RC6 and the AES

M.J.B. Robshaw

16d Stowe Rd, London, W12 8BN, UK.
mrobshaw@supanet.com

January 9, 2001

1 Introduction

On October 2, 2000 the National Institute of Standards and Technology (NIST) announced that Rijndael [4] had been chosen to become the Advanced Encryption Standard (AES) [12].

Rijndael is a very elegant and novel cipher. Relatively simple, with a very rich mathematical structure, Rijndael has a versatile performance characteristic. The designers Daemen and Rijmen are to be unreservedly congratulated on a wonderful achievement!

In the final round another four algorithms were under consideration to become the AES. One of these was RC6 [14] which had been submitted by RSA Laboratories. NIST made the following comments about the five finalists:

> *Each of the finalist algorithms appears to offer adequate security, and each offers a considerable number of advantages. Any of the finalists could serve admirably as the AES. However, each algorithm also has one or more areas where it does not fare quite as well as some other algorithm; none of the finalists is outstandingly superior to the rest.*
> – Nechvatal, Barker, Bassham, Burr, Dworkin, Foti, Roback [12]

Now we are at the close of the AES process, it makes an interesting final exercise to compare RC6 and the AES and to review some of the design decisions that were made during the development of RC6. As NIST observed, no single AES finalist provided the best fit in all circumstances. Rijndael is superior to RC6 in several important application areas. But there are also places where the two algorithms appear to be equally suitable and there are niche application areas where RC6[1] might offer an interesting alternative to Rijndael.

The observations in this note have already appeared in technical submissions to NIST as part of the AES process. They can be found in the notes *RC6 as the AES* [15] and *The Case for RC6 as the AES* [16] which are archived at www.rsalabs.com/rc6/ .

[1] Licensing terms for RC6 can be obtained by contacting RSA Security directly.

2 RC6 as an AES Candidate

RC6 was designed by Rivest, Robshaw, Sidney, and Yin and submitted by RSA Laboratories as a candidate for the AES. Adhering closely to the philosophy that an algorithm should be simple so that it can be analyzed, RC6 was based around RC5 [13] which was published in December 1994.

Since the time for security assessment of the AES was anticipated to be short, an early design decision was to build as closely as possible on RC5 and to try and re-use much of the security analysis and independent cryptanalysis that had already taken place over the intervening four years. The simplicity of RC5 made it an attractive object for research.

During the design of RC6, the design team considered dozens of alternatives and subjected them to intense cryptanalysis. We finally decided that RC6 best captured our three goals of high security, exceptional simplicity, and good performance.

The most significant working assumptions we made were the following:

1. Security would be the over-riding requisite for a credible AES candidate.
2. Performance on older 8- and 16-bit environments would probably be less important over the lifetime of the AES than current 32- and future 64-bit environments. Today most 8-bit processors are to be found in cheaper smart cards. Many of the more versatile smart cards use more advanced processors. It was our belief that the fraction of smart cards with 8-bit processors would rapidly decline with the inevitable drop in the cost of more advanced cards. Furthermore, we felt that it was important to recognize the range of applications that might require encryption on a smart card. Such applications would typically not be performance-critical. (Certainly a smart card would not be used for the bulk of encryption of data!) So, in the interests of compromise, we aimed the performance of RC6 at 32- and 64-bit environments in the belief that over the 20-30 year life-time of the AES this would be the most important environment of use.
3. Software implementations would probably be more important than hardware implementations. For most applications, we believed that an implementation of RC6 in software would likely be the best choice. We felt that RC6's primitive operations were typically well-supported on modern microprocessors and that this would increasingly become the norm[2]. Thus one would benefit from the exceptional effort that has gone into the design of such processors. Furthermore, it would allow one to easily ride the familiar technology curve that results in Moore's Law (a factor of two improvement of performance every 18 months). However, once again in the interests of compromise, we recognized

[2] The very poor support for multiplication [21] in the IA-64 architecture was a great surprise to many observers (especially ourselves).

that it could sometimes be desirable to have a custom integrated circuit. Because RC6 uses familiar primitive operations, we felt that one could take advantage of existing expertise in designing circuit modules for implementing these primitives [14]. While one could implement RC6 using standard gate-array technology, we felt that this would not utilize the tremendous design efforts that have gone into efficient multiplication circuitry. Indeed, we recognized [14] that a gate-array implementation could perform less well than a processor-based implementation. But we felt that this wasn't an atypical situation, and one could easily design circuits incorporating the best available multiplication circuitry as submodules.

