VOLUME 3, NUMBER 2 — AUTUMN 1997

RSA LABORATORIES'

CryptoBytes

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

Contents

1
On the Foundations of Modern Cryptography

2
Editor's Note

6
Efficient DES Key Search — An Update

8
The RSA Data Security DES Challenge II

9
The Cryptographic Hash Function RIPEMD-160

15
PKCS: The Next Generation, Chapter 2

16
Announcements

On the Foundations of Modern Cryptography

Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel

In our opinion, the Foundations of Cryptography are the paradigms, approaches and techniques used to conceptualize, define and provide solutions to natural cryptographic problems. We survey some of these paradigms, approaches and techniques as well as some of the fundamental results obtained using them. Special effort is made in attempt to dissolve common misconceptions regarding these paradigms and results.

> **It is possible to build a cabin with no foundations, but not a lasting building.**
>
> *– Eng. Isidor Goldreich (1906-1995)*

Cryptography is concerned with the construction of schemes which are robust against malicious attempts to make these schemes deviate from their prescribed functionality. Given a desired functionality, a cryptographer should design a scheme which not only satisfies the desired functionality under "normal operation," but also maintains this functionality in face of adversarial attempts which are devised after the cryptographer has completed his/her work.

The fact that an adversary will devise its attack after the scheme has been specified makes the design of such schemes very hard. In particular, the adversary will try to take actions other than the ones the designer had envisioned. Thus, our approach is that it makes little sense to make assumptions regarding the specific *strategy* that the adversary may use.

The only assumptions which can be justified refer to the computational *abilities* of the adversary. Furthermore, it is our opinion that the design of cryptographic systems has to be based on *firm foundations*; whereas ad-hoc approaches and heuristics are a very dangerous way to go. A heuristic may make sense when the designer has a very good idea about the environment in which a scheme is to operate, yet a cryptographic scheme has to operate in a maliciously selected environment which typically transcends the designer's view.

Providing firm foundations to Cryptography has been a major research direction in the last two decades. Indeed, the pioneering paper of Diffie and Hellman [8] should be considered the initiator of this direction. Two major (interleaved) activities have been:

1) **Definitional Activity:** The identification, conceptualization and rigorous definition of cryptographic tasks which capture natural security concerns; and
2) **Constructive Activity:** The study and design

(continued on page 3)

Oded Goldreich is currently visiting the Laboratory for Computer Science of MIT, and is partially supported by DARPA grant DABT63-96-C-0018. He can be contacted via e-mail at oded@wisdom.weizmann.ac.il.

RSA Laboratories
A Division of RSA Data Security

Editor's Note

As years come to a close it is natural to look back at past achievements. Some achievements in 1997 caught the attention of the popular press, but other arguably more significant developments went unnoticed outside of the research community.

We encourage any readers with comments, opposite opinions, suggestions or proposals for future issues to contact the CryptoBytes *editor.*

The RSA sponsored secret-key challenges very publicly demonstrated a valuable point. However, the challenges provided little surprise to cryptographers. In fact many believe that the software-based search efforts characterized by the challenges do not pose the main threat. In 1993 Michael Wiener described a design for a dedicated machine that could be built for $1,000,000 and which would find a DES key in 3.5 hours on average. To many, this is the real threat. In this issue Michael updates those estimates to allow for progress over the intervening four years. Those that take comfort in the time required to find a DES key by software-based exhaustive search might find this article particularly interesting.

Perhaps the most remarkable cryptanalytic developments over the last year or two have been the advances made in the analysis of hash functions by Hans Dobbertin. The net result of this work has been a lack of options in the hash functions that are available for long-term use. In Europe, however, RIPEMD-160 has been gaining in popularity and the designers of this algorithm provide us with a summary of its features in this issue of the newsletter.

Finally, at the 1997 *Crypto* conference attendees honored the work of Oded Goldreich. As one of the pioneers in establishing a theoretical framework to today's cryptography Oded's invited lecture was one of the highlights of the conference. In our lead article Oded provides us with his perspective on the foundations of modern cryptography.

The future success of *CryptoBytes* depends on input from all sectors of the cryptographic community, and as usual we would like very much to thank the writers who have contributed to this second issue of the third volume. 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*.

About RSA Laboratories

An academic environment within a commercial organization, RSA Laboratories is the research and consulting division of RSA Data Security, the company founded by the inventors of the RSA public-key cryptosystem. Its purpose is to provide state-of-the-art expertise on cryptography and information security for the benefit of RSA Data Security and its customers. RSA Data Security is a Security Dynamics company.

CryptoBytes Feedback Requested

The *CryptoBytes* publication schedule is currently being revised. As part of that process, we are interested to know if readers have a preference for electronic or paper versions of the newsletter. Please direct any comments via e-mail to *rsa-labs@rsa.com*. Also, to help with the distribution of the newsletter and to provide updates on the publication schedule, a mailing list will be maintained. To put yourself on this list, send e-mail to *majordomo@rsa.com* with the line "subscribe cryptobytes-information" in the message body. To remove yourself from this list, send e-mail to *majordomo@rsa.com* with the line "unsubscribe cryptobytes-information" in the message body.

Newsletter Availability and Contact Information

CryptoBytes is a free publication and all issues, both current and past, are available via the World-Wide Web at <*http://www.rsa.com/rsalabs/pubs/cryptobytes.html*>.

For each previous issue a limited number of copies were printed. While available, copies of these printed issues can be requested by contacting RSA Laboratories though a nominal fee to cover handling costs may be charged for individual requests.

RSA Laboratories can be contacted at:

RSA Laboratories
100 Marine Parkway, Suite 500
Redwood City, CA 94065
650/595-7703
650/595-4126 (fax)
rsa-labs@rsa.com

Continued from page 1

of cryptographic schemes satisfying definitions as in (1).

The definitional activity

An arcitypical example of this activity is the definition of the most classical of all cryptographic notions — the notion of secure encryption [18]. The reader may be surprised: *what is there to define* (beyond the basic setting formulated in [8])? Let us answer with a question (posed by [18]): *should an encryption scheme which leaks the first bit of the plaintext be considered secure?* Clearly, the answer is negative and so some naive conceptions regarding secure encryption (e.g., "a scheme is secure if it is infeasible to obtain the plaintext from the ciphertext when not given the decryption key") turn out to be unsatisfactory. The lesson is that even when a natural concern (e.g., "secure communication over insecure channels") has been identified, work still needs to be done towards a satisfactory (rigorous) definition of the underlying concept.

The definitional activity also undertook the treatment of unforgeable signature schemes [20]: One result of the treatment was the refutation of a "folklore theorem" (attributed to Ron Rivest) by which "a signature scheme that is robust against chosen message attack cannot have a proof of security". The lesson here is that unclear/unsound formulations (i.e., those underlying the above folklore paradox) lead to false conclusions.

