VOLUME 1, NUMBER 2 — SUMMER 1995

RSA LABORATORIES'

CryptoBytes

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

Contents

Alfred Menezes
120 Math Annex
Auburn University
Auburn, AL 36849 USA

col, which allows two parties Alice and Bob to establish a secret key through an exchange of public messages, works as follows. Let p be a large prime number, and let α be a generator of the multiplicative group Z_p^*; in layman's terms this means that the powers $\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{p-2}$ of α, each reduced modulo p, yield all the integers between 1 and $p-1$. The parameters p and α are public knowledge. Alice then generates a random integer a, $0 \le a \le p-2$, and transmits $\alpha^a \bmod p$ to Bob. Bob similarly generates a random integer b, $0 \le b \le p-2$, and transmits $\alpha^b \bmod p$ to Alice. Both parties can now compute the number $\alpha^{ab} \bmod p$, which serves as their secret key.

The security of the Diffie-Hellman key agreement protocol is based on the apparent intractability of the *discrete logarithm problem* in Z_p^*: given a prime p, a generator α of Z_p^*, and an element $\beta \in Z_p^*$, determine the integer a, $0 \le a \le p-2$, such that $\alpha^a \equiv \beta \pmod p$. The best algorithm known for this problem is the general number field sieve by Dan Gordon, and has an asymptotic running time of

$$\exp((1.923 + o(1))(\log p)^{1/3}(\log \log p)^{2/3}). \qquad (1)$$

The security of the RSA public-key cryptosystem is based upon the difficulty of the problem of factoring integers. The number field sieve is the best algorithm known for this problem, and has an asymptotic running time of

$$\exp((1.923 + o(1))(\log n)^{1/3}(\log \log n)^{2/3}) \qquad (2)$$

RSA LABORATORIES

CRYPTOGRAPHIC RESEARCH AND CONSULTATION

Editor's Note

Welcome to the second issue of *CryptoBytes*. During May of this year, we launched this newsletter with the aim of providing a forum for "results and opnions that, while of great cryptographic interest, would not appear at any of the more classical outlets because of their format." The initial response to the general idea of a technical newsletter has been very encouraging, and we are pleased to bring a second issue which further fulfills the aims set out in the first.

By providing a suitable forum, we hope that reports on major algorithm developments, be they cryptographic, cryptanalytic or implementational, can be brought together for the benefit of the cryptographic community.

In this second issue of *CryptoBytes*, we are presenting two invited articles.

The first, *Elliptic Curve Cryptosystems* by A. Menezes, provides us with an intriguing glimpse at what are potentially very efficient alternatives to cryptosystems based on the discrete logarithm problem. This article is nicely complemented by news that the IEEE P1363 standardization effort is moving quickly towards standards for algorithms based around the use of elliptic curves.

Our second invited article, *The Future of Integer Factorization* by A. Odlyzko, provides a valuable contemporary assessment of factoring ability. The article also provides some thoughts on possible trends for the future and potential developments well into the next century. Because of the obvious relevance of this article to the continuing safe use of the RSA cryptosystem, we conclude the article with the current RSA Laboratories minimum key-size recommendations for users of RSA.

Following on pieces in the last issue of *CryptoBytes* we include an article providing details of an initial analysis of the RC5 encryption algorithm. In addition, readers interested in last issue's article *Message Authentication with MD5* will find a short letter by P. van Oorschot and B. Preneel essential reading.

We also include a short review of recent analysis into the performance of MD5. Since one of the original aims of *CryptoBytes* was to provide a basic, reliable cryptographic news service, we are hoping that this item will be the first of an occasional but ongoing feature, *Algorithms Update*. By providing a suitable forum, we hope that reports on major algorithm developments, be they cryptographic, cryptanalytic or implementational, can be brought together for the benefit of the cryptographic community.

To include all these items, we have already had to increase the number of pages in *CryptoBytes*. We would very much like to thank all the writers who have contributed to this second issue, and since the future success of *CryptoBytes* depends on input from all sectors of the cryptographic community, we encourage any readers with comments, opposite opinions, suggestions or proposals for future issues to contact the *CryptoBytes* editor at RSA Laboratories or by E-mail to *bytes-ed@rsa.com*.

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 and development 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.

Design and layout for CryptoBytes are provided by CodaGraphics, Oakland, CA.

Continued from page 1

Comparing these two running times, one can conclude that with our current state of knowledge, it is roughly as difficult to compute discrete logarithms modulo a prime p as it is to factor an integer n of the same size.

There are many cryptographic protocols whose security is based on the discrete logarithm problem in $\mathbf{Z}_p^*$, for example the ElGamal public-key encryption and signature schemes, the Digital Signature Algorithm (DSA), and the Schnorr signature scheme. Originally these protocols were all described in thealgebraic setting of the multiplicative group $\mathbf{Z}_p^*$. However, they can equally well be described inthe setting of any finite group, for example the multiplicative group of the finite field F_{2^m} of characteristic 2.

Why consider other groups?

There are two primary reasons for this. Firstly, other groups may be easier to implement in software or hardware. Secondly, the discrete logarithm problem in the group may be harder than the discrete logarithm problem in $\mathbf{Z}_p^*$. Consequently, one can use a group G that is smaller than $\mathbf{Z}_p^*$ while maintaining the same level of security. This results in cryptosystems with smaller key-sizes, bandwidth savings, and possibly faster implementation.