The most significant design decisions we made were the following:

1. The design had to remain simple.
2. We would use a 32-bit multiplication operation within RC6. This was perhaps the most significant change in moving from RC5 to RC6 and resulted from a careful evaluation of the additional security that might be offered by the multiplication operation. This operation is well-supported on many modern 32-bit processors and we felt that it was likely to be increasingly well-supported in the future.
3. We would re-use the key schedule that was used in RC5. This already had an excellent track record as a rock-solid key schedule and had been studied widely for more than four years. We expected this to be a significant factor during the AES process.

How well did we fare? RC6 is a very simple cipher with excellent security credentials. In many situations and environments its performance is at least equal to, and in several places better than, the other AES finalists. However to better appreciate the successes of RC6, it is important to recognize some of the hesitations that were voiced by some commentators.

1. On 8-bit processors, RC6 does not generally perform as well as some of the other AES finalists.
2. The ultra-secure key schedule had an impact on the key agility of the cipher.
3. The key schedule and the use of multiplication added some complexity to hardware implementations and impacted hardware performance [19].
4. The performance profile of RC6 on 64-bit machines was volatile. On some processors, RC6 easily out-performed all other finalists [2]. Yet, on Intel's IA-64 architecture, surprisingly poor support for multiplication meant that the performance of RC6 was hindered [21].

In line with our design decisions, we were not unduly concerned about cases **1** and **3** since we held them to be of relatively minor significance. With regards to cases **2** and **4** we provided technical submissions [16] to NIST that went some considerable way to addressing any legitimate concerns that had arisen. Of course, the relative importance of these factors is subjective and others may place a different emphasis on these attributes!

3 The Security of RC6 and Rijndael

First we should consider the most important question of how the security of the two algorithms compare.

RC6 was the simplest of the AES finalists. Rijndael was a close second, and as a result both algorithms received considerable cryptanalytic scrutiny. NIST adjudged both algorithms to have "an adequate security margin" [12]. Thus, with the current state of knowledge of both algorithms, there is no need to feel that a choice between Rijndael and RC6 is a choice in the level of security offered.

4 The Performance of RC6 and Rijndael

Here we separate the different performance aspects of RC6 and Rijndael into three sections.

4.1 Where RC6 is potentially inferior to Rijndael

We have already listed some of the areas where Rijndael would typically be expected to out-perform RC6. Generally these include 8-bit environments and hardware implementations. It should be noted that while Rijndael might out-perform RC6 in these environments, the performance of RC6 is not necessarily poor. For instance, with regards to an ASIC (Application-Specific Integrated Circuit) implementation, speeds of over 2 Gbits/second were reported for some hardware configurations [19].

4.2 Where RC6 and Rijndael are roughly comparable

DSPs (Digital Signal Processors). There seems to be little to choose between the performance of RC6 and Rijndael on these devices. However one area of advantage for RC6 seems to be the small memory requirements [22].

FPGAs (Field Programmable Gate Arrays). Here RC6 and Rijndael again seem to offer roughly equivalent performance. The evidence is a bit mixed, since during the AES process two papers on FPGA implementations [7,8] seemed to offer contradictory evidence while a third [18] provided detailed and informative observations though they weren't the results of actual implementations. RC6 generally offers good performance, but it does so with exceptionally compact implementation. According to Elbirt et al. [7] Rijndael out-performs RC6 when used in a feedback mode. But when non-feedback modes are used RC6 out-performs Rijndael.

4.3 Where RC6 is potentially superior to Rijndael

Pentium II, Pentium Pro, Pentium III, and PowerPC. In 'C' implementations, and hand-optimized assembly on these processors, RC6 generally out-performs Rijndael. At times the performance figures are roughly comparable, but the difference in performance can sometimes amount to a factor of two or more [12]. When we look to future 64-bit architectures the situation becomes muddled. On some processors RC6 appears to be penalized [21], in this particular case due to how the multiplication operation is supported. On others, such as the SGI R12000, RC6 performs at up to a factor of two faster than Rijndael [2]. Support for the 32-bit multiplication seems to most determine the relative performance of RC6 and Rijndael. As an example of this, when considering performance on 32-bit processors we find that RC6 does not perform as well as Rijndael on the Pentium processor.

