Elastic Block Ciphers

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

Technical Report 2/24/2004

Abstract

We introduce a new concept of *elastic block ciphers,* symmetric-key encryption algorithms that for a variable size input do not expand the plaintext, (*i.e.,* do not require plaintext padding), while maintaining the diffusion property of traditional block ciphers and adjusting their computational load proportionally to the size increase. Elastic block ciphers are ideal for applications where length-preserving encryption is most beneficial, such as protecting variable-length database entries or network packets.

We present a general algorithmic schema for converting a traditional block cipher, such as AES, to its elastic version, and analyze the security of the resulting cipher. Our approach allows us to "stretch" the supported block size of a block cipher up to twice the original length, while increasing the computational load proportionally to the block size. Our approach does not allow us to use the original cipher as a "black box" (*i.e.,* as an ideal cipher or a pseudorandom permutation as is used in constructing modes of encryption). Nevertheless, under some reasonable conditions on the cipher's structure and its key schedule, we reduce the security of the elastic version to that of the fixed size block cipher. This schema and the security reduction enable us to capitalize on secure ciphers and their already established security properties in elastic designs. We note that we are not aware of previous "reduction type" proofs of security in the area of concrete (*i.e.,* non "black-box") block cipher design. Our implementation of the elastic version of AES, which accepts blocks of all sizes in the range 128 to 255 bits, was measured to be almost twice as fast when encrypting plaintext that is only a few bits longer than a full block (128 bits), when compared to traditional "pad and block-encrypt" approach.

Keywords: Cipher Design, Variable Length Block Cipher, Encryption Algorithm, Security Proofs.

1 Introduction

Block ciphers typically support a small number of block sizes, usually just one. Since the length of the data to be encrypted is often not a multiple of the block size, plaintext-padding schemes are necessary. Although widely used in practice, padding imposes additional computational and space overheads to the encryption process. Furthermore, certain applications require that the length of the protected data remain the same. A natural alternative, using a stream cipher, is not always attractive since it sacrifices data and key diffusion and it further requires synchrony between sender and receiver, which is an unsuitable assumption for many applictions. The ideal solution combines the length-preserving aspects of stream ciphers with the diffusion properties of block ciphers and should use existing and well analyzed components or algorithms, to leverage prior work as much as possible.

We introduce a new concept of an *elastic block cipher,* which allows us to "stretch" the supported block size of a block cipher up to a length double the original block size, while increasing the computational load proportionally to the block size. This permits block sizes to be set based on an application's requirements, allowing, for example, a non-traditional block size to be used for all blocks or a traditional block size to be used for all but the last block. Such as cipher will be very useful in database and network traffic encryption, as well as other applications of encryption. We propose elasticity of block size as a criterion for cipher design, creating areas for future work, including the analysis of elastic versions of ciphers and security against cryptanalytic techngiues not addressed here.

In this paper, we propose and analyze a general method for creating an elastic block cipher from an existing block cipher. Our intent is not to design an *ad-hoc* new cipher, but to systematically build upon existing block ciphers. We neither modify the round function of the block cipher nor decrease the number of rounds applied to each bit, but rather create a method by which bits beyond the supported block size can be interleaved with bits in the supported block size. Our method takes a novel approach that permits a reduction to be formed between the elastic and original cipher, allowing us to relate the security of the elastic version to that of the fixed-size "traditional" version. We deal both with security against key recovery and security against distinguishability (from a random permutation) attacks. We are not aware of existing proof methods that argue about sub-ciphers in the area of concrete block cipher design. The importance of such a proof is that it allows one to exploit the established or provable properties of existing ciphers.

We have implemented elastic versions of AES [2], MISTY1 [18] and RC6 [20]. We include a description, a discussion of security including differential cryptanalysis, and performance results of the elastic version of AES accepting blocks of size 128 to 255 bits. The elastic version of AES is almost twice as fast as regular AES using padding when the data to be encrypted is only slightly longer than one block. Naturally, as the length of the plaintext increases towards twice the block size, the speedup of the elastic AES decreases relative to two applications of regular AES (using plaintext-padding to encrypt two full blocks). We briefly discuss some applications where the gains in space and performance translate to improved bandwidth or storage utilization.

There has been little previous work on variable-length PRFs. The focus has been on variable-length inputs with fixed length outputs as applicable to MACs and hash functions [3, 4, 6, 9] and, more recently, on modes that work on multiples of the original block length [12, 13]. While there have been proposals for variable-length block ciphers in the past, such as [23], we take a different approach in that we do not wish to design a new cipher but rather provide a mechanism for converting existing block ciphers into variable length block ciphers. A noteworthy proposal for a variable length-block cipher created by converting any existing block cipher into one that accepts variable size block lengths is [5], which demonstrates that the problem we deal with has been noticed. The method in [5] involves two applications of the cipher for block lengths between the original length and less than twice that length. Therefore, the resulting cipher requires twice the work of the original block cipher regardless of the amount of data actually encrypted (*e.g.,* even if the data is one bit longer than the original block). In comparison, since we use a concrete design that modifies the structure of the cipher adapting it to the size, the workload in our construction gradually expands to twice that of the original block cipher as the block length expands (which was one of our goals). Unlike our construction, [5] does not modify the original block cipher but rather adds operations around it, treating the original cipher as a PRP and analyzes it as a black box. In contrast, we cannot treat the original cipher as a black box but need to add to its internals. This is, perhaps, the major methodological difference between the works. While we do not modify the round function of the original block cipher in our construction (and thus allows for reduction-type proofs), we alter the inputs to each round, add whitening steps when not already present and extend the key schedule.

The remainder of this paper is organized as follows. Section 2 explains our approach for constructing elastic block ciphers from existing block ciphers. Section 3 presents a security analysis for our scheme. Within this we introduce the concept of a reduction between ciphers to relate the security of the elastic and original versions of a cipher. Section 4 discusses our analysis and implementation of elastic AES, including bounds on differentials and preliminary performance measurements. Section 5 briefly describes some practical applications of elastic ciphers and Section 6 describes a new mode of encryption using the elastic version of a block cipher. Section 7 concludes the paper.

2 Elastic Block Cipher Construction

2.1 Algorithm

We begin with a description of the algorithm 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 resulting block cipher can accept blocks of length $2b$, but the original block cipher can be applied to two blocks without padding at this point.) We neither modify the round function of the block cipher nor decrease the number of rounds applied to each bit, but rather create a method by which bits beyond the supported block size can be interleaved with bits in the supported block size. The reasoning behind specific steps is presented afterwards. It is assumed the appropriate amount of key material is available. The exact key expansion algorithm will depend on the specific block cipher so we skip that discussion for now; we describe its properties later on. In this paper we focus on the basic algorithm independent of the original block cipher. Subtleties specific to particular types of block ciphers, such as those using Feistel networks, are noted and are explained more fully in subsequent work. Figure 1 illustrates the resulting cipher when the modifications are applied to the version of AES that accepts 128-bit inputs.

The following notation and terms will be used in the description and analysis 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.

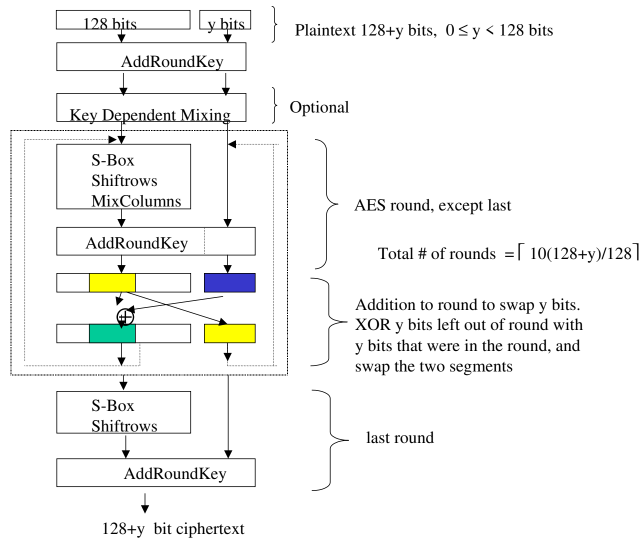

Figure 1: **Elastic Version of AES**

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 (the original b bits "inner block" whitening for the leftmost bits and also "side whitening" of the y bits). If G does not have an initial whitening step, this step is added to G'. In either case, additional bits of expanded key material are required beyond the amount needed for G (for the side whitening and perhaps the added inner block whitening).

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.

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

5. If the round function includes XOR 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 (inner-block as well as side whitening). 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. In either case, additional bits of expanded key material are required beyond the amount required by G.

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 is dependent on the specific block cipher. This is discussed further in Section 2.2.
   (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. We also point out a subtlety exists regarding which rounds the swap step is appended to that simplifies the selection of which of the leftmost b bits to swap. We discuss this briefly in Section 2.2.

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 Explanation of Algorithm

The method was designed for G' to be equivalent to G (with the possible addition of a mixing step and whitening) when the data is an integral number of b bits blocks, while accommodating a range of b to $2b - 1$-bit blocks. The following is an explanation of why specific steps are included in the construction.

Step 1: Each bit position of the input is required to be active in the same number of rounds in G' as the number of rounds in which each bit is active in G. The reason for this requirement is to avoid a reduced round attack on G from being applied to G'. Consider what happens if $y = b - 1$ and no rounds were added to G when creating G': $b - 1$ bits would be active in only $\frac{1}{2}$ of the rounds in which a bit is normally active in G.

Step 2: The initial whitening is performed on all $b+y$ bits in order to prevent an adversary from learning y input bits to the second round in a known-plaintext attack.

Step 3: A key dependent permutation or mixing of bytes prior to the first round increases the extent to which the first round contributes to preventing a differential attack. In Section 4, Fact 4 in the differential analysis of elastic AES illustrates the outcome without this step. The mixing step will need to take less time than a single round; otherwise, an additional round can be added instead to decrease the probability of a specific differential occurring. A trivial mixing that prevents the attacker from knowing with probability 1 which y bits are excluded from the first round is a key dependent rotation. This quarantees any particular bit is within the y bits with probability $< \frac{1}{2}$.

