VOLUME 1, NUMBER 1 — SPRING 1995

RSA LABORATORIES'

CryptoBytes

The technical newsletter of RSA Laboratories, a division of RSA Data Security, Inc.

Contents

The Impending Demise of RSA?

Gilles Brassard
Département d'informatique et de R.O.
Université de Montréal
C.P. 6128, Succursale Centre-Ville
Montréal (Québec)
CANADA H3C 3J7

In August 1977, Rivest, Shamir and Adleman issued a ciphertext challenge worth one hundred dollars to *Scientific American* readers when Martin Gardner described their revolutionary RSA cryptographic system in his monthly "Mathematical Games" column. This sounded very safe because it was estimated at the time that the fastest existing computer using the most efficient known algorithm could not earn the award until it had run without interruption for millions of times the age of the Universe. This particular challenge involved the factorization of a 129-figure number *r*, which appeared to be so out-of-reach at the time that Martin Gardner reported: "Rivest and his associates have no proof that at some future time no one will discover a fast algorithm for factoring composites as large as *r* [...]. They consider [the] possibility extremely remote." Nevertheless, the $100 reward was cashed last year—and donated to the Free Software Foundation—after a mere eight months of intensive computation led by Derek Atkins and Arjen Lenstra. What happened?

Professor Gilles Brassard, Université de Montréal, is interested in all aspects of cryptology but perhaps his best-known contribution is as a co-developer of Quantum Cryptography. He can be contacted at brassard@iro.umontreal.CA. *This essay was written while the author was on sabbatical at the University of Wollongong, Australia. Research supported in part by Canada's* NSERC *and Québec's* FCAR.

The increase in raw computing power during those years cannot be discounted, nor can the fact that hundreds of workstations around the world spent all their otherwise idle cycles on the task for the better part of one year. Indeed, the several thousand MIPS-years that were spent on the calculation might not have been available to poor academics back in 1977. But in the Preface of my new textbook *Fundamental of Algorithmics*, soon to be released by Prentice-Hall with Paul Bratley as coauthor, I am quick to point out that far more significant was the discovery of more sophisticated factorization algorithms. When I put on my hat as teacher of algorithmics, I like to use this example to illustrate that more efficient algorithms can produce much more dramatic results than better hardware. To make life more interesting, however, I am wearing quite a different hat as I write this essay!

Even though the "double large prime multiple polynomial variation of the quadratic sieve" algorithm was successful in factoring the 129-figure number relevant to the RSA challenge, the age of the Universe would still not suffice to factor a 500-figure number by this or any other known classical algorithm (such as the number field sieve) even if all the world's computers were put to contribution. Therefore, one should not infer from the fate of the 1977 challenge that RSA has been broken: the lesson is that bigger numbers should be used and that overconfident claims should be avoided. Clearly, even more remarkable advances in algorithmics will be required if RSA is to fail completely or even if it is to fail on keys merely twice the size used in the *Scientific American* challenge. Progress in hardware

(continued on page 3)

RSA LABORATORIES
CRYPTOGRAPHIC RESEARCH AND CONSULTATION

Welcome to CryptoBytes

Welcome to the first issue of *CryptoBytes,* the technical newsletter on cryptography from RSA Laboratories.

One of the nicest features of cryptographic research is the speed with which developments occur. This is a primary reason why so many researchers find working in the field so rewarding. More often than not, however, new results or second-hand accounts of someone's valued opinion circulate for months by word of mouth or by E-mail before appearing in journals or in conference proceedings. In fact, it might be convincingly argued that a great deal of interesting information is never actually published, never cited, and never properly referred to by other researchers.

Prof. Ronald Rivest is co-inventor of the RSA public-key cryptosystem, a co-founder of RSA Data Security Inc. and a distinguished associate of RSA Laboratories.

A newsletter can alleviate this situation. The aim would be to circulate interesting news as it happens, thereby providing a reliable distribution method for substantial 'bites of crypto'. In addition, a newsletter might provide a forum for results or opinions that, while of great cryptographic interest, would not appear at any of the more classical outlets because of their format.

Such a newsletter should be of interest to all those involved in cryptography; from those implementing cryptographic techniques and designing cryptographic products to those in the academic development of cryptographic knowledge. Often these groups are viewed as being somewhat exclusive of each other; instead we suggest that there is an important symbiosis. One goal of a newsletter on cryptography must be the transfer of information across these artificial divides. In this way researchers will hear about continuing efforts to implement the fruits of their research efforts and implementers can keep track of the latest cryptographic innovations.

With the first issue of *CryptoBytes* in your hands, we are hoping to achieve some of these goals. We are also hoping to provide a complement to current newsletters such as IEEE's *Cipher,* the IACR newsletter and the TIS *Data Security Letter,* among many others.

Much of the future success of *CryptoBytes* will depend on input from outside of RSA Laboratories. Such input might range from invited articles and researchers providing notification of recent results and developments, through letters and opposite opinions from readers. While RSA Laboratories will coordinate *CryptoBytes,* the intention is for it to become a useful resource for the whole cryptographic community. To help in this process, back issues of *CryptoBytes* will be available free of charge via the World-Wide Web.

We hope that you'll agree that this first issue of *CryptoBytes* is a step towards our goals. We would very much like to thank the writers who have contributed to this first issue, and we welcome any comments, suggestions or proposals for future issues.

— Ron Rivest

Editor's note: Suggestions and contributions for future issues of *CryptoBytes* can be sent to *bytes-ed@rsa.com* or to RSA Laboratories by any of the methods given below.