Another existing concept which was re-examined is the then-fuzzy notion of a "pseudorandom generator." Although ad-hoc "pseudorandom generators" which pass some ad-hoc statistical tests may be adequate for some statistical samplings, they are certainly inadequate for use in Cryptography: For example, sequences generated by linear congruential generators are easy to predict and endanger cryptographic applications even when not given in the clear. The alternative suggested in [7,18,28] is a robust notion of pseudorandom generators — such a generator produces sequences which are *computationally indistinguishable* from truly random sequences, and thus, can replace truly random sequences in any practical application. The approach was further extended to pseudorandom functions [14].

The definitional activity has identified concepts which were not known before. One well-known example is the introduction of zero-knowledge proofs [19]. A key paradigm crystallized in making the latter definition is the *simulation paradigm*: A party is said to have gained nothing from some extra information given to it if it can generate (i.e., simulate the receipt of) essentially the same information by itself (i.e., without being given this information). The simulation paradigm plays a central role in the related definitions of secure multi-party computations as well as in different settings.

The definitional activity is an on-going process. Its more recent targets have included mobile adversaries, Electronic Cash, Coercibility, Threshold Cryptography and more.

The constructive activity

As new definitions of cryptographic tasks emerged, the first challenge was to demonstrate that they can be achieved. Thus, the first goal of the constructive activity is to *demonstrate the plausibility* of obtaining certain goals. Thus, standard assumptions such as that the RSA is hard to invert were used to construct secure public-key encryption schemes [18,28] and unforgeable digital schemes [20]. We stress that assuming that RSA is hard to invert is different from assuming that RSA is a secure encryption scheme. Furthermore, plain RSA (alike any deterministic public-key encryption scheme) is not secure (as one can easily distinguish the encryption of one *predetermined* message from the encryption of another). Yet, RSA can be easily transformed into a secure public-key encryption scheme by using a construction which is reminiscent of a common practice (of padding the message with random noise). The resulting scheme is not merely believed to be secure, but rather its security is linked to a much simpler assumption (i.e., the assumption that RSA is hard to invert). Likewise, although plain RSA signing is vulnerable to "existential forgery" (and other attacks), RSA can be transformed into a signature scheme which is unforgeable (provided RSA is hard to invert). Using the assumption that RSA is hard to invert, one can construct pseudorandom generators [7,28], zero-knowledge proofs for any NP-statement [16], and multi-party protocols for securely computing any multi-variant function [29,17].

[...] should an encryption scheme which leaks the first bit of the plaintext be considered secure?

Using the assumption that RSA is hard to invert, one can construct pseudorandom generators, zero-knowledge proofs for any NP-statement, and multi-party protocols for securely computing any multi-variant function.

A major misconception regarding theoretical work in Cryptography stems from not distinguishing work aimed at demonstrating the plausibility of obtaining certain goals from work aimed at suggesting paradigms and/or constructions which can be used in practice. For example, the results concerning zero-knowledge proofs and multi-party protocols [16,29,17] mentioned above are merely *claims of plausibility*: What they say is that any problem of the above type (i.e., any protocol problem) can be solved in principle. This is a very valuable piece of information. Thus, if you have a specific problem which falls into the above category then you should know that the problem is solvable in principle. However, if you need to construct a real system then you should probably construct a solution from scratch (rather than employing the above general results). Typically,

Bibliographic abbreviations

- **STOC** is ACM Symposium on the Theory of Computing.
- **FOCS** is IEEE Symposium on Foundations of Computer Science.

References

[1] M. Bellare, O. Goldreich and S. Goldwasser. Incremental Cryptography: the Case of Hashing and Signing. In *Proceedings of Crypto94*, Springer-Verlag LNCS (Vol. 839), pages 216-233, 1994.

[2] M. Bellare, O. Goldreich and S. Goldwasser. Incremental Cryptography and Application to Virus Protection. In *27th STOC*, pages 45-56, 1995.

[3] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Proceedings of Crypto93*, Springer-Verlag LNCS (Vol. 773), pages 232-249, 1994.

[4] M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: The Three Party Case. In *27th STOC*, pages 57-66, 1995.

[5] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability. In *20th STOC*, pages 113-131, 1988.

[6] M. Blum and S. Goldwasser. An Efficient Probabilistic Public-Key Encryption Scheme which hides all partial information. In *Proceedings of Crypto84*, LNCS (Vol. 196) Springer-Verlag, pages 289-302, 1985.

[7] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. on Comput.*, Vol. 13, pages 850-864, 1984.

[8] W. Diffie, and M.E. Hellman. New Directions in Cryptography. *IEEE Trans. on Info. Theory*, IT-22 (Nov. 1976), pages 644-654.

[9] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. In *23rd STOC*, pages 542-552, 1991. Full version available from authors.

[10] C. Dwork, and M. Naor. An Efficient Existentially Unforgeable Signature Scheme and its Application. To appear in *J. of Crypto*. Preliminary version in *Proceedings of Crypto94*, LNCS (Vol. 839) Springer-Verlag, pages 234-246, 1994.

[11] S. Even, O. Goldreich and S. Micali. On-line/Off-line Digital signatures. *J. of Crypto.*, Vol. 9, 1996, pages 35-67.

[12] A. Fiat. Batch RSA. *J. of Crypto.*, Vol. 10, 1997, pages 75-88.

[13] O. Goldreich. *Foundation of Cryptography - Fragments of a Book*. February 1995. Available from http://theory.lcs.mit.edu/~oded/frag.html.

[14] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *J. of the ACM*, Vol. 33, No. 4, pages 792-807, 1986.

[15] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM J. on Comput.*, Vol. 25, No. 1, February 1996, pages 169-192.

[16] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *J. of the ACM*, Vol. 38, No. 1, pages 691-729, 1991. See also preliminary version in *27th FOCS*, 1986.

[17] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game - A Completeness Theorem for Protocols with Honest Majority. In *19th STOC*, pages 218-229, 1987.

[18] S. Goldwasser and S. Micali. Probabilistic Encryption. *J. of Comp. and Sys. Sci.*, Vol. 28, No. 2, pages 270-299, 1984. See also preliminary version in *14th STOC*, 1982.

[19] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. on Comput.*, Vol. 18, pages 186-208, 1989.

[20] S. Goldwasser, S. Micali, and R.L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. on Comput.*, April 1988, pages 281-308.

[21] R. Impagliazzo and S. Rudich. Limits on the Provable Consequences of One-Way Permutations. In *21st STOC*, pages 44-61, 1989.

[22] R.C. Merkle. Protocols for public key cryptosystems. In *Proc. of the 1980 Symposium on Security and Privacy*.

[23] R.C. Merkle. A Certified Digital Signature Scheme. In *Crypto89*, Springer-Verlag LNCS (Vol. 435), pages 218-238, 1990.

