Elastic Block Ciphers

Debra L. Cook, Moti Yung, and Angelos D. Keromytis
Department of Computer Science
Columbia University
{*dcook,moti,angelos*}@*cs.columbia.edu*
May 30, 2004

Abstract. We introduce the new concept of *elastic block ciphers,* symmetric-key encryption algorithms that (1) for a variable-size input do not expand the plaintext (*i.e.,* do not require plaintext padding) and (2) adjust their computational load *proportionally* to the size increase. Contrary to stream ciphers, elastic block ciphers maintain the diffusion property and non-synchronicity of traditional block ciphers. Elastic block ciphers are ideal (when combined with encryption modes) for applications where length-preserving encryption is most beneficial, such as protecting variable-length database fields or network packets.
We present a general algorithm for converting a traditional block cipher, such as AES, to its elastic version, and analyze the security of the resulting cipher against key recovery attacks. 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 expanded 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 certain key recovery attacks of the elastic version to such attacks on 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 developing 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 work puts forth the notion of elasticity in block cipher design.

Keywords: Cipher Design, Variable Length Block Cipher, Elastic Block Ciphers, Encryption Algorithm, Key Recovey Attacks, 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. For example, most data fields in a database do not interact well with the typical block sizes: 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. Similar issues arise when considering network traffic protection, *e.g.,* in IPsec

[1]. 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 the sender and the receiver, which is an unsuitable assumption for many applications. The ideal solution combines the length-preserving aspects of stream ciphers with the diffusion properties of block ciphers, and uses existing and well analyzed components or algorithms, to leverage prior work as much as possible.

We introduce the 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 "stretched" block size. This, together with modes of operation, permit 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 in a given mode of operation. Such a cipher will be very useful in database and network traffic encryption, as well as other applications of encryption.

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 base block cipher nor decrease the number of rounds applied to each bit (position), 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 utilize the reduction to evaluate the security of elastic block ciphers against certain key recovery attacks. We are not aware of existing proof methods that argue about sub-ciphers in the area of concrete block cipher design when the basic block cipher is not treated as a black box (due to efficiency requirements in our case). The importance of such a proof is that it allows one to exploit the established and provable properties of existing ciphers in establishing certain properties of the new design.

There has been little previous work on variable-length pseudo-random functions (PRFs). The focus has been on variable-length inputs with fixed-length outputs as applicable to MACs and hash functions [2–5] and, more recently, on modes that work on multiples of the original block length [6, 7]. While there have been proposals for variable-length block ciphers in the past, such as [8], 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 [9], which demonstrates that the problem we deal with has been noticed. The method in [9] 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 per block, regardless of the block size. In comparison, 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 design goals). For example, when both schemes are applied to 128-bit AES, our scheme requires one extra round to encrypt a 136-bit plaintext; whereas, the method in [9] requires ten extra rounds. Unlike our construction, [9] does not modify the original block cipher but adds operations around it, treating the original cipher as a pseudo-random permutation (PRP) and analyzes it as a black box. In our method, 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 allow for reduction-type proofs), we alter the inputs to each round, add whitening steps (when not already present), and extend the key schedule. *A novelty of our methodology is that while modifying the cipher in this fashion, we are able to maintain a reduction between the original and elastic versions which permits us to link their security against key recovery attacks.*

Our work proposes elasticity of block size as a criterion for cipher design, creating areas for future work. These areas include the analysis of elastic versions of concrete ciphers and their security against cryptanalytic techniques not addressed here (since security of block ciphers need to consider various types of specific attacks).

The remainder of the 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. To this end, we introduce the concept of a reduction between ciphers to relate the security of the elastic and original versions of a cipher. Section 4 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 $2b - 1$ bits, 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 in that case.) We neither modify the round function of the block cipher nor decrease the number of rounds applied to each bit; instead, we create a method by which bits beyond the supported block size can be interleaved with bits in the supported block size. We explain the reasoning behind the specific steps in Section 2.2.

We assume that 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 left for subsequent work. Figure 1 illustrates the resulting cipher when the modifications are applied to the version of AES [10] 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 [11], half 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.

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. $r' = r + \lceil yr/b \rceil$.
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, as well as "side whitening" of the extra 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 is viewed as the identity function if it is omitted. Similarly, a key dependent mixing step may be added at the end of the last round.
4. Use the leftmost b-bits that are output from the mixing step as the input to 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 (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.