Subscription Information

CryptoBytes is published four times annually; printed copies are available for an annual subscription fee of U.S. $90. To subscribe, contact RSA Laboratories at:

RSA Laboratories
100 Marine Parkway, Suite 500
Redwood City, CA 94065
415/595-7703
415/595-4126 (fax)
rsa-labs@rsa.com

Back issues in electronic form are available via the World-Wide Web at http://www.rsa.com/rsalabs/cryptobytes/.

RSA Laboratories is the research division of RSA Data Security, Inc., the company founded by the inventors of the RSA public-key cryptosystem. RSA Laboratories reviews, designs and implements secure and efficient cryptosystems of all kinds. Its clients include government agencies, telecommunications companies, computer manufacturers, software developers, cable TV broadcasters, interactive video manufacturers, and satellite broadcast companies, among others.

Continued from page 1

will at best be a minor factor in the eventual success of future attacks against RSA. Right? Wrong!

Quantum computing, an emerging branch of computer science, may well prove the above conventional wisdom false. For the first time, revolutionary new concepts may hold the key to a computer that would go *exponentially* faster than conventional computers, at least for some computational tasks. This means that the speed-up would be increasingly spectacular as the size of the input gets larger, which is precisely the type of claim that had been the prerogative of algorithmic improvements until now. In particular, building on the work of David Deutsch and Richard Jozsa, Ethan Bernstein and Umesh Vazirani, Daniel Simon, and Don Coppersmith, Peter Shor has discovered that quantum computers can factor an n-figure number in a time asymptotically proportional to $n^{2+\varepsilon}$ for arbitrarily small ε. This means that it would take about the same time to crack an RSA key as to use it legitimately! In other words, quantum computers spell complete disaster on RSA. This has not (yet) forced RSA Laboratories to file for Chapter 11 because there are formidable technological difficulties before the first quantum computer can be built, but the possibility should not be underestimated.

What is a quantum computer? This theoretical notion emerged from the work of Paul Benioff, Richard Feynman and David Deutsch in the first half of the eighties. I cannot say much in this short essay but I shall try to sketch the basic principles. For more detail and references to the work mentioned here—such as Peter Shor's quantum factorization algorithm—I invite you to read my forthcoming paper in *Current Trends in Computer Science,* Jan van Leeuwen (Ed.), Lecture Notes in Computer Science, Volume 1000 (special anniversary volume), Springer-Verlag, 1995. Until this volume has appeared, you may wish to read my earlier account in *Sigact News,* Volume 25, number 4, December 1994, pp. 15 – 21.

Let us begin with a quantum bit, or *qubit* (a word coined by Benjamin Schumacher). In classical digital computing, a bit can take either value 0 or value 1. Nothing in between is allowed. In quantum computing, a qubit can be in linear *superposition* of the two classical states. If we denote the classical states by $|0\rangle$ and $|1\rangle$, then a qubit can be in state $\psi = \alpha|0\rangle + \beta|1\rangle$ for arbitrary complex numbers α and β subject to $|\alpha|^2 + |\beta|^2 = 1$. The coefficients α and β are called the *amplitudes* of $|0\rangle$ and $|1\rangle$, respectively. A qubit is best visualized as a point on the surface of a unit sphere whose North and South poles correspond to the classical values. If state ψ is *observed* in the sense that the qubit is asked to assume a classical value, it will *collapse* onto $|0\rangle$ with probability $|\alpha|^2$ and onto $|1\rangle$ with complementary probability $|\beta|^2$. So far, it looks as if we have but reinvented analogue computation. Things become more interesting when we consider quantum *registers* composed of n qubits. Such registers can be set to an arbitrary quantum superposition of states $\Psi = \sum_{x \in X} \alpha_x |x\rangle$ subject to $\sum_{x \in X} |\alpha_x|^2 = 1$, where X denotes the set of all classical n-bit strings. If this register is asked to assume a classical value, each x in X will be obtained with probability $|\alpha_x|^2$, and the register will collapse onto the observed value.

In principle, it is possible to compute on such registers. If a quantum computer is programmed to compute some function $f : X \to Y$ and if it is started with superposition $\Psi = \sum_{x \in X} \alpha_x |x\rangle$ in its input register, then it will produce superposition $\Psi' = \sum_{x \in X} \alpha_x |f(x)\rangle$ in its output register *in the time needed to compute f on a single input.* In other words, this provides for exponentially many computations to take place simultaneously in a single piece of hardware, a phenomenon known as *quantum parallelism.* We are far from analogue computing now. The good news is that we have obtained exponentially many answers for the price of one. The bad news is that Heisenberg's uncertainty principle forbids us from looking at the output register for fear of spoiling it! More precisely, if we ask the output register to assume a classical value, it will collapse to the value of $f(x)$ for an x randomly chosen in X with probability $|\alpha_x|^2$, and the quantum superposition Ψ' will be destroyed by the measurement. So far, it looks as if quantum computing is not only impractical but useless as well.

What makes quantum computing interesting is the notion of *quantum interference,* which is exactly the principle behind Young's celebrated double-slit experiment. In a classical probabilistic calculation, it is possible to program the computer to select one of several possible computation paths according to the laws of probability. In any specific instance of the calculation, however, only one of the potential paths is actually taken, and what-could-have-happened-

Progress in hardware will at best be a minor factor in the eventual success of future attacks against RSA. Right? Wrong!

Revolutionary new concepts may hold the key to a computer that would go exponentially faster than conventional computers.