[24] S. Micali. Fair Public-Key Cryptosystems. In *Proceedings of Crypto92*, Springer-Verlag LNCS (Vol. 740), pages 113-138, 1993.

[25] M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Application. In *21st STOC*, pages 33-43, 1989.

[26] M.O. Rabin. Digitalized Signatures. In *Foundations of Secure Computation* (R.A. DeMillo et. al. eds.), Academic Press, 1977.

[27] R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *CACM*, Vol. 21, Feb. 1978, pages 120-126.

[28] A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd FOCS*, pages 80-91, 1982.

[29] A.C. Yao. How to Generate and Exchange Secrets. In *27th FOCS*, pages 162-167, 1986.

Efficient DES Key Search — An Update

Michael J. Wiener
Entrust Technologies
750 Heron Road, Suite E08
Ottawa, Ontario
Canada K1V 1A7

[...] just 21 of these chips would give the same key searching power as the entire set of computers used by the team who solved the DES challenge.

An exciting moment in the history of DES was reached this past June when a group coordinated by Rocke Verser solved RSA Data Security's DES challenge by exhaustive key search on a large number of computers. This result was useful because it served to underscore in a public way how vulnerable DES has become. However, it may also have left the false impression that one cannot do much better than attacking DES in software with a large distributed effort. The design of DES is such that it is fairly slow in software, but is compact and fast when implemented in hardware. As a result, using software to attack DES gives poor performance compared to what can be achieved in hardware. This applies not only to DES, but also to most other block ciphers, attacks on hash functions, and attacks on elliptic curve cryptosystems. Avoiding efficient hardware-based attacks requires the use of algorithms with sufficiently long keys, such as triple-DES, 128-bit RC5 [3], and CAST-128 [1].

In this article we assess the cost of DES key search using hardware methods and examine the effectiveness of some proposed methods for thwarting attacks on DES.

Today's version of the $1 million machine could find a DES key in, on average, about 35 minutes.

Advancing technology

The best known way to attack DES is to simply try all of the possible 56-bit keys until the correct key is found. On average, one expects to go through about half of the key space. In 1993, a design for an exhaustive DES key search machine including a detailed chip design was published [4]. A $1 million version of this machine used 57600 key search chips, each capable of testing 50 million keys per second. Overall, the machine could find a DES key in, on average, three and a half hours.

About four and a half years have passed since this design was completed, and according to Moore's Law, processing speeds should have doubled three times in that period. Of course, estimating in this fashion is a poor substitute for the careful analysis and design effort that went into the earlier design. The original chip design was done in a 0.8 micron CMOS process, and with the geometries available today, it is possible to fit four instances of the original design into the same silicon area. In keeping with the conservative approach to estimates in the 1993 paper, we assume here that the updated key search chip's clock speed would increase to only 75 MHz from the original 50 MHz, making the modern version of the chip six times faster for the same cost. It is interesting to note that just 21 of these chips would give the same key searching power as the entire set of computers used by the team who solved the DES challenge.

Today's version of the $1 million machine could find a DES key in, on average, about 35 minutes (one-sixth of 3.5 hours). This time scales linearly with the amount of money spent as shown in the following table.

Key Search Machine Cost	Expected Search Time
$10,000	2.5 days
$100,000	6 hours
$1,000,000	35 minutes
$10,000,000	3.5 minutes

Note that the costs listed in the table do not include the cost to design the chip and boards for the machine. Because the one-time costs could be as high as half a million dollars, it does not make much sense to build the cheaper versions of the machine, unless several are built for different customers.

This key search engine is designed to recover a DES key given a plaintext-ciphertext pair for the standard electronic-codebook (ECB) mode of DES. However, the machine can also handle the following modes without modification: cipher-block chaining (CBC), 64-bit cipher feedback (CFB), and 64-bit output feedback (OFB). In the case of OFB, two consecutive plaintexts are needed. The chip design can be modified to handle two other popular modes of DES, 1-bit and 8-bit CFB, at the cost of a slightly more expensive chip. Fewer chips could be purchased for a $1 million machine causing the ex-

Michael Wiener is a senior cryptologist at Entrust Technologies. He can be contacted by e-mail at wiener@entrust.com.

pected key search time to go up to 40 minutes for all modes, except 1-bit CFB, which would take 80 minutes, on average.

Programmable hardware

The costs associated with chip design can present a significant barrier to small-time attackers and hobbyists. An alternative which has much lower start-up costs is the use of programmable hardware. One such type of technology is the Field Programmable Gate Array (FPGA). One can design a circuit on a PC and download it to a board holding FPGAs for execution. In a report in early 1996 [2], it was estimated that $50000 worth of FPGAs could recover a DES key in, on average, four months. This is considerably slower than what can be achieved with a chip design, but is much more accessible to those who are not well funded.

Another promising form of programmable hardware is the Complex Programmable Logic Device (CPLD). CPLDs offer less design freedom and tend to be cheaper than FPGAs, but the nature of key search designs seems to make them suitable for CPLDs. Further research is needed to assess whether CPLDs are useful for DES key search.

Avoiding known plaintext

The designs described to this point have relied on the attacker having some known plaintext. Usually, a single 8-byte block is sufficient. One method of preventing attacks that has been suggested is to avoid having any known plaintext. This can be quite difficult to achieve. Frequently, data begins with fixed

References

[1] C. Adams, "Constructing Symmetric Ciphers Using the CAST Design Procedure", Designs, Codes and Cryptography, vol. 12, no. 3, pp. 283-316, Nov. 1997. Also available as "The CAST-128 Encryption Algorithm", RFC 2144, May 1997.

[2] M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener, "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security", currently available at http://www.bsa.org/policy/encryption/cryptographers.html.

[3] R. Rivest, "The RC5 Encryption Algorithm", Fast Software Encryption—Lecture Notes in Computer Science (1008), pp. 86-96, Springer, 1995.

[4] M. Wiener, "Efficient DES Key Search", presented at the Rump session of Crypto '93. Reprinted in Practical Cryptography for Data Internetworks, W. Stallings, editor, IEEE Computer Society Press, pp. 31-79 (1996). Currently available at ftp://ripem.msu.edu/pub/crypt/docs/des-key-search.ps.

The RSA Data Security DES Challenge II

In January of 1997, RSA Laboratories launched what came to be known as the DES challenge. The aim of the challenge was to demonstrate that 56-bit security, such as that offered by the government's Data Encryption Standard (DES), offers only marginal protection against a committed adversary. On June 17, 1997 the key that was used for the encryption of the DES challenge was recovered, thereby demonstrating the intended point. Nevertheless it is widely acknowledged that faster exhaustive search efforts are possible (see for instance the article by Michael Wiener in this issue of *CryptoBytes*) and to help assess the true viability of exhaustive search attacks, RSA Laboratories announces an ongoing series of contests.