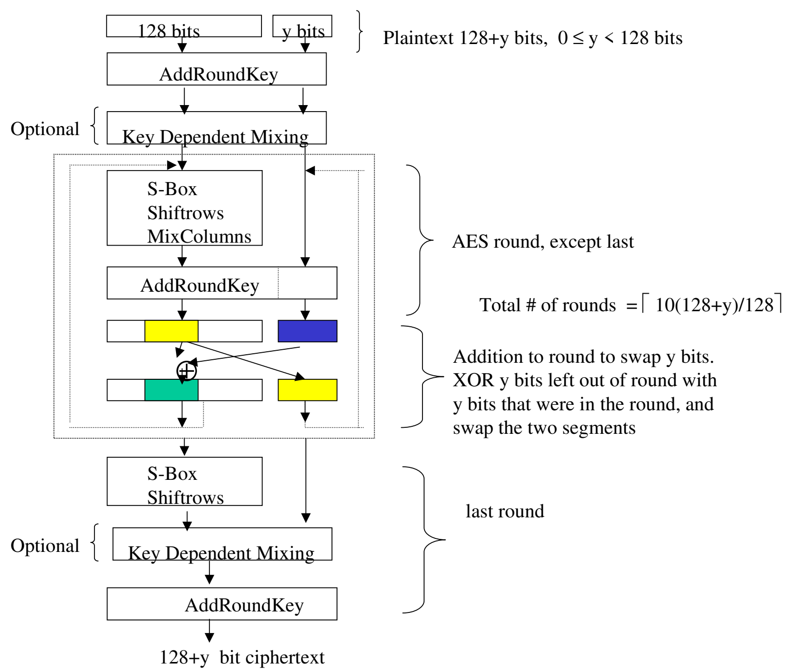

Fig. 1. Elastic Version of AES

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. We discuss this further in Section 2.2.
   (c) Set $Y \leftarrow X \oplus Y$.
   (d) Swap X 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 provide 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.

2.2 Explanation of Algorithm

The method was designed for G' to be equivalent to G (with the possible addition of whitening and the optional key-dependent mixing steps) when the data is an integral number of b-bit 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. This requirement allows the computational workload to increase proportionately to the block size while avoiding 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. As y increases, the number of rounds increases gradually from $r+1$ when $0 < \lceil \frac{ry}{b} \rceil \leq 1$ to $2r$ when $r-1 < \lceil \frac{ry}{b} \rceil \leq r$.

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. 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 guarantees any particular bit is within the y bits with probability $< \frac{1}{2}$. Similarly, a key-dependent mixing step after the last round will prevent a single round differential from occurring with a probability of 1 in the first round of decryption.

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.

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 that 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 is sufficient to select X to be consecutive bits starting at position $a_1 + a_2 * i (\bmod b)$ 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 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 the second round'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.

An issue that is left to subsequent work is how to select the bits to be swapped (or adding the swap step less often) 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.

Decryption: The inverse of the round function, if it is not its own inverse, must be used for decryption. While the structure is similar to an unbalanced Feistel network, it is not a Feistel network due to bits output from the round function in the i^{th} round becoming the bits omitted from the $i+1^{st}$ round. In constrast, in an (unbalanced) Feistel network bits input to the round function in the i^{th} round become the bits omitted from the round function in the $i+1^{st}$ round. [1] Thus, it is not possible to perform decryption by simply running the ciphertext through the encryption function of G' with the round keys used in reverse order. Designing the elastic cipher in this manner increases the diffusion rate compared to that of an unbalanced Feistel network.

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, to generate 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. Using a stream cipher may result in the expanded key being more (pseudo-)random than the output of the key schedule of G and the key schedule for G' (as a schema) need not be changed for new choices of G.

We make the following assumption regarding the expanded keys used for the elastic version of the cipher. In our analysis, we embed a copy of G inside a prefix of G' and assume the round keys bits used outside the embedded G are independent of (do not give any information about) the round key bits used inside the embedded copy of G. We call this independence property needed for the analysis, "*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.,* use a stream cipher to get the "independent" portion of the expanded key bits, and model them in the proof

[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 [12]. 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.

as partial round keys that can be known/ controlled without affecting the security of the other round keys), we feel that this is reasonable assumption for initiating analytic methods that validate the security of a design against given attacks via "reduction-type" proofs in this area. (We note that we can embed G in a place different from the prefix of G' and obtain the same relationship between the security of G' and G, but our analysis in Section 3 concentrates on one embedding of G within the prefix of G'.)

3 Security of G'

3.1 Overview

For any concrete block cipher used in practice (and not treated as a pseudorandom permutation or a member of a family of functions), the cipher cannot be proven secure in a theoretical sense but rather is proven secure against known types of attacks. Thus, we can only do the same for the elastic version of such a cipher. In order to provide a general understanding of the security of our construction, we 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 and the proper expansion assumption on the key schedule. Specifically, we exploit the fact that an instance of G is embedded in G'.