It is possible to program a quantum computer so that all potential computation paths are taken simultaneously in quantum superposition.

but-did-not has no influence whatsoever on the actual outcome. In contrast, it is possible to program a quantum computer so that all potential computation paths are taken simultaneously in quantum superposition. What makes this so powerful—and mysterious—is the exploitation of constructive and destructive interference phenomena, which allows for the reinforcement of the probability of obtaining desired results at the expense of the probability of obtaining spurious results. This happens because the amplitude of reaching any given state is the sum of the amplitudes of reaching this state by all possible computation paths. Because amplitudes can be negative (even complex), the amplitude of reaching an undesirable result from one path can be annihilated by the amplitude of reaching it from another path. In the words of Richard Feynman, "somehow or other it appears as if the probabilities would have to go negative." It is not easy to put such interference phenomena to practical use, but Peter Shor's *tour de force* proves that it is possible, at least in principle.

The actual implementation of a quantum computer will be very challenging. It may even turn out to remain forever out of reach of human technology. One difficulty is to keep *quantum coherence* in the computer. The problem is that coherent superpositions are very fragile: they spontaneously turn into incoherent statistical mixtures—which are unsuitable for quantum parallelism—because of residual interactions with the environment. Consequently, quantum information disappears gradually from the computer. Rolf Landauer has pointed out the additional problem that quantum computation may be susceptible to spontaneous reversals, which would unpredictably cause the computation to go backwards. Yet another difficulty is that of error correction despite early work by Asher Peres and more recent work by André Berthiaume, David Deutsch and Richard Jozsa: in classical digital computing, discrete states are continuously snapped back onto the proper voltages for 0 and 1; no similar process is available for quantum computing even if the program is such that the legitimate states have discrete amplitudes. As a result, the computer may drift away from its intended state.

I like to think that I shall see a special-purpose quantum factorization device in my lifetime.

Nevertheless, Seth Lloyd has promising ideas on how to build "a potentially realizable quantum computer." Furthermore, it was discovered by David DiVincenzo that universal quantum computation can be based on simple 2-bit quantum gates such as the natural quantum extension of the classical exclusive-or. Experimental physicists such as Serge Haroche and Jean-Michel Raimond are already attempting to build simple quantum gates based on atomic interferometry and microwave cavities capable of trapping single photons for a significant fraction of one second. Even though a full-fledged quantum computer may take a long time to come, I like to think that I shall see a special-purpose quantum factorization device in my lifetime. If this happens, RSA will have to be abandoned. Most other practical public-key systems will be compromised as well because Peter Shor has also devised a quantum algorithm for extracting discrete logarithms.

But do not despair for the fate of cryptography: there will always be quantum cryptography to come to the rescue! Quantum cryptographic systems take advantage of the uncertainty principle, according to which measuring a quantum system in general disturbs it and yields incomplete information about its state before the measurement—which is precisely what makes quantum computers difficult to program. When information is encoded with non-orthogonal quantum states, any attempt from an eavesdropper to access the information necessarily entails a probability of spoiling it irreversibly, which can be detected by the legitimate users. Using protocols that I have designed with Charles H. Bennett, building on earlier work of Stephen Wiesner, this phenomenon can be exploited to implement a key distribution system that is secure even against an eavesdropper with unlimited computing power, indeed even against an eavesdropper who has a quantum computer at her disposal! Several prototypes have been built in recent years. In particular, British Telecom announced in September 1994 the successful completion of their fully working apparatus, perfected by Paul Townsend, capable of implementing quantum key distribution over 10 kilometres of ordinary optical fibre. For more information on quantum cryptography, please consult my article in the March 1995 sesquicentennial Anniversary Edition of *Scientific American* on "The Computer in the 21st Century" (reprinted with updates from their October 1992 regular issue).

To paraphrase the Book of Job,

The quantum taketh away
and the quantum giveth back!

Message Authentication with MD5

Burt Kaliski and **Matt Robshaw**
RSA Laboratories
100 Marine Parkway, Suite 500
Redwood City, CA 94065 USA

Message authentication is playing an important role in a variety of applications, especially those related to the Internet protocols and network management, where undetected manipulation of messages can have disastrous effects.

There is no shortage of good message authentication codes, beginning with DES-MAC, as defined in FIPS PUB 113 [7]. However, message authentication codes based on encryption functions such as DES, which were designed for hardware implementation, may be somewhat limited in performance for software, and there is also the question of U.S. export restrictions on encryption functions.

In standards efforts such as the Simple Network Management Protocol [5] and proposals for Internet Protocol security, a more practical solution seemed to be to base the authentication codes not on DES but on hash functions designed for fast software implementation which are widely available without restriction, such as the MD5 message-digest algorithm [9].

A more practical solution seemed to be to base the authentication codes [...] on hash functions.

But how to do it? Hash functions are intended to resist inversion — finding a message with a given hash value — and collision — finding two messages with the same hash value. Message authentication codes, on the other hand, are intended to resist forgery — computing a message authentication code without knowledge of a secret key. Building a message authentication code on an encryption function thus seems a logical choice (and the security relationship has been recently settled — in work by Mihir Bellare, Joe Kilian and Phillip Rogaway [3]). Building one on a hash function, however, is not as simple, because the hash function doesn't have a key.

Burt Kaliski is chief scientist and Matt Robshaw is a research scientist at RSA Laboratories. They can be contacted at burt@rsa.com or matt@rsa.com.

A hash function can provide message authentication in a most satisfying manner when combined with a digital signature algorithm, which does have a key. But typical digital signature schemes have some performance overhead, which while acceptable for the periodic setup of communications sessions, is often too large on a message-by-message basis. Thus, the focus is on message authentication based on a shared secret key, which is ideally integrated into the hash function in some manner.

As an illustration of the challenges, consider the "prefix" approach where the message authentication code is computed simply as the hash of the concatenation of the key and the message, where the key comes first and which we denote as MD5 ($k \cdot m$).

