VOLUME 4, NUMBER 2 — WINTER 1999

RSA LABORATORIES'

CryptoBytes

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

Contents

1
Breaking DES

2
Editor's Note

3
On the Transition from DES to AES

6
First Advanced Encryption Standard (AES) Candidate Conference

15
Attacking Elliptic Curve Cryptosystems Using the Parallel Pollard rho Method

16
Announcements

RSA Laboratories
A Division of RSA Data Security

Breaking DES

Paul C. Kocher
Cryptography Research, Inc.
870 Market Street, Suite 1088
San Francisco, CA 94102, USA

Introduction

There are 72,057,594,037,927,936 possible DES keys. On July 13, 1998 RSA Data Security published a secret DES-encrypted message. Using a custom-built keysearch machine nicknamed Deep Crack, our team found the correct key, decoded the message, and claimed RSA's $10,000 prize less than three days later.

Several academic papers have estimated the cost of building a machine to break DES using a brute force search. Critics trying to justify short key length restrictions imposed by export controls have claimed that these papers do not properly account for failures, heat dissipation problems, and development costs.

John Gilmore of the Electronic Frontier Foundation (EFF) and I had several discussions in early 1997 about actually building such a machine. Approaches using FPGAs, microprocessor arrays, and networks of low-cost PCs were prohibitively expensive, but our projections showed that low-volume ASIC fabrication processes were well-suited to the task. John then arranged $250,000 in financing for the project from the EFF - a significant feat. My company, Cryptography Research, donated time for the system architecture design, chip emulator, and control software implementation. We contracted with AWT, a chip design company, to build the machine, while John provided project leadership. Our final budget included $80,000 for AWT's labor and $130,000 for chip fabrication and other materials.

Machine Architecture

The system design is built around "search units," DES circuits that independently test keys and stop whenever a possible match is found. Each chip contains 24 identical search units operating at 40MHz. Each DES circuit performs one round per clock cycle with no overhead, so a chip can test 60 million keys per second. We made 28 custom-built circuit boards, each containing 64 chips plus glue logic to manage the computer interface, provide the system clock, etc. A simple bus connects the boards to an ordinary Pentium PC running Linux, which can read and write the registers of each chip by selecting the appropriate board, chip, and address.

To run a search, the computer performs a set of diagnostic tests then loads the search array with a problem. As the chips are searching, the computer

Paul Kocher is President and Chief Scientist of Cryptography Research, Inc. He can be contacted by e-mail at paul@cryptography.com. His web page is http://www.cryptography.com/paul.

(continued on page 3)

Editor's Note

About RSA Laboratories

An academic environment within a commercial organization, RSA Laboratories is the research and consulting division of RSA Data

Breaking DES

Continued from page 1

rapidly polls each search unit to see if it has halted on a candidate key. If a search unit is stopped, the PC tests whether its key is correct and, if not, restarts the unit. This architecture makes it easy to add new search units or to bypass defective ones, making the system highly scalable and fault-tolerant.

To test a key, a search unit loads its DES engine with the value from a 56-bit key register and decrypts the first ciphertext. The chip's search parameters specify which byte values can appear in the plaintext. If any unselected values appear in the computed plaintext, the key register is incremented and the search continues. If all 8 bytes are valid, a second ciphertext is tested using the same key. If the second plaintext also matches the search criteria, the search unit stops. In the actual chip, decryption, plaintext testing, and key updates are performed in parallel to keep the DES engine running at full speed if no matches are found. The plaintext testing process is customizable to accommodate DES modes such as cipher block chaining.

For example, a message consisting of encrypted ASCII text can contain only about 70 of the 256 ASCII characters. The 70 bits corresponding to these characters can be selected in each chip's 256-bit plaintext vector. Decrypting with an incorrect key produces an effectively random plaintext, so the expected number of incorrect keys that will decrypt both ciphertexts to form plaintexts containing only these characters is $(\frac{70}{256})^{16}(2^{56})$, or about 70 million. These keys are then tested by the PC.

