VOLUME 2, NUMBER 3 — WINTER 1997

RSA LABORATORIES'

CryptoBytes

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

Contents

How Exhausting is Exhaustive Search?

Germano Caronni
Computer Engineering & Networks Laboratory
ETH, Zurich, Switzerland

Matt Robshaw
RSA Laboratories
Redwood City, CA 94065, USA

On January 28, 1997, a series of new cryptographic contests were launched by RSA Laboratories. The goal was to quantify the security of the Data Encryption Standard (DES) and secret-key ciphers in general when keys of different sizes are used.

It is widely agreed that 56-bit keys, as offered by the DES standard [5], offer marginal protection against a committed adversary. Indeed, theoretical studies have been performed showing that it is possible to build a specialized "DES cracker" computer that could crack keys in mere hours by exhaustive search [9]. However, it is unknown whether any such machine has been built, and DES is still very widely used, in part because of its continued resistance to sophisticated cryptanalytic attacks.

For those concerned about the length of the keys used in DES there are a variety of options available, such as triple-DES [3] or an alternative cipher like IDEA[4], Ron Rivest's DESX [8] or RC5 [7] with which longer keys can be used. However, for certain countries, among them the United States, the export of cryptographic products is a sensitive issue and the cryptographic length of the encryption key may be limited. One generally-used threshold that has been applied to exportability in the United States has been that the strength of the encryption should be equivalent to that offered by a 40-bit symmetric encryption key.

Germano Caronni is a postgraduate researcher and can be contacted at gec@acm.org. Matt Robshaw is principal research scientist at RSA Laboratories and can be contacted at matt@rsa.com.

There is no need for theoretical studies in this case since 40-bit encryption has been acknowledged for some time to offer little more than token resistance to an adversary. Just how much resistance this is was dramatically confirmed a mere three and a half hours after the launch of the 40-bit contest on January 28!

Exhaustive Key Search

Many problems require a large amount of computational power to derive a solution. Some problems, though, are amenable to an extremely high level of parallelization, and with today's Internet it is possible to broaden the reach of any large-scale effort to previously unanticipated levels. The factorization of RSA-129 [1] in 1994 was remarkable for the coordination via E-mail of the combined efforts of 600 people and 1600 computers, including two fax machines. The more recent factorization of RSA-130 [2], which incidentally demonstrated the increasing effectiveness of an alternative factoring technique, shows the opportunities of coordinating a global factoring effort with the World-Wide Web. In fact, an earlier *CryptoBytes* article by Odlyzko [6] has considered both the growth in the Internet and also the increasing power of the constituent computers as a part of his assessment for the future of factorization.

(continued on page 3)

Editor's Note

In this issue of *CryptoBytes* we provide details and some of the latest news from the recently launched RSA Data Security Secret-Key Challenge. Aimed at quantifying the resistance of symmetric ciphers to exhaustive key search, substantial cash prizes are on offer for the recovery of the secret keys used to construct a variety of challenge encryptions.

The first two challenges have already been solved, with considerable surrounding publicity, and in our lead article for this issue we summarize the data provided during these challenges by the efforts of both Ian Goldberg and Germano Caronni.

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

Interest in the Secret-Key Challenge is two-fold. While the 48-bit challenge has provided us with insight into the capabilities of distributing simple but large computational tasks among many users, the results of the 40-bit challenge have already provided useful information to the continuing debate on the size of cryptographic keys and national export restrictions. It will also be interesting to see the impact on the community when a 56-bit RC5 or DES key has been recovered. This is particularly the case when refinements to US cryptographic export laws now allow the export of 56-bit technology, though only under particular circumstances.

In the second invited article of this issue Peter Gemmell offers us an overview of some different threshold technologies available and the wide range of properties they offer. These techniques are very new, and have only received limited attention outside of the cryptographic community, but the properties offered are very appealing and they seem ideally matched to the realities of today's cryptographic world. In particular, by avoiding the storage of a single secret in one place, or by ensuring that a computation never requires the complete reconstruction of sensitive and secret information, the opportunity for a compromise in security can often be greatly reduced.

As always, the newsletter contains the latest news from the world of standards and algorithm development. In particular, we include an overview of the status of RC5. Published two years ago, RC5 is gaining in popularity and results from the research community point to a cipher that delivers the security the designer had originally hoped for. While only time will serve to tell how RC5 will ultimately fare, the two-year assessment we offer here demonstrates that RC5 has made a very strong start.

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

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 issue a limited number of copies are printed. They are distributed at major conferences and through direct mailing. While available, additional copies of the newsletter can be requested by contacting RSA Laboratories though a nominal fee to cover handling costs might be charged for individual requests.

RSA Laboratories can be contacted at:

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

How Exhausting is Exhaustive Search?

Continued from page 1

Shifting our emphasis back to exhaustive key search, it is clear that this is a problem that is perfectly suited to massive parallelization. (This has already occurred in an earlier attack on 40-bit SSL.) With the proper coordination of a large number of computers distributed around the Internet, it might well be possible to provide sufficient computational power to exhaustively search the DES key space (or the key space of a cipher with comparable key size) in a matter of weeks. And as an important by-product for researchers and developers alike, we might hope to obtain information about the full potential of distributing problems among a large community of connected computers.

Solving RC5-32/12/5

In 1992, the first author and Werner Alemsberger wrote software that would coordinate such a distributed search effort. The goal was to illustrate the dangers of short and badly chosen local administrator UNIX passwords and the software had been successful in this, using up to 350 workstations for the task. With the announcement of the Secret-Key Challenge and its impending launch on January 28, 1997, another opportunity for putting this work to the test presented itself.

During January the password-specific features were cut out of the software (especially the handling of different possible key space subsets) and RC5 was added. The implementation of RC5 was optimized for an Ultra 1/170 without using assembler and required 6.15 seconds to search 1,000,000 RC5-32/12/5 keys. In the week before the challenge, access to local student pools was organized and three days before the challenge more colleagues were contacted to provide additional computing power.

The search for RC5-32/12/5 started at 18:07 MEZ (9:07 PST), but a very subtle flaw in the server, found by Christian Schneider at 20:30, caused the search to crash multiple times (around 40 server restarts were required). Despite that, the key was recovered after 800 machines from ETH, Germany, Norway, and the United States had searched 51% of the space in 231 minutes with a key testing rate of about 40 million keys per second.

But it was 21 minutes too late. UC Berkeley graduate student Ian Goldberg had already found the correct key, in a search of about 31% of the key space that required 210 minutes.[1] More details on Goldberg's dramatic demonstration of the inadequacy of 40-bit encryption keys are given below, but it is interesting to observe that the approaches of Goldberg and Caronni are essentially the same.

The Solution to the RC5-32/12/5 Contest

The 40-bit key used to encrypt the message "This is why you should use a longer key" was recovered after 210 minutes of searching while using a total of 259 machines (287 processors). All resources were available at the start and the rate of key testing was a steady 27 million keys per second.

- 97 UltraSPARCs
- 4 8-processor UltraSPARCs
- 120 HP workstations
- 8 Pentium Pros
- 30 SPARCStation 20s

The search used a centralized key server. Every time a client needed a piece of the key space, it would do a short timing test, and report its speed to the server. The server would then give it about 20 minutes' worth of key space. When the client was finished, it would report back that it was done, and repeat. The server kept track of what key ranges were currently unallocated, in use, and done. If the "unallocated" list were to become exhausted, the "in use" list would be copied into the "unallocated" list. This would handle the case of a client dying while working on something, though this feature wasn't actually needed during the RC5-32/12/5 contest.

