The Security of Elastic Block Ciphers Against Key-Recovery Attacks

Debra L. Cook[1], Moti Yung[2], Angelos D. Keromytis[2]

[1] Alcatel-Lucent Bell Labs, New Providence, New Jersey, USA
dcook@alcatel-lucent.com**
[2] Department of Computer Science, Columbia University, New York, NY, USA
{moti,angelos}@cs.columbia.edu

Abstract. We analyze the security of elastic block ciphers against key-recovery attacks. An elastic version of a fixed-length block cipher is a variable-length block cipher that supports any block size in the range of one to two times the length of the original block. Our method for creating an elastic block cipher involves inserting the round function of the original cipher into a substitution-permutation network. In this paper, we form a polynomial-time reduction between the elastic and original versions of the cipher by exploiting the underlying network structure. We prove that the elastic version of a cipher is secure against a given key-recovery attack if the original cipher is secure against such an attack. Our analysis is based on the general structure of elastic block ciphers (*i.e.,* the network's structure, the composition methods between rounds in the network and the keying methodology) and is independent of the specific cipher.

keywords: variable-length block ciphers, security analysis, reduction proof, key recovery attacks.

1 Introduction

Elastic block ciphers are variable-length block ciphers created from existing block ciphers [5]. The elastic version of a block cipher supports any block size between one and two times that of the original block length, and results in a computational workload for encryption that is proportional to the actual block size. Our method for creating elastic block ciphers consists of a substitution-permutation network that uses the round function from the existing fixed-length block cipher as a black box. Elastic block ciphers, in turn, can be combined with modes of encryption to support encryption of any size cleartext.

Traditionally, block ciphers are designed to support a specific block size, with the security analysis and design optimized for the supported block size. For a variable-length block cipher, a more general analysis is required to avoid evaluating the cipher separately for each supported block length. Furthermore, for elastic block ciphers it is preferable to be able to analyze the ciphers as a category as opposed to evaluating each one individually against specific attacks to which the fixed-length versions have previously been proven to be immune.

** This work was performed while the author was at Columbia University.

We have extensively analyzed both the underlying structure used to create elastic block ciphers and practical examples of elastic block ciphers. Our analysis has ranged from proving that elastic block ciphers, in theory, provide variable length pseudorandom permutations (PRPs) and strong PRPs to creating and analyzing concrete examples [4]. In this work, we present our analysis of the security of elastic block ciphers against practical attacks. These attacks typically attempt to recover the keys or the round keys of the block cipher. Differential cryptanalysis [3, 7], linear cryptanalysis [9] and exhaustive search methods are instances of such attacks (but other key-recovery attacks exist [2, 13]).

We prove, in general, that the elastic version of a block cipher is secure against attacks that attempt to recover key bits if the original, fixed-length version of the cipher is secure against such attacks. *Our method is unique in that we show how to convert such an attack on the elastic version directly into an attack on the original version with a polynomially related time complexity.* Unlike generic design methodologies, where the component from which security is derived is a well defined black-box building block [6], our proof requires identifying the presence of a fixed-length instance of the block cipher embedded inside the elastic design even though it is the round function and not the original block cipher in its entirety that is used as a black box. As a result of our proof, if the original cipher is (assumed, shown heuristically, or proven to be) immune to a certain type of attack (such as linear or differential cryptanalysis) then the elastic version is also (respectively assumed, shown heuristically, or proven to be) immune to the attack in the same sense (with polynomially related parameters that we concretely calculate).

The use of the round function of the original block cipher as a black box in the elastic version, together with the methods by which we compose rounds and schedule key material, is what enables us to relate the security of the elastic version of a block cipher directly to the security of the original cipher. Our general approach is motivated by reduction-oriented proofs of security. Such proof techniques are not typical in symmetric-key cryptography, especially in concrete designs (for a survey of proof techniques in this area, see [12]:Chapter 4), and are more common in generic designs that assume strong secure components (*e.g.*, assuming a component is a random function or a pseudorandom function [8]).

Our elastic block cipher design exploits existing components of a cipher to gain efficiency and avoids using the entire fixed-length cipher as a black-box (as was done in earlier work, [1, 11]). Thus, it may appear at first that the ability to perform a reduction-based proof is lost. However, the methodology presented in this work demonstrates that even concrete designs that use components of a cipher may resort to reduction-like proof techniques if the components' properties and the composition methods are carefully chosen, even with respect to concrete key-recovery attacks as opposed to only distinguishability attacks, which are more typical in investigations of a formal theoretical nature. To the best of our knowledge, this type of methodology is new in this area. While it is not common in block cipher design, we believe it will be a useful analysis tool in settings that employ cipher components within extended contexts, and may also be of independent interest.