Step 5: Including all $b + y$ bits in the whitening performed at the end of a round prevents an adversary from learning any of the output bits of the next-to-last round. Suppose the additional y bits were not included in the whitening; then the ciphertext, C, for the block would include the y bits that were excluded from the last round. Since no bit is excluded from 2 consecutive rounds, an adversary would be provided with y bits of output from the next to last round, potentially aiding in cryptanalysis. The placement of whitening when not already present (as opposed to expanding exisitng end of round whitening from b to $b + y$ bits) is linked to the placement of the swap step since it is the set of y bits left out of the last round that an adversary would otherwise learn.

Step 6: $X \oplus Y$ is performed instead of merely swapping X and Y in order to increase the rate of diffusion. If G does not have complete diffusion in one round, then at the end of the first round there is some subset S of bits output from the round that have not been impacted by some of the bits in X. While the bits in Y may impact S in the second round, swapping X and Y would result in the bits in X having no impact in the second round; whereas, swapping X with $X \oplus Y$ will allow the bits in X to impact the second round. The selection of X depends on the round function of the specific block cipher. The bit positions selected for X should vary amongst the rounds to ensure all bit positions are involved in both the b bit and y bit components as opposed to always selecting the same y positions for use in X. If all input bits to the round are utilized in the same manner, the bit positions chosen for X can be rotated across the rounds. For example, in AES all bytes are processed by the same functions within the round. In that case, it may be sufficient to select X to be consecutive bits starting at position $a_1 + a_2 * i (\mathrm{mod}\ b)$ in round i for some constants a_1 and a_2. If the input bits are treated differently in the round (for example in RC6 the input consists of 4 words of which one pair is operated on differently than the other pair), then swap the bits in a manner such that each bit participates in each pair the same number of times. Another benefit of the XOR is a reduction in the ability to specify a y-bit differential in the input to the second round. If the optional key dependent mixing step is omitted, then without the XOR a differential in round 2's input of y bits can be obtained with probability 100% regardless of the round function by choosing the rightmost y bits of the original input appropriately.

In subsequent work we elaborate on how to select the bits to be swapped (or adding the swap step less frequently) when the original cipher's round function is structured such that only a subset of the b bits are processed by the round function in each round or subsets of the b bits are processed differently by the round function. Here we briefly consider the case of block ciphers structured as Feistel newtorks. 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 end of round whitening, if needed, and swap step by determining the series of steps through which each bit has the round function applied to it and add the swap step and whitening, if not already present in the cipher, at this point. The same number of rounds is required in the elastic version when inserting the swap step and whitening in this manner as when inserting the swap step after each round. In the case of a Feistel network, the swap step and whitening is added after every two rounds so a bit participates in the actions applied to each half once prior to potentially being swapped out regardless of its position. In contrast, if the swap step is added after every round the bits swapped out will have to be swapped back into the same half they were in when swapped out. Since 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.

Decryption: The inverse of the round function must be used for decryption. While the structure is similar to an unbalanced Feistel network utilizing the round function of the block cipher, it is not a Feistel network because the bits omitted from the round function in the i^{th} round are XORed with the round function's output to form the input to the $(i+1)^{st}$ round; whereas, in the i^{th} round of a Feistel network the bits omitted from the round function are XORed with the input bits to the round function to form the input to the round function in round $i + 1$. [1] Thus, it is not possible to perform decryption by running the ciphertext through the encryption function of G' with the round keys used in reverse order. Incorporating the round function's output from the i^{th} round into the input to round $i + 1$ instead of using the input of the round function from round i was done to increase the diffusion rate.

[1] Within the context of our work, the term unbalanced Feistel network refers to a Feistel network in which the left and right parts are not of equal length as defined in [21]. The term "unbalanced Feistel network" has been used in at least one other context to refer to Feistel networks in which the input and output are of different lengths.

2.3 Key Schedule

Our obvious options in creating the key schedule for G' include modifying the key schedule of G to produce additional bytes or increasing the original key length and running the key schedule multiple times. A third, less obvious option is to use an existing efficient stream cipher that is considered secure in practice as all or part of the key schedule, independent of the choice of G. The stream cipher can either serve as the entire key schedule, replacing that of G, or provide only the additional key bits needed for whitening and the mixing step while using the original key schedule of G for all other expanded key bits. There are benefits to using a stream cipher as the key schedule. First, the output of the stream cipher may be more (pseudo)-random than the output of the original key schedule. If G's key schedule is highly structured in a manner that allows either key bytes or expanded key bytes to be easily determined from another part of the expanded key, replacing the key schedule with a stream cipher can eliminate this weakness. Second, the key schedule for G' (as a schema) would not have to be changed for new choices of G. Third, the stream cipher may provide a faster key expansion than G's original key schedule.

It should be noted that if the round keys are truly random, then for a single plaintext, P, there is more than one key that will produce the same ciphertext, C, from the plaintext. For example, arbitrarily select a set of round keys and encrypt P. To create a second set of round keys, arbitrarily select a set of round keys for all rounds except the last round. Encrypt P but do not perform the XOR with the round key in the last round. Let C' denote the output. Set the last round key to be $C \oplus C'$. This does not imply for a series of (P, C) pairs corresponding to one set of round keys that there will be another set of round keys which will produce the same (P, C) pairs for all of the plaintexts (the probability of this event is negligible).

For our analysis, we embed a copy of G inside a prefix of G' and assume the round keys outside the embedded G are independent of (do not give any information about) the round keys used inside the embedded copy of G. We also require that the round keys inside the embedded G are such that the side whitening bits are independent of the rest of the keys used in the rounds. We call this property "*proper expansion*" of the key schedule of the elastic cipher. Given the lack of any reduction proofs in the area of concrete block cipher design and given that such idealization of the key schedule can be achieved by certain scheduling methods that can be adopted (*e.g.,* applying a stream cipher to the keys to get the "independent" portion of the round keys, and model them in the proof as partial round keys that can be known/ controlled without affecting the security of the other round keys), we feel that these are reasonable assumptions for initiating analytic methods that validate the security of a design via "reduction-type" proofs in this area.

2.4 Efficiency and Performance

The elastic block cipher provides efficiency when data is not an integral number of blocks by eliminating padding and reducing the number of applications of the block cipher's round function from what would occur when padding is required. For most values of y, the number of rounds will be less than if the y bits were padded to a full b bit block and encrypted as a second block, and the number of rounds will never be greater than the number required when the data is padded. The XORing and swapping of y bits to form the input to each round after the first will increase the number of operations, but since these operations are quick, the impact is minimal. The extent of the impact is influenced by how the bits to be swapped are selected, *e.g.,* selecting y consecutive bits versus selecting non-consecutive bits. Also recall that the key expansion may be faster if a stream cipher is used in place of the G's key schedule. We describe the relative performance of an elastic version of AES compared to that of AES itself in Section 4.2.

3 Security of G'

3.1 Overview

For any block cipher used in practice, the cipher cannot be proven secure in the theorectical sense but rather is proven secure against known types of attacks, thus we can only do the same for the elastic version of a cipher. For now we do not focus on the elastic version of specific ciphers but rather, in order to provide a general understanding of the security of our construction, provide a method for reducing the security of the elastic version to that of the original version, showing that a security weakness in G' implies a weakness in G. Our security analysis of G' exploits the relationship between G' and G, specifically that an instance of G is embedded in G'. We discuss two types of attack: key recovery and distinguishability attacks. In order to focus on the core components of the algorithm for creating G' from G, for both types of attacks we consider G' without the optional initial key dependent mixing step which, if present, increases the security of G'. First, we show how to reduce G' to G in a manner that allows an attack which finds the round keys of G' to find the round key bits for G. Security against key-recovery attacks does not by itself imply security (*e.g.,* the identity function which ignores the key is insecure). However, all concrete attacks (differential, linear, *etc.*) attempt key recovery and thus practical block ciphers should be secure against key recovery. Second, we provide an overview of how the pseudorandomness of G' relates to that of G.

3.2 Round Key Recovery Attack

We use the fact that an instance of G is embedded in G' to create a reduction from G' to G. As a result of this reduction, an attack against G' that allows an attacker to determine some of the round keys implies an attack against G itself which is polynomially related in resources to the attck on G'. Assuming that G itself is resistant to such attacks, we conclude that G' does not reveal round-key bits to the attacker. The reduction requires a set of plaintext, ciphertext pairs. This is not considered a limiting factor because in most types of attacks, whether they are known plaintext, chosen plaintext, adaptive plaintext, chosen ciphertext *etc.,* the attacker acquires a set of plaintext, ciphertext pairs. We also assume that G has end of round whitening and that the key scheduling and expansion method is input independent. Again, these assumptions apply to many ciphers, or versions of ciphers that contain whitening and expand the key.

We first draw attention to the fact that the operations performed in $G\ '$ on the leftmost b bits is an application of G as shown in Figure 2. This relationship can be used to convert an attack on G' to an attack on G which finds the round keys for G. Recall G_{rk} denotes G using round keys rk. Specifically, if $G'_k(p \parallel x) = c \parallel z$, a set of round keys, rk, for G such that $G_{rk}(p) = c$ can be formed from the round keys and the round outputs in G' by collapsing the end-of-round whitening and swapping steps in G' into a whitening step. The leftmost b bits of the round key for the initial whitening are unchanged, and the rightmost y bits are dropped. While the resulting round keys provide good whitening to the rounds of the copy of G (due to the proper expansion property), they will vary in rounds 1 to r per plaintext, ciphertext pair due to the previous round's output impacting the end-of-round whitening step. However, it is possible to use these keys to solve for the round keys of G.