Twice a year, a new challenge will be posted on the RSA Data Security WWW home page. The challenge will consist of the ciphertext that was produced by DES-encrypting some unknown plaintext message. While the text of the plaintext message will clearly be known to a few employees of RSA Data Security, the secret key used for the encryption will be generated at random and is destroyed within the challenge-generating software. The key will never be revealed to anyone — not even the challenge administrators.

The goal of each contest is for participants to recover the secret randomly-generated key and to do so in a faster time than that required for earlier challenges in the series.

The goal of each contest is for participants to recover the secret randomly-generated key and to do so in a faster time than that required for earlier challenges in the series. As in previous contests, prizes will be awarded for the first correct entry received and validated as being correct at RSA Data Security. However the prize money paid will depend on the CALENDAR time required to recover the correct key.

If the time required to solve the challenge for the current contest is less than or equal to 25% of the previous best time, then a prize of $10,000 will be paid to the finder of the correct key. If the time required is greater than 25% but less than or equal to 50% of the previous best time, then a prize of $5,000 will be paid. Finally, if the time required to solve the challenge is greater than 50% but less than or equal to 75% of the previous best time, then a prize of $1,000 will be paid.

If at the end of the contest period there is no solution within 75% of the previous best time, then the contest will be closed. At the start of the next contest period a new challenge will be posted with the same conditions attached to the payment of prizes.

New challenges will be posted at 9:00 am (PST) on January 13th and July 13th of each year, as long as the contest runs. The first challenge will be launched on January 13th, 1998. The previous best exhaustive search time for a DES key will be considered to be 90 days and so the prize structure for the first challenge will be:

Time for solution	Prize
Less than or equal to 540 hours	$10,000
Greater than 540 hours but less than or equal to 1080 hours	$5,000
Greater than 1080 hours but less than or equal to 1620 hours	$1,000

More information on this challenge and ongoing results are available from *http://www.rsa.com/rsalabs/*.

Bart Preneel
Katholieke Universiteit Leuven,
ESAT-COSIC K. Mercierlaan 94
B-3001 Heverlee, Belgium

Antoon Bosselaers
Katholieke Universiteit Leuven,
ESAT-COSIC K. Mercierlaan 94
B-3001 Heverlee, Belgium

Hans Dobbertin
German Information Security Agency
P.O. Box 20 03 63
D-53133 Bonn, Germany

Introduction

RIPEMD-160 is a fast cryptographic hash function that is tuned towards software implementations on 32-bit architectures. It has evolved from the 256-bit extension of MD4, which was introduced in 1990 by Ron Rivest [21, 22]. Its main design feature are two different and independent parallel chains of computation, the result of which are combined at the end of every application of the compression function. As suggested by its name, RIPEMD-160 offers a 160-bit result. It is intended to provide a high security level for the next 10 years or more. RIPEMD-128 is a faster variant of RIPEMD-160, which provides a 128-bit result. Together with SHA-1, RIPEMD-160 and RIPEMD-128 have been included in the International Standard ISO/IEC 10118-3, the publication of which is expected in late 1997 [18]. The goal of this article is to motivate the existence of RIPEMD-160, to explain the main design features, and to provide a concise description of the algorithm.

Applications of hash functions

The main application of hash functions in cryptography is the digital 'fingerprinting' of information before applying a digital signature algorithm. Hash functions have also been used to design Message Au-

constants. At the end of the compression function, the words of left and right halves are added to yield a 128-bit result. RIPEMD was believed to be stronger than extended MD4, which consisted of two parallel versions of MD4 with a 256-bit result [21]. RIPEMD was used in several European banking projects, but did not enjoy the same commercial success as MD4 and MD5.

Hash function cryptanalysis

On January 31, 1992, NIST (National Institute for Standards and Technology, USA) published in the Federal Register a *proposed* Secure Hash Standard (SHS) that contains the description of the Secure Hash Algorithm (SHA) [16]. While SHA borrows many of its design features from MD4 and MD5, it also has some remarkable differences in the message processing: instead of reordering message blocks in the different rounds, they were processed through a linear function, which at bit level can be described as a shortened cyclic code. Moreover, it has 80 steps compared to 48 for MD4 and 64 for MD5. On July 11, 1994 NIST announced a revision of FIPS 180, under the name SHA-1, which *"corrects a technical flaw that made the standard less secure than had been thought. The algorithm is still reliable as a security mechanism, but the correction returns the SHS to the original level of security"* [17].

P. van Oorschot and M. Wiener demonstrated in [23] a design for a $10 million collision search machine for MD5 that could find a collision in 24 days.

In 1992 Th. Berson tried to cryptanalyze MD5 using differential cryptanalysis [2]. A new cryptanalytic result on MD4 was obtained in 1994 by S. Vaudenay [25]. One year later, the 2nd author started his cryptanalytic work on the MD4-type hash functions. This resulted in collisions for MD4 [9, 12], and collisions for the compression function of MD5 [14] and extended MD4 [13], in both cases for different messages and fixed initial values. Moreover, he developed collisions for 2 out of the 3 rounds of RIPEMD [10]. Early 1997 he showed that it is also possible to compute a preimage for 2 rounds out of 3 for MD4 [15]. The results on RIPEMD were of some concern to the members of the RIPE consortium, as RIPEMD was designed to withstand the partial attacks developed by the consortium on MD4 and MD5.

An independent reason to upgrade RIPEMD is the limited resistance against a brute force collision search attack. P. van Oorschot and M. Wiener demonstrated in [24] a design for a $10 million collision search machine for MD5 that could find a collision in 24 days. It is clear that these results extend easily to any similar hash function with a 128-bit result. Taking into account 'Moore's law' (the cost of computation and memory is divided by four every three years), a 128-bit hash-result does not offer sufficient protection for the next ten years.

As a consequence, it was decided to upgrade RIPEMD. RIPEMD-128, with a 128-bit result was designed as a plug-in substitute for RIPEMD, while RIPEMD-160 was intended to provide long term security (10 years or more) with a 160-bit result. In addition, it was decided to stay as close as possible to RIPEMD, in order to capitalize on the evaluation effort for this algorithm. Moreover, all design criteria and evaluation results should be public. Finally, note that both designs are rather conservative: RIPEMD-128 has four double rounds, and RIPEMD-160 has five double rounds, while breaking even three double rounds would require a substantial improvement of existing cryptanalytic techniques. This means that RIPEMD-160 can provide the long term security required for digital signatures; we believe that this is worth the small penalty paid in terms of performance.

Description of RIPEMD-160

Like all MD4-variants, RIPEMD-160 operates on 32-bit words. Its primitive operations are:

- left-rotation (or "left-spin") of words;
- bitwise Boolean operations (AND, NOT, OR, exclusive-OR);
- two's complement modulo 2^{32} addition of words.