For a traditional known plaintext attack, both ciphertext registers are loaded with the encrypted message, and the bit corresponding to each plaintext byte value is set in the plaintext vector. Even if all 8 plaintext bytes are different, only $(\frac{8}{256})^{8}(2^{56})$, or 65536 false positives are expected. In general, the search parameters need to be chosen such that search units will stop at the correct key without overloading the bus and host PC with false positives.

Building the Machine

AWT implemented the chip using VHDL, a com-

On the Transition from DES to AES

A Report Prepared by the X9F3 Working Group

Editor's note: As part of encouraging a broader dialogue about the impact of recent DES-cracking efforts, RSA Laboratories has asked the X9F3 committee, which develops DES-based standards for the financial services industry, to comment in this issue of CryptoBytes. *X9F3's statement follows.*

The X9 Standards Committee

ASC X9 is an Accredited Standards Committee that produces standards, guidelines, and technical reports for the financial services industry. The oversight of American National Standards Institute (ANSI) provides the independent accreditation process needed to ensure that X9's American National Standards (ANS) are consensus documents. X9 represents the Financial Industry in the United States and serves as the official representative to the International Standards Organization (ISO) for a similar financial industry standards organizations within ISO called Technical Committee 68, Banking and Related Services.

The X9F Standards Subcommittee

The Data and Information Security subcommittee of X9 is X9F. X9F produces standards, guidelines, and technical reports that are incorporated into products and services offered by the financial services industry (such as payment card ATM and POS systems, electronic funds transfer networks, and the Automated Clearing House). Banks across the US implement X9F standards for data and information security within their enterprises and in support of external business relationships (e.g., customer-to-bank, and bank-to-bank communications. X9F's domestic standards form the basis of many ISO TC68 security standards.

The X9F subcommittee and its working groups are composed of volunteers from financial services organizations, cryptographic security vendors, consultants, and the U.S. government. Active liaison groups include the American Bankers Association, IETF, ISO, and IEEE.

Migration to Triple DES and AES

In response to the needs of the financial services industry, and more recently, new brute force methods of attack on the data encryption algorithm, the X9F subcommittee will accelerate its strategy to:

1) Require algorithm independence, and dynamic algorithm and key management in any new X9F standards so that they can accommodate the triple Data Encryption Algorithm (TDEA) as well as the algorithm to be chosen from the Advanced Encryption Standard (AES) development effort. The AES is a National Institute of Standards and Technology (NIST) initiative aimed at producing the successor algorithm to the Data Encryption Algorithm (DEA). See the AES home page at *http://csrc.nist.gov/encryption/aes/aes_home.htm.*

2) Update certain single DEA based standards (such as ANS X9.26, Secure Sign On to provide a triple Data Encryption Algorithm (TDEA) option for interoperability).
3) Withdraw certain single DEA based standards (such as X9.23, Encryption), probably at the time they would normally be due for 5 year review, without causing disruption for current business applications
4) Promote the early adoption of triple Data Encryption Standards (TDES) and guidelines, as well as migration to AES as required.
5) Promote the continued use of Unique Key Per Transaction combined with single DEA PIN encryption as specified in X9.24, Key Management, used for POS and ATM transactions. X9.24 was recently modified to require the use of TDEA for encrypting cryptographic keys.

X9's Triple DES Standards

The following is a list of TDES standards and guidelines:

1) ANS X9.52, Triple Data Encryption Algorithm (TDEA), Modes of Operation shows how to securely and efficiently perform multiple iterations of the DEA.
2) Draft ANS X9.65, Triple Data Encryption Algorithm (TDEA) Implementation will show how to securely implement TDEA and apply TDEA to other X9 standards, such as ANS X9.17, Key Management.
3) Draft ANS TG-19, Modes of Operation Validation System for Triple Data Encryption Algorithm (TMOVS) contains validation procedures needed to test implementations of ANS X9.52. It is expected that the three NIST validation laboratories will use TG-19 in providing testing services for the vendors implementing TDES standards.