Theorem I: *If there exists an attack on G' which allows the round keys to be determined for the first r rounds, then there exists a polynomially related attack on G with r rounds, assuming:*

- *G contains end-of-round whitening.*
- *No message-related round keys. Namely, if there are expanded key bits utilized in G aside from the initial and end-of-round whitening steps, these expanded key bits depend only on the key and do not vary across inputs.*

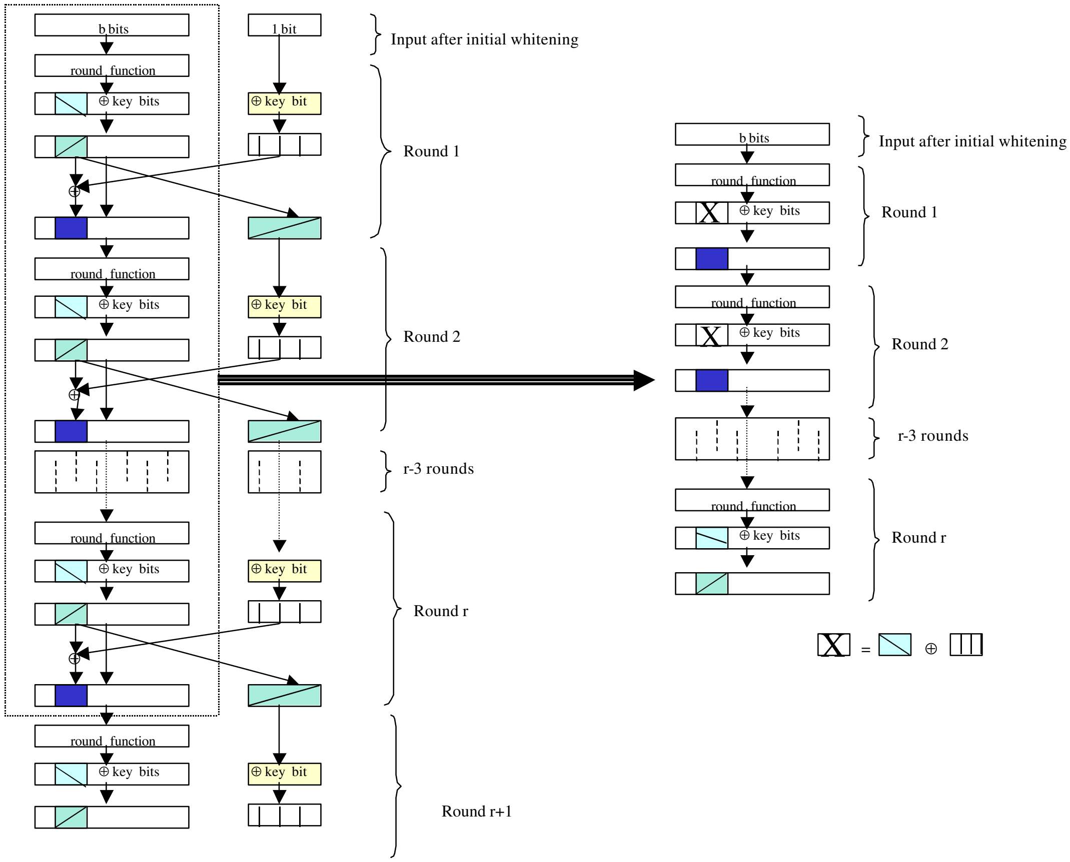

Figure 2: G **within** G'

- *The expanded key bits are done via proper expansion (defined in Section 2.3).*

With respect to the first condition placed on G in *Theorem I,* the condition may be removed if the attack on G' involves solving for the round key bits directly and allows the bits used in the whitening steps to be set to 0 for bit positions not swapped and to 0 or 1, as necessary, for bit positions swapped to insure the whitening on the leftmost b bits is equivalent to XORing with 0, which is the same as having no whitening in G. If the attack on G' finds all possible keys or sets of round keys (since it is the expanded key bits which are required to decipher texts, the attacks we are concerned with may find the expanded key bits directly as opposed to finding the original key then expanding it), the attack must find the key(s) or set(s) of round keys corresponding to round keys that are equivalent to XORing with 0.

In proving the theorem, we will describe two methods of utilizing the attack on G' to attack G. Before beginning, we prove a claim which will assist the reader in understanding the linkage between G and G'. The claim shows that for any set of plaintext, ciphertext pairs encrypted under some set of round keys in G', there exists a corresponding set of plaintext, ciphertext pairs for G where the round keys used in G' for the round function and the leftmost b bits of each whitening step are the same as those used in G, the plaintexts used in G are the leftmost b bits of the plaintexts used in G', and the ciphertexts for G are the same as the leftmost b bits of output of the r^{th} round of G' prior to the swap step.

Claim I: Let $\{(pi, ci)\}$ denote a set of n plaintext, ciphertext pairs and let $|w| = |vi| = y$. If $G_k(pi) = ci$ then there exists n sets of round keys for the first r rounds of G' that are consistent with inputs $pi \parallel w$ producing $ci \parallel vi$ as the output of the r^{th} round prior to the swap at the end of the r^{th} round, for $i = 1$ to n, such that the following condition applies:

Condition I: The leftmost b bits used for whitening in each round are identical across the n sets and any bits used internal to the round function are identical across the n sets.

Furthermore, y may be any valid value. The bits in vi are not used and thus no restrictions are placed on their values.

Proof: Let $rk = \{rk_j$ for $j = 0$ to $r\}$ be the set of round keys corresponding to key k for G. rk_0 denotes the key bits used for initial whitening. For (pi, ci), form a set of the first r round keys for $G\,'$ as follows: Pick a constant string, w, of y bits, such as a string of $0's$. Let $pi \parallel w$ be the input to G'. Let $rki' = \{rki'_j$ for $j = 0$ to $r\}$ denote the round keys for G' through the r^{th} round for the pair (pi, ci). Set any bits in rki'_j used internal to the round function to be the same as the corresponding bits in rk_j. Set the leftmost b bits used for whitening in rki'_j to the b bits used for whitening in rk_j. Set the rightmost y bits used for whitening in rki'_j to be the same as the y bits left out of the round function in round j of G'. This is illustrated in Figure 3. Notice that the leftmost b bits used for whitening in each round are identical across the n sets and any bits used internal to the round function are identical across the n sets, specifcally, they correspond to rk in each case, and the rightmost y bits used in each whitening step differ based on (pi, ci) across the n sets.

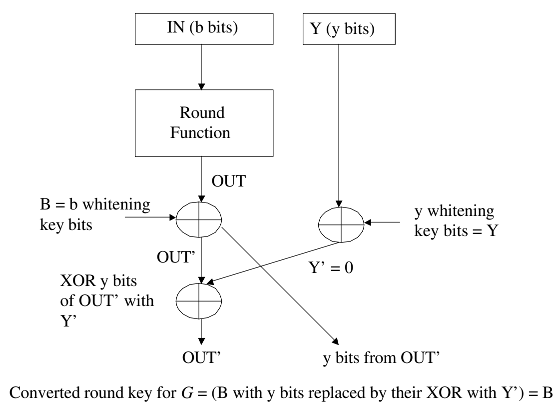

Figure 3: **Converted Key Unchanged in Left b Bits**

The operations of G' on the leftmost b bits through round r, prior to the last swap, are identical to the operations in $G_k(pi)$ because the swap step in G' results in XORing y bits of a round function's output with y $0's$. Therefore, the leftmost b bits output from the r^{th} round prior to the swap in the r^{th} round is ci. Therefore, for $i = 1$ to n there exists a set of round keys, rki' for $G'_{rki'}$ such that $G'(pi)$ produces ci as the leftmost b bits in the r^{th} round prior to the swap step and Condition I holds.

Proof of Theorem I: We now describe the reduction and two attacks. The attacks are presented in terms of solving for the round keys from round 0 to r, but may also be performed by working from round r back to the initial whitening. The first method's efficiency is dependent on y, $|k|$, and r, and may be the least useful of the two. We present it first in order to illustrate the method by which round keys for G' can be converted into round keys for G.

First method:

This method produces an attack on G that runs in time polynomial in the attack on G' and r. It is more efficient than an exhaustive key search when $y < \frac{|k|}{r-2}$. The attack works as follows: Assuming there exists a known plaintext, ciphertext pair attack on G' which produces the round keys either by finding the original key than expanding it or finding the round keys directly, using round keys for rounds 0 to r of G', convert the round keys into round keys for G one round at a time. For each round, extract the largest set of plaintext, ciphertext pairs used in the attack on G' that have the same converted round key. Each round may reduce the size of the set of pairs by 2^y. The end result is a set of round keys for G that are consistent with a set of $\frac{n}{2^{y(r-2)}}$ b bit plaintext, ciphertext pairs for G. We then describe how to take a set of plaintext, ciphertext pairs for G, convert them into a set of plaintext, ciphertext pairs for G' in order to run the attack on G' to find the round keys for G. Finally, we discuss the bounds on y for which this attack is more efficient than an exhaustive key search.

Let $\{(P,C)\} = \{(pi \parallel xi, ci \parallel zi)\}$ (for $i = 1$ to n) denote a set of n known $b+y$ bit (plaintext, ciphertext) pairs for G', where $|pi| = |ci| = b$ and $|xi| = |zi| = y$.

Assume the existence of an algorithm $A_{G'}$ that finds all possible keys, $\{k_j\}$, corresponding to $\{(P,C)\}$ in time less than a exhaustive search for the key. Let m denote the number of keys founds. Without loss of generality, it is assumed the keys are available in expanded form. The key bits for the initial whitening will be referred to as "round key 0."

Let $S = \{ek_j\}$ for $j = 1$ to m be the set of expanded keys used for whitening for which ek_j is from the expansion of key k_j and $G'_{k_j}(pi \parallel xi) = ci \parallel zi$ for $i = 1$ to n.

Let R_{int} denote any key material utilized within the round function. The values found for such key bits will be the same for the solutions derived by the attack for G' and G.

Let $\{(P,U)\} = \{(pi||xi, ui||vi)\}$ such that $ui||vi$ is the output of the r^{th} round of G', where $|ui| = b$ and $|vi| = y$.

Let $S' = \{ek'_j | ek'_j =$ bits of $ek_j \in S$ corresponding to rounds 0 to r used for whitening $\}$ be the set of expanded key bits used for whitening in rounds 0 to r of G'.

For each $ek_j \in S'$ and each $(pi \parallel xi, ui \parallel vi) \in \{(P,U)\}$, convert the round keys to round keys for G. Let ek'_{ij} be the converted key corresponding to the i^{th} element of $\{(P,U)\}$ and the j^{th} element of S'. The part of ek'_{ij} corresponding to round 0 will be identical across all elements. When the round keys are converted, at most y bits change in the leftmost b bits. Thus, the resulting round keys for round q, $0 < q \leq r$ can be divided for each of the y impacted bits into those that have a 0 in the affected bit and those that have a 1 in the affected bit. For $q = 1$ to r, define S'_{rnd_q} as the maximum-sized set of $ek'_{ij}s$ from $S_{rnd_{q-1}}$ that have identical round key(s) for round q, where $S'_{rnd_0} = S'$. Let $\{(P,U)_{rnd_q}\}$ be the corresponding elements of $\{(P,U)\}$. When forming $\{(P,U)_{rnd_q}\}$, at least $2^{-y}|\{(P,U)_{rnd_{q-1}}\}|$ of the elements from $\{(P,U)_{rnd_{q-1}}\}$ are included.

