Tatra Mt. Math. Publ. **37** (2007), 59–74

TATRA MOUNTAINS
Mathematical Publications

ON THE STATE OF THE AES AND CRYPTOGRAPHIC HASHING

Matthew J. B. Robshaw

ABSTRACT. The Advanced Encryption Standard (AES) is an elegant and popular block cipher. In this short summary we provide an overview of the state of cryptanalysis for this important primitive. We will also consider the problems surrounding some widely-used hash functions and we note the potential role of the AES as the basis for a hash function.

1. Introduction

This paper is based on an invited presentation given at MoraviaCrypt'05. At the conference, the opportunity to survey the state of the art in the cryptanalysis of the AES seemed timely, it being nearly five years since the adoption of the block cipher Rijndael [10] as the Advanced Encryption Standard (AES) [35]. During the presentation the author mentioned that when adopting Rijndael as the AES, NIST had decided to restrict the range of permitted block sizes. It was suggested that this may have reduced the usefulness of the AES as the basis for a cryptographic hash function.

During the time since the presentation, there has been an opportunity to consider this issue in more detail. While still essentially true, that the restriction of the block size of Rijndael limits its versatility, there still remain competitive ways to use the AES as the basis of a hash function. This paper, therefore, provides an opportunity to summarise some of these issues. In particular, given the current crisis in cryptographic hash functions, it seems that the AES could be well-suited to providing, at least in the interim, a reasonably standardised solution to the problem of cryptographic hashing.

2000 Mathematics Subject Classification: 94A60.
Keywords: Advanced Encryption Standard, AES, survey, hash functions.
The author wishes to thank the organisers of MoraviaCrypt'05 for their kind invitation to speak at the conference.

The paper is organised in the following way. After a brief description of the AES, there follows a summary of the state of the art. There have now been so many summaries of the AES that this section will be reduced to a basic overview. We will then be able to devote more space to considering how we might use the AES as the basis for a hash function.

2. The Advanced Encryption Standard

On October 2, 2000, the National Institute of Standards and Technology (NIST) announced that Rijndael, a block cipher designed by Joan Daemen and Vincent Rijmen, would be the Advanced Encryption Standard and hence the successor to DES [33]. After more than three years of AES development, this was not a big surprise. Offering both good security and an attractive performance profile, Rijndael was a popular choice.

2.1. The Description of the AES

The AES is now well-known and only the shortest description of the necessary details will be required. Indeed, much of our discussion with regards to hashing with the AES will be concerned with the global characteristics of the algorithm as given by the block size and key size, rather than specific details of the encryption process. But for completeness, we present the essential characteristics here.

Like any block cipher, the AES instantiates an indexed family of permutations. When a key is specified, the bijective transformation of every input string to every output string is determined. If the block cipher has a block length of b bits then the block cipher provides a permutation of all possible 2^b inputs. As we change encryption key a different permutation is instantiated and, in all, the AES will generate at most 2^k permutations where k is the key length.

While we will return to the question of block length a little later, the AES is specified with 128-bit blocks and three key sizes of 128, 192, and 256 bits. During encryption the 128-bit input is viewed, conceptually at least, as a (4×4)-array of bytes and the following operations are used:

`SubBytes`: A byte-wise S-box substitution of all 16 state array elements in parallel. The S-box transformation is carefully chosen to provide good local resistance to differential [3] and linear [27] cryptanalysis.

`ShiftRows`: A row-wise left-rotation of the rows of the state array by (starting from the top) zero, one, two, and three positions respectively.

`MixColumns`: A column-wise mixing of the state array that uses a specially chosen transformation. This transformation helps to amplify the local resistance to differential and linear cryptanalysis that is provided by the S-box, into the global resistance of the block cipher to these attacks.

`AddRoundKey`: The byte-wise addition of a 128-bit round key to all 16 state array elements in parallel. The round keys are derived from the user-supplied encryption key using a reasonably lightweight *key schedule*.

Encryption over r rounds begins with an application of `AddRoundKey`, there are $r-1$ internal rounds [`SubBytes`→`ShiftRows`→`MixColumns`→`AddRoundKey`] and a final round given by [`SubBytes`→`ShiftRows`→`AddRoundKey`]. More details can be found in many sources but particularly FIPS 197 [35] and the Rijndael book [11].

The number of rounds r used during encryption varies depending on the key size. In fact, if we look at the original specifications of Rijndael we note that these allow the block size to vary as well as the key. In the most general sense, therefore, there are two parameters determining the number of rounds; the block size b and the key size k. The choices for certain versions of Rijndael are given in Table 1 with values for the AES (in bold) taken from the first column for $b = 128$.