The remainder of the paper is organized as follows. Section 2 summarizes the construction of elastic block ciphers. Section 3 defines the relationship between the security of the elastic version of a block cipher against key recovery attacks to the security of the original cipher against such attacks. Section 4 concludes the paper.

2 Elastic Block Cipher Review

2.1 Overview

We briefly review our method for creating elastic block ciphers [5]. The method converts the encryption and decryption functions of any existing block cipher to accept blocks of size b to $2b$ bits, where b is the block size of the original block cipher. The general structure of an elastic block cipher is shown in Figure 1. An elastic version of a block cipher is created by inserting the cycle of the original fixed-length block cipher into the network structure to form the round function of the elastic version. In each round the leftmost b bits are processed by the round function and the rightmost y bits are omitted from the round function. Afterwards, the rightmost y bits are XORed with a subset of the leftmost b bits and the results swapped. This swapping of bits may be omitted after the last round. The number of rounds in the elastic version is set such that the round function is applied to each bit position at least the same number of times as in the fixed-length version. The elastic version also includes initial and end-of-round whitening, and an initial and final key-dependent permutation. The key-dependent permutations are present to prevent an attacker from knowing with a probability of 1 exactly what y bits are omitted from the first application of the round function when encrypting or decrypting. Decryption is performed by applying the network in reverse with the round function of G' replaced by its inverse, specifically the inverse of the cycle in G.

We use the following notation from the definition of elastic block ciphers [5] throughout the remainder of this paper.
Notation:

- G denotes any existing block cipher with a fixed-length block size that is structured as a sequence of rounds. By default, any block cipher that is not structured as a sequence of rounds is viewed as having a single round.
- A cycle in G refers to the point at which all b-bits of the block have been processed by the round function of G. For example, if G is a Feistel network, a cycle is the sequence of applying the round function of G to the left and right halves of the b-bit block. In AES [10], the round function is a cycle.
- r denotes the number of cycles in G.
- b denotes the block length of the input to G in bits.
- y is an integer in the range $[0, b]$.
- G' denotes the modified G with a $(b + y)$-bit input for any valid value of y. G' will be referred to as the elastic version of G.
- r' denotes the number of rounds in G'.
- The round function of G' will refer to one entire cycle of G.
- The swap step will refer the step in which the rightmost y bits are XORed with y bits from the leftmost b bits and the results swapped.

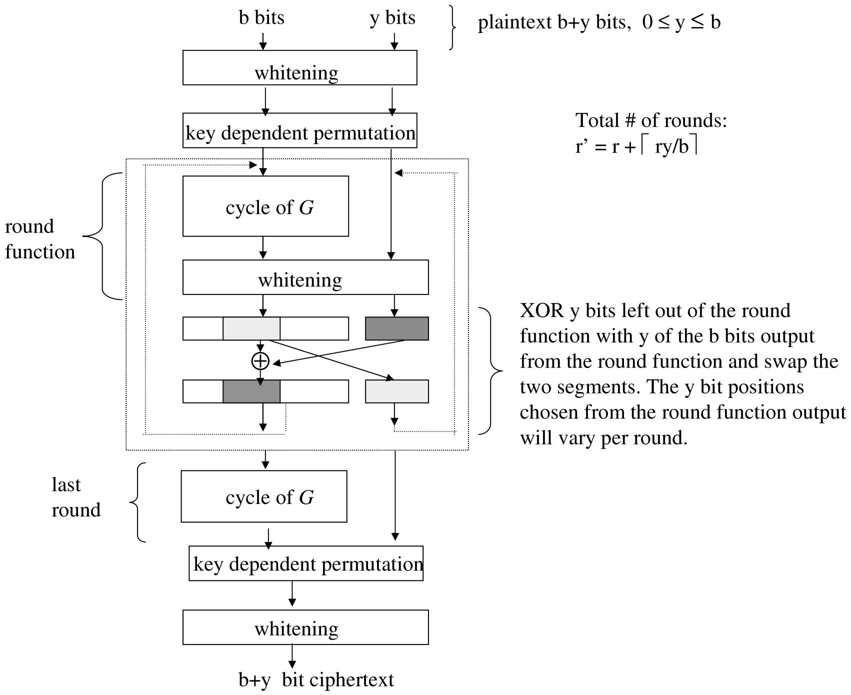

b bits
y bits
plaintext b+y bits, 0 ≤ y ≤ b
whitening
key dependent permutation
Total # of rounds:
r' = r + ⌈ ry/b ⌉
round function
cycle of *G*
whitening
XOR y bits left out of the round function with y of the b bits output from the round function and swap the two segments. The y bit positions chosen from the round function output will vary per round.
last round
cycle of *G*
key dependent permutation
whitening
b+y bit ciphertext

Fig. 1. Elastic Block Cipher Structure

The elastic version of a block cipher requires a greater number of expanded key bits than the original, fixed-length version. In practice, options for the key schedule include using a stream cipher to generate all expanded key bits, applying the original key schedule multiple times, or using the original key schedule for some expanded key bits and a stream cipher or other algorithm for the additional key bits. We note that the use of a stream cipher for the key schedule allows for a generic key schedule across all elastic block ciphers and increases the pseudorandomness of the expanded key bits when compared to existing key schedules, although in practice this incurs the cost of a decrease in the rate of key expansion [4]. The acceptable relationships between the expanded key bits of the elastic version and the original key bits are expressed in the security analysis below.