MD5 follows the Damgård/Merkle [4,6] iterative structure, where the hash is computed by repeated application of a compression function to successive blocks of the message. (See Figure 1.) For MD5, the compression function takes two inputs — a 128-bit chaining value and a 512-bit message block — and produces as output a new 128-bit chaining value,

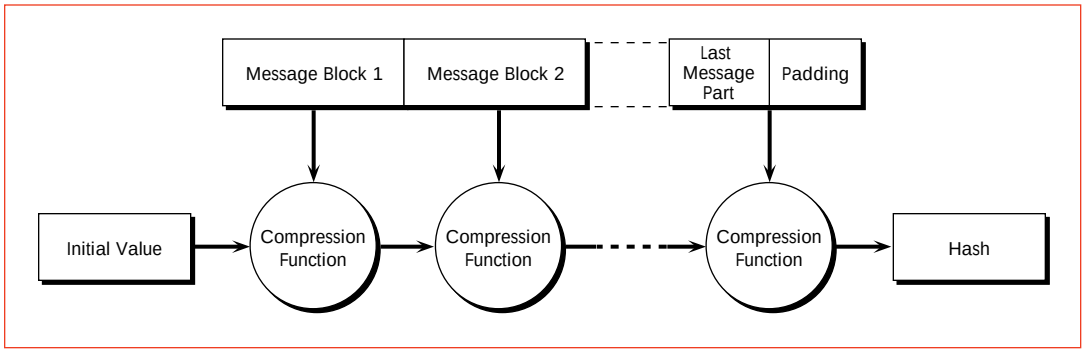

Figure 1. Damgård/Merkle iterative structure for hash functions.

which is input to the next iteration of the compression function. The message to be hashed is first padded to a multiple of 512 bits, and then divided into a sequence of 512-bit message blocks. Then the compression function is repeatedly applied, starting with an initial chaining value and the first message block, and continuing with each new chaining value and successive message blocks. After the last message block has been processed, the final chaining value is output as the hash of the message.

Because of the iterative design, it is possible, from only the hash of a message, to compute the hash of longer messages that start with the initial message

and include the padding required for the initial message to reach a multiple of 512 bits. Applying this to the prefix approach, it follows that from MD5 ($k \cdot m$), one can compute MD5 ($k \cdot m'$) for any m' that starts with $m \cdot p$, where p is the padding on $k \cdot m$. In other words, from the message authentication code of m, one can forge the message authentication code of $m \cdot p \cdot x$ for any x, without even knowing the key k, and without breaking MD5 in any sense. This is called a "message extension" or "padding" attack [10].

Other hash functions with an iterative design, such as NIST's Secure Hash Algorithm [8], are also vulnerable to the message extension attack, and similar attacks can also be mounted on tree-structured designs.

(Note also that if only part of the hash were output, say only 64 bits, this attack would not be possible; however, this is not a completely satisfying solution because of other concerns raised below. In SNMP, the message extension attack is not a problem because messages are a fixed length. Another way to avoid the attack is to include an explicit length field at the beginning of the message.)

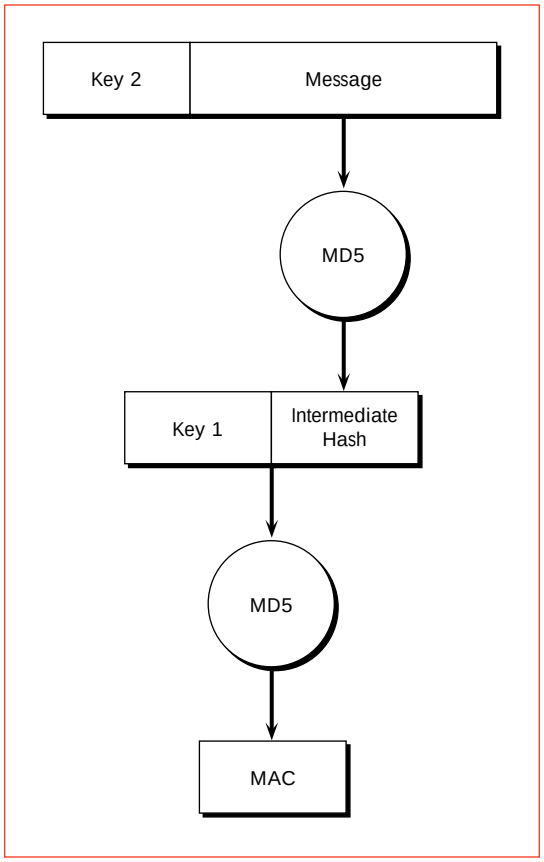

Figure 2. A recommended approach to message authentication with MD5. Here, the keys are each 128 bits long. They may be the same, although different keys are preferable.

Because of the message extension attack on the prefix approach, the "suffix" approach, MD5 ($m \cdot k$), would seem to be preferred. But another problem arises: the key may be vulnerable to cryptanalysis, depending on the properties of the compression function. This is because the message authentication code is a function of known values and the key, assuming the key is passed entirely to the last iteration of the compression function. (The known values are the next-to-last chaining value, which by assumption depends only on the message; the last part of the message; and the padding.)

An opponent who sees the message authentication codes for many messages thus sees the result of applying the compression function to many different known values and the same key, which may reveal information about the key. While our analysis suggests MD5's compression function is unlikely to reveal information about the key, other hash functions may not fare as well, and so we prefer a more robust design.

The prefix approach is also affected by these issues, but only when the message is very short and there is only a single iteration of the compression function.