	Block size b				
Key size k	**128**	160	192	224	256
128	**10**	11	12	13	14
160	11	11	12	13	14
192	**12**	12	12	13	14
224	13	13	13	13	14
256	**14**	14	14	14	14

TABLE 1. The proposed number of rounds for different AES (in bold) and Rijndael block/key-size options [10].

2.2. The Security of the AES

There are several overviews and summaries of the security of the AES. A brief overview is provided in [16] while *The State of the Art of AES Cryptanalysis* [17] which is published as part of the *ECRYPT Network of Excellence* is exhaustive. In general terms, attacks on the AES are typically summarised in the following way.

Statistical cryptanalysis

This type of cryptanalysis relies on the cryptanalyst being able to analyse the inputs and outputs to repeated interactions with the block cipher. The most important classes of attacks here are differential [3] and linear [27] cryptanalysis along with some increasingly sophisticated extensions. Due to the design principles of the AES, the current forms of these attacks are particularly ineffective against the AES.

Structural cryptanalysis

Attacks in this group rely on the specific design of the AES and, in particular, on the very structured way the operations of `SubBytes`, `ShiftRows`, and `MixColumns` interact. Perhaps unsurprisingly, attacks specifically aimed at the AES are, in general terms, more effective than the more general statistical attacks. The most important attacks are the *square attack* [12] and the *bottleneck attack* [19] along with some extensions [18]. While they work very well over a limited number of rounds they tend to break down after around 60% of the encryption process. Sometimes the headline figures to these attacks can be somewhat misleading. For instance, extensions to the basic attacks might gain an extra round by assuming that a larger encryption key is used. This allows the cryptanalyst a greater work effort and a greater margin within which to claim a "break". However, by moving to a longer key, the cryptanalyst is faced with more encryption rounds (as shown in Table 1). It is therefore instructive to summarise the effectiveness of the best structural attacks as the percentage of the cipher that can be successfully attacked. The *reach* of the best structural attacks on the AES are summarised in Table 2. The versions that are "broken" are in the literature are indicated along with the proportion of the AES the attack represents for different key lengths. By considering Table 2 we observe that, in one way, we are losing ground with the more sophisticated structural attacks! No attack has yet managed to cover more than 60% of the appropriate encryption process. That said, these sophisticated attacks are far from irrelevant and they reveal much about the inner workings of the cipher.

Algebraic cryptanalysis

Of all prominent block ciphers the AES is notable for its clean design and mathematically simple operations. Together these provide an ideal environment in which to attempt to represent the process of encryption as a sequence of equations that might be solved by algebraic means. This approach has been termed algebraic cryptanalysis. However with our current knowledge it is unknown how

	Number of rounds being attacked		
Key size k	6	7	8
128	60%		
192	50%	58%	
256	43%	50%	57%

TABLE 2. The *reach* of structural attacks on the AES. While basic structural attacks can be extended when the cryptanalyst has the luxury of longer keys, the proportion of the cipher covered by the attack is reduced since extra encryption rounds are required.

such an attack might be carried out. Two papers have described how the process of AES encryption might be expressed by a sparse system of overdefined equations. One set of equations is given over GF(2) [9] and the other over GF(2^8) [30]. A discussion of the derivation of these equations along with their relative merits would take us out of the scope of this paper. Suffice it to say that solving these equations with only a handful of plaintext-ciphertext pairs would immediately give the encryption key.

Since the initial proposal for the algebraic cryptanalysis of the AES [9] there has been some progress in this area. But this progress has only served to moderate early claims. The method of solution known as XSL, which was proposed by Courtois and Pieprzyk [9], was viewed somewhat sceptically even at its publication. Now, recent work by Cid and Leurent [6] has concluded that the method as described cannot succeed. A more basic method of solution, using Gröbner basis techniques, is always available. However the algorithms to hand are exceptionally inefficient [7]. Tests on small-scale versions of the AES [8] have demonstrated the very limited progress such techniques currently make.

Thus we find ourselves in an interesting position. A method for solving the system of AES equations would be devastating. Yet, there is no indication of where such an efficient solution method might be found, or whether one might even exist. Consequently algebraic attacks can only be described as speculative.

Side-channel cryptanalysis

The importance of side-channel cryptanalysis has been recognised for years [24, 25]. Issues such as the time required for certain operations on different processors and the power consumed during some cryptographic operation are well-known. Indeed, these were an important issue during the AES process and the claimed resistance of Rijndael to side-channel cryptanalysis was one of the points given in favour of its selection [34].