Many thanks to Ian Goldberg for this information.

Solving RC5-32/12/6

With RC5-32/12/5 now closed, attention turned to the 48-bit RC5 challenge. Server and client software was updated and colleagues launched requests for more people to join the mailing lists. The definitive start of attack on RC5-32/12/6 was at 18:00 on January 29 (with a six-hour test run beforehand).

There are of course different approaches to an exhaustive search. The major consideration is perhaps whether the search is coordinated by some central server, or whether multiple processes start at random

[1] While the contest was available on the World-Wide Web, it had still not been announced at the first day of the 1997 RSA Data Security Conference before the 40-bit contest was solved!

[...] exhaustive key search [...] is a problem that is perfectly suited to massive parallelization.

positions in the search space and run independently until a key is found.

The use of a central server poses some difficulties.

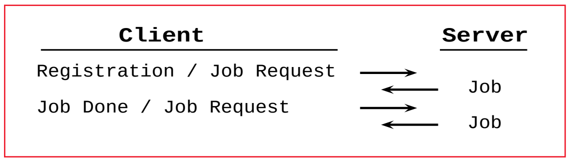
The use of a central server poses some difficulties. As well as providing a single point of failure, there is also the potential of network congestion and failures. Unintentionally malfunctioning clients could cause havoc to a centrally-coordinated search effort, but even worse are the potential risks if deliberately malevolent clients are encountered. (Such a situation was not experienced in the attack on RC5-32/12/6, however.)

A variety of precautions can be taken. Servers can be networked into a hierarchy, or perhaps replicated, if resources allow, so that points of failure are less catastrophic. In addition, clients can test themselves to provide some level of assurance against malfunction and more explicit testing can be conducted on the clients by servers to provide assurance against malevolent clients. The server can have the client report on server-fabricated problems which can be checked at a very small cost. Alternatively, a client could calculate a checksum over all attempted solutions in the range examined and this could be checked by another client of the same architecture.

Unintentionally malfunctioning clients could cause havoc to a centrally-coordinated search effort, but even worse are the potential risks if deliberately malevolent clients are encountered.

Software Architecture

The design of the server client was intentionally very simple. A trivial UDP protocol was used for clients to register with the server, and for the server to send jobs to the client. The protocol was extended when additional needs became obvious. The following graph depicts a typical exchange between a client and the server:



`Registration` contains version, machine, and host/user information. `Job Request` gives the number of keys that the client wants to search, and how long it expects to take for this job. If the number of keys communicated is zero, then the client plans to become idle, and the server confirms this by a simple `Idle Confirm` message. Additional messages from the client to the server report a successful key guess, or the fact that the client wishes to re-awake after an idle time. If the server is terminated, and a new one is started, `Job Done` messages are replied with `Request Registration` so that the client registers itself with the new server.

Additionally, the server can send `Kill` messages to clients, which causes them to terminate. The UDP protocol is used for data transfer, doing an idle repeat-request with the client as the driving side. The server manages the key space in distinct chunks, and check points to disk every few minutes in case of software failure or machine reboots. Clients register with the server, which deals out part of the key space and remembers the recipient. If the time-out on the key space segment expires, then the block is rescheduled, and eventually assigned elsewhere. If a client reports that a block has been searched, it is assigned a new one.

Optimizations were mostly native assembler implementations of RC5 for the client of the software packet. This was done for INTEL and RS6000. Finding the best compiler and the best optimization options inevitably lead to a significant increase in performance. Eventually the following architectures were supported, though others did not find their way back into the central repository: AIX, FreeBSD, HP-UX, IRIX, Linux, NEXTSTEP, NetBSD, OS2, OSF1, OpenBSD, SCO_SV, SunOS, ULTRIX, WINDOWS (NT & 95), AMIGA MacOS on MIPS, ALPHA, (Ultra) SPARC, [45]86 Intel, Pentium Pro, Paragons and massive parallel systems (16000 CPUs).

The Growth of Available Resources

As news of the attempt on RC5-32/12/6 spread, the resources available for the search effort increased tremendously. While the peak rate of key testing achieved was 440 million keys/second, and the peak number of parallel hosts was 4500 (with more than 7500 known workers), it is interesting to look at the average growth of the available resources.

In Figure 1 we provide a measure of the growth of the rate of key testing on a day by day basis. After a slower build up, the rate of key testing roughly doubled over the last two three-day periods with the rate of February 6th being almost double that of February 3rd and the rate for February 9th offering another doubling of the rate of key testing.

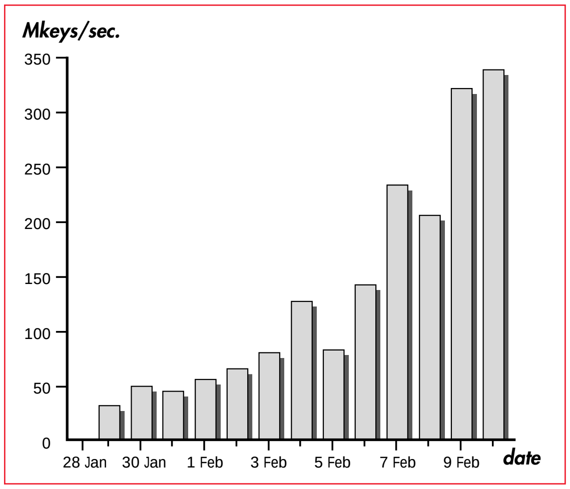


Figure 1: The average rate of key testing in Mkeys/sec during each day of the challenge.

Note that this growth in computational power can be compared to the constant rate of key testing (27 million keys per second) that the non-distributed effort of Ian Goldberg provided in the solution to the RC5-32/12/5 challenge.

This key testing translates into a percentage of the total key space that has been examined during the course of the challenge. In Figure 2, we illustrate this percentage on a day-by-day basis.

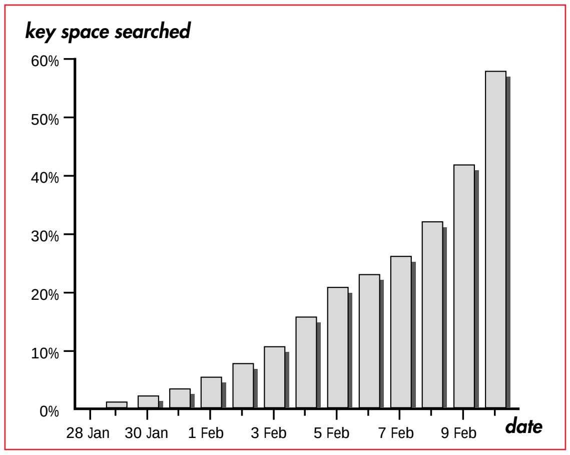


Figure 2: The total percentage of the key space searched by the conclusion of each day of the challenge. At the conclusion of the challenge 57.6% of the key space had been searched.

These crude graphs give no indication of any form of saturation point being reached in the available computational power had the challenge on RC5-32/12/6 not ended on February 10th. Indeed, the more detailed information available from the many logs kept during the challenge (information that is available via *http://www.ee.ethz.ch/challenge/*, *http://www.klammeraffe.org/challenge/* and *http://www.42.org/challenge/*) corroborate this view.

