Elastic AES

Debra L. Cook, Moti Yung, Angelos D. Keromytis
Department of Computer Science
Columbia University
{*dcook,moti,angelos*}*@cs.columbia.edu*
June 6, 2004

Abstract. Recently an algorithmic schema was proposed for converting any existing block cipher into one which excepts variable length inputs with the computational workload increasing in proportion to the block size. The resulting cipher is referred to as an elastic block cipher. The initial work presented immunity to certain key recovery attacks, and left open further analysis of the method and its possible instantiations. In order to provide a concrete example of an elastic block cipher, we design and implement an elastic version of the Advanced Encryption Standard (AES) cipher which accepts all block sizes in the range 128 to 255 bits. To validate the design we perform differential cryptanalysis on elastic AES which confirms that the cipher does not introduce potential differential attacks as a result of a subset of bits being omitted from each round (which is at the heart of the elastic design). We also compare the performance of software implementations of elastic AES and regular AES on variable size inputs, as a step in determining the practicality of the elastic version.

Keywords: Block Cipher Design, Elastic Block Cipher, Variable Length Block Cipher, Differential Cryptanalysis.

1 Introduction

The concept of an elastic block cipher was introduced in [3] and provides a method by which an existing block cipher can be modified to create a variable length block cipher accepting all block lengths between one and two times its original block size. We use this method to create an elastic version of AES [1] in order to provide a concrete example of an elastic block cipher. In [3], Cook, *et al.* define the general construction of an elastic block cipher and generically relate the security of the elastic version to the security of the original cipher, proving the existence of any attack recovering key material for the elastic version implies a similar security weakness in the original cipher. However, specific examples of elastic ciphers are not discussed and many issues such as concrete design and cryptanalysis are left as open issues.

The manner in which the main idea behind the elastic block cipher, that of interleaving the bits between rounds, is precisely incorporated into a cipher depends on the specific cipher. In the elastic version of a block cipher, bits beyond the normal block size are left out of the round function then XORed and swapped with bits output from the round function in order to become part of the input to the next round. The bits output from the round function involved in the XOR become the set left out in the next round. The number of rounds are increased such that the round function is applied to each bit position the same number of times as in the original block cipher. However, the general algorithmic schema does not specify exactly how to select the bits to involve in the XOR and omit from the next round. Instead, it allows the selection to depend on the particular cipher (which it must, as mentioned in [3], due to the fact that a cipher's round function may process subsets of input bits differently).

In this work we design a specific elastic version of AES and perform differential cryptanalysis on it to confirm that the swapping can be implemented in a manner which does not introduce the potential for differential attacks as a result of a subset of bits being omitted from each round. This result validates the strength of elastic ciphers design against differential attacks.

Elastic ciphers do not increase the size of the cleartext, since no padding techniques are needed. We also compare the time performance of software implementations of elastic AES and a regular AES when encrypting increased size blocks. This is a step in determining the practicality of the elastic version. Specifically, we show that the amount of overhead added to AES to create the elastic version is small. This allows the elastic version to offer a substantial performance benefit when encrypting plaintext that is only slightly longer than AES's normal block size of 128 bits compared to padding the plaintext to two full blocks. Naturally, as the length of the plaintext increases towards twice the block size, the speedup of the elastic AES decreases relative to two applications of regular AES (using plaintext-padding to encrypt two full blocks). This demonstrates the proportionality of the increase of work in the elastic version which was a design goal of the schema.

The remainder of this paper is organized as follows. In Section 2 we step through the algorithm for constructing an elastic version of a block cipher and explain how each step is implemented in the elastic version of AES. We provide a brief overview of the security of elastic AES and a derivation of upper bounds for differentials in Section 3. In Section 4 we compare the performance of elastic AES to the performance of AES. Section 5 concludes the paper.

2 Elastic AES

In [3], Cook *et al.* presented an algorithm for modifying the encryption and decryption functions of existing block ciphers to accept blocks of size b to $2b-1$, where b is the block size of the original block cipher. We apply this algorithm to AES to create an elastic version of AES accepting block sizes of 128 to 255 bits. Figure 1 illustrates the general structure elastic AES. We recall that the algorithm in [3] was designed such that it neither modifies the round function of the block cipher nor changes the number of rounds applied to each bit, but rather creates a method by which bits beyond the supported block size can be interleaved with bits in the supported block size. In order to clearly explain how the algorithm from [3] is applied to AES, we list the steps of the algorithm and indicate for each step how we applied it to AES. We will use the following 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.