However two bodies of work have pointed out potential difficulties when implementing the AES [2, 40]. These depend on the fact that implementations of the AES that deliver good performance do so by using look-up tables. The use of look-up tables can lead to dependencies in the time required to access arrays in memory. Such dependencies can reveal sensitive state information and this can be exploited remotely or by independent processes that access the same shared resources.

The attacks that result are dramatic and powerful. While it is unclear that any cipher would be immune to such concerns, it seems that implementations of the AES can show a particular vulnerability. A range of countermeasures have been considered [2, 40] but none are unaccompanied by caution or caveat.

2.3. The Status of the AES

Apart from the issue of side-channel cryptanalysis which seems to pose particularly tricky issues for implementations of the AES and other block ciphers, the best attack that does not attack the implementation remains brute-force key search. When considering brute-force attacks it is worth keeping in mind the spectrum of time-memory trade-offs that might be available to the cryptanalyst. An analysis of the different aspects to this problem is given by Biryukov *et al.* [4] and one trade-off highlighted by their work applies to all block ciphers with 128-bit keys and yields the following: With a pre-computation effort of 2^{96} operations and 2^{56} words of memory, one 128-bit key can be recovered in 2^{80} operations from a set of 2^{32} encryptions that were formed using distinct keys.

However such attacks represent an improved understanding of time-memory trade-offs rather than any improvements in the cryptanalysis of the cipher. In fact, such attacks represent the best that any cipher designer can hope for. Brute-force attacks take no account of the specific structure of the cipher and therefore they always apply. However, if they remain the best attack available to the cryptanalyst, then specific choices made by the designers have not introduced any exploitable weakness.

For those seeking more information about the AES and/or its implementation, the formal description of the AES is given in FIPS 197 [35], the accompanying modes of operation are specified in SP 800-38A [36], and the official status and co-existence of DES, the AES, and Triple-DES are described in SP 800-67 [37]. The *AES Lounge* at *www.iaik.tu-graz.ac.at/research/krypto/AES/* provides many links to the specification, analysis, and implementation of the AES.

3. Cryptographic Hash Functions

Hash functions are useful cryptographic primitives with interesting properties. They have a particularly prominent role within the deployment of public-key infrastructures since they are often used in public-key signature schemes. But they also feature in a wide range of protocols [15] and cryptographic constructions. An n-bit cryptographic hash function $h(\cdot)$ takes an arbitrary-length input and generates a fixed-length output of n bits. There are three main properties of such functions;

Pre-image resistance: Given an output y it is hard to find any input x such that $h(x) = y$.

Second pre-image resistance: Given an input-output pair x and y with $h(x) = y$, it is hard to find an input x' with $x' \neq x$ such that $h(x') = y$.

Collision resistance: It is hard to find any two inputs x and x' such that $h(x) = h(x')$.

For a good hash function, the minimum work effort to defeat these three properties is expected to be 2^n, 2^n, and $2^{n/2}$ hash function operations, respectively.

3.1. Current Hash Function Solutions

Over the years there have been many different hash function proposals. Currently, most deployed hash functions are either dedicated designs or they are based around the use of a block cipher.

The most common hash functions in deployment today are likely to be MD5 [43] and SHA-1 [38]. However recent cryptanalytic advances [45, 46] have shown weaknesses that allow collisions to be computed for these hash functions much faster than by brute force. A second family of hash functions [38] has been standardised by NIST. But these are closely related to SHA-1 and, like the vast majority of dedicated hash functions, they are built around the same design principles that go back to MD4 [42]. Thus confidence in their construction is somewhat undermined by the recent cryptanalytic work on MD5 and SHA-1. Other hash function constructions that are widely used are built around the use of a block cipher such as DES [33]. Two prominent constructions are MDC-2 and MDC-4 (see for example [28]).

A common design approach for hash functions over the past decades has been to adopt what is termed the Merkle-Damgård construction [14, 29]. Again, details would take us out of the scope of this survey article, but such a construction requires the definition of a *compression function*. At iteration i of the compression function there are two inputs, the *chaining variable* v_i and some *message input* m_i. There is one output, namely, the value of the chaining variable v_{i+1} for the next iteration. By starting from a fixed value to the chaining variable and processing the message serially, the hash output for a message of any length can be computed.