3 Security Analysis

3.1 Overview

For any concrete block cipher used in practice, as opposed to a theoretical construction of a pseudorandom permutation (PRP), the cipher cannot be proven secure in a formal sense (is not proven to be a PRP or strong PRP) but rather is proven or shown under certain assumptions to be 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 elastic block ciphers, 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 fact that there is an instance of G embedded in G' and is independent of the specific block cipher used for G.

We prove that G' is secure against any attack that attempts to recover the key or the expanded-key bits if G is secure against the attack, under certain assumptions on the independence of the expanded-key bits in G'. This is accomplished by showing how to convert such an attack on G' to an attack on G. We believe this result is important because it implies that G' does not have to be analyzed against any practical attack to which G is immune (unless a more refined analysis than the reduction is required). Our approach is novel because we show how to convert an attack on the variable-length version of a block cipher directly into an attack on the fixed-length version of the block cipher, and, in general, it points out at a direction of identifying embedded ciphers inside ciphers when the design is not purely of a black box fashion.

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 impossible). However, all concrete attacks against real ciphers (linear, differential, higher order differential, impossible differential, related key attacks, *etc.*) attempt key or expanded-key recovery and thus practical block ciphers should be secure against such attacks. We note that if there is a relationship between the plaintext and ciphertext bits that does not involve the key bits, this relationship would either manifest itself in the results of statistical tests on whatever versions of the block cipher (original and/or elastic) for which the relationship holds, and/or as algebraic equations relating the plaintext and the ciphertext.

3.2 G within G'

Before stating our theorem, we provide some preliminary analysis that assists us in conveying the linkage between the original and elastic versions of a block cipher. For simplification of terminology only, we will refer to the fixed-length block cipher G as if the round function of G is a cycle and omit using the term ”cycle”. For any G in which a cycle involves multiple applications of the round function, such as in a Feistel network, our analysis holds by referring to a cycle of G instead of the round function of G.

We first draw attention to the fact that the operations performed in G' on the leftmost b-bit positions in r consecutive rounds is an application of G. This is depicted intuitively in Figure 2. We note that we are concerned only with r consecutive rounds of G' and do not include either the initial or final key-dependent permutation present in the definition of elastic block ciphers. This relationship between G' and G can be used to convert an attack which finds the round keys for G' to an attack which finds the round keys for G. Let G_{rk} denote G using round keys rk and let $G_k{}'$ denote G' using key k. Let (p, c) be a b-bit (plaintext, ciphertext) pair, and let x and z each be of length y. $\|$ denotes concatenation. 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 swap steps in G' into a whitening step. The leftmost b bits of the initial whitening in G' are used as the initial whitening in G and the rightmost y bits of the initial whitening in G' are dropped. The resulting end-of-round whitening key bits for G will vary in up to y positions across the (plaintext, ciphertext) pairs when collapsing the steps from G'; however, it is possible to use these keys to solve for the round keys of G.

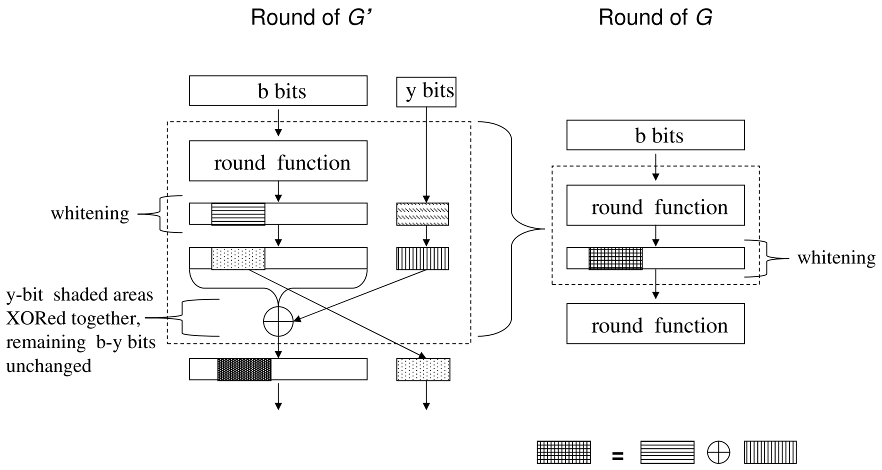


Fig. 2. G within G'

The following claim shows that for any set of (plaintext, ciphertext) pairs encrypted under sets of round keys in G' where the rightmost y bits used for whitening in each round may vary amongst the sets and all other key bits are identical amongst the sets, 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 leftmost b bits of output of the r^{th} round of G' prior to the swap step.