But does the single server approach generate a potential bottleneck for such large search efforts? Under normal conditions, no. By looking at the peak conditions experienced towards the end of the RC5-32/12/6 project we might assume that there are 5000 clients that report every 30 minutes to the server. That means that around two clients talk to the server every second, and the 300 bytes of traffic this generates per second is negligible. The server load on the coordinating Sun Ultra 1/170 for RC5-32/12/6 was below 2%, and the memory consumption for the server was below 6 Mbytes. It appears that handling up to 100,000 clients with a single server is perfectly feasible. Any real bottlenecks would be the network congestion that might be experienced on international communication links.

Finally it is interesting to estimate the computing power accumulated in the search for the 48-bit RC5 key. If we assume that 2,000 instructions are required to test an RC5 key (this figure falls between two of the implementations that were actually used) then we find that the key search required 3.24 x 10^{17} instructions. Since a MIPS-year (MY), the number of operations completed by a machine that runs for one year at the rate of one million instructions per second, contains around 3.15 x 10^{13} operations, the 48-bit search effort managed to accumulate around 10,000 MY of computing power over 10 calendar days. This demonstrates the increase in computing power that might be available over the Internet. In comparison, the RSA-129 factoring effort [1] in 1994 required 5,000 MY of computing power, and this was accumulated over nine calendar months.

56-bit RC5, DES, and Other Efforts

What lessons can we take from the results of the RC5-32/12/5 and RC5-32/12/6 challenges? At first sight, the final rate of key-testing for the RC5-32/12/6 (330 million keys per second) doesn't allow the recovery of a 56-bit key anytime soon. At such a rate, the recovery of the key used in RC5-32/12/7 might be expected to require around 2^{55} / 330 x 2^{20} seconds, which is a little under three and a half years.

After a slower build up, the rate of key testing doubled over the last two three-day periods [...]

[...] does the single server approach generate a potential bottleneck for such large search efforts? Under normal conditions, no.

But this assumes that we have the same resources at hand as we had at the conclusion of the attack on 48-bit RC5. Instead we might estimate from Figure 1 that the speed of the search effort was doubling roughly every three days and showed little imminent sign of slackening. With the combined publicity of both the 40- and the 48-bit challenge let us suppose that we can maintain this level of increasing involvement for more than another nine days.[2] Under this assumption a level of computing power around 10 times the final rate of key testing obtained for RC5-32/12/6 seems reasonable. With this kind of computational power available, an exhaustive search effort on RC5-32/12/7 is certainly within reach and might require around four calendar months.

[...] the strength of symmetric encryption using keys of either 40 or 48 bits can only be described as "slight".

And what are the implications for an exhaustive key search attempt on DES? While the key is of the same length as that used for RC5-32/12/6, testing a DES key is faster than testing an RC5 key. Depending on the techniques used, a factor of between two and four improvement in time when testing a DES key offers a reasonable guide. (Some implementations can test half a million DES keys per second on a Pentium 120.) With such a factor in hand, and if our estimate on the available power for such a search holds, then the hope of being able to achieve the recovery of a DES key in software within months (or potentially even weeks) seems to be quite reasonable.

Conclusions

The implications of the searches for both RC5-32/12/5 and RC5-32/12/6 are immediate and obvious. Against adversaries with access to even a modest amount of computing power, the strength of symmetric encryption using keys of either 40 or 48 bits can only be described as "slight".[3]

It is also clear that a 56-bit encryption key as provided for in new proposals for United States export regulations (see *http://www.bxa.doc.gov/encstart.htm*) offers too little protection when we consider the potential computational power available to a committed adversary. When we consider the phenomenon known as Moore's Law [6] which predicts a doubling of computing power every 18 months, then even the partial resistance offered today by the use of 56-bit keys will soon be eroded by computational progress.

In short, these recent exhaustive search attacks give support to the conclusions drawn by an ad hoc group of cryptographers and computer security specialists during 1996 (see *http://www.bsa.org/policy/encryption*). In this report, it is concluded that security for the near future could only be achieved with symmetric key sizes of between 75 and 90 bits as a minimum. Indeed, the more conservative recommendation given there, and supported by the authors here, is that 128-bit symmetric keys be used instead. As attacks go, exhaustive search is not elegant and prevention is easy. But, incredibly for today's world of international business, existing legal restrictions often ensure that exhaustive search remains practical.

[2] Among other incentives in a renewed search effort could be the opportunity for material gain if future efforts allow the finder of the correct key to keep the prize money. By contrast, from the start of the 48-bit challenge it was clear that any prize would be donated to project Gutenberg (http://promo.net/pg/).

References

[1] D. Atkins, M. Graff, A.K. Lenstra and P.C. Leyland. The magic words are squeamish ossifrage. In *Advances in Cryptology - Asiacrypt '94*, pages 263-277, Springer-Verlag, 1995.

[2] J. Cowie, B. Dodson, R.M. Elkenbracht-Huizing, A.K. Lenstra, P.L. Montgomery and J. Zayer. A world wide number field sieve factoring record: On to 512 bits. In *Advances in Cryptology - Asiacrypt* '96, pages 382-394, Springer-Verlag, 1996.

[3] B.S. Kaliski Jr. and M.J.B. Robshaw. *Multiple encryption: weighing up security and performance. Dr. Dobb's Journal*, #243, pages 123–127, January 1996.

[4] X. Lai, J.L. Massey and S. Murphy. Markov ciphers and differential cryptanalysis. In *Advances in Cryptology — Eurocrypt '91*, pages 17–38, Springer-Verlag, 1992.

[5] National Institute of Standards and Technology. *FIPS Publication 46-2: Data Encryption Standard.* December 1993.

[6] A. Odlyzko. The Future of Integer Factorization. RSA Laboratories' *CryptoBytes*, Vol. 1, No. 2, pages 5-12.

[7] R. Rivest. The RC5 encryption algorithm. In *Proceedings of 2nd Workshop on Fast Software Encryption*, pages 86–96, Springer-Verlag, 1995.

[8] P. Rogaway. The Security of DESX. RSA Laboratories' *CryptoBytes*, Vol. 2, No. 2, Summer 1996, pages 8-11.

[9] M.J. Wiener. Efficient DES key search. Technical Report TR–244, School of Computer Science, Carleton University, Ottawa, Canada, May 1994.

[3] One press release contained the colorful phrase that "40-bit encryption isn't worth the electrons it's carried on" (*Computing*, February 20, 1997).

An Introduction to Threshold Cryptography

Peter S. Gemmell
Sandia National Laboratories
MS1110, PO Box 5800
Albuquerque, NM 87185-1110, USA

Any application requiring high security for information is a potential customer of threshold protocols. In some applications, such as electronic commerce and certification systems, the security of cryptographic keys is a major system design issue, a potential bottleneck for system performance, and a primary target for would-be adversaries.

Single-use data can be protected by threshold *secret sharing* in which one piece of secret information is derived probably at most once from a set of secret shares. Multiple-use keys such as the private keys of a certicate authority or a mint for electronic money can be protected by threshold *function sharing* in which a distributed key is employed repeatedly while maintaining its security.