We concentrate on key-recovery attacks. We show how to reduce G' to G in a manner that allows an attack that finds the round keys of G' to find the round keys for G. Security against key-recovery attacks does not by itself imply security (*e.g.,* the identity function which ignores the key is insecure while key recovery is imposible). However, all concrete attacks against real ciphers (differential, linear, *etc.*) attempt key recovery and thus practical block ciphers should be secure against such attacks.

In order to focus on the core components of the algorithm for creating G' from G, we consider G without the optional key-dependent mixing steps described in step 3 of the algorithm. If present, these intuitively only serve to increase the security of G' since they prevent an attacker from knowing with probability 1 which bits are omitted from the first application of the round function. Furthermore, since the mixing steps are added steps (as opposed to modifications to components of G) using key material that is independent of the round and whitening key material (by our assumption on the key schedule), they do not impact our analysis which is entirely dependent on the fact that G' contains an instance of G in a manner that permits a reduction from G' to G.

3.2 Round Key Recovery Attack

As mentioned above, 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 (with a concrete polynomial) in resources to the attack 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 such 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.

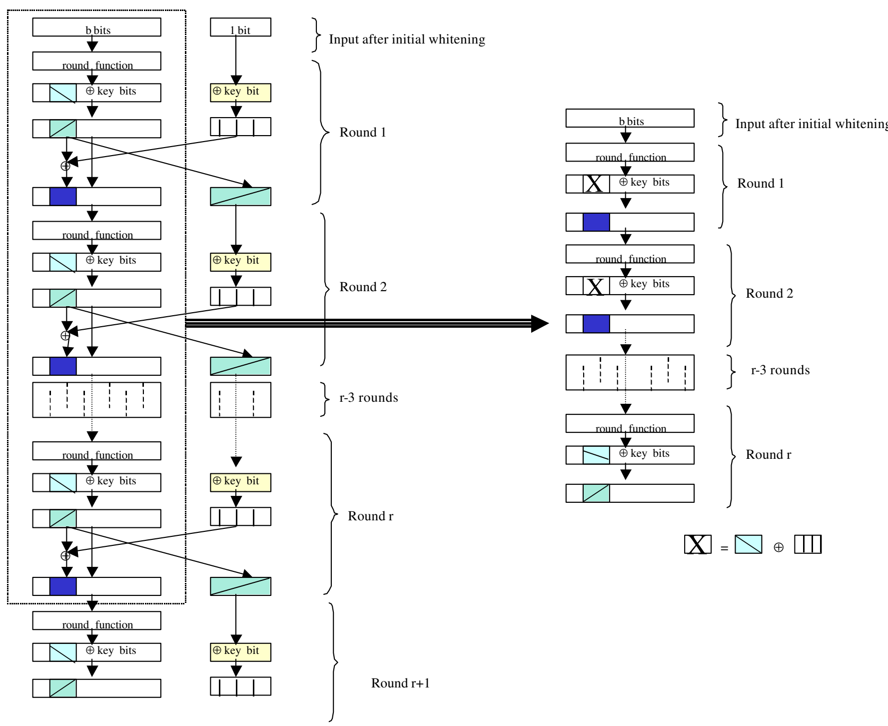

Fig. 2. G **within** G'

Before we formally claim the security, we first draw attention to the fact that the operations performed in G' in its prefix until the leftmost b bit positions go through as many rounds as in G, can be shown to be an application of G, as depicted intuitively in Figure 2. This relationship can be used to convert an attack which finds the round keys for G' to an attack which finds the round keys for G. Recall that 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 as will be argued below.

Theorem I: *If there exists an attack on* $G\,'$ *that 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.*
- *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 ensure 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, 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 exist 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

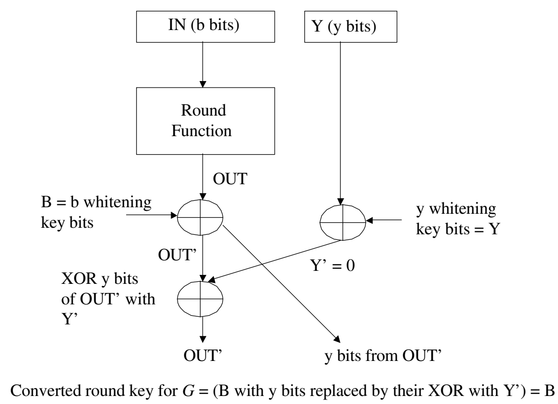

Fig. 3. Converted Key Unchanged in Left b Bits

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 by 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; specifically, 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.

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, thus proving Claim I.