Elliptic curves

In 1985, Neal Koblitz and Victor Miller independently proposed using the group of points on an elliptic curve in existing discrete-log cryptosystems. For an introduction to this subject, the reader is referred to the books by Koblitz [2] and Stinson [4]. A more complete treatment is given by Menezes [3]. Without going into all the details, an *elliptic curve* over a finite field F is the set of all solutions (also called points) (x,y), x F, y F, to an equation of a special form. For example, if the finite field is $F = Z_p$, the integers modulo a prime p, then the equation has the form $y^2 = x^3 + ax + b$, where a,b Z_p and $4a^3 + 27b^2 \neq 0 \pmod{p}$. There is a simple group law for adding two points on the curve to produce a third point. The addition rule simply involves a few operations (addition, subtraction, multiplication and inversion) in the underlying finite field F, and thus can be efficiently implemented.

The main attraction of elliptic curve cryptosystems arises because the analogue of the discrete logarithm problem in these curves is apparently much harder than the discrete logarithm problem in Z_p^* and the integer factorization problem. More precisely, let E be an elliptic curve defined over a finite field F_q, where q is a prime or prime power. Let n denote the number of points on E; it is always the case that n is roughly equal to q. If n is a prime (or at least divisible by a large prime) then the best method known for computing discrete logarithms in E are the fully exponential algorithms of Shanks and Pollard, which have an expected running time of about $\sqrt{q}$ elliptic curve operations. What should be noted here is that the function $\sqrt{q}$ grows much faster than the running time functions for discrete logs in Z_p^* (equation (1)) and factoring integers (equation (2)).

As a concrete example, let E be an elliptic curve over the field $F_{2^{160}}$. As an optimistic estimate, suppose that a machine rated at 1 mips can perform 40,000 elliptic curve operations per second. (This estimate is indeed very optimistic — an ASIC built by MÖBIUS Encryption Technologies for performing elliptic curve operations over the field $F_{2^{155}}$ has a 40 MHz clockrate and can perform roughly 40,000 elliptic operations per second.) Then the computing power required to compute a single discrete logarithm in E is about 10^{12} MY (mips-years). By contrast, based on the most recent implementation of the number field sieve for factoring by Dodson and Lenstra [1], the computing power required to factor a 1024-bit integer is estimated to be about $3 \cdot 10^{11}$ MY (see Andrew Odlyzko's article in this issue). Thus an elliptic curve cryptosystem over the 160-bit field $F_{2^{160}}$ offers the same level of security as RSA or DSA with a modulus size of 1024 bits.

The advantage of the elliptic curve system is that arithmetic in the field $F_{2^{160}}$ is far easier to implement, both in hardware and software, than arithmetic modulo a 1024-bit integer. For example,

There are many cryptographic protocols whose security is based on the discrete logarithm problem in Z_p^* [...]. However, they can equally well be described in the setting of any finite group.

[...] elliptic curve cryptosystems offer the most security per bit of any known public-key scheme [...]

the ASIC mentioned above for performing ellip In addition, the encryption and signature schemes based on elliptic curves are an order of magnitude faster than 1024-bit DSA and RSA. Similar speedups are now possible in software due to recent advances in the software implementation of elliptic curve cryptosystems over F_{2^m}.

In summary, elliptic curve cryptosystems offer the most security per bit of any known public-key scheme. This will tend to increase their attractiveness relative to other cryptosystems as computing power improvements warrant general key size increases. The smaller key sizes result in smaller system parameters, smaller public-key certificates, bandwidth savings, and faster implementations. Elliptic curve systems are particularly beneficial in applications where computational power and integrated circuit space is limited, such as smart cards, PCMCIA cards, and wireless devices.

tic curve arithmetic over $F_{2^{155}}$ has only 12,000 gates, and would occupy less than 5% of the area typically designated for a smart card processor.

References

[1] B. Dodson and A. Lenstra, "NFS with four large primes: an explosive experiment", *Advances in Cryptology — CRYPTO'95*, to appear.

[2] N. Koblitz, *A Course in Number Theory and Cryptography*, Springer-Verlag, New York, 1994.

[3] A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, Boston, 1993.

[4] D. Stinson, *Cryptography — Theory and Practice*, CRC Press, Boca Raton, 1995.

STANDARDS UPDATE

Elliptic Curves in Draft IEEE Standard

Elliptic curve cryptosystems are taking another step forward from theory to practice as part of a draft standard being prepared by IEEE's P1363 working group, *Standard for RSA, Diffie-Hellman and Related Public-Key Cryptography*.

Four cryptographic mechanisms based on elliptic curves are currently being considered:

- Elliptic Curve Encryption Scheme (ECES), an analog of the ElGamal public-key cryptosystem;
- Elliptic Curve Signature Scheme (ECSS), an analog of a variant of the ElGamal signature scheme;
- Elliptic Curve Digital Signature Algorithm (ECDSA), an analog of NIST's Digital Signature Algorithm; and
- Elliptic Curve Key Establishment Protocol (ECKEP), an extension of Diffie-Hellman key agreement that provides implicit key authentication.

Elliptic curve cryptosystems are taking another step forward from theory to practice.

Other algorithms to be included in the standard are RSA, Diffie-Hellman and ElGamal. The standard will also have sections on random number generation and hardware support for public-key cryptography.

ISO/IEC JTC 1/WG 2/SC 27 (Security Techniques — Techniques and Mechanisms) is also drafting a standard for cryptography based on elliptic curves.

For more information on P1363, contact the working group's chair, Burt Kaliski of RSA Laboratories. Meeting notices and draft materials are available by anonymous ftp to ftp.rsa.com in the pub/p1363 directory. The next meeting of P1363 will review the elliptic curve material and is scheduled for August 31-September 1 at the University of California, Santa Barbara, following the CRYPTO '95 conference.