To illustrate how the sets S'_{rnd_q} and $\{(P,U)_{rnd_q}\}$ are created, consider the example in Figure 4 where $b = 4$, $y = 2$, and the leftmost 2 bits are swapped with the y bits in the swap step. The round number is q and $\{(P,U)_{rnd_{q-1}}\}$ contains three plaintext, ciphertext pairs. Suppose the outputs of the round function in the q^{th} of G' are $100101, 110011$ and 111111 and the whitening bits in the q^{th} round are 011010. The converted round keys corresponding to the three cases are $0110, 1110$ and 1110. Since 1110 occurs in the majority of the cases, set the q^{th} round key of G to 1110. S'_{rnd_q} contains the round keys for rounds 0 to $q-1$ from $S'_{rnd_{q-1}}$ and 0010, and $\{(P,U)_{rnd_q}\}$ contains the second and third plaintext, ciphertext pairs from $\{(P,U)_{rnd_{q-1}}\}$.

Let $\{(P,C)_G\} = \{(pi, ci) | (pi \parallel yi, ui \parallel vi) \in \{(P,U)_{rnd_r}\}\}$. $|\{(P,C)_G\}| \geq n/2^{yr}.\{(P,C)_G\}$ is a

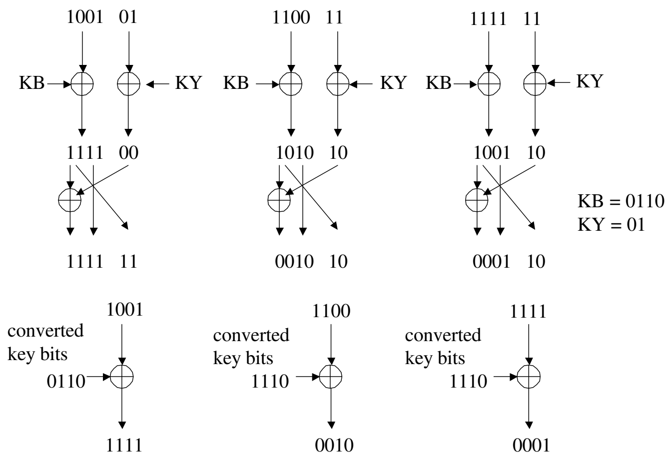

Figure 4: **Forming** S'_{rnd_q}

set of (plaintext, ciphertext) pairs for which $G_{rk}(pi) = ci \, \forall \, (pi, ci) \in \{(P, C)_G\}$ with the whitening round keys of $rk \in S'_{rnd_q}$ and any additional key material utilized by the rounds is the same as that for G', namely R_{int}.

In order to produce the largest set of (plaintext, ciphertext) pairs with a common key, every possible S'_{rnd_q} can be formed for each round, and the iteration for the $q + 1^{st}$ round applied to each. This will create a tree of depth r with at most 2^y children of each node. The leaf with the maximun size S'_{rnd_r} contains the keys for G with the highest probability.

Let t_r denote the time to run r rounds of G', and t_A denote the time to run $A_{G'}$. In the case of obtaining at least one set $\{(P, U)_{rnd_r}\}$ of size $\geq n/2^{yr}$, the time required beyond t_A consists of: nmt_r time to obtain the outputs of the first r rounds for each $\{(P, U)\}$, $O(nmr)$ time to perform the conversion of the round keys from G' to round keys for G and $O(nmr)$ time to form the S'_{rnd_r} sets. Thus, the additional time required to attack G (beyond the time required to attack G'_{b+y}) is $nmt_r + O(nmr)$. The only unknown value is m, the number of keys produced by the attack on G'_{b+1}. If m is large enough, to the extent that it approaches the average number of keys to test in a brute force attack on $G\,'$, then this contradicts the assumption that an efficient attack exists on $G\,'$ because the attacker is left with a large set of potential keys for decrypting additional ciphertexts.

To perform the attack on G when given a set of (plaintext, ciphertext) pairs for G, convert the pairs into a set of (plaintext, ciphertext) pairs for $G\,'$ and find the round keys for $G\,'$ then for G as follows: Let rk_{r+l}, where $1 \leq l \leq r' - r$, denote a set of randomly chosen round keys for rounds $r + 1$ to the last round of $G\,'$ that will be held constant, and let RF_{end} denote the last $r' - r$ rounds of $G\,'$ using these round keys. Given a set $\{(P^*, C^*)\} = \{(pi^*, ci^*)\}$ for $i = 1$ to n known (plaintext, ciphertext) pairs for G, create the set $\{(P, C)\}$ of (plaintext, ciphertext) pairs to use in the attack on $G\,'$ by setting $pi \parallel xi = pi^* \parallel 0$ and $ci \parallel zi = RF_{end}(ci^* \parallel 0)$ for $i = 1$ to n. This choice of $ci \parallel zi$ corresponds to an output of $ci^* \parallel 0$ in the r^{th} round of $G\,'$. For the set of (P, C) pairs are created, $\{P, U)\} = \{(pi^* \parallel 0, ci^* \parallel 0)\}$. Apply the attack on $G\,'$ to solve for the round keys of $G\,'$ then produce the sets $\{P, U)_{rnd_r}\}$ and S_{rnd_r}. The sets of round keys in S_{rnd_r} will be consistent with the plaintext,ciphertext pairs in $\{P, U)_{rnd_r}\}$.

We now discuss how the number of plaintext, ciphertext pairs required (i.e. the number of encryptions required) compares to that of an exhaustive key search. Recall the size of resulting set of plaintext, ciphertext pairs which are consistent with the round keys is $\geq \frac{n}{2^{yr}}$. When $y > \frac{|k|}{r}$ the number of plaintexts encrypted, n, must be $> 2^{|k|}$ to guarantee at least one plaintext, ciphertext pair is in $\{(P,C)_G\}$, which is more encryptions than that required by an exhaustive key search. Without changing either r or the length of k, this bound on y can be slightly increased to $y \geq \frac{|k|}{r-2}$ by using $b+y$ bit plaintexts which are the same in the rightmost y bits and by defining the ui values representing the ciphertext output of G in the r^{th} round of G' to be the output of the r^{th} round prior to the swapping step. This will result in $|S'_{rnd_1}| = n$ and $|S'_{rnd_r}| = |S'_{rnd_{r-1}}|$, thus in first and r^{th} rounds the set of plaintext, ciphertext pairs is not reduced. The number of plaintext, ciphertext pairs produced for G that are consistent with the round keys for G is $\geq \frac{n}{2^{y(r-2)}}$. Notice that increasing the number of rounds in G increases the number of plaintext, ciphertext pairs required to guarantee $\{(P,C)_G\}$ is non-empty and will prevent the attack from being more efficient than an exhaustive key search. While we prefer to not alter G in this manner, the efficiency of the attack being based on the number of rounds is useful when setting $G = G'_{b+y}$, in which case we are willing to adjust the rounds of G'_{b+y}, then creating a G' for the new G.

We define a direct attack on G'_{b+y} to be an attack which finds the key or round keys for G'_{b+y} without attacking $G'_{b+y'}$, for $y' > y$, and converting the round keys from $G'_{b+y'}$ to round keys for G'_{b+y}. We define an indirect attack on G'_{b+y} to be an attack which finds the round keys for $G'_{b+y'}$ for some $y' > y$, and uses them to find the round keys for G'_{b+y}, and $y' - y < \frac{|k|}{r^*-2}$, where r^* is the number of rounds in G'_{b+y}. Our analysis implies that if a direct attack exists on G'_{b+y} for $y < \frac{|k|}{r-2}$, then an attack requiring less time than an exhaustive key search exists on G. However, it does not imply G'_{b+y} is secure for all $y < \frac{|k|}{r-2}$ because there may be a direct attack on $G'_{b+y'}$ for some y' such that $y' - y < \frac{|k|}{r^*-2}$, thus implying an indirect attack on G'_{b+y}. In the worst case, $y' = y + 1$ and r^* must be increased to $|k| + 2$ for the attack to be more inefficient than an exhaustive key search. If the length of k can be changed (as part of the design of G'), setting $|k|$ to $2b(r-2)$ results in the bound being $y < 2b$, thus allowing all $y \in [0, b-1]$.

Second method:

This attack runs in time quadratic in the number of rounds of G and avoids the decrease in the number of plaintext, ciphertext pairs that occurs in the first method. The attack on G' is used to solve for round keys 0 and 1 for G, then repeatedly solves for one round key of G at a time, using the output of one round of G as partial input to a reduced round version of G', running the attack on G' and converting the 1^{st} round key of G' to the round key for the next round of G. We assume if an attack on G' with r rounds exists, then a reduced round attack on G' exists for any number of rounds $< r'$.