Claim 1: Let G be a b-bit block cipher and G' be its elastic version. Let $\{(pi, ci)\}$ denote a set of n (plaintext, ciphertext) pairs such that $|pi| = |ci| = b$. Let $b + y$ be the variable block size for G' where $0 \leq y \leq b$. Let w be a y-bit constant. Let vi be a y bit string that may vary per i, for $i = 1$ to n. Under the following assumptions regarding the key schedules:

– The rightmost y bits of each whitening step in G' can take on any value and are independent of any other expanded-key bits within the round and in other rounds.

– There are no message-related expanded keys. Any expanded-key bits utilized in G depend only on the key and do not vary across plaintext or ciphertext inputs.
– Any expanded-key bits used in the round function of the r consecutive rounds of G' can take on the same values as the expanded-key bits used in the round functions of G.
– If G contains initial and end-of-round whitening, any expanded-key bits used for the leftmost b bits of each whitening step in r consecutive rounds of G' can take on the same values as the whitening bits in G.

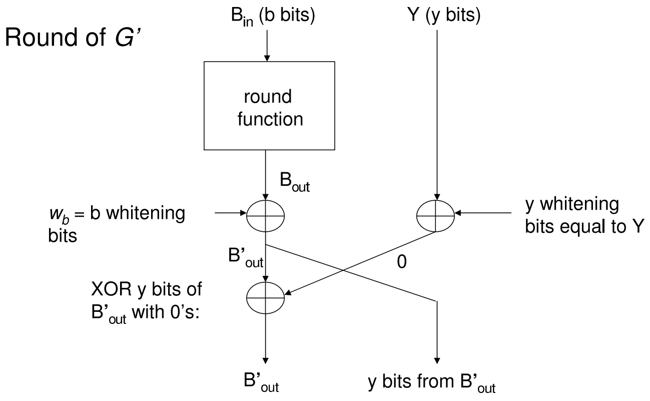
if $G_k(pi) = ci$ then there exists n sets of round keys for the first r rounds of G' that are consistent with inputs $pi \parallel w$ producing $ci \parallel vi$ as the output of the r^{th} round prior to the swap step at the end of the r^{th} round, for $i = 1$ to n, such that the leftmost b bits used for whitening in each round are identical across the n sets and any expanded-key bits used internal to the round function are identical across the n sets.

Proof. Let $rk = \{rk_0, rk_1, ...rk_r\}$ be the set of round keys corresponding to key k for G. rk_0 denotes the key bits used for initial whitening. For each (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 $rki' = \{rki'_0, rki'_1, ...rki'_r\}$ 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 of round keys formed, 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 case in which G does not contain whitening steps corresponds to using 0's for the leftmost b bits of each whitening step in G'.

The operations of G' on the leftmost b bits of rounds 1 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 $0's$. Thus, the leftmost b bits in the output of the r^{th} round prior to the swap step 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, thus proving the claim.

3.3 Reduction Between the Original and Elastic Versions of a Cipher

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 that is polynomially related in resources to the attack on G'. Assuming that G itself is resistant to such attacks, we conclude that G' is also resistant to such attacks. We note that if an attack finds the key as opposed to the expanded-key bits (the round keys) then the attacker can apply the key schedule to the key to obtain the round keys. Therefore, in our analysis, we view any



The b whitening key bits for *G* will be w_b when converting the round key for *G'* to a round key for *G* because the XOR in the swap step involves XORing with 0's.

Fig. 3. Converted Key Unchanged in b Whitening Bits

key recovery attack as providing the round keys 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 chosen plaintext, chosen ciphertext *etc.*, the attacker acquires a set of such pairs.

In our analysis, we consider G' without the initial and final key-dependent permutations. This allows us to focus on the core components of the elastic block cipher algorithm. If present, the initial and final permutations only serve to increase the security of G' since they prevent an attacker from knowing with probability one which bits are omitted from the first application of the round function when encrypting or decrypting. Furthermore, since these permutations are added steps (as opposed to modifications to components of G) using key material that is independent of the round and whitening key bits, they do not impact our analysis.