RIPEMD-160 compresses an arbitrary size input string by dividing it into blocks of 512 bits each. Each block is divided into 16 strings of 4 bytes each, and each such 4-byte string is converted to a 32-bit word using the little-endian convention, which is a.o. used on the Intel 80x86 architecture; MD4, MD5 and RIPEMD use the same convention, while SHA-1 uses the big-endian convention.

In order to guarantee that the total input size is a multiple of 512 bits, the input is padded in the same way as for all the members of the MD4-family: one appends a single 1 followed by a string of 0s (the number of 0s lies between 0 and 511); the last 64

bits of the extended input contain the binary representation of the input size in bits, least significant byte first.

The result of RIPEMD-160 is contained in five 32-bit words, which form the internal state of the algorithm. The final content of these five 32-bit words is converted to a 160-bit string, again using the little-endian convention.

This state is initialized with a fixed set of 5 32-bit words, the initial value. The main part of the algorithm is known as the compression function: it computes the new state from the old state and the next 16-word block. The compression function consists of five parallel rounds, each containing 16 steps. The total number of steps is thus $5 \times 16 \times 2 = 160$, compared to $3 \times 16 = 48$ for MD4 and $4 \times 16 = 64$ for MD5. First, two copies are made from the old state (five left and right registers of 32-bits). Both halves are processed independently. Each step updates one of the registers based on the other four registers and one message word. At the end of the compression function, we compute the new state by adding to each word of the old state one register from the left half and one from the right half (see Figure 1). Pseudocode for RIPEMD-160 is given in Appendix A.

1. **Operations in one step.** $A := (A + f(B,C,D) + X + K)^{<<s} + E$ and $C := C^{<<10}$. Here $<<s$ denotes cyclic shift (rotation) over s bit positions.

2. **Ordering of the message words.** Take the following permutation ρ:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho(i)$	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8

Further define the permutation π by setting $\pi(i) = 9i + 5 \pmod{16}$. The order of the message words is then given by the following table:

Line	Round 1	Round 2	Round 3	Round 4	Round 5
left	id	ρ	ρ^2	ρ^3	ρ^4
right	π	$\rho\pi$	$\rho^2\pi$	$\rho^3\pi$	$\rho^4\pi$

RIPEMD-160 derives its strength from a judicious choice of the parameters, combined with the fact that the processing of the two halves is much more different than for RIPEMD: the order of the message blocks in the two iterations is completely different and the order of the Boolean functions is reversed.

The operation for RIPEMD-160 on the A register is related to that of MD5 (but five words are involved); the rotate of the C register has been added to avoid the MD5 attack which focuses on the most significant bit [8]. SHA-1 has two rotates as well, but in different locations. The value of 10 for the C register was chosen since it is not used for the other rotations.

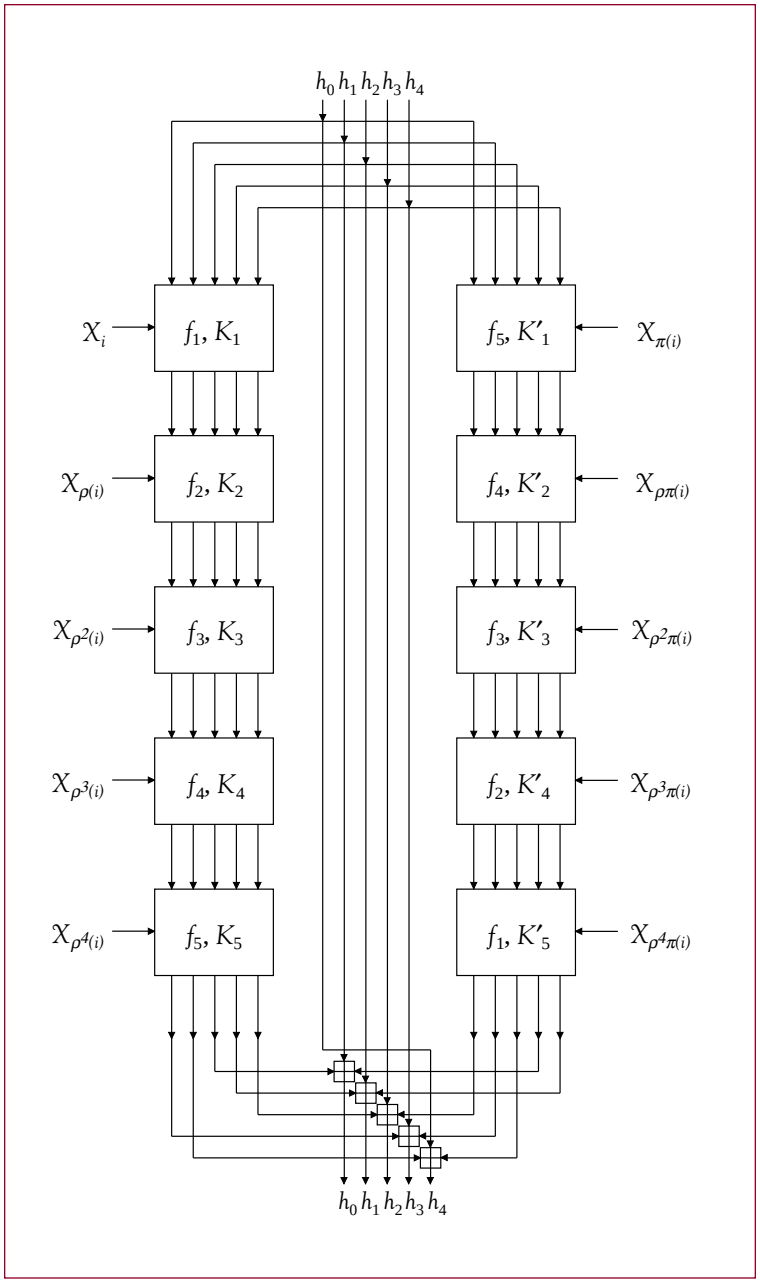

Figure 1: *Outline of the compression function of RIPEMD-160. Inputs are a 16-word message block X_i and a 5-word chaining variable $h_0h_1h_2h_3h_4$, output is a new value of the chaining variable.*

RIPEMD-160 has been put in the public domain by its designers so that anyone can use it.

The permutation of the message words of RIPEMD was designed such that two words that are 'close' in round 1-2 are far apart in round 2-3 (and vice versa). This principle has been extended to RIPEMD-160, but it required a small modification to the permutation ρ. The permutation π was chosen such that two message words which are close in the left half will always be at least seven positions apart in the right half.

For the Boolean functions, it was decided to eliminate the majority function because of its symmetry properties and a performance disadvantage. The Boolean functions are now the same as those used in MD5. As mentioned above, the Boolean functions in the left and right half are used in a different order.