Applications for threshold cryptography include *key escrow* situations, such as those that could occur in communication, commerce, and other applications. In key escrow protocols, two or more escrow agents or trustees hold parts of a key or keys. Periodically, the escrow agents may be called upon to employ a key that they jointly hold. At this point, if the key is shared via a threshold cryptography protocol, the key may be employed distributedly in a robust fault-tolerant way. Furthermore, if the key is distributed using a threshold function sharing protocol, it might be employed repeatedly without revealing the key itself.

Threshold cryptography protocols address a variety of adversaries and a variety of attacks. They maintain appropriate security against hackers, insiders, disgruntled ex-employees, computer viruses, and other agents of data espionage and destruction.

The types of security that threshold cryptography can provide include:

- confidentiality against outsiders and coalitions of shareholders.
- data integrity and availability in the presence of adversarial shareholders.
- verification that the data being shared is correct.
- verifiable robust distributed cryptographic function computation without revealing the key.
- robust confidentiality and integrity in the presence of an adversary that, over a period time, infiltrates many, possibly all, shareholders.

First we present two important models for threshold cryptography — single-secret sharing and function sharing. Then we describe three single-secret sharing protocols and their underlying mathematical techniques. After presenting a sharing protocol for the RSA function we present three additional dimensions of security that improve threshold protocols.

Any application requiring high security for information is a potential customer of threshold protocols.

The Basic Model

Threshold protocols aim to achieve the two somewhat divergent goals of data secrecy and data integrity/availability.

If integrity were the only goal, then simple duplication of the full data among n parties would prevent coalitions of up to $n - 1$ parties from erasing the secret. However, this also would ensure that any one party could disclose the secret to an adversary.

If secrecy were the only goal, then solutions might include splitting the data into n pieces that are $(n - 1)$-wise independent random variables and giving one piece to each of the n parties. This would require all n parties to divulge the secret. However, the destruction or alteration of any one piece would erase the distributed secret.

Threshold protocols aim to achieve the two somewhat divergent goals of data secrecy and data integrity/availability.

Threshold protocols maintain secrecy in the face of up to $k - 1$ adversaries and yet achieve data integrity and availability with the cooperation of k shareholders. In the protocols in the literature (see the attached bibliography for a partial set of references) and in those that we describe, k can generally be any number and is generally independent of n.

Single-secret threshold paradigm

A basic k-of-n threshold protocol involves n parties called *shareholders*, possibly one party called a *dealer*, and possibly one party called a *combiner*.

Peter Gemmell is a researcher at Sandia National Laboratories and is supported by the U.S. Department of Energy. He can be contacted at psgemme@cs.sandia.gov.

The shareholders all hold pieces of data called *shares*. Any k shareholders can combine their shares to deduce a secret s. No $k - 1$ shares yield any significant information about the value s. The combiner gathers information from the shareholders and computes the secret. The combiner may be one or more of the shareholders, or, in some protocols, a distinct party.

In some protocols, the shareholders themselves collectively choose the secret s (without knowing individually what it is) and determine their shares.

In other protocols, the shares are distributed initially by a dealer who chooses the secret. This dealer is trusted not to reveal the secret to others — including the shareholders. In some implementations, the dealer may be a tamper-resistant device jointly and securely created by the shareholders. This device may produce a random secret and shares, distribute the shares confidentially among the shareholders, and subsequently erase its copy of the secret and shares.

Threshold sharing for functions

One shortcoming of threshold sharing of single-secrets is that to employ the secret in a useful way shareholders have to reveal it among themselves. In this case, the combiner will learn the key to a shared cryptographic function the first time that it is applied.

In addition to the single-secret and threshold paradigms, researchers have added notions including verifiable secret sharing, robustness, and proactive sharing.

De Santis, Desmedt, Frankel, and Yung [10] introduced the notion of *threshold sharing for functions*. There had been a great deal of previous work in this area, including work by [23], [8], [16], [12], and [14]. In [10]'s model, they described how to share a key to a cryptographicly secure function f in such a way that:

- Any k shareholders can collectively compute f.
- Even after taking part in the computation of f on some inputs, no set of up to k - 1 shareholders can compute f on other inputs.

Fault-tolerance extensions to threshold secret sharing

In addition to the single-secret and threshold paradigms, researchers have added notions including *verifiable secret sharing*, *robustness*, and *proactive sharing*. These three ideas are discussed in later sections.

Mathematical Techniques and Single-Secret Protocols

Shamir's protocol and interpolating polynomials

Many threshold protocols, including Shamir's original protocol, rely on an interesting property of polynomials:

Observation 1: If $p(x) = \sum_{i=0}^{d} a_i x^i$ is a degree d polynomial with random coefficients over a finite field F with more than d elements, (e.g. the integers modulo a prime greater than d) then

1. Values $\{p(x_i)\}_{i=1}^{d+1}$ for distinct values of x_i determine p on *all* points.

In particular, $p(x) = \sum_{i=1}^{d+1} (p(x_i) \Pi_{j \neq i} \frac{x - x_j}{x_i - x_j})$

Computing a polynomial this way is known as LaGrange interpolation.

2. d or fewer values $\{p(x_j)\}_{j=1}^{d}$ for distinct values of x_i yield *no information* about $p(y)$ for any single value of $y \notin \{x_1 \dots x_d\}$.

Shamir's protocol and many others use the above properties by creating a degree $k - 1$ polynomial p that has random coefficients other than that $p(0) = a_0 = s$. Each shareholder gets a value of p evaluated at a distinct non-zero point. To compute the secret s, k shareholders need only perform LaGrange interpolation.

Shamir's original protocol is as follows:

Shamir's basic k-of-n single-secret sharing

The *initial set-up*:

Dealer: Given secret s of l bits:

1. Choose prime $q > max\{2^l, n\}$.
2. Define coefficient $a_0 = s$.
3. Choose coefficients $a_1, a_2, \dots a_{k-1}$ independently at random from Z_q.
4. For all x, define $p(x) = \sum_{i=0}^{k-1} a_i x^i \bmod q$
5. Give $p(i)$ to shareholder i for $i \in \{1, \dots, n\}$.

The *reconstruction* of s by k cooperating shareholders:

- The jth cooperating share-holder x_j provides his or her share $p(x_j)$ to the others.
- Each cooperating shareholder can then interpolate the secret $s = a_0 = \sum_{j=1}^{k} (p(x_i) \Pi_{l \neq j} \frac{0-l}{j-l}) \bmod q$.

Blakley's protocol and facts about plane geometry

Another tool for threshold cryptography, employed by Blakley, is that of intersecting hyperplanes defined over cross-products of a finite field.

Let $\mathcal{F}$ be a finite field (for example $\mathbb{Z}_q$, the integers modulo a prime q). First, consider the two-dimensional case:

Observation 2: Let $\mathcal{L}$ be a set of n distinct random lines in $\mathcal{F} \times \mathcal{F}$ (not defined to be equal to $x_1 = s$) all intersecting in an unknown single point (s,t). Then no *single* line $l \in \mathcal{L}$ reveals any information about the value s. Furthermore, any 2 non-parallel lines determine the point (s,t).

A similar observation is true in the general case:

Observation 3: Let $\mathcal{L}$ be a set of n random hyperplanes in $\mathcal{F}^k$ (not defined to be equal to $x_1 = s$) all intersecting in an unknown single point $(s, t_1, \ldots, t_{k-1})$ and defined so that the set of hyperplanes consisting of $\mathcal{L}$ and the plane $x_1 = s$ is not degenerate. Then no set of $k-1$ hyperplanes in $\mathcal{L}$ reveals any information about the value s. Furthermore, any set of k different hyperplanes in $\mathcal{L}$ determines $(s, t_1, \ldots, t_{k-1})$.