Thus, the design of a secure hash function is often recast as the design of a good compression function. For the dedicated design the architecture and contents of the compression function are completely open. When using a block cipher as the basis for a hash function (and therefore a compression function) we are restricted by the form of the block cipher. It has been shown that there are a limited number of ways to build the compression function for a secure b-bit hash function out of a b-bit block cipher [5, 28, 41]. The most significant constructions are typically referred to as *Davies-Meyer*, *Matyas-Meyer-Oseas*, and *Miyaguchi-Preneel*. These are illustrated in Figure 1. One property of each of these constructions when built around a b-bit block cipher, is that a brute-force collision search would require a time proportional to $2^{b/2}$. Thus, if we were using DES as the underlying block cipher, the time for collision search would be 2^{32} operations. For this reason, the double block-length constructions of MDC-2 and MDC-4 were introduced in the 1970's. Today, when we routinely aim at a security level of 2^{128} and our standardised block cipher (the AES) has a block length of 128 bits, such constructions may still have some use.

4. Hashing With The AES

Given current misgivings around the use of a dedicated hash function, it is worth considering how we might build a hash function out of the AES. This issue was briefly mentioned by the Rijndael designers in [11]. The first thought might be to use the single block length construction of Figure 1. Since the block size of the AES is fixed to 128 bits, the security offered against brute-force attacks would be at most 2^{64} operations. This is typically viewed as being insufficient for many deployments and so we will explore the implications of two approaches to dealing with this.

(1) We might use Rijndael with 256-bit blocks in a single-block length construction. We will denote a version of Rijndael with 256-bit blocks and key size k by Rijndael_k^{256}.

(2) We might use the AES in a double-block length construction identical to MDC-2. We will denote versions of the AES with k-bit keys by AES_k.

Of course, this is a survey article and it is not the first time that such issues have been raised. Indeed, the use of the AES as the basis for a hash function is

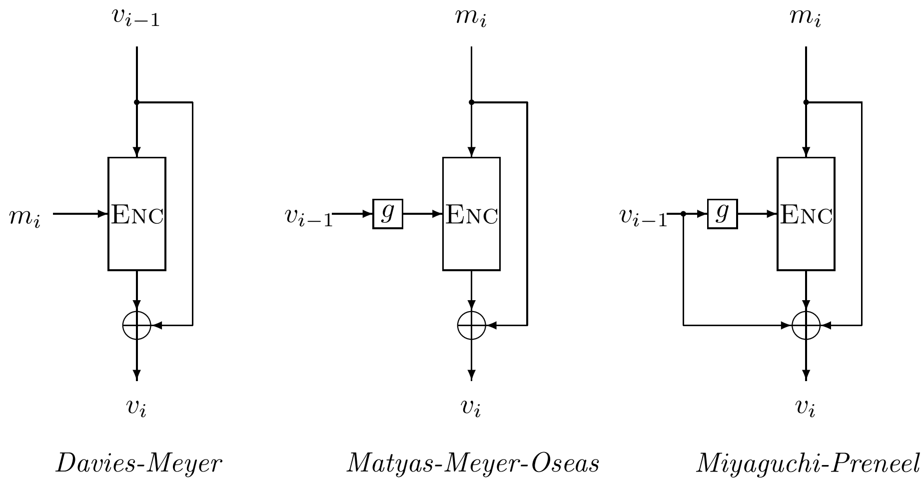

FIGURE 1. Three compression functions from a single iteration of a block cipher. The input to the top of ENC is the message while the input from the left is the key. A function g might be required for technical reasons.

an active area of research with [23, 26, 31, 32] providing just a sample of some ongoing work.

4.1. A single block-length construction using Rijndael

It is worth considering Figure 1 in a little detail before proceeding. In each construction we note the two inputs; the *chaining variable* and part of the *message* or input being hashed. The relative sizes of the block length and the key length will have an impact on the performance of these constructions.

In the case of *Davies-Meyer* the message block is used as the key to encrypt the current value of the chaining variable. Thus, at each invocation of the block cipher, k bits of the message can be hashed. For the best performance it would therefore be advantageous to take $k = 256$. For the remaining single-block length constructions the sizes of the message input and the chaining variable must be essentially the same. Thus for all three constructions we would probably use $\text{Rijndael}^{256}_{256}$.

At this point we might refer to recent results on finding second pre-images for an iterated hash function [22]. These attacks rely on a range of observations and to protect against these attacks we might use some additional mechanism [22, 44].