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: Assume there exists a known (plaintext, ciphertext) pair attack on G' which produces the round keys either by finding the original key and then expanding it, or by 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 found. 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 shown 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 $\{(P,U)_{rnd_{q-1}}\}$.

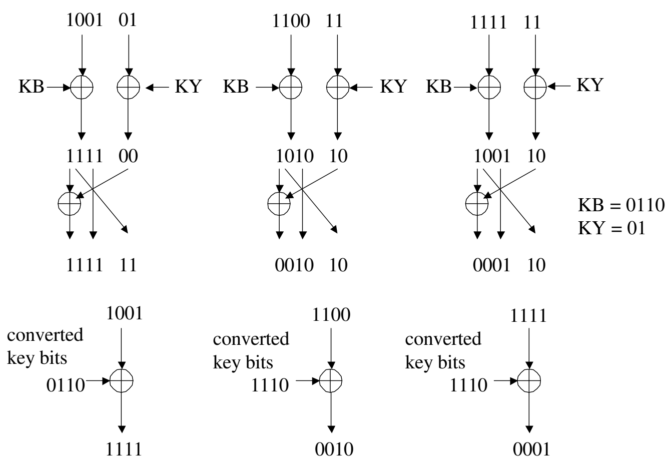

Fig. 4. Forming S'_{rnd_q}

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 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}.

For now, we are only concerned with obtaining a set, and not necessarily the largest, of (plaintext, ciphertext) pairs corresponding to a common key. 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.

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 $\geq \frac{n}{2^{yr}}$, 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 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 that 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 that 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 that 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 quadratic time 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 that 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 rightmost 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.

$\}$

4 Conclusions

We have introduced a new concept, that of an *elastic block cipher* and presented a general method for converting an existing block cipher that is based on a round function into an elastic block cipher. The ability to create an elastic version of a block cipher allows us to "stretch" the supported block size of the cipher up to twice the original length while increasing the computational load proportionally to the block size. We show that the existence of an attack on the elastic version that produces certain round keys implies a key recovery attack exists on the original block cipher by creating a reduction between the elastic and original versions of the block cipher; this methodological contribution appears to be *unique* in the area of block cipher design. Practical applications of elastic ciphers include database and network traffic encryption. Our work suggests that the notion of elasticity can be a new design criterion for block ciphers. Our work also suggests numerous questions for future work. Open issues include how to extend the methodology presented here and how to further analyze elastic designs. Regarding concrete instances of elastic ciphers, since the security of block ciphers is an involved issue and requires consideration of various potential attacks, it is left open how to validate these designs by concrete cryptanalysis of specific attacks, by performance analysis and by considering their applications in various real contexts.

References

1. Kent, S., Atkinson, R.: Security Architecture for the Internet Protocol. Request for Comments (Proposed Standard) 2401, Internet Engineering Task Force (1998)
2. An, J., Bellare, M.: Constructing VIL-MACs from FIL-MACs: Message Authentication Under Weakened Assumptions. In: Proceedings of Advances in Cryptology - Crypto '99, LNCS 1666, Springer-Verlag. (1999)
3. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom Functions Re-Visited: The Cascade Construction and its Concrete Security. In: Proceedings of Foundations of Computer Science, IEEE. (1996)
4. Bernstein, D.: How to Stretch Random Functions: The Security of Protected Counter Sums. In: Journal of Cryptology, Vol. 12(3), Springer-Verlag. (1999) 185–192
5. Black, J., Rogaway, P.: CBC MACs for Arbitrary-Length: The Three-Key Constructions. In: Proceedings of Advances in Cryptology - Crypto 2000, LNCS 1880, Springer-Verlag. (2000)
6. Halevi, S., Rogaway, P.: A Tweakable Enciphering Mode. In: Proceedings of Advances in Cryptology - Crypto 2003, LNCS 2729, Springer-Verlag. (2003)

7. Halevi, S., Rogaway, P.: A Parallelizable Enciphering Mode. Cryptology ePrint Archive, Report 2003/147 (2003) `http://eprint.iacr.org/`.
8. Schroeppel, R.: Hasty Pudding Cipher. `http://www.cs.arizona.edu/rcs/hpc` (1998)
9. Bellare, M., Rogaway, P.: On the Construction of Variable Length-Input Ciphers. In: Proceedings of Fast Software Encryption (FSE), LNCS 1636, Springer-Verlag. (1999)
10. FIPS 197: Advanced Encryption Standard (AES) (2001)
11. FIPS 46-3: Data Encryption Standard (DES) (1999)
12. Schneier, Kelsey: Unbalanced Feistel Networks and Block Cipher Design. In: Proceedings of Fast Software Encryption (FSE), LNCS 1039, Springer-Verlag. (1996)