Given a set $\{(P^*, C^*)\} = \{(pi^*, ci^*)\}$ of n plaintext,ciphertext pairs for G, create a set $\{(P,C)\} = \{(pi^* \parallel 0, ci \parallel vi)\}$ of n plaintext,ciphertext pairs for an r round version of G'. Note: we only require that the y bits appended to each pi^* when forming $\{(P,C)\}$ be a constant; we choose to use 0. The vi values appended to the $ci's$ are arbitrary and do not need to be identical. Solve G' for round keys 0 and 1. By the pseudo-randomness of the round keys described in Section 2.3, sets of round keys exist that correspond to $\{(P,C)\}$ and which are identical in at least the first two rounds (the round keys across all n pairs may be identical in all but the last round, but we are only concerned with the first two rounds). Denote these as rk'_0 and rk'_1. Use the leftmost b bits of rk'_0 as round key 0, rk_0, for G. Since the rigthmost y bits are identical across all inputs to G', when rk'_1 is converted to a round key for G, the result will be the same across all n elements of $\{(P,C)\}$. Use the converted round key as round key 1, rk_1, for G. For each pi^*, apply the initial whitening and first round of G using the two converted round keys. Let $p1i$ denote the output of the first round of G for $i = 1$ to n. Using a reduced round version of G' with $r-1$ rounds and the initial whitening removed, set $\{(P,C)\} = \{(p1i \parallel 0, ci \parallel vi)\}$ and solve for the first round key of G'. As before, convert the resulting round key(s) to a round key for G. Again the converted round keys for G will be identical across all n values. Use the converted round key as the second round key for G. Repeat the process for the remaining rounds of G, each time using the outputs of the last round of G for which the round key has been determined as the inputs to G' and reducing the number of rounds in G' by 1, to sequentially find the round keys for G. This attack requires work equivalent of applying n rounds of G when deriving the outputs of the n inputs to each round of G, $\frac{n(r+1)r}{2}$ rounds of G' in the worst case if $A'_{G'}$ requires knowing the output of each round of G' to find the first round key and r applications of $A'_{G'}$ on $\frac{n(r+1)r}{2}$ rounds of G' when solving for the round keys of G'.

In summary, the attack on G described in this second method can be written as:

Input $\{(P^*, C^*)\} = \{(pi^*, ci^*)$ for $i = 1$ to $n\}$.
Create $\{(P, C)\} = \{(pi^* \parallel 0, ci^* \parallel vi)$ for $i = 1$ to $n\}$ for a r round version of G', where the $vi's$ are arbitrary.
Using $A_{G'}$, solve a r reduced round version of G' for rk'_0 and rk'_1.
Convert rk'_0 to rk_0 and rk'_1 to rk_1.
Set $p1i =$ first round output of G using rk_0 and rk_1, for $i = 1$ to n.
For $j = 1$ to $r - 1\{$
  $\{(P, C)\} = \{(p1i \parallel 0, ci^* \parallel vi)$ for $i = 1$ to $n\}$.
  Solve a $r - j$ reduced round version of G' for the first round key, rk'_1.
  Convert rk'_1 to form rk_{j+1}.
  $p(j + 1)i =$ output of round $j + 1$ of G on pji using rk_{j+1} for $i = 1$ to n.
$\}$

3.3 Distinguishability attack:

The focus so far has been on relating the security of the elastic version, G', of a block cipher G to the security of G in a concrete sense of (round) key recovery. Namely, in that ciphertexts cannot be decrypted without the key material and an attack cannot produce the key material for G' if the same is true of G. It is also of interest as to whether or not the output of G' can be distinguished from random. Typically, such proofs have a black box nature and show the pseudorandomness of G' based on the same assumption applied to G;however, our formation of G' from G is not based on a black box construction, thus we need a stronger assumption on G. Distinguishing the output of a cipher from that of a random permutation does not by itself imply the existence of an attack which finds key material or otherwise provides the ability to decrypt ciphertexts, however the fact that ciphers should approximate ideal cipher (where the function table for each key is a random permutation) is desirable property of any strong block cipher.

We now discuss the relationship between the distinguishability of outputs from G' from random (namely, from a truly random permutation) and a related key attack on G. In a related key attack, the relationship, but not the actual values, between two or more keys can be specified [7]. Encryption or decryption of known texts are then requested using these related keys. Typically in related key attacks, the relationship takes advantage of the cipher's key schedule to produce a relationship between the round keys of the cipher. For our purposes, it is the relationship between the round keys which is of interest and which we specify directly. We further weaken the attack by also specifying that the inputs are related such that one input is given and the second input is specified to have a relation with the first one based on the cipher's structure and the unknown keys. It is this attack which we consider: related keys - related inputs attack.

We define how the output of G'_{b+y} is the output of one instance of G concatenated with y bits from a second overlapping instance of G with the inputs of the two instances being related and the round keys of the two instances being related. The relation between the pairs of inputs and between the pairs of round keys is a function of the cipher structure only and treats the key and message variables symbolically. We then show if there is a distinguisher for $G\,'$, this distinguisher can be used to distinguish related key instances of G from a random permutation. From the pairs of outputs from G, $b+y$ bit strings are formed to which the distinguisher for $G\,'$ is applied and will recognize the bias bounded away from the assumed random permutation. Indistinguishability from a random permutation (with only a negligible probability of finding a bias) should hold even under related key attacks in ciphers which are ideal (are truly random permutation for each key). We note that this attack assumes that the cipher, G, is (an approximation of) an ideal cipher, so that even when relations in input and keys are given based on the internal structure of the cipher, we can still treat the cipher instances as ideal. While this may be viewed as a strong assumption indeed, the result is nevertheless an indication for the indistguishability of the elastic cipher from a pseudorandom permutation under the assumption that the basic cipher, G, is an ideal cipher.

Indeed, if G is an approximation to a random cipher (an ideal cipher in the Shannon sense where each key defines a truly random permutation table), no polynomial time distinguisher should exist that takes samples of input, output pairs and can tell whether they come from G or from a truly random permutation on strings of the same length. In fact even if we probe two instances of G with two different unknown keys that are of known relationship (related key probing) and on related cleartexts (relation may involve applying a fixed function on the known input portion and fixed unknown random values), we should not be able to distinguish the output set of a polynomially many such pairs from the same sample size set over two randomly chosen permutations. Using this assumption we will next claim that:

Theorem II: *The existence of a distinguisher for the elastic cipher G ′ implies the existence of a related key - related input distinguisher for G so that G cannot be an approximation of an ideal cipher, assuming:*

- *G contains initial and end of round whitening.*
- *The components of the round keys in G used for whitening are pseudorandom bit strings.*

Proof: We first formally describe how the output of G'_{b+y} is the output of one instance of G concatenated with y bits from a second instance of G with the inputs and round keys of the two instances related. Let:

- rkj_i denote the i^{th} round key of the set of round keys rkj with rkj_0 referring to the key bits used for the initial whitening.
- IRF_{rk_0} denote initial whitening and the round function, excluding the end of round whitening, of G using key material rk_0.
- x^* be a $b+y$ bit string.
- x' be the $b+y$ bit output of the $(r'-r-1)^{th}$ round of G' prior to the end of round whitening. Recall that r' and r are the number of rounds in G' and G, respectively.
- $x1, x2$ be two b bit strings such that $x1$ is the leftmost b bits of x' and $x2 = IRF_{rk_0}(x1)$.
- $k1, k2$ and $rk1, rk2$ be two keys for G and their corresponding sets of round keys, respectively, such that:
  - $rk1_{i+1} = rk2_i$ for $i = 0$ to $r-1$.
  - $rk2_0 =$ the key bits from $rk1_1$ used for the end of round whitening.
  - $rk1_0 =$ the leftmost key bits from rk_0 used for the inital whitening step in IRF.
- $c1 = G_{k1}(x1)$ and $c2 = G_{k2}(x2)$

- q be the y bits from $c1$ in the same position as the y bits from the leftmost b bits which are involved in the swap in round $r' - 1$ of G'_{b+y} when forming the input to round r' of G'_{b+y}.
- $v \leftarrow \{0,1\}^y$

Define $F_k(x) = q \oplus v$. The output of $G'_{(b+y)_k}(x^*)$ is $G_{k2}(x2)||F_{k1}(x1)$.

For clarity, we point out that the pair of related inputs $(x1, x2)$ to G are defined such that $x2$ is equivalent to applying a single round of G' to $x1$ using a round key which is held constant when forming multiple related message pairs of the form $(x1, x2)$. The round keys of G are related in that they are shifted one round.

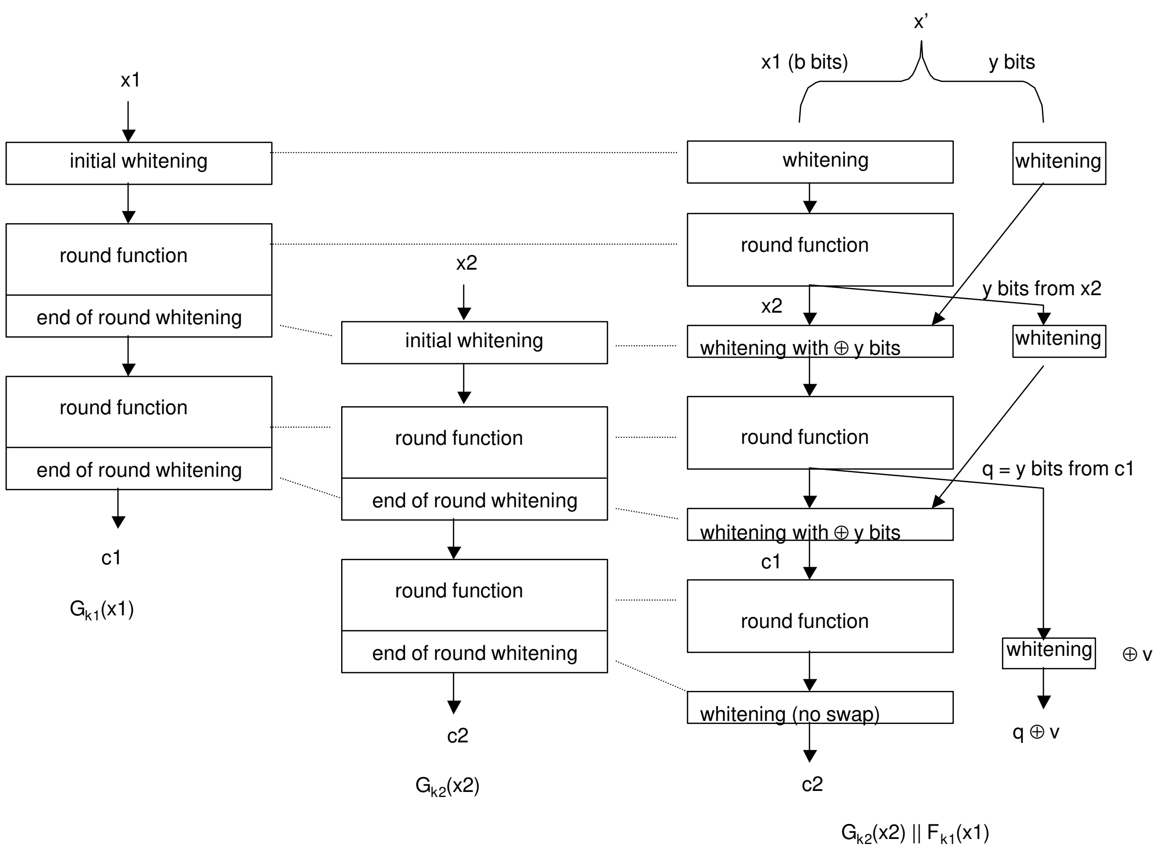

Figure 5: **Related Instances of G within G'**

Figure 5 illustrates the relationship using a G with two rounds. The right part of the diagram reflects the a whitening step followed by 3 rounds of G'. Thus $x1$ can be viewed as either the leftmost b bits of the input to a 3 round version of G' or of the output of a previous round of G' after the swap step. The "whitening with $\oplus y$ bits" and "whitening" within these 3 rounds are an equivalent representation of the end of round whitening and swap steps in $G\,'$.