Theorem 1. *Given a fixed-length block cipher, G, that works on b-bit blocks and its elastic version, G', that works on $(b+y)$-bit blocks, where $0 \leq y \leq b$, if there exists an attack, $A'_{G'}$, on G' that allows the round keys to be determined for r consecutive rounds of G' using polynomial (in b and/or r) time and memory, then there exists an attack on G with r rounds that finds the round keys for G and that uses polynomial (in b and/or r) many resources as $A'_{G'}$, assuming:*

- *There are no message-related expanded keys. Any expanded-key bits utilized in G depend only on the key and do not vary across plaintext or ciphertext inputs.*
- *An attack on r' rounds of G' implies a reduced-round attack on r rounds of G' for $r \leq r'$.*

- *$A'_{G'}$ finds all possible sets of round keys, if more than one set exists.*
- *Any expanded-key bits used in the round function of r consecutive rounds of G' can take on the same values as the expanded-key bits used in the round functions of G.*
- *If G contains initial and end-of-round whitening, any expanded-key bits used for the leftmost b bits of each whitening step in r consecutive rounds of G' can take on the same values as the whitening bits in G.*

Before beginning the proof, we have a few comments on the theorem and assumptions. We first note that for an attack on G' to be computationally feasible, it must involve $< 2^b$ (plaintext, ciphertext) pairs because otherwise an exhaustive search on G would be possible, implying G is insecure against practical attacks. The first assumption is typical of existing block ciphers and is true of the elastic versions of block ciphers. The second assumption is true of block ciphers used in practice. The last two assumptions mean that the key schedule of G' is defined such that a subset of the expanded-key bits can have the same values as if they were generated by the key schedule of G. These assumptions are easily satisfied in practice by using the key schedule of G to generate a subset of the round key bits and a separate algorithm to generate the expanded-key bits required in G' for the additional $r' - r$ rounds and any whitening present in G' that is not present in G. Another option is if the key schedule of G' generates pseudorandom expanded-key bits such that it is possible the expanded-key bits for the round function and leftmost b bits of whitening in r consecutive rounds can take on the same values generated by the key schedule of G. In practice, given an expanded-key, it is feasible to check if the expanded-key adheres to a specific block cipher's key schedule. A subset of the expanded-key bits being tested can be inserted into the key schedule to generate additional key bits which can be checked against the bits in the value being tested.

The theorem holds by default for the case when $y = 0$, since G' is just G (with the possible addition of whitening which can be set to 0's when applying the attack if G does not contain whitening). We view G as having whitening steps in the proof to Theorem 1. This is not an issue for the following reason. 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, then 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. Setting a subset of bits in each whitening step in G' to 0's is equivalent to using a weaker version of G'. Any attack that works on G' will work on the weaker version. This is merely the case where the attacker knows certain bits of each whitening step are 0's.

We note that Theorem 1 only states that an attack on G' can be converted to an attack on G and not the reverse. This is because, in general, the claim that an attack on G can be converted into an attack on G' does not hold. Consider the case when G contains the initial and end-of-round whitening steps. When $y = 0$, G' is G with the initial and final key-dependent permutations added and the key schedule replaced (such as by a stream cipher). If the attack on G is due to the original key schedule, the attack does not necessarily hold if the key schedule is changed to generate pseudorandom bits when creating G'. For any attack not due to the key schedule, in order to claim that an attack on G implies an attack on G', it is necessary that the attack on G be such that the addition of the initial and final key-dependent permutations, the addition or expansion of the whitening steps and the addition of the swap steps do not result in the attack becoming inapplicable or computationally infeasible. In general, the conversion of an attack from G' to G works because there is a decrease in the complexity of the block cipher being attacked when going from G' to G; whereas, the reverse is not true because there is an increase in the complexity of the block cipher when converting G to G'.

To prove Theorem 1, we must show for any value of y, where $0 \leq y \leq b$, that if an attack exists on G' it can be converted into an attack on G using polynomial time and memory. We define the steps for converting a round-key recovery attack on G' to an attack on G. We describe two ways of performing the conversion. The first method works for any value of y, where $0 \leq y \leq b$. The second method is is applicable for values of y satisfying $r(y-2) < b$, where r is the number of rounds in the original cipher. We include the second method because it requires fewer computations than the first method and thus is useful for small values of y. The methods treat whitening key bits as if they are pseudorandom in that the whitening key bits can take on any value. In G, if there is a relationship amongst the whitening key bits and/or between whitening key bits and key material used within the round function due to the key schedule of G, such keys will be a subset of all the possible sets of round keys found using the attack on G'. Then the set of round keys that satisfies the key schedule of G can be determined by checking which of the potential keys corresponds to the key schedule. If the number of potential sets of round keys found by the attack on G' is large enough such that it is computationally infeasible to determine which ones adhere to the key schedule of G, then the attack on G' is not computationally feasible. This is because the number of potential sets of round keys it finds for a set of (plaintext, ciphertext) pairs will also be large enough such that it is computationally infeasible for an attacker to determine which set to use to decrypt additional ciphertexts.