ARM™ (Advanced RISC Machine). During the AES process, two independent teams of researchers implemented the AES finalists on the ARM processor. Hachez et al. [9] show that in raw encryption speed RC6 is twice as fast as Rijndael and Messerges [10] also shows that RC6 out-performs Rijndael in terms of basic encryption speed[3]. In fact, Hachez et al. [9] comment on the suitability of RC6 to the ARM processor.

> *RC6 was beyond a doubt the easiest candidate to implement on a 32 bits (sic) machine, as is illustrated by its incredibly short code [...]. On a speed point of view, RC6 is impressive too.*
> – Hachez, Koeune, and Quisquater [9]

It is the superior performance of RC6 on processors such as these that make RC6 particularly suited to the high-end smart cards of today. We believe that these will become the low-end smart cards of tomorrow.

Advanced processors and non-feedback modes. Many modern processors offer opportunities for parallelism. The design of Rijndael is such that much of this parallelism is exploited within the encryption of a block. With RC6 this parallelism cannot be exploited so easily. Thus, when performance between the two algorithms is compared, the Rijndael speed is typically taking advantage of the available parallelism whereas the RC6 speed is not. To which some would reply "So what? That's surely a problem for RC6?"

Well, on one level this might be true. However it is important to take a more enlightened view. There are many applications that offer additional opportunities for parallelism. Most obvious among them might be using non-feedback modes

[3] Note, however, that the key schedule requirements for Rijndael will be considerably less than those for RC6.

of operation, but other applications might require simultaneous encryption of different inputs, or the simultaneous computation of encryption and MAC.

Those implementers and researchers that considered this issue during the AES process, typically came to the same conclusions. Rijndael typically gained the least among the finalists, if anything at all, when moving from a feedback mode (say) to a non-feedback mode of operation. But for performance-critical uses we might expect to use a non-feedback mode of operation, at which point the extensive opportunities for parallelism with RC6 are no longer overlooked. As has been observed:

> *On advanced CPUs the relative performance of candidates may differ widely between feedback and non-feedback (or interleaved) modes.*
> – Clapp [3]

As an example, Clapp [3] reports on the possible implications of using non-feedback modes with the TriMedia VLIW Media Processor.

> *RC6 shows the greatest benefit from interleaved modes, considerably outperforming the other candidates.*
> – Clapp [3].

Projections there indicate that RC6 can perform at around twice the speed of Rijndael when taking full advantage of the parallelism available with a non-feedback mode.

Similar observations have been made elsewhere. On the Alpha 21264 [20] the AES finalists were compared and estimates given for the time required to encrypt two blocks simultaneously. The authors conclude:

> *RC6 has the most potential for parallelism when multiple streams are processed on the same processor simultaneously in a single thread.*
> – Weiss and Binkert [20].

Of course, the full extent of any gains will be specific to the processor, the environment, and the mode of use. But in general it can be expected that RC6 will be particularly well-suited to non-feedback modes of encryption. At the recent workshop *AES Modes of Operation Workshop* on October 20th, in Baltimore, it appeared that NIST were strongly considering including a non-feedback mode of operation in the forthcoming new standard on modes of operation for the AES. Judging from comments received during the AES process this could be welcomed by a good many implementers.

> *High performance applications, such as high speed network encryption, will require the increase in output, and as a result, often focus on a non-feedback mode of operations such as counter mode to obtain performance.*
> – Weeks, Bean, Rozylowicz, Ficke [19]

However we believe that for many applications which require high encryption rates, non-feedback modes [...] will be the modes of choice. Note that the Counter mode grew out of the need for high speed encryption of ATM networks which required parallelization of the encryption algorithm.
– Elbirt, Yip, Chetwynd, Paar [7]

[...] using current standards does not permit to fully utilize the performance advantage of the hardware implementations of secret key cryptosystems, based on parallel processing of multiple blocks of data.
– Gaj, Chodowiec [8].

Java implementations. The suitability of RC6 to Java implementation has been consistently noted [5,6,17]. RC6 offers excellent performance both in raw encryption speed and in the amount of memory required. From Table 1 in [17] we see that RC6 has the smallest code size by a factor of three over the closest finalist and uses the least amount of dynamic RAM.

Although the least time was spent on optimizing RC6 it still comes out as the fastest algorithm on almost all platforms.
– Sterbenz and Lipp [17].

Code size and memory requirements. RC6 has exceptionally compact code and requires little additional working memory. Sometimes the good performance of Rijndael is attained by the use of look-up tables.