S/MIME Standardized

A group of leading networking and messaging vendors, in conjunction with RSA Data Security, Inc., recently endorsed a specification that will enable encrypted messages to be exchanged between E-mail applications from different vendors. Vendors participating in the announcement included ConnectSoft, Frontier, FTP Software, Qualcomm, Microsoft, Lotus, Wollongong, Banyan, NCD, SecureWare, and VeriSign.

The specification — Secure/Multipurpose Internet Mail Extensions (S/MIME) — is based on the popular Internet MIME standard (RFC 1521) with the secure portion of the message defined by the public key cryptography standards PKCS #7 and PKCS #10.

Developers interested in S/MIME can get more information in "What's New" on RSA's web page (http://www.rsa.com).

The Future of Integer Factorization

Andrew M. Odlyzko
AT&T Bell Laboratories
Murray Hill, NJ 07974 USA

Introduction

How large should be the moduli in public key cryptosystems such as RSA and DSS (the US Digital Signature Standard)? The answer depends on the anticipated threats. Even if those are known, there is no way to provide a definitive answer, since progress in integer factorization and discrete logarithm algorithms is not predictable. (Furthermore, there is still the possibility that RSA and DSS could be broken by other methods than factoring or discrete log approaches.) However, since choices of moduli have to be made, it is necessary to make some estimates, and this note attempts to do so, taking into account both the increase in available computing power and future algorithmic developments. The projections made below suggest that RSA and DSS moduli might have to be unpleasantly large. In particular, the 512-bit moduli that have been adopted in a variety of cryptosystems are already unsafe for all applications except those with very modest security requirements.

I would like to stress that while the assertion about insecurity of 512-bit moduli is easy to support with solid technical evidence, the projections about the future are much less certain, and should not be treated as firm forecasts, but as possible ways that computational number theory might develop.

Only conventional algorithms and standard integrated circuit technologies will be considered. If either quantum computers (à la Shor) or DNA computers (à la Adleman) become practical, the outlook might change, and even larger moduli might become necessary.

Computing power will be measured in units of MY, or mips-years. By convention, a 1 mips machine is equivalent to the DEC VAX 11/780 in computing power, and so 1 MY is one year on a VAX 11/780. This measure has many defects, and nowadays a wide variety of other benchmarks are used in preference to mips measures. Still, given the uncertainty in any projection far into the future, this measure seems adequate.

43D will refer to an integer of 43 decimal digits.

Discussion will be restricted to integer factorization. Discrete logarithms are, with the present state of knowledge, slightly more difficult to compute modulo an appropriately chosen prime than it is to factor a "hard" integer of the same size, but the difference is not large [11]. Therefore to be on the safe side in designing cryptosystems, one should assume that all the projections about sizes of integers that it will be possible to factor will also apply to sizes of primes modulo which one can compute discrete logarithms.

Factorization records and historical estimates

There is a long record (see Appendix A) of estimates of sizes of integers that could be factored. They have uniformly turned out to be too low, primarily because of unanticipated algorithmic improvements. Faster than expected growth in available computing resources also played a role, though. Table 1 summarizes the progress that has occurred in the last few decades. Table 2 shows how much the computing power available for integer factorizations has increased.

Table 1 (see Appendix B)

Historical records in integer factorization	
year	**record factorizations**
1964	20D
1974	45D
1984	71D
1994	129D

Table 2 (see Appendix C)

Computing power used to achieve record factorizations	
year	**MY**
1974	0.001
1984	0.1
1994	5000

Projections of computing power available in the future

The dramatic increase in computing power used for factorization between 1984 and 1994 resulted largely

There is a long record of estimates of sizes of integers that could be factored. They have uniformly turned out to be too low.

Andrew M. Odlyzko is Head of the Mathematics of Communication and Computer Systems Department at AT&T Bell Laboratories in Murray Hill, New Jersey. His interests include computational complexity, cryptography, number theory, combinatorics, coding theory, analysis, and probability theory. He can be contacted at amo@research.att.com.

from the introduction of distributed computing, using the idle time on a network of workstations. This trend was started by Bob Silverman. (An earlier instance was Richard Schroeppel's attempt to factor F_8, but that work did not attract much public attention.) It was fully developed by Arjen Lenstra and Mark Manasse. The RSA129 factorization used idle time on around 1600 computers around the world during an 8 month period. Most modern factoring methods do lend themselves to distributed implementations.

In the remainder of this discussion, we will only consider factoring attempts similar to that on RSA129, namely ones that use idle time on networks of computers. Unlike attacks on DES, where effective attacks appear to require a special purpose machine (see Appendix J), these attacks do not require any major commitment of financial or technical resources, and can even be mounted surreptitiously.

Another projection of the future of integer factorization has been made by Ron Rivest [14]. Rivest's expectations for progress in computer technology and algorithms do not differ much from mine, but he considers what can be done with a fixed amount of money

computers in 2004, and up to 10^6 in 2014. Thus they might have 10^8 MY at their disposal in 2004, and 10^{10} - 10^{11} in 2014. (Would they attempt to attack a single cryptographic system, or would they attempt to attack 1000 different systems? This would likely depend on what moduli are used, and on the potential payoff.)

We summarize the projections above in Table 4.

Table 4

Computing power available for integer factorization		
year	**covert attack**	**open project**
2004	10^8 MY	$2 \cdot 10^9$ MY
2014	10^{10} - 10^{11} MY	10^{11} - 10^{13} MY

Factorization using current algorithms

Of the methods that are currently known and apply to generally hard integers, the general number field sieve (gnfs) has the best asymptotic running time estimate. It is also practical, and runs faster than previous algorithms on generally hard integers of more than about 115D. Since a 119D integer was factored recently using gnfs with 250 MY (see Appendix C), we can project, using standard methods (see Appendix H), the amount of computing that is likely to be required to factor various integers. The results are shown in Table 5.