This latter observation suggests Blakley's threshold sharing protocol:

Blakley's basic *k*-of-*n* single-secret sharing

The *initial set-up*:

Dealer: Given secret s of l bits:

1. Choose prime $q > max\{2^l, n\}$.
2. Compute a set $\mathcal{L}$ of n distinct random hyperplanes, not defined equal to $x_1 = s$, in $\mathbb{Z}_q^k$ that intersect in a point $(s, t_1, \ldots, t_{k-1})$ where $t_1, \ldots, t_{k-1}$ are random points in $\mathbb{Z}_q$ and $\mathcal{L}$ is defined so that the set of hyperplanes consisting of $\mathcal{L}$ and the plane $x_1 = s$ is not degenerate.
3. Give the coefficients of one hyperplane to each of the shareholders.

The *reconstruction* of s by k cooperating shareholders:

- The k shareholders each provide one hyperplane — a linear equation in k variables — and solve the resulting set of k equations in k variables, keeping the first coefficient as the secret.

Alon, Galil, and Yung's Combinations of Families and Committees

Alon, Galil, and Yung [1] have recently added another combinatorial tool to the threshold cryptography arsenal.[1]

They divide the secret s into shares in several different ways, creating a number of different sets of shares. They assign each share of each set of shares to several different shareholders and each shareholder holds shares from various sets of shares.

More specifically: there are a number, m_{fam}, of families and a number, m_{com}, of committees per family. Each committee has m_{mem} shareholders on that committee.

For each family, the secret s is broken up into m_{com} shares and share i is given to all the members of the ith committee. All the shares of any one family are necessary and sufficient to reconstruct the secret s.

The trick then is how to choose the values m_{fam}, m_{com}, and m_{mem} and how to assign the shareholders to committees.

Let ε be a small constant fraction (like $^1/_{10}$). [1] (with a slight generalization by [20]) showed that for reasonably sized values of m_{fam}, m_{com}, and m_{mem}, most assignments of shareholders to committees have the following property:

1. For every set of k shareholders, the shareholders have a representative on every committee in most of the families.

2. For every set of fewer than $k - \varepsilon n$ shareholders, there is no committee for which the shareholders have a representative on every committee.

We can now describe a share assignment strategy for the dealer.

Another tool for threshold cryptography, employed by Blakley, is that of intersecting hyperplanes [...]

[1] Technically, the families and committees of [1] do not yield threshold protocols for large values of n, because there is a small grey zone between the number of shareholders who always can not compute the secret and the number of shareholders who always can. [20] determined [1]-sharing configurations that yield exact threshold protocols for modest values of k and n.

Call the share assigned to the jth committee of the ith family the share $share_{i,j}$.

Alon-Galil-Yung basic single-secret sharing

The *initial set-up*:

Dealer: Given secret s: For each family i:

1. Choose the shares $\{share_{i,j}\}_{j=1}^{m_{com}-1}$ as independent random variables chosen from a large interval.
2. Let $share_{i,m_{com}} = s - \Sigma_{j=1}^{m_{com}-1} share_{i,j}$ so that $s = \Sigma_{j=1}^{m_{com}} share_{i,j}$.
3. Determine a good assignment of shareholders to committees and assign the shares accordingly.

The *reconstruction* of s by k cooperating shareholders:

- The shareholders determine a family i_0 for which they have a member on each committee and then compute $s = \Sigma_{j=1}^{m_{com}} share_{i_0,j}$.

[...] the shareholders might still be assured individually that their shares could be combined to form the correct secret [...]

De Santis, Desmedt, Frankel, and Yung's Threshold Function Sharing of RSA

[10] presented an elegant threshold function sharing protocol for RSA. Their protocol allows the shareholders to evaluate repeatedly an RSA signature or decryption function in a distributed way without affecting the confidentiality of the private key.

To accomplish their goals, [10] adopted the LaGrange interpolation approach to sharing.

They had to enable the combiner to compute $h(M)^d$ mod N without learning d, the RSA private key. Here, $h(M)$ refers to the hash of the message being signed and N is the RSA modulus.

To do this, they had the combiner and shareholders perform the LaGrange interpolation implicitly in the exponent of the message:

$$h(M)^d = h(M)^{\Sigma_{i=1}^{k}\left(p(x_i)\Pi_{j\neq i}\frac{0-x_j}{x_i-x_j}\right)} = \Pi_{i=1}^{k}\left(h(M)^{p(x_i)}\right)^{\Pi_{j\neq i}\frac{0-x_j}{x_i-x_j}} \bmod N$$

This notion, called verifiable secret sharing, [...] is used in many threshold protocols.

Note that this process implies that some party has to be able to compute inverses of some elements in the exponent. However, if one person knew non-trivial elements a and a^{-1} mod(ϕ (N)), then they could factor N. [10] solved this difficult problem by extending the group of RSA exponents to a larger set of operators. This set contains special invertable elements that do not compromise the RSA key.

Verifiable Secret Sharing

When a dealer provides the shareholders with the shares of a secret s, no set of $k - 1$ shareholders are supposed to know what the secret is.

However, the shareholders might still be assured individually that their shares *could* be combined to form the correct secret, or at least one unambiguous secret. This notion, called verifiable secret sharing, was pioneered by [6] and [15] and is used in many threshold protocols.

Robustness

Some threshold sharing protocols are vulnerable to attacks from adversarial shareholders whose goal is to prevent good shareholders from reconstructing the secret.

For example, in Shamir's original protocol, a group of k shareholders wishing to determine the secret s may contain one adversary who lies about the value of his or her share. Not only will this lying prevent the reconstruction of the secret, but the other shareholders will not know who to blame for the failure to reconstruct the secret. In fact, they may not realize that the result of the protocol is not the real secret.

These complications imply that, without modification, some protocols may be unable to produce a consistent secret when a large number of good shareholders is contaminated with a relatively small number of anonymous adversarial shareholders.

A solution is as follows. Shareholders prove that their computations and/or communications follow protocols correctly. Furthermore these shareholders do not compromise the confidentiality of their shares in the process.

Fortunately, there is a large body of algorithmic theory to wield on this problem. *Zero-knowledge* proofs address this issue allowing one party to prove the correctness of computation to another party while ensuring that no additional information is released. There are a number of variations on this theme.

Recently, [22] and [20] described robust function sharing protocols for RSA and [21] and [24] described robust sharing protocols for DSS and related signature functions.

Proactive Sharing

Ostrovsky and Yung [32] recently introduced the notion of *proactive* secret sharing protocols. Proactive protocols protect against a *mobile* adversary that may occupy and vacate shareholders and then move on to others. This adversary may occupy only up to $k - 1$ shareholders at once, but it may occupy any or all shareholders over the lifetime of the system. One example of such an adversary could be realized by a sequence of computer virus infections. Another could be a sequence of disgruntled ex-employees who have had access to share information.

A sequence of proactive threshold protocols has been announced in the last couple of years, including proactive threshold protocols for single-secret sharing [28], proactive threshold protocols for signatures such as DSS, Schnorr, and El Gamal [24], and for the RSA function [20].