5 Conclusions

By choosing a single AES algorithm NIST looked for an algorithm that fitted a wide-range of environments and applications. In this note we have provided a top-level comparison between RC6 and Rijndael highlighting some of the different characteristics of these two modern ciphers.

References

1. L.E. Bassham III. Efficiency Testing of ANSI C Implementations of Round 1 Candidate Algorithms for the Advanced Encryption Standard. October 13, 1999. National Institute of Standards and Technology. Available from www.nist.gov/aes/round1/round1.htm .
2. L.E. Bassham III. Efficiency Testing of ANSI C Implementations of Round 2 Candidate Algorithms for the Advanced Encryption Standard. National Institute of Standards and Technology. Proceedings of 3rd AES conference, New York, pages 136-148, April 2000.
3. C. Clapp. Performance of AES candidates on the TriMedia VLIW Media-processor. Available from www.nist.gov/aes .
4. J. Daemen and V. Rijmen. AES Proposal: Rijndael. June 11, 1998.

5. J. Dray. Report on the NIST Java AES candidate algorithm analysis. National Institute of Standards and Technology. Available from www.nist.gov/aes . November 8, 1999.
6. J. Dray. Report on the NIST Java AES candidate algorithm analysis. National Institute of Standards and Technology. Proceedings of 3rd AES conference, New York, pages 149-160, April 2000.
7. A. Elbirt, W. Yip, B. Chetwynd, and C. Paar. An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists. Proceedings of 3rd AES conference, New York, pages 13-27, April 2000.
8. K. Gaj and P. Chodowiec. Comparison of the hardware performance of the AES candidates using reconfigurable hardware. Proceedings of 3rd AES conference, New York, pages 40-54, April 2000.
9. G. Hachez, F. Koeune, and J.J. Quisquater. cAESar results: Implementation of four AES candidates on two smart cards. Proceedings of 2nd AES Candidate Conference, Rome, pages 95-108, March 1999.
10. T. Messerges. Securing the AES finalists against power analysis attack. In B. Schneier, editor, Fast Software Encryption, Lecture Notes in Computer Science, Springer-Verlag. To appear.
11. J. Nechvatal, E. Barker, D. Dodson, M. Dworkin, J. Foti, and E. Roback. Status Report on the First Round of the Development of the Advanced Encryption Standard. National Institute of Standards and Technology. August 9, 1999. Available from www.nist.gov/aes .
12. J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback. Report on the Development of the Advanced Encryption Standard (AES). October 2, 2000.
13. R.L. Rivest. The RC5 encryption algorithm. In B. Preneel, editor, Fast Software Encryption, Lecture Notes in Computer Science Volume 1008, pages 86-96, Springer Verlag, 1995. Available at theory.lcs.mit.edu/~rivest/ .
14. R.L. Rivest, M.J.B. Robshaw, R. Sidney, and Y.L. Yin. The RC6 Block Cipher. v1.1, August 20, 1998. Available at www.rsalabs.com/rc6/ .
15. R.L. Rivest, M.J.B. Robshaw, and Y.L. Yin. RC6 as the AES. Provided at 3rd AES conference, New York, April 2000. Available at www.rsalabs.com/rc6/ .
16. R.L. Rivest, M.J.B. Robshaw, and Y.L. Yin. The Case for RC6 as the AES. May 15, 2000. Available at www.rsalabs.com/rc6/ .
17. A. Sterbenz and P. Lipp. Performance of the AES candidate algorithms in Java. Proceedings of 3rd AES conference, New York, pages 161-165, April 2000.
18. N. Weaver and J. Wawrzynek. A comparison of the AES candidates amenability to FPGA implementation. Proceedings of 3rd AES conference, New York, pages 28-39, April 2000.
19. B. Weeks, M. Bean, T. Rozylowicz, and C. Ficke. Hardware performance simulations of Round 2 Advanced Encryption Standard algorithms. Proceedings of 3rd AES conference, New York, pages 286-304, April 2000.
20. R. Weiss and N. Binkert. A comparison of AES candidates on the Alpha 21264. Proceedings of 3rd AES conference, New York, pages 75-81, April 2000.
21. J. Worley, B. Worley, T. Christian, and C. Worley. AES finalists on PA-RISC and IA-64: Implementations & Performance. Proceedings of 3rd AES conference, New York, pages 57-74, April 2000.
22. T. Wollinger, M. Wang, J. Guajardo, and C. Paar. How well are high-end DSPs suited for the AES algorithms? Proceedings of 3rd AES conference, New York, pages 94-105, April 2000.