Table 5

Computing power required to factor integers with current version of gnfs	
bits of n	**MY required**
512	$3 \cdot 10^4$
768	$2 \cdot 10^8$
1024	$3 \cdot 10^{11}$
1280	$1 \cdot 10^{14}$
1536	$3 \cdot 10^{16}$
2048	$3 \cdot 10^{20}$

Thus, based on tables 4 and 5, moduli of 1280 bits are likely to be safe for well over 20 years, and even 1024-bit moduli are not likely to be vulnerable, unless they conceal extremely valuable information. However, Table 5 assumes that the current version of gnfs will remain the best algorithm. This would be an extremely imprudent assumption, as history shows that algorithmic improvements are often crucial.

It is important to note that 512-bit integers, which are used in a variety of commercial implementations of RSA, can already be factored with the available computing power. There is no need for new algorithms or faster or more computers, just the effort to find enough people willing to let their workstations be used in their idle time. The machines at Silicon Graphics alone could factor a single 512-bit integer in about half a year total time (under the assumption that they are idle about two thirds of the time).

Algorithmic improvements

The special number field sieve (snfs) applies to integers such as the Fermat numbers. Based on the recent factorization of a 162D special integer by Boender et al. in about 200 MY, we can estimate how long snfs takes to factor various integers, and the results are presented in Table 6.

Table 6

Computing power required to factor integers with the snfs	
bits of n	**MY required**
768	$1 \cdot 10^5$
1024	$3 \cdot 10^7$
1280	$3 \cdot 10^9$
1536	$2 \cdot 10^{11}$
2048	$4 \cdot 10^{14}$

In particular, it appears that it might be possible to factor $F_{10} = 2^{1024} + 1$ by the year 2000 (continuing the tradition in which F_7 was factored in 1970, F_8 in 1980, and F_9 in 1990).

Is it reasonable to expect a breakthrough that would enable generally hard integers to be factored in about the same time that the snfs factors integers of comparable size? I feel that it is prudent to expect even larger improvements. It is impossible to predict scientific breakthroughs. However, just as in other disciplines (cf. "Moore's Law"), one can observe a steady progress in integer factoring. Every few years a new algorithm appears that allows for factoring much larger integers, and then there is a steady stream of incremental improvements. As one example, there were wide concerns that the linear algebra step that plays a crucial role in most of the fast integer factorization algorithms might become a serious bottleneck when factoring large integers, since it could not be executed easily in a distributed way. However, new algorithms were developed in the last decade that have allayed these concerns.

Is it reasonable to expect a breakthrough that would enable generally hard integers to be factored in about the same time that the snfs factors integers of comparable size?

To factor a 129D integer with the continued fraction method that was used in the 1970s would have required about $6 \cdot 10^{11}$ times more computing power than to factor a 45D integer, but the factorization took only about $5 \cdot 10^{6}$ times as much because a much better algorithm was used. Thus here the algorithmic improvement was comparable (on a logarithmic scale) to the hardware one. This is similar to what has been reported in other areas, such as numerical analysis, where again better mathematical ideas have contributed about as much as faster computers.

Let us assume that algorithmic improvements will continue to be comparable to those from increasing computing power. Then we can expect that even a covert attack in 2004, which with present algorithms could only factor about a 768-bit integer, will instead be able to factor a 1024-bit one. For the year 2014, the threshold of what might be achievable rises to 1500 bits or even more.

Conclusions

For many applications, the conjectured progress in factorization is not a serious threat. For example, for digital signatures, time-stamping (à la Haber and Stornetta) provides a way to maintain their validity (although in a somewhat cumbersome way, requiring recourse to a document trail) even as secret moduli are factored. (This assumes, or course, that there are no totally unexpected breakthroughs, so that it is possible to estimate at any given time what are the largest integers that might be factorable in the next year, say.) Also, for many records, the security requirements are not serious, in that loss of secrecy after 10 years or sometimes even 10 days is acceptable. Hardware improvements favor the cipher designer, since a 100-fold speedup in commonly used processors allows for a 10-fold increase in the modulus in DSS (assuming, as seems reasonable, that the auxiliary prime q stays at 160 bits, or is increased to at most 240 bits), for example. However, for some records, even 20-year protection is not sufficient. In those cases extremely large moduli appear to be required for many of the most popular public key systems, such as RSA and the various ElGamal-type algorithms, such as the DSA. For extremely sensitive information, it might sometimes be prudent to use 10,000-bit moduli.

The main reason that the projections for lengths of safe moduli are growing so fast is that the asymptotic running time estimates for the latest factoring algorithms are subexponential. In particular, the growth rate for the running time of the number field sieve is not fast. By comparison, there are problems where the only known algorithms have exponential running times, such as DES and related ciphers, and also many public key schemes on elliptic curves (see Appendix I). It might therefore be prudent to consider even more seriously elliptic curve cryptosystems.

Acknowledgements

be surprised if anyone regularly factors numbers of size 10^{80} without special form during the present century." He was also shown to be too cautious in a few years. The choice of a 129D integer for the RSA challenge number in 1977 [5] was apparently motivated by an estimate of Ron Rivest that to factor such an integer would require more than "40 quadrillion years" on a computer much faster than any that exist even today. However, this challenge integer was factored in 1994.