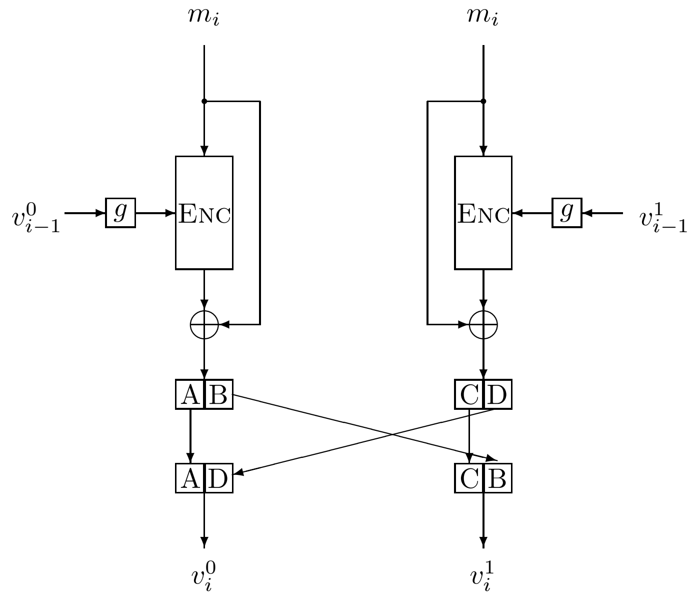

FIGURE 2. The compression function for the MDC-2 hash function built from two iterations of the underlying block cipher. At iteration i the chaining variable is given by $v^0_i || v^1_i$.

One structural property that has been identified as being immediately useful in this work is that of fixed points for the compression function. In particular it can be shown that finding fixed points for *Davies-Meyer* is trivial while finding fixed points in other block cipher based constructions may not be so easy for a good block cipher. While fixed points are not the only issue highlighted in [22], these may provide a reason for prefering the *Matyas-Meyer-Oseas* or *Miyaguchi-Preneel* constructions. In what follows we will consider the *Matyas-Meyer-Oseas* construction since this also forms the basis for MDC-2 and allows an easier comparison.

4.2. An AES-based MDC-2 construction

The MDC-2 construction is illustrated in Figure 2. Again, there are two inputs to each iteration of the compression function, the chaining variable and the message being hashed. The obvious use of the MDC-2 construction is to substitute AES_{128} for DES. The result would be a compression function that hashes 128 bits of message at each iteration, at the cost of two instantiations of AES_{128}.

The function g merits discussion. First, it is well-known that there are *weak keys* and *semi-weak keys* for DES (see for example [28]). These can lead to structural weaknesses in the hash function. Second, if we have the same key used in both the left and right strands of encryption, then the output from both computations would be the same. This property would then perpetuate throughout the rest of the hash computation. To prevent both issues, the function g specified within MDC-2 ensures that the two keys used in the compression function are different and free from structural defects. Given that there is no equivalent concept of weak keys for the AES, we would perhaps use an equivalent function g to make sure that the two keys in the compression function are distinct. Despite these precautions, however, the MDC-2 construction is not without its structural defects, and these have some impact on the security delivered [28, 41]. Nevertheless, it is a construction that is used and at least provides a reasonable basis for comparison.

4.3. Performance Comparison

In this section we provide a rough comparison of the performance of these two options. We will concentrate on software performance and we will use the performance figures derived during the NESSIE project [39]. These provide the range of figures we require under a suitably consistent testing framework. While the performance of encryption and key setup for different variants of AES and Rijndael are measured on different platforms we will use a more generic estimate. In particular, we will denote the time for one block of encryption with AES_{128} as t and this allows us to make the following rough comparison.

First we observe that $\text{Rijndael}_{256}^{256}$ has 14 rounds instead of 10 as is the case for AES_{128}. Further we note that the state array for $\text{Rijndael}_{256}^{256}$ is twice the size as that for AES_{128}. We can therefore estimate the relative encryption performance for the two ciphers in Table 3.

For the key setup time, we turn to [39] and observe that key setup for AES_{128} for some platforms can require up to $\frac{3}{2}$ the cost of encrypting a single block, though on other platforms they can be comparable. We will assume the worst case. Meanwhile key setup for $\text{Rijndael}_{256}^{256}$ appears to require two to three times the effort required for encryption, though it is unclear from the documentation how well-optimised the code for $\text{Rijndael}_{256}^{256}$ might be [39]. We will therefore estimate that key setup requires around $\frac{5}{2}$ times the cost of encryption.