We call a bit (position) input to a block cipher *active* in a round if the bit is input to the round function. In AES all bits are active in each round.

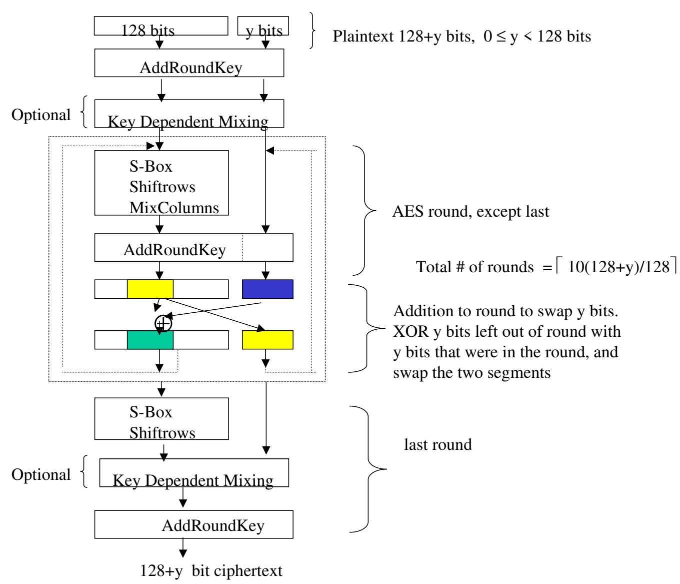

Fig. 1. Elastic Version of AES

Given G and a plaintext P of length $b+y$ bits, where $0 \le y < b$, the following modifications to G's encryption function are made to create the encryption function of $G\,'$. In our case, G is AES and b is 128.

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. $r' = r + \lceil yr/b \rceil$. For elastic AES, $r = 10$ and r' ranges from 10 (when $y = 0$) to 20 (when $b+y \geq 244$).
2. XOR all $b+y$ bits with key material as the first step. This requires including y extra bits in AES's AddRoundKey step.
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. Similarly, a key dependent mixing step may be added as a final step. We implemented these by using a simple key dependent rotation. After the rotation we also XOR the rightmost y bits with the leftmost y bits so that every bit has some influence in the first round. The mixing steps are included in order to avoid a one round differential that occurs with a probability of 1.
4. Input the leftmost b bits output from the mixing step into the round function.
5. At the end of each round, whitening is applied to all $b+y$ bits by XORing all $b+y$ bits with key material. In the case of AES, this requires including the extra y bits in the AddRoundKey step after each round.
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. This step is referred to as "swapping" or the "swap step". 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. The general elastic block cipher algorithm does not specify exactly how to select X but allows the selection to depend on the specific block cipher.
   (c) Set $Z = X \oplus Y$.
   (d) Swap Z and Y to form the input to the next round.

   We implement this swap step by selecting y sequential bits from the leftmost b bits, wrapping around from the right end to the left as needed. The starting position is varied by moving one byte to the right each round to avoid the same bit positions from being used in each swap. This avoids any complex selection process for choosing the y bits that would decrease performance. Due to the nature of AES's round function involving all bits equally (in contrast consider a Feistel network where only half the bits may be acted upon in a given round and RC6 [7] where bits are acted upon differently in a round) and the XOR operation in the swap step allowing bits swapped out to still impact the round, it is permissible to use sequential bits. We also note that the security analysis of the general elastic block cipher algorithm in [3] and our differential analysis in Section 3 are independent of the exact method for selecting the bits to swap and apply even to the most trivial means of selection.

In elastic AES, 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.

The elastic version of AES requires additional key bits for each whitening step compared to AES. It also requires key bits for the two mixing steps. Instead of modifying the key expansion to produce $128 + y$ bit keys for the initial whitening and round keys, all of the key material was generated by a stream cipher, with the 128-bit key used as the key for the stream cipher. Specifically, we used RC4 with the first 512 bytes discarded [6, 8]. This allows us to produce a more random set of round keys than if the AES key schedule was used.