Appendix B: Historical factorization records

In 1964, there was practically no organized activity in factoring integers, and so this entry is based on the remarks in [2].

The largest generally hard integer that had been factored by 1974 was F_7, the 7-th Fermat number, $2^{128} + 1$, which is 39D. (Today it would not be classified as "generally hard," since the special number field sieve handles numbers of this type much more efficiently than general ones. Also, as another technical point, the integer that was factored was $257 \cdot F_7$, since the use of a multiplier speeded up the algorithm.) F_7 had been factored in 1970 by Mike Morrison and John Brillhart, but their paper describing this work was only published in the 1975 D. H. Lehmer special issue of *Math. Comp.* In those days, integer factorization was not fashionable, and there was not much interest in going after records. Therefore inspection of the published literature is not adequate to assess the state of the art. Experts who were active in this area in the 1970s say that they thought that numbers of 45D were doable with the algorithms and machines available then, and so 45D is the figure used here.

1984: This is the Sandia factorization of Jim Davis, Diane Holdridge, and Gus Simmons.

1994: This is RSA129, the RSA challenge integer of 1977, which was factored by a world-wide collaborative effort organized and led by Derek Atkins, Michael Graff, Arjen Lenstra, and Paul Leyland [1].

If we also consider integers of a special form, then the 162D integer $(12^{151} - 1) / 11$ provides another record.

Peter Montgomery, Herman te Riele, Robby Robson, Russell Ruby, and Dik Winter.

Appendix C: Computing power of historical factorizations

According to John Brillhart, integer factorizations in the early 1970s usually used up only about an hour of cpu time on the mainframes of those days, which are typically rated at 1-10 mips.

The 1984 Sandia factorization used 9.5 cpu hours on a Cray X-MP, which is on the order of 100 mips.

1994: This is the Atkins et al. estimate [1] for the computation, which used the ppmpqs algorithm. Since then a 119D integer has been factored by Scott Contini, Bruce Dodson, Arjen Lenstra, and Peter Montgomery in about 250 MY using gnfs (the general number field sieve), suggesting that today the RSA129 factorization could be carried out in about 1000 MY instead of the 5000 MY that was used. See [3] for details.

Appendix D: Current computing power

The oft-quoted figure of 30M users of the Internet is questionable, as it is obtained by assuming there are 10 users per computer. However, the estimate that about 3M machines of one kind or another are hooked up to the Internet seems much more solid. Since many of them are old, an average 10 mips rating seems reasonable for them.

There are well over 10^8 PCs in the world. Since several tens of millions of them already have the 486 chips, which are usually rated at tens of mips, the estimate of $3 \cdot 10^8$ mips is conservative. On the other hand, most of these machines are not easily usable for factoring, since they are not networked, do not have enough memory, do not have operating systems that allow for background jobs to run easily in idle time, etc. All these factors will change in a few years, but now it would be hard to harness a large fraction of all the PCs in a factoring project.

We might note that all the supercomputers in the world (about 10^3 in total) have a computing power of about $3 \cdot 10^6$ mips, with several sites having between 5% and 10% of of that capacity. (This estimate equates 1 megaflop with 1 mips, which is not very accurate, and is based on the June 1995 version of the TOP500 report [4].) IBM mainframes shipped in 1994 (which was a record year in terms of mainframe computing power, although not in dollar volume of sales) amounted to only $2 \cdot 10^5$ mips.

Appendix E: Moore's "Law"

There is some skepticism whether this law will continue to hold much longer. Line widths will eventually be so small that entire new technologies might be needed, architectural improvements (speculative execution, etc.) might run into the exponential complexity blowup, and so on. Even if the bare technologies are not a barrier, economics might present one, since the costs of state of the art fabrication facilities are also escalating. However, such concerns are not new, and were already present a decade ago, and yet progress has continued unhindered. Thus it seems imprudent to assume Moore's "Law" will be violated any time soon. Since state of the art microprocessors are already running close to 500 mips, the projection that by 2004 the typical processor will have a rating of 1000 mips (compared to around 10 mips today) seems safe. Beyond that, there are bigger question marks. Even if raw processor speed continues to increase, memories might be more of a bottleneck. Since most fast factorization algorithms

than the Cray-1, the first supercomputer. The new 32-bit video game machines, of which tens of millions are expected to be sold each year, will have similar power.) We might note, as a forerunner of what will be common, that there were two fax machines among the approximately 1600 processors in the RSA129 project.

There is still a question, even if we assume that there will be many computers around, of whether they will all be networked together, and whether their computing power will be easily accessible. Will there be computing-power brokers, selling time on millions of machines? Will people be willing to tolerate somebody else running jobs on their machines, even in spare time?

Appendix G: Running time of algorithms

In a variant of the standard notation, we define

$$L[n, v, a] = \exp(a \cdot (\log n)^{v} \cdot (\log \log n)^{(1-v)}),$$

where $\log n$ refers to the natural logarithm of n.

The heuristic running time (there are no rigorous proofs, but experience and heuristics support the calculations) of the gnfs to factor an integer n is

$$L[n, 1/3, c_0 + o(1)] \text{ as } n \to \infty,$$

where $c_0 = (64/9)^{1/3} = 1.9229\ldots$. There is also a variant, due to Don Coppersmith, which does not seem to be practical, that allows the replacement of c_0 by $c_1 = 1.9018\ldots$. See [8,12] for presentations of the gnfs.

The special number field sieve, which factors efficiently integers of the form $a^k \pm b$, for example, where a and b are small, and k large (k can also be small, but then has to fall into certain ranges) has running time

$$L[n, 1/3, c_2 + o(1)] \text{ as } n \to \infty,$$

where $c_2 = (32/9)^{1/3} = 1.5262\ldots$.