We now show how a distinguisher that distinguishes $G\,'$ from a random permutation can be used to prove G is not an ideal cipher. Assume $G\,'$ is not a pseudorandom permutation. Then there exists a distinguisher $D\,'$ such that if polynomial many, n, inputs are given to a black box that contains either G' with a fixed, but randomly chosen key, or a random permutation, D' can determine from the outputs whether the black box contains G' or a random permutation with non-negligible probability. The n inputs to the black box can be adaptively chosen and any specific input to the black box always produces the same output. Each output from the black box is given to $D\,'$, which then outputs a 0 if it thinks the output came from an instance of $G\,'$ and outputs 1 otherwise. On n inputs, $D\,'$ will answer correctly with probability $\frac{1}{2} + \epsilon$ for some non-negligible ϵ.

D' can be used to construct a distinguisher, D, for G that distinguishes G from an ideal cipher. For G to be an ideal cipher, each instance of G (G with a specific key) must correspond to a random permutation. Thus, two instances of G with related keys must also each correspond to random permutations and a related key attack cannot exist which distinguishes G from a random permutation. Using the notation and relations defined previously, let G_{k1} and G_{k2} be two instances of G with keys $k1$ and $k2$ that are related, but are otherwise unknown. We consider a black box, B, that takes pairs of b bit inputs and contains either a random permutation on $2b$ bits or the two instances of G. The two instances of G are such that G_{k1} is applied to the first b bits and G_{k2} is applied to the last b bits. Without loss of generality, the outputs, $c1$ and $c2$, of the two instances of G are concatenated to form the output from B, and the random permutation treats the pairs of b bits as $2b$ bit strings and outputs $2b$ bits. It should not be possible to determine with non-negligible probability the contents of B on polynomial many queries to it if G is an ideal cipher because doing so implies a related key attack on G. Furthermore, the inputs used in the queries may be adaptively chosen (we will require only that they can be chosen, not necessarily adaptively) Thus even if the pairs of b bit inputs consist of related b bit strings, $x1$ and $x2$ as defined previously, a distinguisher should not exist if G is ideal.

However, if $G\,'$ is not a pseudorandom permutation on $b + y$ bits, we can create a D that succeeds with non-negligible probability and thus G cannot be ideal. Use n pairs of $(x1, x2)$ as the inputs to B (where $x1$ and $x2$ are related) and apply D to the $2b$ bit outputs of B. Let $w1||w2$ denote an output of B, where $|w1| = |w2| = b$. Define D as follows to output 0 if B contains the two instances of G and 1 otherwise:

Form a y bit random string, v, that is constant for all inputs to D: $v \leftarrow \{0,1\}^y$

$D(w1||w2)$ {
  Form a y bit string, q, by taking the y bits from $w1$ that are in the same y positions used in the last swap step in $G\,'$.
  $ans \leftarrow D\,'(w2||q \oplus v)$
  Return ans.
}

We note that the bit positions chosen in each swap step of $G\,'$ are part of the definition of $G\,'$ and depend at most on y (they are neither dependent on the key nor the input), thus they can be known by D. The $b + y$ bit string formed by D and given to $D\,'$ is precisely $G_{k2}(x2)||F_{k1}(x1)$ when the $2b$ bits input to D are from the related key instances of G using related inputs, and thus is the output from an instance of $G\,'$. The $b + y$ bit string formed by D is random when the $2b$ bits input to D are from a random permutation. Since $D\,'$ succeeds in distinguishing $G\,'$ from a random permutation and D returns a 0 whenever $D\,'$ returns a 0, D will succeed with non-negligible probability in determining whether B contains the two instances of G or a random permutation.

If n queries to D' are required to distinguish the output of $G\,'$ from a random permutation, then n queries constructed from $2n$ outputs (n pairs) of inputs of B are required to use D to distinguish the outputs of G (with the related key - related inputs) from outputs of a random permutation. Therefore, by using related keys and related inputs, the outputs of G can be distinguished from random with non-negligible probability on polynomial many queries. Thus if G' is not a pseudorandom permutation, G cannot be the approximation of an ideal cipher, and if G is an approximation of an ideal cipher, G' is a pseudorandom permutation.

4 Elastic Version of AES

4.1 Analysis

We applied our algorithm to the 128 bit version of AES [2] to create an elastic block cipher that accepts blocks of size 128 to 255 bits. The number of rounds will range from 10 when $y = 0$ to 20 when $y \geq 116$. The modifications to the encryption and decryption process consist of an additional y bits in the initial whitening and per-round whitening, and the XORing and swapping of y bits between each round. Instead of modifying the key expansion to produce $128 + y$ bit keys for the initial whitening and round keys, all of the key material was generated by a stream cipher, with the 128-bit key used as the key for the stream cipher. Before describing the implementation, we discuss the security briefly.

In AES, complete diffusion occurs by the end of the second round, meaning every output bit of the second round has been affected by every input bit to the first round (refer to page 41 of [11]). In the elastic version of AES, if $y > 0$, three rounds are required before every bit has affected every other bit.

As we showed in Section 3, any attack on G' which produces the round keys implies an attack on G exists. In the case of our elastic version of AES, the round keys produced by the attack are used only for whitening, since there are no key bits used internally in the round function. When the round keys for G' are converted into round keys for G, the results correspond to solutions for a version of AES with pseudo-random round keys, since the AES key schedule was replaced with a stream cipher. We can easily prune the sets of round keys of keys that do not adhere to AES's key schedule.

It should also be mentioned that with respect to linear cryptanalysis [17], if a set of equations exist relating the round key, plaintext and round output bits for r rounds of G', the equations may be modified in two ways to create a set of equations corresponding to G. The first possibility is to replace variables representing any of the first b bits which are swapped in a round with the XOR of the variables which are XORed in the swap step, and discard any variables representing the last y bits of whitening in the last round key and of the last y bits of the ciphertext to obtain a set of equations relating the round key bits, plaintext and round outputs for G. The second possibility is to replace the variables representing the key bits used for the rightmost y bits of whitening in each round with the variables representing the rightmost y bits of input to the round (i.e. the y bits left out of the round function). Again discard any variables representing the last y bits of whitening in the last round key and the last y bits of the ciphertext.

To gain an understanding of the potential for differential cryptanalysis [8], the impact of a differential that is 0 except for 1 byte was traced through 3 rounds of the elastic version. A 1-byte differential was used because it has the greatest single round probability of 2^{-6} for AES (refer to pages 205-206 of [11]) and thus also has the greatest single round probability for elastic AES since the round function is unmodified. The analysis excludes the key dependent mixing step after the initial whitening. Determing the bounds without this step allows us to determine the extent of mixing required. Including a simple key dependent rotation will decrease the upper bound by $\frac{1}{2}$. The analysis was performed in a manner that applies to $128 + y$ bits for any $y \in [1, 127]$. By the 4^{th} round of the elastic version, at least 12 of the 16 bytes input to the round function will differ. Resulting from the analysis is the following claim:

Claim II: *A 3-round differential for the elastic version of AES (without an initial key dependent mixing) accepting input blocks of 128 to 255 bits occurs with probability* $\leq 2^{-30}$.

Proof:

The following is the analysis resulting in the upper bound for the probability of a differential for G' when G is AES and the swapping step in G' does not break up byte boundaries. A bound will be established on a 3 round differential and used to obtain an upper bound on 11 and 20 round differentials.

For a non-zero differential, the highest single round probability in AES is 2^{-6} and is achieved by a one byte difference in the input. This is due to the S-Box in the SubBytes step of the round. Two single byte differences achieve a probability of 2^{-6}; each of the other 126 possible one byte differences produce a specific output difference with probability 2^{-7} (refer to page 205 in [11]). The only point in the AES round function in which a byte impacts other bytes is the MixColumns step, thus if a certain output of the round is desired that requires more than one specific input byte to the MixColumns step to take on a certain value then each byte can be set with probability at most 2^{-6} and there are at most 128 possible pairs of deltas that produce the result. Thus, in AES the highest probability differential for the round occurs if only one byte differs in the input and the output from the S-Box, and it is one of the cases that occurs with probability 2^{-6}.

Estimating an upper bound on the probability for a differential over multiple rounds by computing the product of the differentials for individual rounds, based on the S-Box alone, the maximum probability possible is $(2^{-6})^r$. However, this rough estimate is higher than the actual upper bound. In AES, the probability after 4 rounds is $\leq 2^{-96}$ (refer to page 205 in [11]). By following how a single byte difference in inputs to a round propagates through the subsequent round in the elastic version of AES, it will be shown the resulting probability is small enough to indicate a differential attack is not feasible.

Terminology:

- Δ indicates the difference between two bit strings.
- Encryption instance refers to the encryption of a particular plaintext. When two plaintexts satisfying a particular Δ are encrypted, the two encryptions will be referred to as two instances.
- Round input and output refers to the entire $128 + y$ bits entering and leaving a round, respectively. The round consists of the AES round function as well as the XOR and swapping of y bits.
- Δ output refers to the difference between the round's output for the two instances of encryption.
- Δ input refers to the difference between the round's input for the two instances of the encryption.
- $a_i \parallel b_i$ denotes the input to round i. a_i is 128 bits and b_i is y bits.
- $\Delta a_i \parallel \Delta b_i$ denotes the difference between two inputs to round i. This also equals the difference between the outputs of round $i - 1$.
- Δa_{out_i} denotes the difference between two outputs of the round function in round i.
- 0 denotes a string of bits that are all 0.
- Controlling a byte in Δ refers to being able to predict what the Δ in the SubBytes outputs for the two instances will be for the particular byte.
- The 128 bits input to the AES round function are treated as a 4 x 4 byte matrix. Rows and columns will refer to the rows and columns of this 4 x 4 matrix.