Recommendations

In joint work with Mihir Bellare and Hugo Krawczyk of IBM, we have considered a number of approaches to message authentication with MD5, settling on three which we recommended to the Internet Protocol Security (IPSEC) working group:

1. MD5 ($k_1 \cdot$ MD5 ($k_2 \cdot m$)), where k_1 and k_2 are independent 128-bit keys
2. MD5 ($k \cdot p \cdot m \cdot k$), where k is a 128-bit key and p is 384 bits of padding
3. MD5 ($k \cdot$ MD5 ($k \cdot m$)), where k is a 128-bit key

The first and third approaches (see Figure 2) are similar, and solve the message extension attack on the prefix approach by the outer application of MD5, which conceals the chaining value which is needed for the attack. The outer MD5 also solves the concerns of cryptanalysis of the suffix approach, because the message authentication code is a function of the unknown secret key and other varying values, which are unknown. These approaches also approximate certain "provably secure" constructions developed by Bellare, Ran Canetti and Krawczyk [1].

(As a disclaimer, we can imagine hash functions for which this construction still doesn't solve the cryptanalytic problems because information from the inner application leaks to the outer one, but this seems more of a pathological case.)

The third approach may be more vulnerable to attack than the first since there is only one key and so

any information revealed from the outer application of the hash function compromises security, but we know of no such attack on MD5.

Although the third approach has a shorter key size than the first, the first could also be implemented with a 128-bit key, without any apparent loss in security. For instance, the keys k_1 and k_2 could be derived from a single 128-bit key k as k_1 = MD5 ($k \cdot \alpha$) and k_2 = MD5 ($k \cdot \beta$), where α and β are distinct constants.

The second approach (see Figure 3) is somewhat like triple encryption, where the first and third keys are the same (the second key is the message). The padding on the key at the beginning ensures that overall, there are at least two iterations of the compression function. Message extension in the prefix approach is solved by the key at the end, and the cryptanalysis of the suffix approach is solved by the key at the beginning. (Without the padding, very short messages might be vulnerable.)

It remains to be seen which, if any, of these three approaches is adopted.

Interestingly, one of the approaches we had been previously promoting is not among the three we recommended to the IPSEC working group, based on our concerns about key exposure. That approach, MD5 ($k \cdot$ MD5(m)), had the advantages that the inner MD5 is applied to the message in the familiar way — as a hash function — and the outer MD5 is applied to a fixed-length value, thereby avoiding message extension. However, since MD5 (m) is known, the door is open for possible cryptanalysis of the outer MD5 to recover the key k. While we once again do not have an attack that recovers the key, we felt as a general design principle that the key should be better concealed.

(Another concern with this approach, observed by some, is that collisions in MD5 — two messages with the same hash — result in collisions in the message authentication code. We do not consider this an intrinsic problem with this option, since MD5 is designed to resist collisions, at least to a certain level of difficulty. Nevertheless, we have no objection if the design of a message authentication code raises that level even further.)

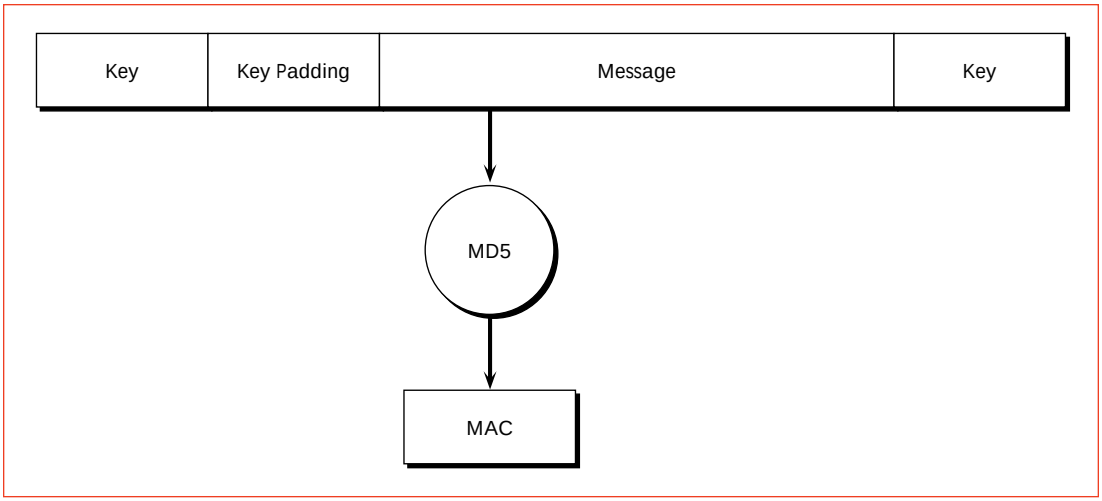

Figure 3. Another recommended approach to message authentication with MD5. Here, the key is 128 bits long and the key padding is 384 bits long.

Yet another approach that we considered was MD5(MD5 ($k \cdot m$)), which again applies MD5 in a familiar way. However, in terms of "provability" under certain assumptions it is less attractive than the three we recommended. (This does not mean that the approach is insecure, simply that the assumptions required for it to be secure are more complicated.)

As the IBM team has pointed out to us, all of the approaches are vulnerable to a chosen message attack involving about 2^{64} chosen messages. This general attack exploits the iterative structure of the message authentication code and applies to MACs based on encryption functions as well. The basic idea is that if two messages $a_i \cdot b$ and $a_j \cdot b$ have the same MAC, then it is possible that the "collision" occured before b was processed, so that for any c, $a_i \cdot c$ and $a_j \cdot c$ have the same MAC. Having found two messages $a_i \cdot b$ and $a_j \cdot b$ with the same MAC, the opponent asks for the MAC of $a_i \cdot c$ for some c, thereby obtaining (fraudulently) the MAC of $a_j \cdot c$.