Previous methods, such as variants of the quadratic sieve algorithm, have running times of the form

$$L[n, 1/2, 1 + o(1)] \text{ as } n \to \infty.$$

The continued fraction method, which was the most widely used method in the 1970s, appears (at least for the most common variant) to have running time

$$L[n, 1/2, c_3 + o(1)] \text{ as } n \to \infty,$$

where $c_3 = 2^{1/2} = 1.4142\ldots$.

Appendix H: Comparison of running times of algorithms

The $o(1)$ terms in the estimates of running times of factorization algorithms are usually not computed explicitly. Instead, to estimate how long an algorithm with asymptotic running time $L[n, v, a + o(1)]$ should take to factor an integer n, one takes the observed running time X on an integer m and computes $X \cdot L[n, v, a] / L[m, v, a]$. This has worked well in practice. This method does ignore various important practical aspects, such as the need for memory and communication capacity as well as cpu cycles, but those have been overcome in the past either through better algorithms or general technological improvements, so it seems reasonable to continue to ignore them.

Appendix I: Exponential algorithms

Public key cryptosystems such as RSA and DSA require use of large moduli because known general algorithms for factoring integers and computing discrete logarithms are subexponential, and so hardware improvements by themselves lead to substantial progress. In addition, there have been steady and rapid improvements in algorithms. On the other hand, there are some problems in which the best algorithms known are exponential, and where there has been no recent algorithmic improvement. We cite here as an example attacks on DSA, the US Digital Signature Algorithm, that do not use the structure of the multiplicative group of integers modulo the large basic prime p. The DSA uses discrete exponentiation modulo a prime p, but the exponents are computed modulo a prime q such that q divides $p - 1$. (This is the Schnorr method that speeds up the algorithm.) Therefore all the integers are inside an abelian group of order q. To break DSA, one can either solve the general discrete log problem modulo p, which can be done using a variant of the number field

There are some problems in which the best algorithms known are exponential, and where there has been no recent algorithmic improvement

sieve, or else work inside the group of order q. The best algorithms for the second attack are those of Dan Shanks and John Pollard, both of which take about $\sqrt{q}$ operations. Since these $\sqrt{q}$ operations involve multiplications modulo p, though, even for q of 160 bits and p of 1024 bits or more, it would take at least 10^{14} MY on general purpose computers to implement either the Shanks or the Pollard algorithm. Hence we can conclude that 160 bits for q is likely to be safe for at least 20 years, and 200 bits (which would require at least 10^6 as much computing power to break) for much longer. (As with all the other projections about algorithms, this one could turn out to be faulty if a breakthrough occurs.)

To find a single DES key will on average take about 300 times as much as the factorization of RSA129 required.

As another example where only exponential attacks are known, we can cite elliptic curve cryptosystems [9], proposed initially by Neal Koblitz and Victor Miller. If the elliptic curve is chosen carefully, only the Shanks and Pollard methods are known for computing the analog of discrete logs on these curves. There is still some reluctance to use elliptic curve cryptosystems, though, since they have not been scrutinized as carefully as integer factorization and ordinary discrete logs.

Appendix J: Attacks on DES

For comparison, we present some estimates of the computing power needed to break DES. We consider known plaintext attacks on cipher codebook mode. We assume that only exhaustive key search will be tried, so that on average 2^{55} keys will have to be tested to find the right one. The best software implementations (written in C) appear to achieve rates of up to about 200 KB/sec on 25 mips machines (such as 50 MHz 80486 PCs), which (since each iteration of DES involves encrypting 8 bytes) corresponds to 25,000 encryptions per second, or about 1,000 encryptions per second on a 1 mips computer. Hence 1 MY allows us to test about $3 \cdot 10^{10}$ encryptions. Therefore to find a single DES key will on average take $1.2 \cdot 10^6$ MY, or about 300 times as much as the factorization of RSA129 required.

DES was made to run fast in hardware, and special purpose machines can provide substantial assistance in breaking it. Michael Wiener [15] has proposed a pipelined parallel computer that could be built for about $1.5M (both development and construction cost) and would find a single DES key in about 4 hours. What this shows is that special purpose hardware can be of great help in breaking DES. On the other hand, general integer factorization and discrete logarithm algorithms do not benefit that much from special designs, and it is the threat of the free computing power on the Internet that seems most serious. Special designs for sieve processors for factoring have been proposed (see [13]), but the economics of the electronics industry favors general purpose computers. (Fast parallel modular multiplication units could be of use in the implementation of the Pollard and Shanks exponential-time algorithms, though, or of the subexponential-time elliptic curve method.)

RSA Laboratories Minimum Key Size Recommendations

It is clear from articles like Andrew Odlyzko's in this issue of *CryptoBytes*, that considerable thought is required in choosing the size of a modulus used in an implementation of RSA. Balancing the issues of security against performance is difficult with any cryptosystem, but it is particularly difficult when one wants to make allowances for potential cryptanalytic developments.

Currently, RSA Laboratories make the following recommendations for the size of RSA moduli:

User keys, short-term security	768 bits
Organizational keys, medium-term security	1024 bits
Root keys, long-term security	2048 bits

Since developments in factoring can be very unpredictable, implementations of RSA should ideally allow for variable keysizes where at all possible.