The design criteria for the shifts are the following:

- the shifts are chosen between 5 and 15 (shifts that are too small or large are considered not very good, and a choice larger than 16 does not help much);
- every message block should be rotated over different amounts, not all of them having the same parity;
- the shifts applied to each register should not have a special pattern (for example, the total should not be divisible by 32);
- not too many shift constants should be divisible by four.

Note that the design decisions require a compromise: it is not possible to make a good choice of message ordering and shift constants for five rounds that is also 'optimal' for three rounds out of five.

We invite the reader to explore the security of RIPEMD-160.

Performance

In this section we compare the performance of RIPEMD-160, RIPEMD-128, SHA-1, MD5, and MD4. Implementations were written in Assembly language optimized for the Pentium processor (90 MHz); the optimizations are tuned to make use of the instruction-level parallelism of this processor. In spite of their serial design, the algorithms can still make use of this feature. More implementation details concerning the MD4-family of hash functions can be found in [3, 5, 6]. The relative speeds coincide more or less with predictions based on a simple count of the number of operations. RIPEMD-160 is about 15% slower than SHA-1 and four times slower than MD4. On a big-endian RISC machine, the difference between SHA-1 and RIPEMD-160 will be slightly larger. Optimized C implementations are a factor of 2.2 to 2.6 slower.

algorithm	performance (Mbit/s)	
	Assembly	*C*
MD4	190.6	81.4
MD5	136.2	59.7
SHA-1	54.9	21.2
RIPEMD-128	77.6	35.6
RIPEMD-160	45.3	19.3

Table 1: Performance of several MD4-based hash functions on a 90 MHz Pentium

Status of RIPEMD-160

RIPEMD-160 has been put in the public domain by its designers so that anyone can use it. Portable C source code and test values are available at: *http://www.esat.kuleuven.ac.be/~bosselae/ripemd160*.

We invite the reader to explore the security of RIPEMD-160. We envisage that in the next years it will become possible to attack one of the two lines and up to three rounds of the two parallel lines, but that the combination of the two parallel lines will resist attacks.

References

[1] M. Bellare, R. Canetti, H. Krawczyk, "Keying hash functions for message authentication," *Advances in Cryptology, Proceedings Crypto'96, LNCS 1109*, N. Koblitz, Ed., Springer-Verlag, 1996, pp. 1-15. Full version: http://www.research.ibm.com/security/.

[2] T. Berson, "Differential cryptanalysis mod 2^{32} with applications to MD5," *Advances in Cryptology, Proceedings Eurocrypt'92, LNCS 658*, R. A. Rueppel, Ed., Springer-Verlag, 1993, pp. 71-80.

[3] A. Bosselaers, R. Govaerts, J. Vandewalle, "Fast hashing on the Pentium," *Advances in Cryptology, Proceedings Crypto'96, LNCS 1109*, N. Koblitz, Ed., Springer-Verlag, 1996, pp. 298-312.

[4] A. Bosselaers, H. Dobbertin, B. Preneel, "The RIPEMD-160 cryptographic hash function," *Dr. Dobb's Journal*, Vol. 22, No. 1, January 1997, pp. 24-28.

[5] A. Bosselaers, R. Govaerts, J. Vandewalle, "SHA: a design for parallel architectures?," *Advances in Cryptology, Proceedings Eurocrypt'97, LNCS 1233*, W. Fumy, Ed., Springer-Verlag, 1997, pp. 348-362.

[6] A. Bosselaers, "Even faster hashing on the Pentium," Presented at the rump session of Eurocrypt'97, Konstanz, Germany, May 12-15, 1997, and updated on November 13, 1997. Available as ftp://ftp.esat.kuleuven.ac.be/pub/COSIC/bosselae/pentiumplus.ps.gz.

[7] B. den Boer, A. Bosselaers, "An attack on the last two rounds of MD4," *Advances in Cryptology, Proceedings Crypto'91, LNCS 576*, J. Feigenbaum, Ed., Springer-Verlag, 1992, pp. 194-203.

[8] B. den Boer, A. Bosselaers, "Collisions for the compression function of MD5," *Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765*, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 293-304.

[9] H. Dobbertin, "Alf swindles Ann," *CryptoBytes*, Vol. 1, No 3, 1995, pp. 5.

[10] H. Dobbertin, "RIPEMD with two-round compress function is not collisionfree," *Journal of Cryptology*, Vol. 10, No. 1, 1997, pp. 51-69.

[11] H. Dobbertin, A. Bosselaers, B. Preneel, "RIPEMD-160: A Strengthened Version of RIPEMD," *Fast Software Encryption, LNCS 1039*, D. Gollman, Ed., Springer-Verlag, 1996, pp. 71-82. Final (corrected) version: http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.

[12] H. Dobbertin, "Cryptanalysis of MD4," *Fast Software Encryption, LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 53-69.

[13] H. Dobbertin, "Cryptanalysis of MD4," submitted to *Journal of Cryptology*.

[14] H. Dobbertin, "The status of MD5 after a recent attack," *CryptoBytes*, Vol. 2, No 2, 1996, pp. 1, 3-6.

[15] H. Dobbertin, "The first two rounds of MD4 are not one-way," *Fast Software Encryption, LNCS*, Springer-Verlag, 1998, to appear.

[16] FIPS 180, *"Secure Hash Standard,"* NIST, US Department of Commerce, Washington D.C., May 1993.

[17] FIPS 180-1, *"Secure Hash Standard,"* NIST, US Department of Commerce, Washington D.C., April 1995.

[18] ISO/IEC 10118, *"Information technology — Security techniques — Hash-functions, Part 1: General (IS, 1994); Part 2: Hash-functions using an n-bit block cipher algorithm," (IS, 1994); Part 3: Dedicated hash-functions (IS, 1997); Part 4: Hash-functions using modular arithmetic, (FCD, 1997)*.

[19] B. Preneel, P.C. van Oorschot, "MDx-MAC and building fast MACs from hash functions," *Advances in Cryptology, Proceedings Crypto'95, LNCS 963*, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 1-14.

[20] RIPE, *"Integrity Primitives for Secure Information Systems. Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040)," LNCS 1007*, Springer-Verlag, 1995.

[21] R.L. Rivest, "The MD4 message digest algorithm," *Advances in Cryptology, Proceedings Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 303-311.

[22] R.L. Rivest, "The MD4 message-digest algorithm," *Request for Comments (RFC) 1320*, Internet Activities Board, Internet Privacy Task Force, April 1992.

[23] R.L. Rivest, "The MD5 message-digest algorithm," *Request for Comments (RFC) 1321*, Internet Activities Board, Internet Privacy Task Force, April 1992.

[24] P.C. van Oorschot, M.J. Wiener, "Parallel collision search with application to hash functions and discrete logarithms," *Proceedings 2nd ACM Conference on Computer and Communications Security*, ACM, 1994, pp. 210-218.