As chosen message attacks go, 2^{64} is quite a large number, and we know of no general way to extend the attack to known messages, except when the known messages are all the same length and end with the same suffix. Full details are given in [1].

Starting over

So far, our research has focused on adapting an existing hash function to message authentication, which is a practical solution, since MD5 is already trusted, and software for MD5 is widely available. For the long term, designing a message authentication code from scratch is perhaps a better solution.

Mihir Bellare, Roch Guérin and Phillip Rogaway [2] describe techniques for such message authentication that are "provably secure," under certain assumptions about the underlying functions. Their techniques are also highly parallelizable, a feature that the iterative approach lacks by definition.

Bellare *et al*'s techniques assume the existence of a pseudorandom function, which takes two inputs, a key and a message block, and produces one output. By assumption, if the key input is fixed and unknown, it is difficult to distinguish the pseudorandom function on the message block from a truly random one in any reasonable amount of time. (This is similar to the idea that it is difficult to find collisions for a hash function — although it is possible because they exist, the amount of time required is large.)

The message authentication code is computed by combining, perhaps by bit-wise exclusive-or, the outputs of the pseudorandom function applied to the blocks of the message. To maintain the ordering of the different blocks, each block is tagged with its position in the message. A random block is also included for technical reasons.

Bellare *et al* show that if an opponent can forge message authentication codes, even with the opportunity to request message authentication codes on many different messages, then the opponent can also distinguish the pseudorandom function from a truly random one. Thus, under the assumption that it is difficult to distinguish the pseudorandom function from a truly random one, the message authentication code is secure.

The independent processing of the message blocks leads to the parallelizability of this approach.

It seems that many of the concerns about designing a message authentication code from a hash function are a consequence of the fact that the key is processed only once, or maybe twice. As a result, the key is isolated, and information about it can be obtained, or other parts of the message can be manipulated independent of the key. By contrast, in message authentication codes based on encryption functions, such as DES-MAC, the key is processed at every step. In Bellare *et al*'s techniques, the key is processed at every step.

We expect that MD5's compression function or a variant of it may be a suitable pseudorandom function for Bellare *et al*'s techniques, something which further research will determine.

References

[1] M. Bellare, R. Canetti and H. Krawczyk. *Keying MD5—Message authentication via iterated pseudorandomness*. In preparation.

[2] Mihir Bellare, Roch Guérin and Phillip Rogaway. *XOR MACs: New methods for message authentication using block ciphers*. Accepted to Crypto '95.

[3] Mihir Bellare, Joe Kilian and Phillip Rogaway. The security of cipher block chaining. In Yvo G. Desmedt, editor, *Advances in Cryptology— Crypto '94,* volume 839 of *Lecture Notes in Computer Science,* pages 341-358. Springer-Verlag, New York, 1994.

[4] I.B. Damgård. A design principle for hash functions. In G. Brassard, editor, *Advances in Cryptology: Proceedings of Crypto '89,* volume 435 of *Lecture Notes in Computer Science,* pages 416-427. Springer-Verlag, New York, 1990.

[5] J. Galvin and K. McCloghrie. *RFC 1446: Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)*. Trusted Information Systems and Hughes LAN Systems, April 1993.

[6] R. Merkle. One way hash functions and DES. In G. Brassard, editor, *Advances in Cryptology: Proceedings of Crypto '89,* volume 435 of *Lecture Notes in Computer Science,* pages 428-446. Springer-Verlag, New York, 1990.

[7] National Institute of Standards and Technology (formerly National Bureau of Standards). *FIPS PUB 113: Computer Data Authentication*. May 30, 1985.

[8] National Institute of Standards and Technology. *FIPS PUB 180: Secure Hash Standard (SHS)*. May 11, 1993.

[9] R. Rivest. *RFC 1321: The MD5 Message-Digest Algorithm*. RSA Data Security, Inc., April 1992.

[10] Gene Tsudik. Message authentication with one-way hash functions. *ACM Computer Communications Review,* 22(5):29-38, 1992.

The RC5 Encryption Algorithm*

Ronald L. Rivest
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139 USA

Introduction

RC5 is a fast symmetric block cipher suitable for hardware or software implementations. A novel feature of RC5 is the heavy use of *data-dependent rotations*. RC5 has a variable-length secret key, providing flexibility in its security level.

RC5 is a parameterized algorithm, and a particular RC5 algorithm is designated as *RC5-w/r/b*. We summarize these parameters below:

- w The *word size*, in bits. The standard value is 32 bits; allowable values are 16, 32, and 64. RC5 encrypts two-word blocks: plaintext and ciphertext blocks are each $2w$ bits long.
- r The number of rounds. Allowable values are 0, 1, ..., 255.
- b The number of bytes in the secret key K. Allowable values of b are 0, 1, ..., 255.

RC5 uses an "expanded key table" S, derived from the user's supplied secret key K. The size t of table S depends on the number r of rounds: S has $t = 2(r+1)$ words.

It is not intended that RC5 be secure for all possible parameter values. On the other hand, choosing the maximum parameter values would be overkill for most applications.

We provide a variety of parameter settings so that users may select an encryption algorithm whose security and speed are optimized for their application, while providing an evolutionary path for adjusting their parameters as necessary in the future. As an example, RC5-32/16/7 is an RC5 algorithm with the number of rounds and the length of key equivalent to DES. Unlike unparameterized DES, however, an RC5 user can easily upgrade the above choice to an 80-bit key by moving to RC5-32/16/10.

As technology improves, and as the true strength of RC5 algorithms becomes better understood through analysis, the most appropriate parameters can be chosen. We propose RC5-32/12/16 as providing a "nominal" choice of parameters. Further analysis is needed to analyze the security of this choice.

