 1. We first consider the case when $0 < y \leq 32$. There will be 5 or 6 rounds, depending on the exact value of y. There can be at most two occurrences of state I, and for any occurence of state I beyond the initial state, a state with probability bounded by 2^{-28} must occur immediately before it. Let "X" denote any state other than state I that has an upper bound of 2^{-14}, "S" denote any state that has an upper bound of 2^{-28} (S $\subset$ X) and "I" denoting state I, the following 5 state sequences are possible in theory:

XXXXX, XXXSI, XXSIX, XSIXX, SIXXX, IXXXX, SIXSI, IXXSI, IXSIX

Therefore, an upperbound on the probability a specific differential occurs in 5 rounds is 2^{-56} due to the sequences IXXXX, IXXSI and IXSIX if no restriction is placed on what bits are involved in the first swap step. If the y bits from the first round's output are taken from the left half, then from state I it is only possible to move to state II, and the sequence begins with IS, decreasing the bound to 2^{-70}. With no restrictions on the bits used for the first swap step, the upper bound for 6 rounds is 2^{-70} because appending one more state to any of the above sequences results in either adding X to any sequence or, when the 5 state sequence ends in XX, changing the 5^{th} state to an S and appending an I. If the first swap is restricted to using the left half of the round's output (starts with first bit of left half) , then the bound decreases to 2^{-84}.

When $32 < y < 64$ there will be 7 or 8 rounds depending on the exact value of y. We determine the upper bounds by considering two state sequences, corresponding to two additional rounds, that can be appended to five and six state sequences. To any sequence ending in XX, the sequences of XX and SI can be appended or the last X can be changed to an S and IX appended. To any sequence ending in I, only XX can be appended. To any sequence ending in IX, the sequences of XX and SI can be appended. In all cases, the addition of two additional states involve a bound of 2^{-28}. Therefore, the upper bound on a differential for 7 rounds is 2^{-98} and for 8 rounds is 2^{-112} when the first swap step starts at the first bit in the left half. Table 2 summarizes the bounds.

5 Key Dependent Permutation

We discuss the benefit of adding a key dependent mixing step after the initial whitening and prior to the final whitening when creating an elastic block cipher. In [5], these steps were listed as optional and not fully

$b+y$	r'	Upper Bound No Restrictions on First Swap	Upper Bound First Swap Starts at Beginning of Left Halve
65 to 80	5	2^{-56}	2^{-70}
81 to 96	6	2^{-70}	2^{-84}
97 to 112	7	2^{-84}	2^{-98}
113 to 127	8	2^{-98}	2^{-112}

Table 2: **Differential Upper Bounds**

explored. Here we explain the need for these steps and illustrate how even a trivial means of mixing the input bits will prevent a first round differential from occurring with a probability of 1. Let B = leftmost b bits of data, and Y = rightmost y bits. In the elastic version of a block cipher, the exclusion of Y from the first round allows for a first round differential that occurs with probability one by using inputs with $\Delta B = 0$ and $\Delta Y \neq 0$. This means the first round does not contribute to preventing a differential attack. The existence of such a differential is due entirely to the fact that the adversary knows what bits are excluded from the first round. Mixing the input bits such that the adversay nolonger knows with a probability of 1 where the non-zero Δ occurs eliminates the problem. Any key dependent mixing step should take less time than the round function; otherwise another round can be added if needed to further decrease the probability of a differential since all bits influence the output of the second round due to the XOR in the swap step.

First we consider a key dependent permutation. If a key dependent permutation is done prior to the first round, the probability of a single byte Δ in $B \parallel Y$ ending up in any specific byte position is $1/\frac{b+y}{8} = \frac{8}{b+y}$ and the probability of it being in Y is proportional to $\frac{y}{b+y}$. Therefore, a key dependent permutation will help most in preventing a $\Delta B = 0$ input to the first round when y is small. If $y = b - 1$, the probability of the single byte Δ remaining in ΔY and not be in ΔB after the permutation is approximately $\frac{1}{2}$.

Second, we consider a trivial means of mixing bytes to ensure a zero differential in B occurs with probability strictly $< \frac{1}{2}$ in the input to the first round. We accomplish this by replacing y bits in B with their XOR with Y. Consider the case where there is a single byte with a non-zero Δ in the input. After the permutation, if the byte with the non-zero Δ was in B, it will still be in B after the XOR. If it was in Y, it will remain in Y and result in $\Delta B \neq 0$. There still exists the potential for $\Delta B = 0$ in the input to round 1, for example if the original data contains some single byte Δz that occurs twice. However, in this case when the mixing operations are performed at the byte level, the probability a Δz is in any byte in B equals $\frac{2b}{b+y}$ and the probability the other Δz ends up in a specific byte in Y equals $\frac{8}{b+y}$. Thus the probability $\Delta B = 0$ after the mixing is less then or equal to $\frac{16b}{(b+y)^2}$. Any inputs that differ in more than two bytes will have a lower chance of creating a zero Δ in the leftmost b bits input to the first round function.

Overall, the permutation combined with the XOR will eliminate the ability of an attacker to set the differential in round 1 with probability 1. The key dependent permutation and XOR collectively are the key dependent mixing step. When the mixing step is present, a single byte non-zero differential in the original input will result in a non-zero Δ in the input to the first round function with probability 1. Any non-zero differential involving more than one byte in the original input will produce a zero Δ in the input to the first round function with probability $\leq \frac{16b}{(b+y)^2}$. For the same reason a key dependent mixing step is needed prior to the first round, one should be added after the last round prior to the final whitening step. This will prevent a differential which occurs with probability of 1 in the last round of encryption (first round of decryption), which is useful for a differential attack which starts with the ciphertexts and works towards the plaintexts. These key dependent mixing steps are only recommended when there is a need to decrease the probability an attacker knows the exact bit positions used in the first rounds of encryption and decryption. Adding rounds will also lower the differential probability and is a viable alternative if the round function can be performed faster than a key dependent permutation and XOR.