Conclusions

The theory of threshold cryptography has benefited from years of model and protocol design and insight into future applications.

With the advent of certification authorities and electronic payment systems on the internet, threshold cryptography could be implemented widely within the next few years.

There is more threshold cryptography research to be done, however, including shared RSA key generation, more efficient protocols, and the development of new models as industrial needs develop.

The theory of threshold cryptography has benefited from years of model and protocol design and insight into future applications.

References

[1] N. Alon, Z. Galil, and M. Yung. Dynamic re-sharing verifiable secret sharing against a mobile adversary. In *Proceedings of 1995 European Symposium on Algorithms*, pages 523-537, Lecture Notes in Computer Science No. 979. Springer-Verlag, 1995.

[2] R. Blakley. Safeguarding cryptographic keys. FIPS Conference Proceedings (v.48), 1979, pages 313-317.

[3] R. Blakley, G.R. Blakley, A.H. Chan, and J.L. Massey. Threshold Protocols with Disenrollment. In Ernest Brickell, editor, *Proceedings of Crypto '92*, Lecture Notes in Computer Science No. 740, pages 540-548, Springer-Verlag, 1993.

[4] J. Boyar, D. Chaum, I. Damgard, and T. Pederson. Convertible undeniable signatures. In A.J. Menezes and S. Vanstone, editors, *Proceedings of Crypto '90*, Lecture Notes in Computer Science No. 537, pages 189-205, Springer-Verlag, 1991.

[5] S. R. Blackburn, M. Burmester, Y. Desmedt, and P. R. Wild. Efficient Multiplicative Sharing Schemes. In U. Maurer, editor, *Proceedings of Eurocrypt '96*, Lecture Notes in Computer Science 1070, pages 107-118, Springer-Verlag, 1996.

[6] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneous Broadcast. In *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, 1985, pages 335-344.

[7] R. Canetti and A. Herzberg. Maintaining security in the presence of transient faults. In Y. Desmedt, editor, *Proceedings of Crypto '94*, Lecture Notes in Computer Science No. 839, pages 425-438, Springer-Verlag, 1994.

[8] Y. Desmedt. Society and group-oriented cryptography: a new concept. In C. Pomerance, editor, *Proceedings of Crypto '87*, Lecture Notes in Computer Science No. 293, pages 120-127, Springer-Verlag, 1988.

[9] Y. Desmedt. Threshold Cryptography. European Transactions on Telecommunications, 5(4):449-457, July, 1994.

[10] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *Proceedings of the 26th ACM Symposium on the Theory of Computing*, pages 522-533, Santa Fe, 1994.

[11] Y. Desmedt, G. Di Crescenzo, and M. Burmester. Multiplicative non-abelian sharing schemes and their application to threshold cryptography. In J. Pieprzyk and R.Safavi-Naini, editors, *Proceedings – ASIACRYPT '94*, Lecture Notes in Computer Science 917, pages 21-32, Springer-Verlag, 1995.

[12] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Brassard, editor, *Proceedings of Crypto '89*, Lecture Notes in Computer Science No. 435, pages 307-315, Springer-Verlag, 1990.

[13] Y. Desmedt and Y. Frankel. Perfect zero-knowledge sharing schemes over any finite Abelian group. In R. Capocelli, A. De Santis, and U. Vaccaro, editors, *Sequences II (Methods in Communication, Security, and Computer Science)*, pages 369-378, Springer-Verlag, 1993.

[14] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In J. Feigenbaum, editor, *Proceedings of Crypto '91*, Lecture Notes in Computer

Science No. 576, pages 457-469, Springer-Verlag, 1990.

[15] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th annual Symposium on the Foundations of Computer Science*, pages 427-437, IEEE, 1987.

[16] Y. Frankel. A practical protocol for large group oriented networks. In J.J. Quisqater and J. Vandewalle, editors, *Proceedings of Eurocrypt '89*, Lecture Notes in Computer Science No. 434, pages 56-61, Springer-Verlag, 1990.

[17] Y. Frankel and Y. Desmedt. Parallel reliable threshold multisignature. Tech. Report TR-92-04-02, Department of EE & CS, University of Wisconsin-Milwaukee, April 1992.

[18] Y. Frankel, Y. Desmedt, and M. Burmester. Non-existence of homomorphic general sharing schemes for some key spaces. In Ernest Brickell, editor, *Proceedings of Crypto '92*, Lecture Notes in Computer Science No. 740, pages 549-557, Springer-Verlag, 1993.

[19] Y. Frankel, P. Gemmell, and M. Yung. Witness-based Cryptographic Program Checking and Robust Function Sharing. In *Symposium on the Theory of Computing, 1996*.

[20] Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Proactive RSA. Draft available at http://www.cs.sandia.gov/ psgemme/crypto/rpro.html.

[21] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In U. Maurer, editor, *Proceedings of Eurocrypt '96*, Lecture Notes in Computer Science 1070, pages 354-371, Springer-Verlag, 1996.

[22] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and Efficient Sharing of RSA Functions. In N. Koblitz editor, *Proceedings of Crypto '96*, Lecture Notes in Computer Science No. 1109, pages 157-172, Springer-Verlag, 1996.

[23] O. Goldreich, S. Micali, and A. Widgerson. How to plan any mental game. In *Proceedings of the 19th ACM Symposium on the Theory of Computing*, STOC, pages 218-229, 1987.

[24] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Public Key and Signature Systems. Available from http://theory.lcs.mit.edu/cis/cis-publications.html.

[25] M. Ito, A. Saito, and T. Nishizeki. Secret sharing schemes realizing general access structures. In *Proceedings of the IEEE Global Telecommunications Conference, Globecom '87*, 1987, IEEE Communications Soc. Press, pages 99-102.

[26] M. Jakobsson and M. Yung. On oblivious, agnostic, and blindfolded provers. In N. Koblitz editor, *Proceedings of Crypto '96*, Lecture Notes in Computer Science No. 1109, pages 186-200, Springer-Verlag, 1996.

[27] S. Jarecki. Proactive secret sharing and public key cryptosystems. Masters thesis, MIT, 1996

[28] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing, or: how to cope with perpetual leakage. In D. Coppersmith editor, *Proceedings of Crypto '95*, Lecture Notes in Computer Science No. 963, pages 339-352, Springer-Verlag, 1995.

[29] W. Jackson, K. Martin, and C. O'Keefe. Multisecret Threshold Schemes. In D. Stinson editor, *Proceedings of Crypto '93*, Lecture Notes in Computer Science No. 773, pages 126-135, Springer-Verlag, 1994.

[30] P. Karger. Limiting the damage potential of discretionary trojan horses. In *IEEE Symposium on Security and Privacy, 1987*, pages 32-37.

[31] H. Krawczyk. Secret Sharing Made Short. In D. Stinson editor, *Proceedings of Crypto '93*, Lecture Notes in Computer Science No. 773, pages 136-146, Springer-Verlag, 1994.

[32] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proceedings of the 10th ACM Symposium on the Principles of Distributed Computing*, 1991, pages 51-61.

[33] T.P. Pedersen. Distributed provers with applications to undeniable signatures. In D. Davies editor, *Proceedings of Eurocrypt '91*, Lecture Notes in Computer Science No. 547, pages 221-242, Springer-Verlag, 1991.