3 Differential Cryptanalysis of Elastic AES

We briefly discuss the security of elastic AES then provide a detailed 3 round differential cryptanalysis to obtain upper bounds on differentials. As was shown in [3], any attack on $G\,'$ which produces the round keys for $G\,'$ implies an attack on G exists and can be used to find the round keys for G. This is due to the ability to convert the round keys for $G\,'$ corresponding to a set, S, of plaintext, ciphertext pairs into a set of round keys for G corresponding to a set of plaintext, ciphertext pairs formed from a subset of S. In the case of our elastic version of AES, the round keys produced by such an attack are used only for whitening, since there are no key bits used internally in the round function. When the round keys for $G\,'$ are converted into round keys for G, the results correspond to solutions for a version of AES with pseudo-random round keys (since we replace the AES key schedule with a stream cipher when creating the elastic version). We can easily prune the sets of round keys of keys that do not adhere to AES's key schedule.

We also note that with respect to linear cryptanalysis [5], if a set of equations exist relating the round key, plaintext and round output bits for r rounds of G', the equations may be modified in two ways to create a set of equations corresponding to G. The first possibility is to replace variables representing any of the first b bits which are swapped in a round with the XOR of the variables which are XORed in the swap step, and discard any variables representing the last y bits of whitening in the last round key and of the last y bits of the ciphertext to obtain a set of equations relating the round key bits, plaintext and round outputs for G. The second possibility is to replace the variables representing the key bits used for the rightmost y bits of whitening in each round with the variables representing the rightmost y bits of input to the round (i.e. the y bits left out of the round function). Again discard any variables representing the last y bits of whitening in the last round key and the last y bits of the ciphertext.

To gain an understanding of the potential for differential cryptanalysis [2], the impact of a differential that is 0 except for 1 byte was traced through 3 rounds of the elastic version. A 1-byte differential was used because it has the greatest single round probability of 2^{-6} for AES (refer to pages 205-206 of [4]) and thus also has the greatest single round probability for elastic AES since the round function is unmodified. The analysis excludes the optional key dependent mixing steps. Determing the bounds without this step allows us to determine the extent of mixing required. Including a simple key dependent rotation for both the initial and final mixing steps will decrease the upper bound by $\frac{1}{4}$. The analysis was performed in a manner that applies to $128+y$ bits for any $y \in [1, 127]$. By the 4^{th} round of the elastic version, at least 12 of the 16 bytes input to the round function will differ. We note that in AES complete diffusion occurs by the end of the second round, meaning every output bit of the second round has been affected by every input bit to the first round (refer to page 41 of [4]). In the elastic version of AES without the initial key dependent mixing step, if $y > 0$, three rounds are required before every bit has affected every other bit. (When including an initial mixing step, at most three rounds are required to obtain complete complete diffusion.) Resulting from the analysis is the following theorem:

Theorem 1: *A 3-round differential for the elastic version of AES (with neither an initial nor final key dependent mixing step) accepting input blocks of 128 to 255 bits occurs with probability* $\leq 2^{-30}$.

Proof:

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

For a non-zero differential, the highest single round probability in AES is 2^{-6} and is achieved by a one byte difference in the input. This is due to the S-Box in the SubBytes step of the round. Two single byte differences achieve a probability of 2^{-6}; each of the other 126 possible one byte differences produce a specific output difference with probability 2^{-7} (refer to page 205 in [4]). The only point in the AES round function in which a byte impacts other bytes is the MixColumns step, thus if a certain output of the round is desired that requires more than one specific input byte to the MixColumns step to take on a certain value then each byte can be set with probability at most 2^{-6} and there are at most 128 possible pairs of deltas that produce the result. Thus, in AES the highest probability differential for the round occurs if only one byte differs in the input and the output from the S-Box, and it is one of the cases that occurs with probability 2^{-6}.