When we refer to converting the round keys of G' into round keys for G, we mean the following: In round j of G', let b_{jl} denote the l^{th} bit of the b bits output from the round function prior to the end-of-round whitening. Let kw_{jl} denote the end-of-round whitening key bit applied to b_{jl}. If b_{jl} is involved in the swap step at the end of round j, let y_{jh} denote the bit from the rightmost y bits with which b_{jl} is swapped and let kw_{jh} denote the whitening key bit applied to y_{jh}. Set the l^{th} whitening bit in round j of G to $kw_{jl} \oplus kw_{jh} \oplus y_{jh}$ when $j \geq 2$. When $j = 1$, the l^{th} whitening bit is set to $kw_{1l} \oplus kw_{1h} \oplus y_{1h} \oplus kw_{0h}$ in order to include the initial whitening on the rightmost y bits in the conversion. Set all other key bits used in G (both whitening and any internal to the round function) to be identical to the key bits used in G'. We refer to the initial whitening as round 0. The initial whitening for G' is converted to initial whitening for G by using the leftmost b expanded-key bits of the initial whitening as the initial whitening in G.

Proof of Theorem I: First Method We describe here a method for converting the attack on G' to an attack on G. Without loss of generality, we use the first r rounds of G' as the r consecutive rounds for which the round keys are found. The conversion is presented in terms of solving for the round keys from the initial whitening to round r, but may also be performed by working from round r back to the initial whitening or by using any consecutive r rounds with whitening applied before the first round as long as the plaintext for G is the leftmost b bits of input to the r rounds and the corresponding ciphertext from G is the leftmost b bits of the output of the r rounds.

This attack runs in quadratic time in the number of rounds of G. The attack, $A'_{G'}$, 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. By the second condition in Theorem 1, if an attack on G' with r' rounds exists, then a reduced round attack on G' exists for any number of rounds $< r'$.

Let P be a set of plaintexts and C be a set of ciphertexts. We use the notation $\{(P, C)\}$ to indicate a set of (plaintext,ciphertext) pairs of the form (pi, ci) with $pi \in P$ and $ci \in C$. Given a set $\{(P^*, C^*)\} = \{(pi^*, ci^*)\}$ of n (plaintext, ciphertext) pairs for G, create a set $\{(P, C)\} = \{(pi^* \parallel 0, ci^* \parallel vi_r)\}$ 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_r values appended to the ci^* values are arbitrary and do not need to be identical. The r subscript in vi_r denotes the number of rounds. Our method runs reduced round attacks on G' and the vi_r's can vary each time. Solve G' for round keys 0 and 1. By the pseudorandomness of the round keys, sets of round keys exist that correspond to $\{(P, C)\}$ and which are identical in at least the initial whitening and first round (the round keys across all n pairs may be identical in additional rounds, but we are only concerned with the initial whitening and first round at this point in the process). 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 pi_1 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 $\{(P, C)\} = \{(pi_1 \parallel 0, ci^* \parallel vi_{r-1})\}$ and solve for the first round key of G'. As before, convert the resulting round key for the first round of G' to a round key for G, but this time use the converted 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 involves applying each round of G to n inputs for a total of rn rounds of G. $\frac{n(r+1)r}{2}$ rounds of G' are computed in the worst case if $A'_{G'}$ requires knowing the output of each round of the reduced round version of G' to find the first round key. r applications of $A'_{G'}$ are needed on the reduced round versions of G'. Let t_A denote the time to run $A'_{G'}$. Let ks_t be the time to check that an expanded-key found by $A'_{G'}$ adheres to the key schedule of G. The time to attack G is $O(nr^2 + rt_A + ks_t)$.

In summary, the attack on G can be written as:
Input $\{(P^*, C^*)\} = \{(pi^*, ci^*)$ for $i = 1$ to $n\}$.
Create $\{(P, C)\} = \{(pi^* \parallel 0, ci^* \parallel vi_r)$ for $i = 1$ to $n\}$ for a r-round version of G',

```
    where the $vi's$ are arbitrary.
Using $A'_{G'}$, solve a $r$-round version of $G'$ for $rk'_0$ and $rk'_1$.
Convert $rk'_0$ to $rk_0$ and $rk'_1$ to $rk_1$.
Set $pi_1 =$ first round output of $G$ using $rk_0$ and $rk_1$, for $i = 1$ to $n$.
For $j = 1$ to $r - 1$ {
        $\{(P, C)\} = \{(pi_j \parallel 0, ci^* \parallel vi_{r-j})$ 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 $rk_{j+1}$.
        $pi_{j+1} =$ output of round $j + 1$ of $G$ on $pi_j$ using $rk_{j+1}$, for $i = 1$ to $n$.
}
```

Proof of Theorem I: Second Method Our second method for proving Theorem 1 requires fewer computations than the first method, but provides round keys for a smaller set of (plaintext, ciphertext) pairs. 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. If there are n_j (plaintext, ciphertext) pairs involved at round j, there will be at least $\frac{n_j}{2^y}$ pairs remaining for which the round keys are consistent after round j. 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.