[34] T.P. Pedersen. A threshold cryptosystem without a trusted party. In D. Davies editor, *Proceedings of Eurocrypt '91*, Lecture Notes in Computer Science No. 547, pages 522-526, Springer-Verlag, 1991.

[35] T.P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, Lecture Notes in Computer Science, No. 576, pages 129-140, Springer-Verlag, New York, 1992.

[36] T. Rabin. Robust sharing of secrets when the dealer is honest or faulty. *Journal of the ACM*, 41(6):1089-1109.

[37] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the 21st annual Symposium on the Theory of Computing*, pages 73-85, ACM, 1989.

[38] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612-613, 1979.

[39] G.J. Simmons. How to (really) share a secret. In S. Goldwasser, editor, *Proceedings of Crypto '88*, Lecture Notes in Computer Science No. 403, pages 390-449, Springer-Verlag, 1989.

RC5 is Published as an Internet RFC

RC5 is now an official Internet Engineering Task Force Informational Standard (RFC).

Invented in 1994 by Professor Ronald L. Rivest of the Massachusetts Institute of Technology, RC5 is a fast and simple block cipher. Among the distinguishing feature of RC5 are the parameterized block size, the variable number of rounds that might be used during encryption, and the variable key length. By the careful selection of these parameters, RC5 can be adjusted to meet different application specific goals of security, performance, and exportability.

The RFC 2040 defines four ciphers with enough detail to ensure interoperability between different implementations. The first cipher is the raw RC5 block cipher. The RC5 cipher takes a fixed size input block and produces a fixed sized output block using a transformation that depends on a key. The second cipher, RC5-CBC, is the Cipher Block Chaining (CBC) mode for RC5. It can process messages whose length is a multiple of the RC5 block size. The third cipher, RC5-CBC-Pad, handles plaintext of any length, though the ciphertext will be longer than the plaintext by at most the size of a single RC5 block. The RC5-CTS cipher is the Cipher Text Stealing mode of RC5, which handles plaintext of any length and the ciphertext length matches the plaintext length.

For more details, the RFC can be fetched via ftp from *ftp://ds.internic.net/rfc/rfc2040.txt.*

Successor to DES Sought

Draft minimum acceptability requirements and draft criteria for the evaluation of candidates for the forthcoming Advanced Encryption Standard (AES) were published in the Federal Register on January 2, 1997. Additionally, the draft submission requirements for potential candidates were also announced for comment. An open, public workshop was scheduled by NIST for April 15, 1997, to discuss issues arising from the announcement.

It is intended that the AES will specify an unclassified, publicly disclosed encryption algorithm capable of protecting sensitive government information well into the next century. The recently published bulletin both describes the process of developing an AES and also invites comments from the public, manufacturers, voluntary standards organizations, and federal, state, and local government users so that their needs can be considered.

It is intended that the AES will specify an unclassified, publicly disclosed encryption algorithm capable of protecting sensitive government information well into the next century.

While the initial publication in the Federal Register highlights many interesting and relevant issues, it seems that both the technical requirements for proposals and the procedures by which the AES effort is intended to move forward still require more detail to avoid potential ambiguities.

As the AES initiative continues, it's progress will be closely reported by *CryptoBytes.*

Progress on P1363 Continues

The IEEE P1363 project, "Standard for Public-Key Cryptography," is moving ahead towards balloting the standard.

The standard provides comprehensive treatment of three families of public-key techniques: those based on the discrete logarithm problem over finite fields (such as Diffie-Hellman), on discrete logarithm problem over elliptic curve groups (such as ECDSA), and on the integer factoring problem (such as RSA). The August 1996 meeting (which followed the Crypto '96 conference) generated a lot of interest in the cryptographic community, and the working group has received much desired feedback, including new proposals for inclusion into the standard.

In order to give the new proposals the thorough consideration they deserve, the working group decided at its November 1996 meeting to issue the standard in two parts. The first part, which is close to completion, was discussed in detail at the March 24-26 meeting in Auburn, Alabama. At the same time, work on the second part, to be issued later as an addendum to the standard, is continuing.

The IEEE P1363 project, "Standard for Public-Key Cryptography," is moving ahead towards balloting the standard.

The working group plans to ballot the first part of the standard during 1997. A draft of the first part will be presented at the May 1997 meeting which follows the Eurocrypt '97 conference. Information on the contents of the first part and the proposals for the addendum, as well as other information on the project, can be found at the working group's website at *http://stdsbbs.ieee.org/groups/1363/.*

The RC5 Encryption Algorithm: Two Years On

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

The RC5 encryption algorithm was designed by Professor Ron Rivest and first published in December 1994 [10]. Since then, RC5 has attracted the attention of many researchers in the cryptographic community in attempts to accurately assess the security offered. In this article, we give a brief summary of the known cryptanalytic results on RC5 and we assess the current status of the cipher.

Features of RC5

Two of the most distinguished features of RC5 are the heavy use of data-dependent rotations and the exceptionally simple encryption routine.

RC5 is a fast block cipher designed to be suitable for both software and hardware implementation. It is a parameterized algorithm with a variable key size, a variable block size, and a variable number of rounds. This provides the opportunity for great flexibility in both the performance characteristics and the level of security offered. Two of the most distinguished features of RC5 are the heavy use of data-dependent rotations and the exceptionally simple encryption routine. The former feature has been shown to be useful in preventing certain advanced types of attack, while the latter feature makes RC5 both easy to implement, and very importantly, more amenable to analysis than many other block ciphers.

Overview of Cryptanalytic Results

Several techniques have been developed for analyzing the security of block ciphers, including *exhaustive key search attack*, *statistical tests*, *differential cryptanalysis* [2] and *linear cryptanalysis* [8]. (See [11] for detailed discussions.) The last two types of attack, both considered substantial advances in recent years, are more sophisticated techniques for block cipher analysis. For differential cryptanalysis, the basic idea is to choose two plaintexts with a certain difference between them so that the resulting ciphertexts have a difference with a specific value with a probability that is better than we might expect. Such a pair of differences (which lead to the concept of a "*characteristic*") is useful in deriving certain bits of the key. For linear cryptanalysis, the basic idea is to find a linear relation among bits of plaintext, ciphertext, and key which hold with a probability that is not equal to $^1/_2$. Such a "*linear approximation*" can potentially be used to obtain information about the key.

Yiqun Lisa Yin is a research scientist at RSA Laboratories. She can be contacted at yiqun@rsa.com.

The first cryptanalytic results on RC5 were given by Kaliski and Yin [3] at Crypto'95 (a summary can be found in [4]). By analyzing the basic structure of the encryption routine as well as the properties of data-dependent rotations, it is possible to construct differential characteristics and linear approximations of RC5 that are useful for mounting differential and linear attacks. Both styles of attack are quite effective on RC5 with a very small number of rounds, but the plaintext requirements increase quickly as the number of rounds grows. Their results, further extended in [5], show that the use of data-dependent rotations and the incompatibility between the different arithmetic operations used in encryption help prevent both differential and linear cryptanalysis.

At Crypto'96, Knudsen and Meier [6] presented nice improvements over Kaliski and Yin's differential attack [3] by a careful analysis of the relations between input, output, and the subkeys used in the first two rounds of encryption. Even though the characteristics used in their attack are essentially the same as in [3], they were able to improve the plaintext requirements by a factor of up to 512 by exploiting the characteristics in an innovative and sophisticated way. They also considered the existence of certain weaker keys for RC5 with respect to which their attack can be further enhanced.