Before beginning the analysis, a few facts should be mentioned.

Fact 1: In AES, a one-byte difference between two inputs to a round will produce a 4 byte difference in the outputs, and these 4 bytes will correspond to a single column in the 4 x 4 byte matrix. When input to the next round, ShiftRows will result in each column having one of the 4 bytes, thus all 16 bytes are impacted after the MixColumns step occurs. Thus, a one byte Δ between two inputs into round i affects the Δ in all 16 bytes of the outputs from round $i+1$.

Fact 2: In the elastic version of AES, once a non-zero Δ occurs in a round's input it is impossible to return to a state where Δ input = 0 for a subsequent round.

Case 1: Suppose Δa is non-zero for a round, then there is a non-zero Δ in the input to the AES round function and a non-zero Δ in the output of the AES round function. If the byte(s) involved in the non-zero Δ are not part of the y bits left out for the next round, there is a non-zero Δ in the input to the next round and it occurs within the first 128 bits. If these bits are part of the y bits left out of the next round, there is a non-zero Δ in the input to the next round and it occurs within the last y bits. There may also be a non-zero Δ in the first 128 bits depending on the result of the XOR.

Case 2: Suppose Δb is non-zero for a round. When the y bits in b are XORed with y bits from the 128 bit output of the AES round function, two general scenarios occur:

a. There is a non-zero Δ in the 128 bits of the AES round function output which results in Δa = 0 into the next round. Then Δb is non-zero for the next round.

b. There is a zero Δ in the 128 bits of the AES round function output. Then the current Δb results in a non-zero Δa for the next round after the XOR step.

Fact 3: If two plaintexts differ in at least one byte then in the first and/or second round the input to the AES round function must differ in at least one byte.

Fact 4: A non-zero differential with probability of 1 exists for a round of the modified AES. If $\Delta a_1 \parallel \Delta b_1 = 0 \parallel x$ for $x \neq 0$, then the difference between the outputs of the first round after the swap is $\Delta a_2 \parallel 0$ where Δa_2 contains the bits from Δb_1. In this case, the first round does not assist in preventing a differential attack. As mentioned in Section 2, a key dependent mixing step prior to the first round will help avoid this case and thus will be useful if there is a need to decrease the probability of a specific differential occurring by a means other than adding an additional round.

Now we proceed with the proof of the claim that the probability a differential occurs across 3 rounds of the modified AES is bounded from above by 2^{-30}. Given that a one byte non-zero Δ in the input to the AES round will result in a 4 byte non-zero Δ in the output, the one byte Δ may result in a 0 to 4 byte non-zero Δ in the y bits that remain out of the next round. Let Δx refer to the 4 non-zero bytes in Δa_{out_i} .Consider what happens in each case of 0 to 4 bytes being swapped into the y bits for round $i+1$.

Case 1: None of the four non-zero bytes in Δa_{out_i} are involved in the XOR and swap step. The following will result:

- Δb_{i+1} contains 0 bytes of Δx and thus is 0.
- Δa_{i+1} contains Δx.
- $\Delta a_{out_{i+1}}$ will be non-zero in all 16 bytes.
- Since Δb_{i+1} is 0, $\Delta a_{out_{i+1}}$ will carry forth into Δa_{i+2}.

Multiplying the probabilities from rounds i (2^{-6}) and $i+1$ ($(2^{-6})^4$) results in an upper bound of 2^{-30} that a specific differential occurs. Furthermore, Δa_{i+2} will be non-zero in every byte.

Case 2: Exactly one of the 4 non-zero bytes is involved in the XOR and swap step. The following will result:

- Δb_{i+1} contains 1 byte of Δx and is 0 in all other bytes.

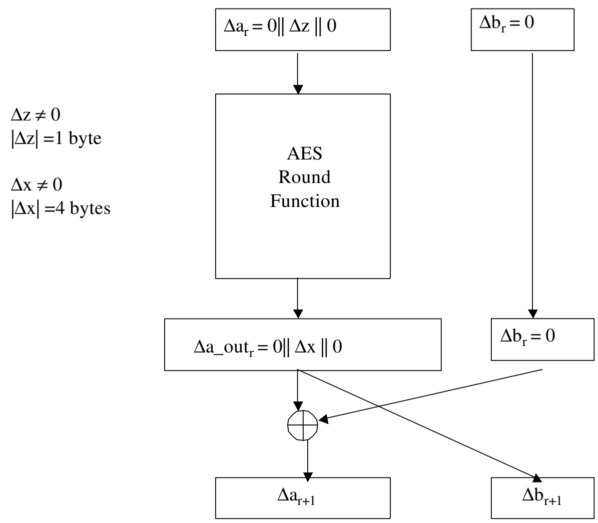

Figure 6: **Elastic AES Round Differential**

- Δa_{i+1} contains Δx. 3 bytes of Δx are not involved in the XOR and swap, and thus remain unchanged when entering the next round. The 4th byte depends on the result of the XOR with Δb_i. However, since Δb_i is 0, this byte will also enter the next round unchanged.

- $\Delta a_{out_{i+1}}$ will be non-zero in all 16 bytes.

Multiplying the probabilities from round i (2^{-6}) and $i+1$ ($(2^{-6})^4$) results in an upper bound of 2^{-30} that a specific differential occurs. Furthermore, Δa_{i+2} will be non-zero in at least 15 bytes because the XOR with Δb_{i+1} can result in at most one byte becoming 0.

Case 3: Exactly two of the 4 non-zero bytes is involved in the XOR and swap step. The following will result:

- Δb_{i+1} contains 2 bytes of Δx and is 0 in all other bytes.

- Δa_{i+1} contains Δx.

- $\Delta a_{out_{i+1}}$ will be non-zero in all 16 bytes.

Multiplying the probabilities from round i (2^{-6}) and $i+1$ ($(2^{-6})^4$) results in an upper bound of 2^{-30} that a specific differential occurs. Furthermore, Δa_{i+2} will contain at least 14 non-zero bytes.

Case 4: Exactly three of the 4 non-zero bytes is involved in the XOR and swap step. The following will result:

- Δb_{i+1} contains 3 bytes of Δx and is 0 in all other bytes.

- Δa_{i+1} contains Δx.

- $\Delta a_{out_{i+1}}$ will be non-zero in all 16 bytes.

As before, 5 bytes must be controlled across the Δ for the two rounds so there is an upper bound of 2^{-30} that a specific differential occurs. Furthermore, Δa_{i+2} will contain at least 13 non-zero bytes.

Case 5: All of the 4 non-zero bytes is involved in the XOR and swap step. The following will result:

- Δb_{i+1} contains Δx and is 0 in all other bytes.
- Δa_{i+1} contains Δx.
- $\Delta a_{out_{i+1}}$ will be non-zero in all 16 bytes.

As before, 5 bytes must be controlled across the Δ for the two rounds so there is an upper bound of 2^{-30} that a specific differential occurs. Furthermore, Δa_{i+2} will contain at least 12 non-zero bytes.

The single byte difference in the inputs to the round must occur by round 2 per Fact 3. Given that the probability a specific differential occurring has an upper bound of 2^{-30} for three rounds and 2^{-6} for one round, and the modified AES requires at least 11 rounds, then calculating the probability for all rounds by multiplying the probabilities for individual rounds, using 2^{-6} for round 11 and 1 for round 1, results in an upper bound of $(1)(2^{-30})^3(2^{-6}) = 2^{-96}$. Extending this to 20 rounds results in an upper bound of $(1)(2^{-30})^6(2^{-6}) = 2^{-186}$. However, notice that in each of the 5 cases, the difference in inputs to the 4th round in the series involves at least 12 non-zero bytes, indicating the probability a specific differential holds for any 3 consecutive rounds after the first 4 rounds will be $< 2^{-30}$.

Using the upper bound of 2^{-30} for the 3 rounds and a bound of 2^{-6} for one round, the probability of an 11-round differential is $\leq 2^{-96}$ and the probability of a 20-round differential is $\leq 2^{-186}$. These bound can further be lowered by calculating the probability for 4 rounds.

4.2 Implementation and Performance

We implemented an elastic version of AES in C that accommodates block sizes of 128 to 255 bits. As we explained previously, we use a stream cipher for the key schedule, rather than modify the AES key schedule to provide the additional bits needed for the elastic version. Specifically, we use RC4 with the first 512 bytes discarded [19, 22]. The key dependent mixing step included is a simple key dependent rotation. The elastic version increased the number of operations beyond the 128-bit version of AES due to the swapping step and an initial key dependent rotation. However, by avoiding the need to pad the data to two full blocks, the elastic version saves processing time in those cases where padding would normally be required for most or all data blocks. Both the elastic version and regular 128-bit version of AES were run on several processors in Linux and Windows environments to compare their performance. In the tests, the data to be encrypted was viewed as individual $b + y$ bit blocks. The elastic version of AES encrypted each block individually with no padding. To encrypt the data with regular AES, the $b + y$ bits were padded to $2b$ bits and encrpyted as two b bit blocks . The results in terms of ratios were similar for each platform tested. The following summarizes the results from a C implementation compiled with Visual C++ 6.0 on a 1.3Ghz Pentium4 processor running Windows XP. When measuring the encryption rate in terms of blocks per second, AES's rate for a single block was based on the time to encrypt 32 bytes to represent the padding required when using AES for $b + y$-bit blocks. The rates are determined by the time to encrypt one million $128 + y$ bit blocks using the elastic version and two million 128 bit blocks using the original version of AES. Table 1 summarizes how the elastic version compares to the original version for 8 bit intervals of $b + y$. The elastic version's rate for encrypting $b + y$ bit blocks ranges from 190% of AES's rate when $y = 1$ to 100% of AES's rate when $y = 97$. The elastic version of AES's rate decreased gradually to a low of 83% of AES's rate when $y = 127$. Thus, for applications where significant data padding is needed, elastic AES can almost double performance.

b+y	Elastic AES's Rate as a Percent of AES's Rate	b+y	Elastic AES's Rate as a Percent of AES's Rate
129 to 136	190	193 to 200	121
137 to 144	182	201 to 208	106
145 to 152	154	209 to 216	101
153 to 160	153	217 to 224	101
161 to 168	143	225 to 232	100
169 to 176	125	233 to 240	88
177 to 184	125	241 to 248	88
185 to 192	122	249 to 255	83

Table 1: **Elastic AES Encryption Rate as a Percentage of AES's Encryption Rate**

5 Applications

In addition to the methodological and design contributions, we note that our design of elastic block ciphers was primarily motivated by two practical applications: database and network traffic protection.