6 Elastic MISTY1 Implementation

We implemented elastic versions, with and without the optional key dependent permutations, and the regular version of MISTY1 in C. The elastic versions accommodate block sizes of 64 to 127 bits. We use RC4 with the first 512 bytes discarded [9, 10] for the key schedule instead of modifying MISTY1's key schedule to provide the additional expanded key bits required by the elastic version. The purpose of this work is to determine the impact on the encryption and decryption rates due to the addition of whitening, key dependent permutations and swap steps required by the elastic version. The exact key schedule is not the focus and can be replaced with another stream cipher, a modification of MISTY1's key schedule or a combination of MISTY1's key schedule to provide key material for the round function and a stream cipher to provide key material for whitening. The swap step was added after every two rounds and the knowledge gained from the differential analysis was used to determine which bits to swap.

Both the elastic and regular versions of MISTY1 were tested in Redhat Linux 9 and Windows XP environments with Intel Centrino, Pentium 3 and Pentium 4 processors varying from 1Ghz to 2Ghz to compare their encryption and decryption rates. The results in terms of how large the block size can be in the elastic version while remaining more efficient than padding a second block and encrypting two blocks in MISTY1 are consistent across the environments. The elastic versions increased the number of operations beyond the 64 bit version of MISTY1 due to the whitening and the swapping steps, and the key dependent permutations in one version.

b+y	Elastic MISTY1 with Permutations	Elastic MISTY1 without Permutations
65 to 72	110	121
73 to 80	103	115
81 to 88	84	91
89 to 96	80	88
97 to 104	66	71
105 to 112	63	67
113 to 120	53	57
121 to 127	51	55

Table 3: **Elastic MISTY1's Encryption Rate as a Percentage of MISTY1's Encryption Rate**

In the tests, the data to be encrypted was viewed as individual $b + y$ bit blocks. The elastic version of MISTY1 encrypted each block individually with no padding. To encrypt the data with regular MISTY1, the $b + y$ bits were padded to $2b$ bits and encrypted as two b bit blocks. The information presented here summarize the results from a C implementation compiled with Visual C++ 6.0 on a 1.8Ghz Pentium 4 processor running Windows XP. When measuring the encryption rate in terms of blocks per second, MISTY1's rate for a single block was based on the time to encrypt 16 bytes to represent the padding required when using MISTY1 for $b + y$-bit blocks. The rates are determined by the time to encrypt one million $64 + y$ bit blocks using the elastic version and two million 64 bit blocks using the original version of MISTY1. The times for the original version of MISTY1 exclude the time to pad the $b + y$ bits to $2b$ bits. Table 3 summarizes how elastic versions with and without the key dependent permutations compare to the original version for 8 bit intervals of $b+y$. Without the key dependent permutations, the elastic version's rate for encrypting $b+y$ bit blocks ranges from 121% of MISTY1's rate when $y=1$ to 55% of MISTY1's rate when $y=63$. By adding the two key dependent permutations, the elastic version's rate decreased slightly compared to the version wihout the permutation, ranging from 110% of MISTY1's rate when $y=1$ to 51% of MISTY1's rate when $y=63$. For each value of $b+y$, the decryption rates in terms of MB/sec are virtually identical to the encryption rates and thus the ratios between encryption rates of the elastic and original versions also represent their decryption ratios.

7 Conclusions

We discussed the construction of an elastic block cipher defined in [5] as it applies to block ciphers based on Feistel networks. For Feistel networks, we explained why adding the swap step after every two rounds is preferable to adding it after every round. MISTY1, which can be viewed as a Feistel network with the round function alternating between even and odd numbered rounds, is used as an example. Using the differentials for MISTY1, we illustrated why it is necessary to carefully select bit positions used for the swap step when creating the elastic version of a cipher. An elastic version of MISTY1 was implemented and its performance compared to that of MISTY1. We also discussed the benefits of initial and final key dependent mixing steps to eliminate the ability to create a single round differential which occurs with a probability of 1 in the first round of encryption and decryption in elastic block ciphers due to the y bits left out of the round. Future work includes analyzing elastic versions of block ciphers against attacks other than differential based attacks.

References

[1] FIPS 46-3 Data Encryption Standard (DES), 1999.

[2] FIPS 197 Advanced Encryption Standard (AES), 2001.

[3] NESSIE Security Report, Version 2. `https://www.cosic.esat.kuleuven.ac.be/nessie`, February 2003.

[4] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, New York, 1993.

[5] D. Cook, M. Yung, and A. Keromytis. Elastic Block Ciphers. Technical Report CUCS-010-04, Columbia University, February 2004.

[6] M. Matsui. New Structure of Block Ciphers with Provable Security Against Differential and Linear Cryptanalysis. In *Workshop on Fast Software Encryption - FSE '96, Lecture Notes in Computer Science 1039, Springer-Verlag*, pages 205–218, 1996.

[7] M. Matsui. New Block Encryption Algorithm MISTY. In *Workshop on Fast Software Encryption - FSE '97, Lecture Notes in Computer Science 1267, Springer-Verlag*, pages 54–68, 1997.

[8] M. Matsui. Specification of MISTY1 - a 64-bit Block Cipher. Manuscript, Mitsubishi Electric Corporation, September 2000.

[9] I. Mironov. (Not So) Random Shuffles of RC4. In *Proceedings of Advances in Cryptology - Crypto 2002, LNCS 2442, Springer-Verlag*, 2002.

[10] B. Schneier. *Applied Cryptography*. John Wiley and Sons, New York, 1996.