Other X9F standards include:

1) Draft X9.66, Security of Cryptographic Modules, which is based on the government's Federal Information Processing Standard, FIPS 140-1.
2) Draft X9.42, Public Key Cryptography for the Financial Service Industry: Agreement of Symmetric Keys on Using Diffie-Hellman and MQV Algorithms
3) Draft X9.44, Management of Symmetric Algorithms Keys Using Reversible Public Key Cryptography
4) Draft X9.63, Key Agreement and Key Management Using Elliptic Curve-Based Cryptography

For information about joining X9, please contact the X9 Secretariat, Cindy Fuller at *cfuller@ABA.com*. Also visit the X9 website at *www.X9.org*.

piled C-like language for ASIC design. (Most ASICs are produced today using high-level languages.) In parallel, Cryptography Research produced a complete software emulator of the chip, which we tested against the VHDL using a simulator before sending the final netlist to the manufacturer.

For a larger machine, we would have dedicated more resources to optimizing the design. For example, we reused a VHDL DES core I had previously helped AWT to develop. A hand-tuned pipelined DES would have increased the search capacity of the chip, but would have also increased design costs. We chose an inexpensive manufacturing process; an organization building several machines or with a larger budget could have obtained larger chips with higher chip density and lower power consumption. As a general rule, I estimate that doubling the amount spent on a keysearch machine should produce a 4-fold increase in performance.

Debugging

Soon after the first sample chips arrived, we had one running and finding known keys. We identified a few minor problems—the analog signal strength of the clock lines was too low at full speed and there was a bug in the chip I/O circuits, which AWT fixed with changes to the circuit board design.

Power consumption estimates for ASICs are often highly inaccurate, and our chip ran dangerously hot when clocked anywhere near full speed. Fortunately, CMOS logic operates well at reduced voltages. Inserting large diodes in series with the power supplies reduced the power consumption by lowering the input voltage. (The machine still overloaded AWT's air conditioning system and blew circuit breakers at EFF's headquarters.)

Additional testing uncovered a manufacturing flaw in most search units on each chip that occasionally causes incorrect results. In the completed machine, there is a slight chance that flaky search units will miss the correct key. If the entire search completes without a match, the host computer has to re-search key regions originally assigned to these search units.

Manufacturers normally guarantee that test vectors that work on a simulator will work with the finished chips, but our fab failed to get any of our test vectors to work. In order to have the machine ready for the RSA challenge, we decided to accept the chips anyway and rely on the host PC's software diagnostic capability to detect and bypass bad search units.

Results

RSA's DES challenge started at 9:00AM on July 13. Within minutes, we started the machine with 26 boards and soon added a 27th, for a total of 1728 chips containing 37050 good search units searching 92.6 billion keys per second.

Fifty-six hours later, after searching almost exactly a quarter of the total key space, the machine found the correct key 3ECDA15E704FB31C and automatically e-mailed it to RSA. (DES keys are normally written with a parity bit in each byte. In 56-bit form, the key is 3F9A82F709EC8E.) We could now read RSA's encrypted message: "It's time for those 128-, 192-, and 256-bit keys".

At Crypto '97, Matt Blaze offered a prize to anyone who could find a DES key and message such that all plaintext bytes are equal and all ciphertext bytes are equal. So far we have found two solutions: encrypting 8787878787878787 with the key 0E329232EA6D0D73 produces the ciphertext 0000000000000000, and encrypting 9E9E9E9E9E9E9E9E with the key 3E1A20832504A7FE produces the ciphertext 0101010101010101.

Conclusions

I predict that the performance/cost ratio for brute force machines should double about every 12 months. Moore's law predicts that transistor densities will double every 18 months, and chip speeds increase at a similar rate. Unlike microprocessors, circuit complexity and design costs for keysearch machines do not increase much at higher clock rates. Scalability and fault tolerance advantages are partially offset, however, by heat dissipation problems. (Cryptographic computations involve a large number of state transitions, which lead to unusually high power consumption.) Programmable chip technologies such as FPGAs are also improving, and will soon be fast enough for projects of this type.