This should be sufficient to estimate the relative performance of the two constructions. For *Matyas-Meyer-Oseas* each iteration of the compression function requires one encryption and one key setup. Since MDC-2 requires two parallel instantiations of *Matyas-Meyer-Oseas*, plus some minor overheads, we will assume that MDC-2 requires two encryptions and two key setups. The net result

	Encryption	*Key setup*		*Hash Function Perf.*	
AES_{128}	t	$3t/2$		MDC-2	$5t$
$Rijndael^{256}_{256}$	$\approx 3t$	$\approx 7t$		MMO	$\approx 10t$

TABLE 3. Estimates of comparative performance for AES_{128} and $Rijndael^{256}_{256}$ along with estimates for compression functions in the style of *Matyas-Meyer-Oseas* and *MDC-2*.

of these figures has been summarised in Table 3. At first it might appear that the MDC-2 construction is twice the speed of *Matyas-Meyer-Oseas*. However, at each iteration of the compression function the *Matyas-Meyer-Oseas* construction hashes 256 bits of message, which is exactly double that hashed in the MDC-2 construction. Somewhat surprisingly, given the very different constructions, the net hashing rate for both proposals will probably be similar.

Thus, unless substantial optimisations to $Rijndael^{256}_{256}$ and the associated key setup time can be delivered, the *Matyas-Meyer-Oseas* construction will be roughly comparable in performance to the MDC-2 construction. However, since the MDC-2 construction is already standardised and deployed, and since it can employ existing AES implementations as a component without any change, there are strong points in favour of the MDC-2 approach. Further, if we were to consider suitability to hardware, then there is also considerable potential for parallelisation within the MDC-2 variant.

But do AES-based alternatives provide attractive alternatives to other recent hash function proposals? It is likely that this will be discussed in the coming years, but one prominent new proposal is Whirlpool [1] which has been proposed for use within ISO/IEC 10118-3:2004. To compare the performance between AES-based MDC-2 and Whirlpool we can use NESSIE figures [39] or those offered in Wei Dai's software performance comparison [13]. While there is considerable dependence on the particular platform being used, the the encryption rate for AES_{128} is roughly five times the hashing rate of Whirlpool. Thus, when considering the estimated factor five performance overhead for an AES-based MDC-2, the construction still appears to be attractive in performance terms and would be based around a primitive that is widely deployed and increasingly trusted. Looking beyond the legacy construction of MDC-2, several strands of research [20, 21, 31, 32] already explore different ways of using a block cipher like the AES as the basis for a hash function. Thus it seems that the case for at least considering the potential of AES-based hashing is quite compelling.

5. Conclusion

The AES is a very elegant cipher that continues to resist all known analytical attacks with considerable room to spare. Given this success, and given the disillusion surrounding dedicated hash functions, it seems to be a particularly interesting time to consider AES-based solutions to the problem of cryptographic hashing.

Acknowlegement

The author wishes to thank the organizers of MoraviaCrypt'05 for their kind invitation to speak at the conference. Since writing this survey the state of hash functions has changed considerably. However an AES-based solution remains an intriguing possibility.

REFERENCES

[1] BARRETO, P. S. L. B.—RIJMEN, V.: *The Whirlpool hash function.* Available via `www.paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html`.

[2] BERNSTEIN, D.: *Cache-timing attacks on AES.* Draft. April 14, 2005. Available via `www.cr.yp.to/antiforgery/cachetiming-20050414.pdf`.

[3] BIHAM, E.—SHAMIR, A.: *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag, New York, 1993.

[4] BIRYUKOV, A.—MUKHOPADHYAY, S.—SARKAR, P.: *Improved time-memory trade-offs with multiple data.* In: Proceedings of SAC 2005 (B. Preneel, ed.), Lect. Notes Comput. Sci., Springer-Verlag, Berlin. (to appear).

[5] BLACK, J.—ROGAWAY, P.—SHRIMPTON, T.: *Black-box analysis of the block-cipher-based hash-function constructions from PGV.* In: Advances in Cryptology–CRYPTO '02 (M. Yung, ed.) Lect. Notes Comput. Sci., Vol. 2442, Springer-Verlag, Berlin, 2002, pp. 320–335.

[6] CID, C.—LEURENT, G.: *An analysis of the XSL algorithm.* In: Advances in Cryptology–ASIACRYPT '05 (B. Roy, ed.), Lect. Notes Comput. Sci., Springer-Verlag, Berlin. (to appear).

[7] CID, C.—MURPHY, S.—ROBSHAW, M. J. B.: *Computational and algebraic aspects of the Advanced Encryption Standard.* In: Proceedings of the 7th International Workshop on Computer Algebra in Scientific Computing, CASC '2004, St. Petersburg, 2004, pp. 93–103.

[8] CID, C.—MURPHY, S.—ROBSHAW, M. J. B.: *Small scale variants of the AES.* In: Proceedings of FSE 2005 (H. Gilbert and H. Handschuh, eds.), Lect. Notes Comput. Sci., Vol. 357, Springer-Verlag, Berlin, 2005, pp. 145–162.