Encryption has been used to ensure database confidentiality for many years. Typically, a block cipher such as DES is used in ECB or CBC mode. Unfortunately, most data fields in a database do not interact well with the typical block sizes (64 or 128 bits): integers are often 32 bits long, doubles are 80 bits, and strings are of arbitrary length. Thus, the amount of space that is "wasted" due to encryption can be significant in a large database. Obviously, elastic block ciphers are of no use when the data are shorter than the block size (*e.g.,* integers). However, for variable-size fields such as names of people or products, which are typically longer than one block, use of an elastic block cipher can reduce the "security overhead" considerably.

A second immediate application includes network traffic protection, especially for protocols such as IPsec [16], which operate at individual packets. Several researchers have observed the tri-modal distribution of network traffic [10]: about 40% of packets are 40 or 60 bytes (the vast majority of these being TCP acknowledgments), with the remainder 60% split mostly between 576 bytes (the default TCP Maximum Segment Size) and 1460 bytes (maximum TCP MSS in an Ethernet-friendly network). IPsec itself adds 2 bytes for a self-describing padding [15]. Thus, for any of these common packet sizes, IPsec will increase the ciphertext length by another full block. Using an elastic block cipher for the last 2 blocks of a packet allows us to eliminate all padding, resulting in a modest performance boost and a reduction in the amount of buffer space and link capacity wasted on "security overhead." We plan to quantify these benefits in future work. Although plaintext padding can be useful for protection against traffic analysis attacks, in practice, this feature is rarely, if ever, used outside the military environment. If such protection is needed, we note that using of an elastic block cipher allows us to employ truly random message padding.

6 Elastic Mode of Encryption

An elastic cipher can in general be used in known modes adapting the last block size to accommodate no padding. We also demonstrate the possible use of the cipher for a new mode, as depicted in Figure 7. The mode is useful in applications where the entity decrypting the ciphertext should or can decrypt starting at the last block (*e.g.,* decryption from storage as opposed to real time communication). Given a block cipher that operates on b-bit blocks, the elastic version can be applied to encrypt $b+y$-bit blocks, where y bits are taken from the previous ciphertext block. It is possible to start with an IV which is used for the first y bits. The output will be the first b bits of each ciphertext for all but the last block, and the entire $b + y$ bits of the last block (or whatever size it actually is). Overall, the ciphertext will be at most y bits longer than the plaintext. If the plaintext is not an integral number of b-bit blocks, the mode can be implemented without padding the last block; whereas, using the non-elastic version of the block cipher would require padding and also produce a ciphertext longer than the plaintext. The mode resembles CBC but rather than simple XORing of the previous block, the influence of the previous block is via concatenation of part of it (a stronger binding that is, of course, achieved by increasing the work per block from working on b bits to working on $b + y$).

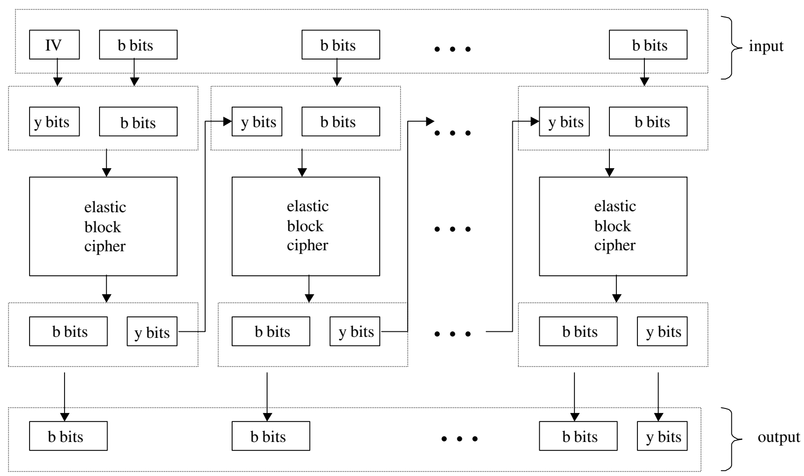

Figure 7: **Mode Using Elastic Block Cipher**

The ciphertext can be decrypted by decrypting the last block, concatenating the y bits from the plaintext block with the previous ciphertext block, and then decrypting the next block. While the IV is not needed for decryption; having it available for decryption provides a type of integrity check in that the first y bits of the resulting plaintext can be verified against the IV (when we use an IV).

The mode offers some security benefits. Incorporating the previous ciphertext block into the current plaintext block when encrypting will help hide plaintext patterns. In the way the cipher is depicted, a single bit toggled in the ciphertext is detectable because it will garble all plaintext prior to and including the altered block. It is possible to append blocks to the end of the ciphertext if the value of y is known (so we may want to keep y secret). A splicing attack, where two ciphertexts are used to generate a third with only a minor corruption (and to which CBC is vulnerable), can be prevented if the IV is secret and is XORed with the last y bits of the last output block. Inserting blocks within the ciphertext requires both knowing y and that the last block inserted encrypts to a value that produces the same y bits used to encrypt the next block of original plaintext. Since the last y bits can be kept inside the cipher device between block encryption, this seems like a measure against the recent block-wise adaptive attacks [14].

7 Conclusions

We introduced a new concept of an *elastic block cipher* and presented a general method for converting any existing block cipher that is based on a round function into an elastic block cipher. Elastic block ciphers allow us to "stretch" the supported block size of a block cipher up to twice the original length, while increasing the computational load proportionally to the block size. In the analysis of our scheme, we showed that the existence of an attack on the elastic version that produces the round keys implies an attack exists on the original block cipher by creating a reduction between the elastic and original versions of the block cipher; this methodological contribution seems to us to be quite unique in the area of block cipher design. We also prove a relationship between the pseudorandomness of the elastic block cipher and that of the original block cipher.

Practical applications of elastic ciphers include database and network traffic encryption. Our prototype "elastic AES" can exhibit a performance improvement of up to 190% compared with regular AES, when applied to data that are only slightly longer than one block. Future work includes cryptanalysis on specific constructions utilizing block ciphers other than AES, extending the analysis of the elastic version of AES, extending the methodology developed herein, and experimentally quantifying the benefits of our elastic AES in database and network encryption.

References

[1] FIPS 46-2 Data Encryption Standard (DES), 1993.

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

[3] J. An and M. Bellare. Constructing VIL-MACs from FIL-MACs: Message Authentication Under Weakened Assumptions. In *Proceedings of Advances in Cryptology - Crypto '99, LNCS 1666, Springer-Verlag*, 1999.

[4] M. Bellare, R. Canetti, and H. Krawczyk. Pseudorandom Functions Re-Visited: The Cascade Construction and its Concrete Security. In *Proceedings of Foundations of Computer Science, IEEE*, 1996.

[5] M. Bellare and P. Rogaway. On the Construction of Variable Length-Input Ciphers. In *Proceedings of Fast Software Encryption (FSE), LNCS 1636, Springer-Verlag*, 1999.

[6] D. Bernstein. How to Stretch Random Functions: The Security of Protected Counter Sums. In *Journal of Cryptology, Vol. 12(3), Springer-Verlag*, pages 185–192, 1999.

[7] E. Biham. New Types of Cryptanalytic Attacks Using Related Keys. In *Proceedings of Advances in Cryptology - Eurocrypt '93, LNCS 0765, Springer-Verlag*, 1993.

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

[9] J. Black and P. Rogaway. CBC MACs for Arbitrary-Length: The Three-Key Constructions. In *Proceedings of Advances in Cryptology - Crypto 2000, LNCS 1880, Springer-Verlag*, 2000.

[10] K. Claffy, G. Miller, and K. Thompson. The nature of the beast: Recent traffic measurements from an Internet backbone. In *Proceedings of the ISOC INET Conference*, July 1998.

[11] J. Daemon and V. Rijmen. *The Design of Rijndael: AES the Advanced Encryption Standard*. Springer-Verlag, Berlin, 2002.

[12] S. Halevi and P. Rogaway. A Tweakable Enciphering Mode. In *Proceedings of Advances in Cryptology - Crypto 2003, LNCS 2729, Springer-Verlag*, 2003.

[13] S. Halevi and P. Rogaway. A parallelizable enciphering mode. Cryptology ePrint Archive, Report 2003/147, 2003. `http://eprint.iacr.org/`.

[14] A. Joux, G. Martinet, and F. Valette. Blockwise-Adaptive Attackers: Revisiting the (In)Security of Some Provably Secure Encryption Models. In *Proceedings of Advances in Cryptology - Crypto 2002, LNCS 2442, Springer-Verlag*, 2002.

[15] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP). RFC 2406, Nov. 1998.

[16] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. Request for Comments (Proposed Standard) 2401, Internet Engineering Task Force, Nov. 1998.

[17] Matsui. Linear Cryptanalysis Method for DES Cipher. In *Proceedings of Advances in Cryptology - Eurocrypt '93, LNCS 0765, Springer-Verlag*, 1993.

[18] M. Matsui. Specification of MISTY1 - a 64-bit Block Cipher, September 18, 2000.

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

[20] Rivest, Robshaw, Sidney, and Yin. RC6 Block Cipher. `http://www.rsa.security.com/rsalabs/rc6`, 1998.

[21] Schneier and Kelsey. Unbalanced Feistel Networks and Block Cipher Design. In *Proceedings of Fast Software Encryption (FSE), LNCS 1039, Springer-Verlag*, 1996.

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

[23] R. Schroeppel. Hasty Pudding Cipher. `http://www.cs.arizona.edu/rcs/hpc`, 1998.