Overview of the Algorithm

RC5 consists of three components: a *key expansion* algorithm, an *encryption* algorithm, and a *decryption* algorithm. These algorithms use the following three primitive operations (and their inverses).

1. Two's complement addition of words, denoted by "+". This is modulo-2^w addition.
2. Bit-wise exclusive-OR of words, denoted by $\geq$.
3. A left-rotation (or "left-spin") of words: the rotation of word x left by y bits is denoted $x \lll y$. Only the $\lg(w)$ low-order bits of y are used to determine the rotation amount, so that y is interpreted modulo w.

Encryption and Decryption

We assume that the input block is given in two w-bit registers A and B. We also assume that key-expansion has already been performed, so that the array $S[0...t-1]$ has been computed. Below is the encryption algorithm in pseudo-code. The output is also placed in registers A and B.

```
A = A + S[0];
B = B + S[1];
FOR i = 1 TO r DO
    A = ((A ≥ B) ⋘ B) + S[2*i];
    B = ((B ≥ A) ⋘ A) + S[2*i+1];
```

We note the exceptional simplicity of this five-line algorithm. We also note that each RC5 round updates *both* registers A and B, whereas a "round" in DES updates only half of its registers. An RC5 "half-round" (one of the assignment statements updating A or B in the body of the loop above) is thus perhaps more analogous to a DES round.

The decryption algorithm can be easily derived from the encryption algorithm.

**RC5 and RSA-RC5 are registered trademarks of RSA Data Security, Inc. Patent pending.*

As technology improves, and as the true strength of RC5 algorithms becomes better understood through analysis, the most appropriate parameters can be chosen.

Professor Ronald L. Rivest is associate director of MIT's Laboratory for Computer Science. He can be contacted at rivest@theory.lcs.mit.edu.
A complete paper on RC5 was presented at the Leuven Algorithms Workshop in December 1994. An on-line version of the complete paper can be obtained by ftp or web. FTP: under pub/rivest/rc5 on theory.lcs.mit.edu; WEB: under http://theory.lcs.mit.edu/~rivest. Parts of this article originally appeared in Dr. Dobb's Journal,

The encryption algorithm is very compact, and can be coded efficiently in assembly language on most processors.

Key Expansion

The key-expansion routine expands the user's secret key K to fill the expanded key array S, so that S resembles an array of $t = 2(r+1)$ random binary words determined by K. The key expansion algorithm uses two "magic constants" and consists of three simple algorithmic parts.

The key-expansion algorithm uses two word-size binary constants P_w and Q_w. They are defined for arbitrary w as follows:

$$P_w = \text{Odd}((e-2)2^w)$$
$$Q_w = \text{Odd}((\phi-1)2^w)$$

where
$e = 2.718281828459...$ (base of natural logarithms)
$\phi = 1.618033988749...$ (golden ratio),

and where $\text{Odd}(x)$ is the odd integer nearest to x (rounded up if x is an even integer, although this won't happen here).

The first algorithmic step of key expansion is to copy the secret key $K[0...b-1]$ into an array $L[0...c-1]$ of $c = \lceil b/u \rceil$ words, where $u=w/8$ is the number of bytes/word. This operation is done in a natural manner, using u consecutive key bytes of K to fill up each successive word in L, low-order byte to high-order byte. Any unfilled byte positions of L are zeroed.

A distinguishing feature of RC5 is its heavy use of data-dependent rotations

The second algorithmic step of key expansion is to initialize array S to a particular fixed (key-independent) pseudo-random bit pattern, using an arithmetic progression modulo 2^w determined by the "magic constants" P_w and Q_w. Since Q_w is odd, the arithmetic progression has period 2^w.

```
S[0] = P_w;
FOR i = 1 TO t-1 DO
    S[i] = S[i-1] + Q_w;
```

The third algorithmic step of key expansion is to mix in the user's secret key in three passes over the arrays S and L. More precisely, due to the potentially different sizes of S and L, the larger array will be processed three times, and the other may be handled more times.

```
i = j = 0;
A = B = 0;
DO 3*max(t,c) TIMES:
    A = S[i] = (S[i] + A + B) <<< 3;
    B = L[j] = (L[j] + A + B) <<< (A+B);
    i = (i + 1) mod(t);
    j = (j + 1) mod(c);
```

The key-expansion function has a certain amount of "one-wayness": it is not so easy to determine K from S.

Speed

The encryption algorithm is very compact, and can be coded efficiently in assembly language on most processors. The table S is accessed sequentially, minimizing issues of cache size. The RC5 encryption speeds obtainable are yet to be fully determined. For RC5-32/12/16 on a 90MHz Pentium, a preliminary C++ implementation compiled with the Borland C++ compiler (in 16-bit mode) performs a key-setup in 220 microseconds and performs an encryption in 22 microseconds (equivalent to 360,000 bytes/sec). These timings can presumably be improved by more than an order of magnitude using a 32-bit compiler and/or assembly language—an assembly-language routine for the '486 can perform each round in eight instructions.

Security

A distinguishing feature of RC5 is its heavy use of *data-dependent rotations*—the amount of rotation performed is dependent on the input data, and is not predetermined.

The encryption/decryption routines are very simple. While other operations (such as substitution operations) could have been included in the basic round operations, our objective is to focus on the data-dependent rotations as a source of cryptographic strength.

Some of the expanded key table S is initially added to the plaintext, and each round ends by adding expanded key from S to the intermediate values just computed. This assures that each round acts in a potentially different manner, in terms of the rotation amounts used. The xor operations back and forth between A and B provide some avalanche properties, causing a single-bit change in an input block to cause multiple-bit changes in following rounds.