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

Terminology:

- Δ indicates the difference between two bit strings.
- Encryption instance refers to the encryption of a particular plaintext. When two plaintexts satisfying a particular Δ are encrypted, the two encryptions will be referred to as two instances.
- Round input and output refers to the entire $128 + y$ bits entering and leaving a round, respectively. The round consists of the AES round function as well as the XOR and swapping of y bits.
- Δ output refers to the difference between the round's output for the two instances of encryption.
- Δ input refers to the difference between the round's input for the two instances of the encryption.
- $a_i \parallel b_i$ denotes the input to round i. a_i is 128 bits and b_i is y bits.
- $\Delta a_i \parallel \Delta b_i$ denotes the difference between two inputs to round i. This also equals the difference between the outputs of round $i-1$.

- $\varDelta a_{out_i}$ denotes the difference between two outputs of the round function in round i.
- 0 denotes a string of bits that are all 0.
- Controlling a byte in $\varDelta$ refers to being able to predict what the $\varDelta$ in the SubBytes outputs for the two instances will be for the particular byte.
- The 128 bits input to the AES round function are treated as a 4 x 4 byte matrix. Rows and columns will refer to the rows and columns of this 4 x 4 matrix.

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

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

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

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

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

a. There is a non-zero $\varDelta$ in the 128 bits of the AES round function output which results in $\varDelta a$ = 0 into the next round. Then $\varDelta$b is non-zero for the next round.

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

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

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

Now we proceed with the proof of the theorem. Given that a one byte non-zero $\varDelta$ in the input to the AES round will result in a 4 byte non-zero $\varDelta$ in the output, the one byte $\varDelta$ may result in a 0 to 4 byte non-zero $\varDelta$ in the y bits that remain out of the next round. Let $\varDelta x$ refer to the 4 non-zero bytes in $\varDelta a_{out_i}$ .Consider what happens in each case of 0 to 4 bytes being swapped into the y bits for round $i + 1$.

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

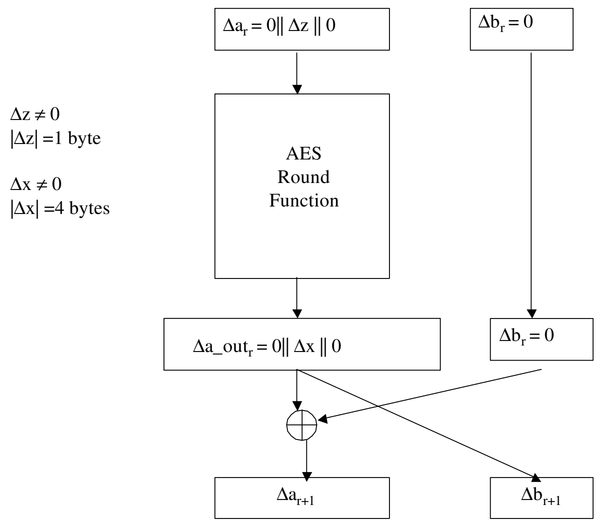

Fig. 2. Elastic AES Round Differential

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

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

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

- Δb_{i+1} contains 1 byte of Δx and is 0 in all other bytes.
- Δa_{i+1} contains Δx. 3 bytes of Δx are not involved in the XOR and swap, and thus remain unchanged when entering the next round. The 4^{th} byte depends on the result of the XOR with Δb_i. However, since Δb_i is 0, this byte will also enter the next round unchanged.
- $\Delta a_{out_{i+1}}$ will be non-zero in all 16 bytes.

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

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

– Δb_{i+1} contains 2 bytes of Δx and is 0 in all other bytes.
– Δa_{i+1} contains Δx.
– $\Delta a_{out_{i+1}}$ will be non-zero in all 16 bytes.

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

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

– Δb_{i+1} contains 3 bytes of Δx and is 0 in all other bytes.
– Δa_{i+1} contains Δx.
– $\Delta a_{out_{i+1}}$ will be non-zero in all 16 bytes.