As a result, 56 bit keys are no longer appropriate for systems that must resist determined adversaries. A machine with the same speed as ours would take an average of just 5.9 seconds to break 40-bit keys - fast enough for real-time decryption of network traffic. Even 64-bit keys, proposed as a replacement for current 56-bit keys, are only 256 times better than DES. Using my assumption that doubling the machine budget yields a 4-fold performance improvement, a machine 256 times as large as ours would cost $3.36 million today, and improvements in chip manufacturing will rapidly reduce this figure.

Our results do not impact triple DES. Breaking a 2-key triple DES system using exhaustive search would require a machine 2^{56} (over 72 million billion) times as large—ignoring the fact that triple DES operations take 3 times longer.

Most in industry have responded appropriately. While it would be irresponsible to deploy new single DES systems for protecting high-value data, the risks of staying with single DES in the short term are often manageable. In some companies, upgrades and new deployments using triple DES or other algorithms are now being combined with Y2K efforts.

Even today, I am thoroughly impressed with the DES algorithm. Although linear and differential attacks are academically important, it is remarkable that brute force is the only practical attack after more than 23 years of research.

Although linear and differential attacks are academically important, it is remarkable that brute force is the only practical attack after more than 23 years of research.

Source code and design information are published by O'Reilly in *Cracking DES* (order online at *http://www.oreilly.com/catalog/crackdes*). For more information see *http://www.cryptography.com/des*, *http://www.eff.org/descracker*, and *http://www.awti.com*.

FIRST ADVANCED ENCRYPTION STANDARD (AES) CANDIDATE CONFERENCE

Edward Roback Morris Dworkin
National Institute of Standards and Technology (NIST)

Introduction

On August 20-22, 1998, two hundred members of the global cryptographic research community gathered in Ventura, CA for the First Advanced Encryption Standard (AES) Candidate Conference (AES1). The conference focused on fifteen cryptographic algorithms being considered for the Federal Government's Advanced Encryption Standard. Sponsored by the National Institute of Standards and Technology's (NIST) Information Technology Laboratory, AES1 provided an opportunity for the submitters of candidate algorithms to brief their proposals and answer initial questions. The purpose of the conference was to introduce participants in the analysis and evaluation process to the various candidate algorithms. This conference served as the formal kick-off of the first AES public evaluation and analysis period ("Round 1"), which runs through April 15, 1999.

[...] with recent successful key exhaustive attacks, the useful lifetime of DES is now drawing to a close.

1. Background and Context: the Advanced Encryption Standard Development Process

Since 1977, NIST's Data Encryption Standard (DES) [4] has been the Federal Government's standard method for encrypting sensitive information. In addition, it has gained wide acceptance in the private sector and has been implemented in a wide variety of banking applications. The algorithm specified in this standard has evolved from solely a U.S. Government algorithm into one that is used globally. However, with recent successful key exhaustive attacks, the useful lifetime of DES is now drawing to a close. Anticipating this eventuality, in 1996 NIST officials began preparing for development of a successor standard. In outlining these plans, NIST sought to construct an open process to engage the cryptographic research community and build confidence in the successor algorithm.

On January 2, 1997, NIST announced the initiation of a process to develop the AES [1], which would specify the Advanced Encryption Algorithm (AEA) and serve as an eventual successor to the venerable DES. Basic criteria that candidate algorithms would have to meet were proposed, in addition to required elements in the nomination packages to be submitted to NIST. Over thirty sets of comments were received from U.S. Government agencies, vendors, academia, and individuals. Additionally, NIST sponsored an AES workshop on April 15, 1997 to discuss the comments received and obtain additional feedback to better define the request for candidate algorithms. This input was of great assistance to NIST in preparing its formal call for algorithms and evaluation criteria.

On September 12, 1997, NIST published its formal call for algorithms. [2] Candidate algorithms had to meet three basic requirements: 1) implement symmetric (secret) key cryptography, 2) be a block cipher, and 3) support cryptovariable key sizes of 128, 192 and 256 bits with a block size of 128 bits. The algorithm could also support additional key and block sizes. In addition to the above requirements, submitters had to provide the following:

1. Complete written specifications of the algorithm,
2. Statements of the algorithm's estimated computational efficiency,
3. Known answer test values for the algorithm, and code to generate those values,
4. Statement of the algorithm's expected cryptographic strength,
5. Analysis of the algorithm with respect to known attacks,
6. Statement of advantages and limitations of the algorithm,
7. Reference implementation of the algorithm, specified in ANSI C,
8. Optimized implementations specified in Java™ and ANSI C, and
9. Signed statements that a) identified any pertinent patents and patent applications and b) provided for the royalty-free use of that intellectual property should the candidate selected be selected for inclusion in the AES.