On the Security of the RC5 Encryption Algorithm

Burt Kaliski and **Yiqun Lisa Yin**
RSA Laboratories
100 Marine Parkway, Suite 500
Redwood City, CA 94065 USA

In this article, we give a brief report on the security of the RC5 encryption algorithm [5] against three different types of attack including exhaustive search, differential cryptanalysis [1], and linear cryptanalysis [3]. RC5 is a new block cipher recently designed by Ron Rivest. It has a variable block size, a variable number of rounds, and a variable-length secret key. The secret key is used to fill an expanded key table which is then used in encryption. A detailed description of the RC5 encryption algorithm was provided in the Spring issue of *CryptoBytes* [6].

To attack RC5, we can try to find either the *original secret key* or the *expanded key table*. Clearly, if the latter approach is used, the attack is independent of the length of the secret key.

The secret key used in RC5 has a variable length with allowed values from 0 to 2,040 bits and the expanded key table for RC5 with r rounds has $2^{5(2r+2)}$ bits (for the 64-bit block size). Hence, if both the length of the secret key and the number of rounds are sufficiently large, RC5 is secure against exhaustive search.

Differential and linear cryptanalysis are two powerful techniques developed in recent years for analyzing the security of block ciphers. For differential cryptanalysis, the basic idea is that two chosen plaintexts P and P^* with a certain difference $P' = P \geq P^*$ provide two ciphertexts C and C^* such that $C' = C \geq C^*$ has a specific value with non-negligible probability; such a "characteristic" (P', C') is useful in deriving certain bits of the key. For linear cryptanalysis, the basic idea is to find linear approximations (parity relations among certain bits of plaintext, ciphertext, and key) which hold with probability $p \neq 1/2$ (i.e., bias $= |p - 1/2| \neq 0$); such approximations can be used to obtain information about the key.

Burt Kaliski is chief scientist and Yiqun Lisa Yin is a research scientist at RSA Laboratories. They can be contacted at burt@rsa.com and yiqun@rsa.com.

We have developed differential and linear attacks [2] on RC5 that are quite effective when the number of rounds is very small. Both attacks recover every bit of the expanded key table. However, the plaintext requirement is strongly dependent on the number of rounds, and the requirement for RC5 with 64-bit block size is summarized in the following table.

Table 1

Plaintext Requirements							
rounds	4	5	6	7	9	12	13
differential cryptanalysis	2^{22}	2^{26}	2^{32}	2^{37}	2^{46}	2^{63}	$>2^{64}$
linear cryptanalysis	2^{37}	2^{47}	2^{57}	$>2^{64}$			

The chosen plaintext requirements for differential cryptanalysis of 64-bit RC5 with the indicated number of rounds are shown in the first row of the table. The known plaintext requirements for linear cryptanalysis are shown in the second row.

One can see that for the 64-bit block size, our differential attack on nine-round RC5 uses 2^{46} chosen plaintexts (about the same as DES [4]); the plaintext requirement becomes increasingly impractical for more rounds. Similarly, our linear attack on five-round RC5 uses 2^{47} known plaintexts (about the same as DES) and the plaintext requirement increases rapidly with additional rounds.

We now briefly describe the idea used in the two attacks on RC5. Since RC5 is iterative, if we can derive the subkey in the last round, we can derive all the subkeys in the expanded key table. In our differential attack, the characteristics for each round have the property that the differences in the pair of inputs do not affect the rotation amounts and they can easily be joined together. Moreover, the characteristics for the last round have a special form, allowing us to recover the subkey in the last round. In our linear attack, the linear approximations relate the least significant bits of the input, the output, and the subkey in each round. When enough plaintext/ciphertext pairs are obtained, the experimental biases of the approximations reveal the subkey in the last round. There is strong evidence that the characteristics and the linear approximations used in our attacks are close to optimal. Thus, Rivest's suggested use of 12 rounds for RC5 with a 64-bit block size is sufficient to make differential and linear cryptanalysis impractical.

Rivest's suggested use of 12 rounds for RC5 with a 64-bit block size is sufficient to make differential and linear cryptanalysis impractical.

Our analysis shows that data-dependent rotations are very helpful for preventing differential and linear attacks.

In sum, we can conclude that RC5 provides good security against all three types of attacks when both the length of the secret key and the number of rounds are sufficiently large. Unlike DES, RC5 is a parameterized algorithm, giving flexibility in the level of security. As a next step, we will consider other possible forms of analysis and study the key expansion algorithm of RC5.

Finally, we want to point out two distinguishing features of RC5. The first feature is the heavy use of *data-dependent* rotations. Our analysis shows that data-dependent rotations are very helpful for preventing differential and linear attacks. The second feature is the exceptional simplicity of the encryption algorithm. Such a simple design makes the analysis easier and will help fully determine the security of RC5 in a rather rapid way.

References

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

[2] B.S. Kaliski and Y.L. Yin. On differential and linear cryptanalysis of the RC5 encryption algorithm. To appear at Crypto '95.

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

[4] National Institute of Standards and Technology (NIST). *FIPS Publication 46-2: Data Encryption Standard*. December 30, 1993.

[5] R.L. Rivest. The RC5 encryption algorithm. In *Proceedings of the Workshop on Cryptographic Algorithms*, K.U. Leuven, December 1994. To appear.

[6] R.L. Rivest. The RC5 encryption algorithm. *CryptoBytes*, 1(1), 9-11, Spring 1995.

ALGORITHMS UPDATE

MD5 Performance for IP Security Questioned

A paper to be presented at ACM SigComm '95 questions whether the MD5 message-digest algorithm is fast enough to handle packets on high-speed Internet links.

The paper, "Performance Analysis of MD5" by Joe Touch of ISI, argues that with today's hardware MD5 can achieve at best a speed of 256 million bits/second (Mbps), not enough to keep up with Asynchronous Transfer Mode (ATM) data rates of up to 622 Mbps. Though hardware performance will likely improve over time, so will network requirements, keeping MD5 behind.