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 $|xi| = |zi| = y$.

Let $A_{G'}$ be an attack on G' that finds the key(s) corresponding to $\{(P, C)\}$ in time less than an exhaustive search for the key. Let m denote the number of keys found. In practice, only one key should be found for any set of (plaintext, ciphertext) pairs. $m > 1$ only impacts the time to perform the attack and not the method itself. Without loss of generality, it is assumed that the keys are available in expanded form.

Let k be one of the m keys found by $A_{G'}$ and let ek be the expanded-key bits corresponding to k. Let $\hat{ek}_i$ be the expanded-key bits for G resulting from the conversion of ek when applied to the i^{th} element of $\{(P, C)\}$. Let R_{int} denote any bits of ek utilized within the round function. The values found for the bits of R_{int} will be the same for G' and G (the same in ek and every $\hat{ek}_i$). For each i, the bits of $\hat{ek}_i$ corresponding to the initial whitening in G (round 0) will be the leftmost b bits of the initial whitening bits from ek.

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

When the round keys from ek are converted to those for $\hat{ek}_i$, at most y bits change in the leftmost b bits of each end-of-round whitening step. Thus, the resulting round keys for round q, $1 \leq q \leq r$ 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 $\hat{ek_i}$ values from $S_{rnd_{q-1}}$ that have identical bits for round q, where $S_{rnd_0} = \{\hat{ek_i}$, for $i = 1$ to $n\}$. 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. There is no swap step after the r^{th} round so $|S_{rnd_r}| = |S_{rnd_{r-1}}|$. Across r rounds, the number of (plaintext, ciphertext) pairs are reduced at most $r-1$ times.

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 whitening bits of 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 elements of $S_{rnd_{q-1}}$ that produced 1110 as the q^{th} round key, and $\{(P,U)_{rnd_q}\}$ contains the second and third (plaintext, ciphertext) pairs from $\{(P,U)_{rnd_{q-1}}\}$.

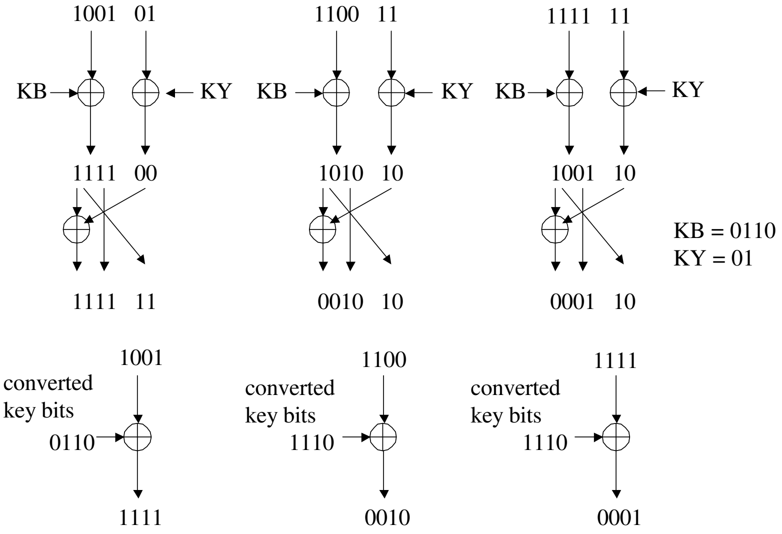


Fig. 4. Forming S_{rnd_q}

Let rk be the contents of S_{rnd_r}. rk is the expanded key bits for G. Let $\{(P,C)_G\} = \{(pi, ci) | (pi \parallel yi, ui \parallel vi) \in \{(P,U)_{rnd_r}\}\}$. $|\{(P,C)_G\}| \geq n/2^{y(r-1)}$. $\{(P,C)_G\}$ is a set of (plaintext, ciphertext) pairs for which $G_{rk}(pi) = ci \; \forall \; (pi, ci) \in \{(P,C)_G\}$.