Edward Roback is a Computer Specialist at NIST. He can be contacted via e-mail at edward.roback@nist.gov. Morris Dworkin is a Mathematician at NIST. He can be contacted via e-mail at dworkin@csmes.ncsl.nist.gov.

In its call for candidates, NIST made clear that security would be the most important criterion by which algorithms are evaluated, followed by efficiency and other characteristics. In the spirit of DES' success, NIST's goal in the AES development effort is to specify an algorithm that will have a lifetime of at least thirty years, that will be used extensively throughout the U.S. Government, and that will be also be available in the private sector, on a royalty-free basis worldwide.

Twenty-one algorithms were submitted to NIST by the June 15, 1998 deadline. After review, NIST determined that fifteen of these met the minimum acceptability requirements and were accompanied by a complete submission package. These algorithms were made public by NIST on August 20, 1998 at AES1 for the first evaluation period. At the conference, submitters of the fifteen candidate algorithms were invited to provide briefings on the candidates and answer any initial questions. NIST also announced its request for comments on the candidates, due April 15, 1999. These comments will help NIST narrow the field of candidates to approximately five or fewer for the second round of public evaluation. The public analysis of the candidates will be the subject of the Second AES Candidate Conference (AES2), scheduled for March 22-23, 1999. Following its study of the second round analysis, NIST intends to select one algorithm (or possibly more than one, if warranted) to be proposed for inclusion in the AES.

2. Conference Purpose, AES Development Overview, Announcement of Candidates and Review of Evaluation Criteria

Mr. Miles E. Smid, Manager of the Security Technology Group of the Computer Security Division in NIST's Information Technology Laboratory, welcomed the AES1 participants and noted that the primary purpose of the conference was to provide an opportunity for each of the submitters to formally present their candidate algorithms and design philosophy. After sketching the history of the AES development process he noted that NIST received and reviewed twenty-one packages. In each case, NIST checked whether: 1) the legal documents were completed; 2) the submissions were responsive to all requirements; and 3) the given code, when run, passed the Known Answer Test." Six of the packages were incomplete; thus, fifteen candidates were formally accepted into the AES development process. Mr. Smid noted NIST did not perform any cryptanalysis and, therefore, acceptance by NIST of an algorithm into the process did not signify anything regarding the strength of a candidate. A list of the six incomplete submissions was read to the audience and posted to NIST's AES website.

Mr. Smid then formally unveiled the accepted candidates, as seen on the following page in Table 1.

Mr. Smid then briefly reviewed the principal goals NIST has for the AES, as discussed above. The AES should be more secure and efficient than Triple DES. He noted that some of the submitters were claiming efficiency performance for their candidate to be greater than that of *single* DES.

Mr. Edward Roback, Computer Specialist in NIST's Computer Security Division, then proceeded to review the AES evaluation criteria, NIST's plans to foster discussion of the candidates, issues regarding submitting formal comments to NIST, and its plans for efficiency testing of the algorithms.

When NIST published its call for algorithms, it included a listing of the evaluation criteria by which NIST intends to make the AES selection. This was done for two reasons: 1) to aid the submitters in understanding the qualities important to NIST, and 2) to ensure that the criteria were well understood and available beforehand to avoid any possible questions of bias. There are three major categories to NIST's

Computational efficiency (i.e., speed) is also a cost consideration. NIST tests the candidate algorithms on a common platform to compare performance characteristics.

provided by NIST identifying any known intellectual property (i.e., patents or patent applications) that may be infringed by the practice of the particular candidate. If such property was identified, the owner of the intellectual property had to agree in writing to allow for its worldwide royalty-free use, should the candidate be included in the AES. (Use of the algorithms *for the purposes of AES evaluation* also had to be granted.) NIST hopes to address any other intellectual property issues that may arise during the public comment process before selecting the AES algorithm.