The use of variable rotations helps defeat differential cryptanalysis (Biham/Shamir [1]) and linear crypt-

analysis (Matsui [3]), since bits are rotated to "random" positions in each round; Kaliski and Yin analyze the security of RC5 against both types of cryptanalysis [2]. For the standard word size $w = 32$, their differential attack can be applied to RC5 with less than 12 rounds and their linear attack can be applied to RC5 with less than six rounds. An assessment of the RC5 encryption algorithm will appear in the Summer issue of *CryptoBytes*; meanwhile, I invite the reader to help determine the strength of RC5.

References

[1] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, New York, 1993.

[2] B. S. Kaliski Jr. and Y. L. Yin. *On differential and linear cryptanalysis of the RC5 encryption algorithm*. Accepted to Crypto '95.

[3] M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In Y. G. Desmedt, editor, *Advances in Cryptology — Crypto '94*, volume 839 of *Lecture Notes in Computer Science*, pages 1-11, Springer-Verlag, New York, 1994.

NEWS AND INFORMATION

X9F1 Considers Triple-DES Standard

The ANSI-accredited X9F1 working group has begun work on a standard for bulk data encryption for financial services based on so-called triple-DES, a method of extending the security of the Data Encryption Standard by encrypting three times with DES.

While triple-DES has been a standard mechanism for several years for encrypting keys as part of ANSI X9.17, attention has turned recently to triple-DES for bulk data encryption, in response to the decreasing security of DES's 56-bit key and the shortage of trusted alternatives to DES.

The specifics of the standard are yet to be determined, but two recommendations by cryptography experts are likely to have strong influence: that the three encryptions involve three different keys (X9.17 involves only two, where the first and third encryption is with the same key), and that modes of operation for bulk data encryption, such as cipher block chaining, be built around triple-DES as a primitive.

Modes involving single-DES instead of triple-DES as a primitive, such as encrypting three times with single-DES in cipher block chaining mode, have been shown by Eli Biham in the past year to be potentially no stronger than single-DES against certain attacks. Encrypting with triple-DES in cipher block chaining mode is not vulnerable to those attacks.

And while two-key triple-DES is significantly stronger than single-DES, it has a certain "certificational weakness" observed by Merkle and Hellman in 1980 which was revealed in 1990 as a known-plaintext attack by Wiener and van Oorschot. No such attacks are known for three-key triple-DES.

Balloting of the standard is expected in 1996.

Modes involving single-DES instead of triple-DES as primitive [...] have been shown to be potentially no stronger than single-DES against certain attacks.

RSA Laboratories Publishes PKCS #11

Culminating a year of development, RSA Laboratories has published the latest in its series of Public-Key Cryptography Standards, *PKCS #11: Cryptographic Token Interface Standard (Cryptoki)*.

PKCS #11 specifies an application programming interface (API) called Cryptoki to devices which hold cryptographic information and perform cryptographic functions, such as ISO smart cards, PCMCIA cards, and the SmartDisk. Cryptoki isolates applications from the device technology, presenting a common, logical view of the device called a "cryptographic token."

The interface supports a wide range of cryptographic mechanisms, including RSA, DSA, Diffie-Hellman, DES, triple-DES, RC2, RC4, MD2, MD5, and SHA; tokens are expected to support subsets of these mechanisms according to application profiles.

Cryptoki is at the algorithm-specific, technology-independent layer in the current cryptographic API standardization effort, and is expected to combine nicely with interfaces at the algorithm-independent, security-service-oriented layer such as the Generic Security Services API (GSSAPI).

Cryptoki [...] is expected to combine nicely with interfaces at the algorithm-independent, security-service-oriented layer.

Copies of PKCS #11 and other PKCS standards can be obtained by anonymous FTP to ftp.rsa.com in the pub/pkcs directory, or by E-mail to pkcs@rsa.com.

1995 RSA Laboratories Seminar Series

RSA Laboratories is pleased to announce details of the 1995 Seminar Series. Now in its third year, the Seminar Series has been expanded again and will now be presented at two US locations.

The Seminar Series is an intensive three-day presentation on all aspects of cryptography. In a slight departure from previous years, the first day of the Seminar Series will be a self-contained overview of the basic ideas and cryptographic techniques that are used today. Building on this introduction, the remainder of the seminar series will provide detailed analysis on many of the algorithms, techniques and theoretical foundations which dominate current cryptographic thinking.

The East Coast Seminar Series will be held at the Columbia Inn in Columbia, MD, from July 19-21; the West Coast Seminar Series will be held at the Hotel Sofitel in Redwood Shores, CA, from August 23-25. Contact RSA Laboratories for more information on how to register.

RSA Laboratories Technical Reports

The RSA Laboratories Technical Reports are now available via a subscription service. These reports offer detailed summaries of current research on a variety of topics and they bring together information from a wide variety of sometimes obscure sources. Subscription to the Technical Reports will be at one of two levels; individual or corporate. As well as receiving all previously written reports, subscribers will receive new reports as they appear as well as current research notes on items of major significance.

Contact RSA Laboratories for more information about the Technical Report subscription service.

Coming in the Summer 1995 CryptoBytes:

- *Elliptic curve cryptosystems*
- *RSA key size recommendations*
- *RC5 update*

For subscription information, see page 2 of this newsletter.

RSA LABORATORIES
CRYPTOGRAPHIC RESEARCH AND CONSULTATION
100 MARINE PARKWAY
REDWOOD CITY
CA. 94065-1031
TEL 415/595-7703
FAX 415/595-4126
rsa-labs@rsa.com

PRESORT
FIRST CLASS
U.S. POSTAGE
PAID
MMS, INC