[9] COURTOIS, N.—PIEPRZYK, J.: *Cryptanalysis of block ciphers with overdefined systems of equations*. In: Advances in Cryptology–ASIACRYPT '02 (Y. Zheng, ed.), Lect. Notes Comput. Sci., Vol. 2501, Springer-Verlag, Berlin, 2002, pp. 267–287.
[10] DAEMEN, J.—RIJMEN, V.: Rijndael. Version 2.0, 1999. Via `www.csrc.nist.gov`.
[11] DAEMEN, J.—RIJMEN, V.: The Design of Rijndael. Springer-Verlag, Berlin, 2000.
[12] DAEMEN, J.—KNUDSEN, L.—RIJMEN, V.: *The block cipher square*. In: Proceedings of FSE '97 (E. Biham, ed.), Lect. Notes Comput. Sci., Vol. 1267, Springer-Verlag, Berlin, 1997, pp. 149–165.
[13] DAI, W.: *Crypto^{++}Benchmarks*. Via `www.eskimo.com/~weidai/benchmarks.html`.
[14] DAMGÅRD, I.: *A design principle for hash functions*. In: Advances in Cryptology–CRYPTO '89 (G. Brassard, ed.), Lect. Notes Comput. Sci., Vol. 435, Springer-Verlag, Berlin, 1990, pp. 416–427.
[15] DIERKS, C.—ALLEN, T.: RFC: 2246. The TLS Protocol. January, 1999.
[16] DOBBERTIN, H.—KNUDSEN, L. R.—ROBSHAW, M. J. B.: *The cryptanalysis of the AES*. In: Proceedings of AES 4 (H. Dobbertin, V. Rijmen, and A. Sowa, eds.), Lect. Notes Comput. Sci., Vol. 3373, Springer-Verlag, Berlin, 2005, pp. 1–10.
[17] ECRYPT Network of Excellence. D.STVL.4: AES Security Report. November, 2005. Available via `www.ecrypt.eu.org`.
[18] FERGUSON, N.—KELSEY, J.—LUCKS, S.—SCHNEIER, B.—STAY, M.—WAGNER, D.—WHITING, D.: *Improved cryptanalysis of Rijndael*. In: Proceedings of FSE 2000, (B. Schneier, ed.), Lect. Notes Comput. Sci., Vol. 1978, Springer-Verlag, Berlin, 2001, pp. 213–230.
[19] GILBERT, H.—MINIER, M.: *A collision attack on seven rounds of Rijndael*. In: Proceedings of the 3rd Advanced Encryption Standard Conference, NIST, eds.), April, 2000. Available via www.csrc.nist.gov, pp. 230–241.
[20] HATTORI, M.—HIROSE, S.—YOSHIDA, S.: *Analysis of double block length hash functions*. In: Proceedings of the 9th IMA International Conference on Coding and Cryptography (K. Paterson, ed.), Lect. Notes Comput. Sci., Vol. 2898, Springer-Verlag, Berlin, 2003, pp. 290–302.
[21] HIROSE, S.: *Provably secure double block length hash functions in a black box model*. In: Proceedings of the 7th International Conference on Information Security and Cryptology (ICISC 2004)(C. Park and S. Chee, eds.), Lect. Notes Comput. Sci., Vol. 3506, Springer-Verlag, Berlin, 2005, pp. 330–342.
[22] KELSEY, J.—SCHNEIER, B.: *Second preimages on n-bit hash functions for much less than 2^n work*. In: Advances in Cryptology–EUROCRYPT '05 (R. Cramer, ed.), Lect. Notes Comput. Sci., Vol. 3494, Springer-Verlag, Berlin, 2005, pp. 474–490.
[23] KNUDSEN, L. R.—MULLER, F.: *Some attacks against a double length hash proposal*. In: Advances in Cryptology–ASIACRYPT '05 (B. Roy, ed.), Lect. Notes Comput. Sci., Vol. 3788, Springer-Verlag, Berlin, 2005, pp. 462–473.
[24] KOCHER, P. C.: *Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*. In: Advances in Cryptology–CRYPTO '96 (N. Koblitz, ed.), Lect. Notes Comput. Sci., Vol. 1109, Springer-Verlag, Berlin, 1996, pp. 104–113.
[25] KOCHER, P. C.—JAFFE, J.—JUN, B.: *Differential power analysis*. In: Advances in Cryptology–CRYPTO '99 (M. Weiner, ed.), Lect. Notes Comput. Sci., Vol. 1666, Springer-Verlag, Berlin, 1999, pp. 388–397.
[26] LUCKS, S.: *Design principles for iterated hash functions* Draft. September 29, 2004. Available via `www.eprint.iacr.org/2004/253`.