So far we have defined a method that produces a set of at least $\frac{n}{2^{y(r-1)}}$ (plaintext, ciphertext) pairs that are consistent with the round keys. This lower bound on the number of (plaintext, ciphertext) pairs can be slightly increased to $\frac{n}{2^{y(r-2)}}$ by using $(b+y)$-bit plaintexts that are the same in the rightmost y bits (which we did by setting these bits to 0). This will result in $|S_{rnd_1}| = n$. Since we also have $|S_{rnd_r}| = |S_{rnd_{r-1}}|$, the set of (plaintext, ciphertext) pairs is not reduced in the first and r^{th} rounds. Then the number of (plaintext, ciphertext) pairs produced for G that are consistent with the round keys for G is $\geq \frac{n}{2^{y(r-2)}}$. The number of possible plaintexts for G is 2^b; therefore, it is necessary for $y(r-2) < b$ to use this method.

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', and then for G as follows: 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 an r-round version of G' by setting $pi \parallel xi = pi^* \parallel 0$ and $ci \parallel zi = ci^* \parallel zi$, for $i = 1$ to n. For the set of (P, C) pairs created, $\{(P, U)\} = \{(pi^* \parallel 0, ci^* \parallel zi)\}$. 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 round keys in S_{rnd_r} will be consistent with the (plaintext, ciphertext) pairs in $\{(P, U)_{rnd_r}\}$. A set of round keys that adheres to the key schedule of G will be found by Claim 1 and the assumption that the attack on G' finds all possible sets of round keys.

Let t_r be the time to run r rounds of G' and t_A be the time to run $A_{G'}$. Recall that m is the number of keys (sets of round keys) found by $A'_{G'}$. In the case of obtaining at least one set $\{(P, U)_{rnd_r}\}$ of size $\geq \frac{n}{2^{y(r-2)}}$, 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. Let ks_t be the time to check that an expanded-key adheres to the key schedule of G. Thus, the additional time required to attack G (beyond the time required to attack G') is $O(nm(r + t_r) + mks_t)$. The only unknown value is m. 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.

4 Conclusions

We have proven that the elastic version of a block cipher is secure against any practical attack that attempts to recover key or expanded-key bits if the original cipher is secure against the attack. This eliminates the need to analyze an elastic version of a block cipher against these types of attacks if the original cipher is secure against such attacks (unless one is interested in improving the concrete work factors and probabilities of success). Our result follows from the network structure used in creating elastic block ciphers and the fact that the round function of the original fixed-length block cipher is used as a black box when forming its elastic version. We note that while reduction-based proofs of security are a cornerstone of cryptographic analysis, they are typical when complete components are used as sub-components in a larger design and used in a black box fashion. We are not aware of the use of such techniques in the case of concrete block cipher designs.

Acknowledgments

This work was partially supported by NSF Grants ITR CNS-04-26623 and CPA CCF-05-41093. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF or the U.S Government.

References

1. M. Bellare and P. Rogaway, On the Construction of Variable Length-Input Ciphers, *Proceedings of Fast Software Encryption 1999*, LNCS 1636, Springer-Verlag, pages 231-244, 1999.
2. E. Biham, New Types of Cryptanalytic Attacks Using Related Keys, *Proceedings of Advances in Cryptology - Eurocrypt 1993*, LNCS 0765, Springer-Verlag, pages 398-409, 1994.
3. E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, New York, 1993.
4. D. Cook, *Elastic Block Ciphers*, Ph.D. Thesis, Columbia University, 2006.
5. D. Cook, M. Yung and A. Keromytis, Elastic Block Ciphers: The Basic Design, *Proceedings of ASIACCS*, ACM, pages 350-355, 2007.
6. L. Knudsen, *Block Ciphers - Analysis, Design and Applications*, Ph.D. Thesis, Aarhus University, `http://www2.mat.dtu.dk/people/Lars.R.Knudsen`, 1994.
7. L. Knudsen, Truncated and Higher Order Differentials, *Proceedings of Fast Software Encryption 1994*, LNCS 1008, Springer-Verlag, pages 196-211, 1995.
8. M. Luby and C. Rackoff, How to Construct Pseudorandom Permutations from Pseudorandom Functions, *Siam Journal of Computing*, vol. 17, no. 2, pages 373-386, April 1988.
9. M. Matsui, Linear Cryptanalysis Method for DES Cipher, *Proceedings of Advances in Cryptology - Eurocrypt 1993*, LNCS 0765, Springer-Verlag, pages 386-397, 1994.
10. NIST, FIPS 197 Advanced Encryption Standard (AES), 2001.
11. S. Patel, Z. Ramzan and G. Sundaram, Efficient Constructions of Variable-Input-Length Block Ciphers, *Proceedings of Selected Areas in Cryptography 2004*, LNCS 3357, Springer-Verlag, pages 326-340, 2004.
12. S. Vaudenay, *A Classical Introduction to Cryptography*, Springer, Berlin, 2006.
13. D. Wagner, The Boomerang Attack, *Proceedings of Fast Software Encryption 1999*, LNCS 1636, Springer-Verlag, pages 156-170, 1999.