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

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

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

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

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

Using the upper bound of 2^{-30} for the 3 rounds and a bound of 2^{-6} for one round, the probability of an 11-round differential is $\leq 2^{-96}$ and the probability of a 20-round differential is $\leq 2^{-186}$. We note that the number of rounds in elastic AES ranges from 10 to 20, with 10 rounds required only when y = 0. Block sizes of 129 to 136 bits require

11 rounds and block sizes of 244 to 255 bits require 20 rounds. Compared to the bounds across the total number of rounds determined by the three round bound, the number of plaintexts for each block size is large enough so as not to preclude the possibility of a differential occurring across all rounds with non-negligible probability. However, the bounds across all rounds can be lowered by calculating the probability for four rounds.

4 Performance

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

Table 1. Normalized Number of Blocks Encrypted by Elastic AES in Unit Time (Regular AES = 100)

We implemented the elastic version of AES in C and compared the performance of Elastic AES to that of AES. The number of rounds in elastic AES will range from 10 when $y = 0$ to 20 when $y \geq 116$. The modifications to the encryption and decryption process consist of an additional y bits in the initial whitening and per-round whitening, key dependent mixing steps, and the XORing and swapping of y bits between each round. The elastic version increases the number of operations beyond the 128 bit version of AES due to the swapping step and the two key dependent rotations. However, by avoiding the need to pad the data to two full blocks, the elastic version saves processing time in those cases where padding would normally be required for most or all data blocks.

Both the elastic version and regular 128 bit version of AES were run on several processors in Linux and Windows environments to compare their performance. In the tests, the data to be encrypted was viewed as individual $b + y$ bit blocks. The elastic version of AES encrypted each block individually with no padding. To encrypt the data with regular AES, the $b + y$ bits were padded to $2b$ bits and encrypted as two b bit blocks. When measuring encryption performance (in terms of blocks per second), AES's performance for a single block was based on the time to encrypt 32 bytes, to represent the padding required when using AES for $b + y$ bit blocks. We measured the time to encrypt one million $128+y$ bit blocks (for all values of y from 0 to 127) using the elastic version and two million 128 bit blocks using the original version of AES. When measuring the time for the original version of AES, we exclude the time to pad the $b+y$ bits to $2b$ bits. The performance ratios of the two algorithms were similar for each platform tested. Table 1 summarizes the results from a C implementation compiled with Visual C++ 6.0 on a 1.3 Ghz Pentium 4 processor running Windows XP. and indicates how the elastic version compares to the original version for 8 bit intervals of $b+y$. The number of $b+y$ bit blocks the elastic version can encrypt per second ranges from 190% of the number of $2b$ blocks AES can encrypt per second when $y = 1$ to 100% when $y = 97$. The elastic version of AES's performance decreased gradually to a low of 83% of AES's rate when $y = 127$. Thus, for applications where significant data padding is needed, elastic AES can almost double performance.

5 Conclusions

We described an elastic version of AES. We derived upper bounds for differentials in elastic AES, showing it withstands differential attacks, a fact that validates our design. Due to the nature of AES's round function, the additional operations required for the elastic version add only minor overhead. Our implementation of elastic AES can even exhibit a performance improvement of up to 190% compared with regular AES when applied to data blocks that are only slightly longer than one block. Thus, this first implementation of the elastic block cipher concept shows its potential in applications that offers inputs of variable size.

References

1. FIPS 197 Advanced Encryption Standard (AES), 2001.
2. E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, New York, 1993.
3. D. Cook, M. Yung, and A. Keromytis. Elastic Block Ciphers. Cryptology ePrint Archive, Report 2004/128, 2004. `http://eprint.iacr.org/`.
4. J. Daemon and V. Rijmen. *The Design of Rijndael: AES the Advanced Encryption Standard*. Springer-Verlag, Berlin, 2002.
5. Matsui. Linear Cryptanalysis Method for DES Cipher. In *Proceedings of Advances in Cryptology - Eurocrypt '93, LNCS 0765, Springer-Verlag*, 1993.
6. I. Mironov. (Not So) Random Shuffles of RC4. In *Proceedings of Advances in Cryptology - Crypto 2002, LNCS 2442, Springer-Verlag*, 2002.
7. Rivest, Robshaw, Sidney, and Yin. RC6 Block Cipher. `http://www.rsa.security.com/rsalabs/rc6`, 1998.
8. B. Schneier. *Applied Cryptography*. John Wiley and Sons, New York, 1996.