Moriai, Aoki, and Ohta [9] have investigated the strength of RC5 against linear cryptanalysis by focusing on the bias of linear approximations for *fixed* keys, rather than the average bias over *all* possible keys which is the customary model for linear cryptanalysis. They also considered a mini-version of RC5 with much reduced word size and computed the percentage of keys that yield ciphers less resistant to linear cryptanalysis than the average case analysis might suggest [3]. While interesting, their work has little practical impact.

As of this writing, the differential attack described in [6] and the linear cryptanalytic attack described in [3] offer the best avenues for the sophisticated cryptanalyst. A summary of the data requirements for a successful attack against RC5 with a varying number of rounds is provided at right. Note that the second row of both tables have been derived from the first by using the simple fact [2] that a differential attack *with m* chosen plaintexts can be converted into a known plaintext attack which requires approximately $2^w(2m)^{1/2}$ known plaintexts where the block size is $2w$. While most of these attacks are impractical anyway, we have used ">" to denote when the attack is impossible even at a theoretical level.

In late 1995, Kocher [7] developed what are called *timing attacks* that are generally applicable to many cryptosystems. In such an attack, an opponent tries to obtain information about the secret key (or private key) by recording and analyzing the time used for cryptographic operations that involve the key. Kocher observed that RC5 may be subject to timing attacks if RC5 is implemented on platforms for which the time for computing a single rotation is proportional to the rotation amount. If RC5 is implemented on platforms with a constant rotation time (which is true for many platforms), then RC5 is actually completely resistant to timing attacks since encryption of any plaintext would take a constant time.

With regards to the less sophisticated brute-force attack of trying each key in turn, the security of RC5 is obviously dependent on the length of the encryption key that is used (as is the case with all ciphers). RC5 has the attractive feature that the length of the key can be varied (unlike the situation with DES for instance) and so the level of security against these attacks can be tuned to suit the application. With the launch of the RSA Data Security Secret-Key Challenge (see elsewhere in this issue of *CryptoBytes* for more details on this challenge) it is hoped that the resistance of ciphers to exhaustive key search attacks can be more accurately gauged in the future. Some of the posted challenges, such as RC5 encryption with a 40- and 48-bit key were solved very quickly, as was expected. But some of the longer key lengths are likely to remain an unsolved challenge for some considerable time to come!

Status of RC5

Early results on the cryptanalysis of RC5 have been very encouraging. With the cipher receiving considerable attention from cryptanalysts worldwide, a picture of the security offered by RC5 has been quick to develop. Acceptance of the cipher is growing, and RC5 has been discussed for inclusion in various standards efforts and has been published by the IETF in RFC 2040 [1]. Two years on, it seems that the RC5 encryption algorithm offers a computationally inexpensive way of providing secure encryption.

64-bit block size ($w = 32$)

rounds	**4**	**6**	**8**	**10**	**12**	**14**	**16**
differential cryptanalysis (chosen plaintext)	2^{17}	2^{24}	2^{35}	2^{46}	2^{54}	2^{63}	>
differential cryptanalysis (known plaintext)	2^{41}	2^{45}	2^{50}	2^{56}	2^{60}	>	>
linear cryptanalysis (known plaintext)	2^{40}	2^{60}	>	>	>	>	>

128-bit block size ($w = 64$)

rounds	**4**	**8**	**12**	**16**	**20**	**24**	**28**
differential cryptanalysis (chosen plaintext)	2^{19}	2^{42}	2^{58}	2^{83}	2^{106}	2^{123}	>
differential cryptanalysis (known plaintext)	2^{74}	2^{86}	2^{94}	2^{106}	2^{118}	>	>
linear cryptanalysis (known plaintext)	2^{47}	2^{95}	2^{119}	>	>	>	>

References

[1] R. Baldwin and R. Rivest. *RFC 2040: The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms*. October 30, 1996. Available at ftp://ds.internic.net/rfc/rfc2040.txt.

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

[3] B.S. Kaliski and Y.L. Yin. On differential and linear cryptanalysis of the RC5 encryption algorithm. In *Advances in Cryptology—Crypto '95*, pages 171–183, Springer-Verlag, 1995.

[4] B.S. Kaliski and Y.L. Yin. On the security of the RC5 encryption algorithm. *CryptoBytes* Vol.1, No.2, pages 13–14, 1995.

[5] B.S. Kaliski and Y.L. Yin. On the security of the RC5 encryption algorithm. RSA Laboratories Technical Report. In preparation, 1997.

[6] L. Knudsen and W. Meier. Improved differential attacks on RC5. In *Advances in Cryptology— Crypto '96*, pages 171–183, Springer-Verlag, 1996.

[7] P. Kocher. Cryptanalysis of Diffie-Hellman, RSA, DSS, and other cryptosystems using timing attacks. In *Advances in Cryptology—Crypto '96*, pages 171–183, Springer-Verlag, 1995.

[8] M. Matsui. Linear cryptanalysis method for DES cipher In *Advances in Cryptology—Eurocrypt '93*, pages 386–397, Springer-Verlag, 1994.

[9] Moriai, K. Aoki, and K. Ohta. Key-dependency of linear probability of RC5. March 1996. To appear in *IEICE Trans. Fundamentals.*

[10] R. Rivest. The RC5 encryption algorithm. In *Proceedings of 2nd Workshop on Fast Software Encryption*, pages 86–96, Springer-Verlag, 1995.

[11] M.J.B. Robshaw. *Block Ciphers.* Technical Report TR-601, version 2.0, RSA Laboratories, July 1995.

Two years on, it seems that the RC5 encryption algorithm offers a computationally inexpensive way of providing secure encryption.

The RSA Data Security Factoring Challenge

RSA Laboratories recently augmented the list of challenge numbers offered as part of the RSA Data Security Factoring Challenge. Launched in 1992, the Factoring Challenge aims to keep track of the progress of factoring by providing financial prizes to those that factor different types of numbers. One set in particular consists of different length numbers of the type that would be suitable for use as RSA moduli. These are considered to be the hardest numbers to factor for a given length.

Recent factoring achievements such as the factorizations of RSA-129 and RSA-130 have focused attention on the RSA Challenge numbers, particularly so since improvements in factoring appear to be bringing a 512-bit RSA number within reach. In reponse to requests to add numbers of "landmark" lengths, RSA Laboratories has added RSA-155 (512 bits), RSA-232 (768 bits), RSA-309 (1024 bits) and RSA-617 (2048 bits) to the list of RSA Challenge numbers. The full list can be obtained by sending E-mail to *challenge-rsa-list@rsa.com* and more information can be found via *http://www.rsa.com/rsalabs/*.

The RSA Data Security Secret-Key Challenge

The RSA Secret-Key Challenge, launched on January 28, 1997, consists of one DES challenge and twelve contests based around the block cipher RC5. The aim is to demonstrate by example the resistance of symmetric ciphers with different key lengths to exhaustive search attack.

The first two RC5 contests, denoted RC5-32/12/5 and RC5-32/12/6, which used a 40- and 48-bit key respectively, have already been solved but the remaining ten RC5 contests and the DES contest remain open. The prize for solving each challenge is $10,000.

More complete contest details are available via *http://www.rsa.com/*.

In this issue:

For contact and distribution 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

FIRST CLASS
U.S. POSTAGE
PAID
MMS, INC