[25] S. Vaudenay, "On the need for multipermutations: cryptanalysis of MD4 and SAFER," *Fast Software Encryption, LNCS 1008*, B. Preneel, Ed., Springer-Verlag, 1995, pp. 286-297.

Appendix: Pseudo-code for RIPEMD-160

All operations are defined on 32-bit words. First we define all the constants and functions.

RIPEMD-160: definitions

nonlinear functions at bit level: exor, mux, -, mux, -

$f(j,x,y,z) = x \oplus y \oplus z \qquad (0 \le j \le 15)$

$f(j,x,y,z) = (x \wedge y) \vee (\neg x \wedge z) \qquad (16 \le j \le 31)$

$f(j,x,y,z) = (x \vee \neg y) \oplus z \qquad (32 \le j \le 47)$

$f(j,x,y,z) = (x \wedge z) \vee (y \wedge \neg z) \qquad (48 \le j \le 63)$

$f(j,x,y,z) = x \oplus (y \vee \neg z) \qquad (64 \le j \le 79)$

added constants (hexadecimal)

$K(j) = \texttt{00000000}_x$	$(0 \le j \le 15)$	
$K(j) = \texttt{5A827999}_x$	$(16 \le j \le 31)$	$\lfloor 2^{30} \cdot \sqrt{2} \rfloor$
$K(j) = \texttt{6ED9EBA1}_x$	$(32 \le j \le 47)$	$\lfloor 2^{30} \cdot \sqrt{3} \rfloor$
$K(j) = \texttt{8F1BBCDC}_x$	$(48 \le j \le 63)$	$\lfloor 2^{30} \cdot \sqrt{5} \rfloor$
$K(j) = \texttt{A953FD4E}_x$	$(64 \le j \le 79)$	$\lfloor 2^{30} \cdot \sqrt{7} \rfloor$
$K'(j) = \texttt{50A28BE6}_x$	$(0 \le j \le 15)$	$\lfloor 2^{30} \cdot \sqrt[3]{2} \rfloor$
$K'(j) = \texttt{5C4DD124}_x$	$(16 \le j \le 31)$	$\lfloor 2^{30} \cdot \sqrt[3]{3} \rfloor$
$K'(j) = \texttt{6D703EF3}_x$	$(32 \le j \le 47)$	$\lfloor 2^{30} \cdot \sqrt[3]{5} \rfloor$
$K'(j) = \texttt{7A6D76E9}_x$	$(48 \le j \le 63)$	$\lfloor 2^{30} \cdot \sqrt[3]{7} \rfloor$
$K'(j) = \texttt{00000000}_x$	$(64 \le j \le 79)$	

selection of message word

$r(j)$	=	j	$(0 \le j \le 15)$
$r(16..31)$	=	7,4,13,1,10,6,15,3,12,0,9,5,2,14,11,8	
$r(32..47)$	=	3,10,14,4,9,15,8,1,2,7,0,6,13,11,5,12	
$r(48..63)$	=	1,9,11,10,0,8,12,4,13,3,7,15,14,5,6,2	
$r(64..79)$	=	4,0,5,9,7,12,2,10,14,1,3,8,11,6,15,13	
$r'(0..15)$	=	5,14,7,0,9,2,11,4,13,6,15,8,1,10,3,12	

$r'(16..31)$ = 6,11,3,7,0,13,5,10,14,15,8,12,4,9,1,2
$r'(32..47)$ = 15,5,1,3,7,14,6,9,11,8,12,2,10,0,4,13
$r'(48..63)$ = 8,6,4,1,3,11,15,0,5,12,2,13,9,7,10,14
$r'(64..79)$ = 12,15,10,4,1,5,8,7,6,2,13,14,0,3,9,11

amount for rotate left (rol)
$s(0..15)$ = 11,14,15,12,5,8,7,9,11,13,14,15,6,7,9,8
$s(16..31)$ = 7,6,8,13,11,9,7,15,7,12,15,9,11,7,13,12
$s(32..47)$ = 11,13,6,7,14,9,13,15,14,8,13,6,5,12,7,5
$s(48..63)$ = 11,12,14,15,14,15,9,8,9,14,5,6,8,6,5,12
$s(64..79)$ = 9,15,5,11,6,8,13,12,5,12,13,14,11,8,5,6
$s'(0..15)$ = 8,9,9,11,13,15,15,5,7,7,8,11,14,14,12,6
$s'(16..31)$ = 9,13,15,7,12,8,9,11,7,7,12,7,6,15,13,11
$s'(32..47)$ = 9,7,15,11,8,6,6,14,12,13,5,14,13,13,7,5
$s'(48..63)$ = 15,5,8,11,14,14,6,14,6,9,12,9,12,5,15,8
$s'(64..79)$ = 8,5,12,9,12,5,14,6,8,13,6,5,15,13,11,11

initial value (hexadecimal)
$h_0 = \texttt{67452301}_x$; $h_1 = \texttt{EFCDAB89}_x$;
$h_2 = \texttt{98BADCFE}_x$; $h_3 = \texttt{10325476}_x$;
$h_4 = \texttt{C3D2E1F0}_x$;

Padding is identical to that of MD4 and MD5 [20, 21, 22]. The message after padding consists of t 16-word blocks that are denoted with $X_i[j]$, with $0 \le i \le t-1$ and $0 \le j \le 15$. The symbol $\boxplus$ denotes addition modulo 2^{32} and rol_s denotes cyclic left shift (rotate) over s bit positions. The pseudo-code for RIPEMD-160 is then given below; an outline of the compression function is given in Figure 1. The final output string then consists of the concatenation of h_0, h_1, h_2, h_3, and h_4 after converting each h_i to a 4-byte string using the little-endian convention.

RIPEMD-160: pseudo-code

for $i := 0$ *to* $t-1$ {
  $A := h_0$; $B := h_1$; $C := h_2$; $D := h_3$; $E := h_4$;
  $A' := h_0$; $B' := h_1$; $C' := h_2$; $D' := h_3$; $E' := h_4$;
  for $j := 0$ *to* 79 {
    $T := \text{rol}_{s(j)}\,(A \boxplus f(j,B,C,D) \boxplus X_i[r(j)] \boxplus K(j)) \boxplus E$;
    $A := E$; $E := D$; $D := \text{rol}_{10}(C)$; $C := B$; $B := T$;
    $T := \text{rol}_{s'(j)}\,(A' \boxplus f(79-j,B',C',D') \boxplus X_i[r'(j)] \boxplus K'(j)) \boxplus E'$;
    $A' := E'$; $E' := D'$; $D' := \text{rol}_{10}(C')$; $C' := B'$; $B' := T$;
  }
  $T := h_1 \boxplus C \boxplus D'$; $h_1 := h_2 \boxplus D \boxplus E'$; $h_2 := h_3 \boxplus E \boxplus A'$;
  $h_3 := h_4 \boxplus A \boxplus B'$; $h_4 := h_0 \boxplus B \boxplus C'$; $h_0 := T$;
}