MD5 has been proposed as a basis for authenticating messages in version 6 of the Internet Protocol (IPv6), where a message and a secret key are hashed together to form an authentication code that prevents intruders from modifying messages. As such, it could be required to handle packets at network speeds.

A significant reason for the limitation in performance is that MD5 has an iterative design where message blocks are processed one after another, each one being combined with the result of previous computation through a compression function. Thus the only hardware optimization possible is within the compression function; it is not possible to process multiple blocks at the same time with additional hardware. Moreover, the compression function itself is complex and somewhat hard to parallelize. Many other hash algorithms including NIST's Secure Hash Algorithm (SHA) are subject to the same limitations.

Alternatives recommended by Touch include tree-based hashing; and internal modifications to the algorithm.

MD5 remains sufficient for many applications, such as hashing as part of a digital signature algorithm. Indeed, Touch reports software performance for MD5 ranging from 31 Mbps on a 66MHz Intel '486 to 87 Mbps on a 190 MHz DEC Alpha. On such processors, files even as large as 10 Mbytes can be hashed with MD5 in only a few seconds.

A report by Touch summarizing these results is available as Internet RFC 1810.

Further Comments on Keyed MD5

This note follows on the excellent overview by Burt Kaliski and Matt Robshaw ("Message authentication with MD5," *CryptoBytes* vol.1 no.1) in which three schemes are recommended to the IPSEC working group. Citing forthcoming work, it was suggested the best attack (forgery) on these schemes required 2^{64} chosen message texts (" ... except when the known messages are all the same length and end with the same suffix").

We have improved this attack in our recent paper(*) "MDx-MAC and building fast MACs from hash functions," *Proc. Crypto'95.* A generic attack is given requiring 2^{64} known text-MAC pairs and a single chosen text, independent of message lengths or suffixes — only for the second recommended scheme we need the messages to be of the same length. (Perhaps more significantly, the attack applied to CBC-MAC requires only 2^{32} known text-MAC pairs for MAC forgery.) The attack requires an additional 2^{64} *chosen* text-MAC pairs if only 64 bits of the hash result are retained; this suggests modifying the method to retain only 64 bits, which also saves bandwidth. The number of text-MAC pairs required can be further reduced if the known messages contain a common (not necessarily chosen) sequence of trailing blocks. The attack also applies if messages are fixed-length or prepended by length fields.

Adapting the same attack strategy allows a divide-and-conquer attack if the envelope method is used with distinct front and tail keys, effectively reducing security to the larger of the two. We also provide analysis of the secret prefix and secret suffix methods, and add here that the secret suffix method is subject to an off-line, memoryless, parallelizable attack requiring 2^{64} operations and a single chosen text (P. van Oorschot and M. Wiener, ACM-CCS'94, Fairfax).

Recent partial attacks on MD4, MD5, and the related RIPEMD, including in particular those of S. Vaudenay (Leuven Algorithms Workshop Dec.'94) and H. Dobbertin (Rump Session, Eurocrypt'95), suggest these functions are susceptible to manipulations of their internal structures. This raises concerns about hash-based MACs being susceptible to attacks exploiting properties of the underlying hash. We therefore advise caution in constructing such MACs, and recommend a design more conservative than the envelope method. We agree customized MACs may be preferable, but are reluctant to discard the experience gained over time with MD4 and MD5.

With exquisite timing, our paper already (as submitted Feb.'95) makes a proposal in line with most of the suggestions of Kaliski and Robshaw: MD5-MAC, a customized MAC involving key processing at every compression function step, and built with only minor modifications from MD5 (to minimize the likelihood of introducing new flaws). The same construction yields MACs based on any of MD5, SHA, or RIPEMD.

In addition to being more conservative than the envelope method, only slightly slower (5-20%, depending on processor and implementation), and easily implemented from MD5, the theoretical underpinnings supporting the security of the envelope method, which assume the compression function of MD5 is pseudorandom, appear to similarly apply to MD5-MAC. We caution, however, that we are aware of no results regarding the pseudorandomness of MD5, and note this property may be independent of collision-resistance, the primary property studied to date.

— *Bart Preneel, Katholieke Universiteit Leuven*
bart.preneel@esat.kuleuven.ac.be
Paul C. van Oorschot, Bell-Northern Research
paulv@bnr.ca

(*) the paper is available by FTP at ftp.esat.kuleuven.ac.be, in the directory pub/COSIC/preneel.

We welcome comments from our readers at any time. In particular we are keen to expand and keep up-to-date issues that have previously appeared in CryptoBytes. Comments can be sent via E-mail to bytes-ed@rsa.com.

We therefore advise caution [...], and recommend a design more conservative than the envelope method.

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.

Technical reports that are currently available include recently updated surveys of block ciphers, stream ciphers and the MD family of hash functions. Other reports offer substantial information on such topics as the RSA Factoring Challenge, fair cryptography and the software implementation of RSA. In immediate preparation are reports on the hardware implementations of RSA, an overview of electronic payment systems and an internal assessment of the security of the new RC5 encryption algorithm.

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

The 1996 RSA Data Security Conference

The 1996 RSA Data Security Conference will be held January 17-19 in the Fairmont Hotel, San Francisco. In addition to daily keynote speeches, the conference will include separate strategic, development and cryptography tracks which the expected 800-1000 attendees will be able to mix and match at will. More information can be found on RSA's web page (http://www.rsa.com) or by contacting the conference organizer, Layne Kaplan Events, at 415/340-9300.

In this issue:

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