[27] MATSUI, M.: *First experimental cryptanalysis of the Data Encryption Standard*. In: Advances in Cryptology–CRYPTO '94 (Y. Desmedt, ed.), Lect. Notes Comput. Sci., Vol. 839, Springer-Verlag, Berlin, 1994, pp. 1–11.

[28] MENEZES, A.—VAN OORSCHOT, P.—VANSTONE, S.: *The Handbook of Applied Cryptography*. CRC Press, 1996.

[29] MERKLE, R.: *One way hash functions and DES*. In: Advances in Cryptology–CRYPTO '89 (G. Brassard, ed.), Lect. Notes Comput. Sci., Vol. 435, Springer-Verlag, Berlin, 1989, pp. 428–446.

[30] MURPHY, S.—ROBSHAW, M. J. B.: *Essential algebraic structure within the AES*. In: Advances in Cryptology–CRYPTO '02 (M. Yung, ed.), Lect. Notes Comput. Sci., Vol. 2442, Springer-Verlag, Berlin, 2002, pp. 1–16.

[31] NANDI, M.: *Designs of efficient secure large hash values*. Draft. April 20, 2004. Available via `www.eprint.iacr.org/2004/296`.

[32] NANDI, M.—LEE, W.—SAKURAI, K.—LEE, S.: *Security analysis of a 2/3-rate double length compression function in the black-box model*. In: Proceedings of FSE 2005 (H. Gilbert and H. Handschuh, eds.), Lect. Notes Comput. Sci., Vol. 3557, Springer--Verlag, Berlin, 2005, pp. 243–254.

[33] National Institute of Standards and Technology. FIPS 46-3: Data Encryption Standard, October 1999. Available via `www.csrc.nist.gov`.

[34] National Institute of Standards and Technology. Report on the development of the Advanced Encryption Standard (AES). October 2, 2000. Available via `www.csrc.nist.gov`.

[35] National Institute of Standards and Technology. FIPS 197: Advanced Encryption Standard, November 2001. Available via `www.csrc.nist.gov`.

[36] National Institute of Standards and Technology. Recommendation for Block Cipher Modes of Operation. December, 2001. Available via `www.csrc.nist.gov`.

[37] National Institute of Standards and Technology. Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher. May, 2004. Available via `www.csrc.nist.gov`.

[38] National Institute of Standards and Technology. FIPS 180-2: Secure Hash Standard, August 2002. Available via `www.csrc.nist.gov`.

[39] NESSIE Consortium. Final report of European project IST-1999-12324: New European Schemes for Signatures, Integrity, and Encryption. April 19, 2004. Available via `www.cosic.esat.kuleuven.ac.be/nessie/`.

[40] OSVIK, D. A.—SHAMIR, A.—TROMER, E.: *Cache attacks and countermeasures: The case of the AES*, Draft. October 8, 2005.
Available via `www.wisdom.weizmann.ac.il/~tromer/papers/cache.pdf`.

[41] PRENEEL, B.: *Cryptographic Hash Functions*, Ph.D. Thesis. 1993.

[42] RIVEST, R. L.: *The MD4 Message-Digest Algorithm*. In: Advances in Cryptology–CRYPTO '90 (A. Menezes and S. Vanstone, eds.), Lect. Notes Comput. Sci., Vol. 537, Springer-Verlag, Berlin, 1991, pp. 303–311.

[43] RIVEST, R. L.: *RFC 1321: The MD5 Message-Digest Algorithm*, April 1992. Available via `www.ietf.org/rfc/rfc1321.txt`.

[44] RIVEST, R. L.: *Abelian square-free dithering for iterated hash functions*. Preprint. Available via `www.theory.lcs.mit.edu/~rivest/`.

[45] WANG, X.—YU, H.—YIN, Y. L.: *Efficient collision search attacks on SHA-0*. In: Advances in Cryptology–CRYPTO '05 (V. Shoup, ed.), Lect. Notes Comput. Sci., Vol. 3621, Springer-Verlag, Berlin, 2005, pp. 1–16.

[46] WANG, X.—YIN, Y. L.—YU, H.: *Finding collisions in the full SHA-1.* In: Advances in Cryptology–CRYPTO '05 (V. Shoup, ed.), Lect. Notes Comput. Sci., Vol. 3621, Springer--Verlag, Berlin, 2005, pp. 17–36.

Received October 28, 2005

France Télécom, R&D Division
38–40, rue du Général Leclerc
92794 Issy Moulineaux, Cedex 9
FRANCE
E-mail: matt.robshaw@francetelecom.com