Test Values for RIPEMD-160

```
Hash of “” = 0x9c1185a5c5e9fc54612808977ee8f548b2258d31
Hash of “a” = 0x0bdc9d2d256b3ee9daae347be6f4dc835a467ffe
Hash of “abc” = 0x8eb208f7e05d987a9b044a8e98c6b087f15a0bfc
Hash of “message digest” = 0x5d0689ef49d2fae572b881b123a85ffa21595f36
Hash of “abcdefghijklmnopqrstuvwxyz” =
0xf71c27109c692c1b56bbdceb5b9d2865b3708dbc
Hash of “abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq” =
0x12a053384a9c0c88e405a06c27dcf49ada62eb2b
Hash of “ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789”
= 0xb0e20b6e3116640286ed3a87a5713079b21f5189
Hash of 8 times “1234567890” =
0x9b752e45573d4b39f4dbd3323cab82bf63326bfb
Hash of 1 million times “a” = 0x52783243c1697bdbe16d37f97f68f08325dc1528
```

PKCS: The Next Generation, Chapter 2

Burt Kaliski
RSA Laboratories
20 Crosby Drive
Bedford, MA 01730 USA

RSA Laboratories' Public-Key Cryptography Standards (PKCS), first published in 1991, were established to provide a catalyst for interoperable security based on public-key cryptographic techniques, and they have become the basis for many formal standards and are implemented widely. In late 1996, RSA Laboratories launched the development of "the next generation" of PKCS, which has continued through 1997, and will move into another phase in 1998.

Major revisions

RSA Laboratories plans to launch or continue major revisions to three PKCS documents in 1998.

- *PKCS #1: RSA Encryption Standard* will be revised to v2.0. Currently, PKCS #1 defines specific encryption and signature algorithms based on the RSA public-key cryptosystem. Since the publication of PKCS #1, several alternative RSA-based algorithms have emerged, including those in SET, ISO/IEC 9796, ANSI X9.31, ANSI X9.44, and IEEE P1363. The revised version will summarize and provide usage guidelines for each of these alternatives. Key size recommendations may also be added, as well as object identifiers (OIDs) for the various techniques.

- *PKCS #5: Password-Based Encryption Standard* will also be revised to v2.0, following the discussions at the June 1997 PKCS workshop, as summarized to the pkcs-tng mailing list. The revised version will support password-based message authentication as well as encryption. Development will be in conjunction with PKCS #14.

- *PKCS #7: Cryptographic Message Syntax Standard* will be revised to v2.0, following discussions at the June 1997 PKCS workshop and drafts of syntax presented to the pkcs-tng list. The revised version incorporates a number of design improvements suggested over the years including support of multiple algorithms and key management techniques and a more flexible format.

Burt Kaliski is chief scientist at RSA Laboratories and can be contacted at burt@rsa.com.

Impending publication

Two projects that were the major focus of work in 1997 will be completed in early 1998.

- *PKCS #11: Cryptographic Token Interface Standard (Cryptoki)*, which defines a technology-independent programming interface for cryptographic devices, will be published as v2.01. The document is now nearly complete, incorporating significant implementation experience.

- *PKCS #12: Public-Key User Information Syntax Standard*, an interchange syntax for users' private keys and certificates, will be published as v1.0. Consensus on syntax has been reached on the pkcs-tng mailing list and implementations are already available.

New documents

Expanding the suite of PKCS documents, RSA Laboratories will launch two new documents in 1998.

- *PKCS #13: Elliptic Curve Cryptography Standard*, will be a counterpart to PKCS #1 v1.5 for encryption and digital signatures based on elliptic curve cryptosystems. Intended to be compatible with ANSI X9.62 and IEEE P1363 drafts, it will provide a reference for incorporating elliptic curve cryptography into other PKCS-based applications including those based on PKCS #11 and #7.

- *PKCS #14: Pseudorandom Generator Standard* will define a standard pseudorandom generator for a variety of cryptographic applications including key generation, parameter generation, and generation of random bits for encryption and digital signature algorithms. Pseudorandom generation has been left as an appendix in many standards, without a general reference; this standard will help establish common, secure techniques in this important area.

Development process

The PKCS development process is an open one, moderated by RSA Laboratories and involving the contributions of many security developers and users. The activities listed above will be carried on on mailing lists as well as at workshops. More information on the mailing lists, the workshops, and the PKCS documents is available on the Web at *www.rsa.com/rsalabs/pubs/PKCS/*, or from *pkcs-editor@rsa.com*.

RSA Laboratories' Public-Key Cryptography Standards (PKCS), first published in 1991, were established to provide a catalyst for interoperable security based on public-key cryptographic techniques, and they have become the basis for many formal standards and are implemented widely.

A N N O U N C E M E N T S

The RSA Data Security Conference '98

The seventh annual RSA Data Security Conference is scheduled to be held in San Francisco on January 13-16, 1998.

Virtually all of San Francisco's Nob Hill will be dedicated to the event, including the Masonic Auditorium, the Fairmont Hotel, the Stanford Court, and the Ritz Carlton.

The conference will deliver four full days of coverage of the latest trends in cryptographic research, product development, market analysis and social thought in the field of cryptography, all presented by some of the leading minds in the industry. An annual pilgrimage for the world's cryptography systems experts, policy-makers, business people and technology developers, the RSA Conference delivers breadth and depth far beyond any other computer security gathering.

Computerworld called it "the *sine-qua-non* event of the crypto community". From very humble beginnings in 1991, when 50 developers gathered to discuss the state of the nascent crypto industry, the annual RSA Conference has grown to become the biggest event on the crypto-circuit. Planners are projecting that over 3,000 cryptographers, policy-makers, business people and technology developers will attend the 1998 conference.

An increasingly significant element of the conference is the RSA Data Security Conference Partner Fair. Scheduled to take place January 13-15 at the Fairmont Hotel, this exhibit hall provides an unparalleled opportunity for attendees to see the very latest developments in cryptographic and computer security products.

For more information or to register, please visit *http://www.rsa.com/conf98/*.

In this issue:

- ***On the Foundations of Modern Cryptography***
- ***Efficient DES Key Search — An Update***
- ***The Cryptographic Hash Function RIPEMD-160***
- ***PKCS: The Next Generation, Chapter 2***

For contact and distribution information, see page 2 of this newsletter.

FIRST CLASS
U.S. POSTAGE
PAID
MMS, INC

RSA Laboratories
A Division of RSA Data Security

100 Marine Parkway, Suite 500
Redwood City, CA. 94065-1031

Tel 650/595-7703
Fax 650/595-4126

rsa-labs@rsa.com
http://www.rsa.com/rsalabs

Copyright © 1997 RSA Laboratories, a division of RSA Data Security, Inc., a Security Dynamics Company. All rights reserved.