Computational efficiency (i.e., speed) is also a cost consideration. NIST tests the candidate algorithms on a common platform to compare performance characteristics. In the first AES evaluation round, this will focus primarily upon the 128-bit key size, while in the second round (with about fivecandidates), this will be expanded to include the 192- and 256-bit key sizes and hardware performance estimates. Memory requirements (e.g., for code, necessary memory, and so forth) will also be measured. While NIST will conduct some of this analysis, it also welcomes the submission of such analysis by other parties.

[...] NIST has established electronic discussion pages for each candidate as well as other relevant AES topics

Algorithm and implementation characteristics include flexibility, hardware and software suitability, and additional features offered by a candidate algorithm. For example, an algorithm may support block sizes other than the required 128-bits and key sizes other than the required 128-,192-, and 256-bits Additionally, some candidates may be designed to facilitate efficient implementation on a wider variety of platforms or in diverse applications. For example, the ability to use the AES in 8-bit processor smart cards with strict memory limitations has often been cited by potential users as desirable. Simplicity of design is also a factor. If an algorithm's construction is straightforward and easier to analyze, it will likely have an edge over an unnecessarily complex design.

In order to facilitate informal discussion of the candidates and to aid NIST in following the expected on-going analysis, NIST has established electronic discussion pages for each candidate as well as other relevant AES topics (e.g., intellectual property). These are intended to aid interaction among parties evaluating particular algorithms or discussing other aspects of the AES process. It is also intended to provide a focal point for each of the fifteen submitters to monitor public review of their candidates. The groups should also provide a way for evaluators to receive feedback on their ideas prior to submitting official formal public comments to NIST. Mr. Roback encouraged submitters to participate in these discussions at their discretion. NIST also welcomes suggestions for other topical discussion groups. All postings to these discussion groups will be publicly available on-line at *http://www.nist.gov/aes*.

Turning to NIST's solicitation of public analysis and comments of the algorithms, Mr. Roback said that NIST seeks comments on all aspects of the candidates. Comments on the algorithms as viewed against the evaluation criteria are anticipated to be the subject of a majority of the public comments. Intellectual property is another area in which comments would be useful to NIST, especially claims of intellectual property that were not known to the submitters. Analysis of the entire field candidates would also be useful (e.g., comparison of all fifteen algorithms against a particular cryptanalytic attack or efficiency testing on a common platform). Finally, NIST is seeking overall recommendations with justifying comments regarding which candidates should be selected as finalists. NIST intends to invite the submitters of particularly useful, novel or insightful comments to brief at AES2. NIST will accept formal public comments through April 15, 1999; however, comments should be received by February 1, 1999 for consideration for the AES2 program. All formal comments will be part of the official public record. E-mail comments will be accepted at *AESFirstRound@nist.gov*.

In order to have *at least* one set of comparable efficiency test values for all fifteen candidates, NIST will measure the efficiency of the optimized ANSI C and Java implementations on a IBM-compatible PC/ Intel Pentium-pro Processor (200 MHz), with 64-MB RAM. NIST will conduct tests on other platforms with various compilers, as time and resources permit. NIST also intends to test ciphertext for randomness to measure the timings of algorithm setup, key setup, key change, encryption, and decryption, where applicable to each algorithm. Mr. Roback emphasized that NIST is conducting these tests to ensure the existence of at least one set of efficiency measures of the entire field of candidates. Other such measurements, on different platforms, including dif-

ferent computer languages or using different compilers, would be welcomed by NIST.

Next, Mr. James Foti, a mathematician with NIST's Security Technology Group, explained the contents of the two CDROMs published by NIST. The first, entitled *CD-1: Documentation* contains algorithm specifications, supporting documentation, and intellectual property information. It is not subject to U.S.

Country of Origin	Algorithm	Submitter(s)
Australia	LOKI97	Lawrie Brown, Josef Pieprzyk, Jennifer Seberry
Belgium	RIJNDAEL	Joan Daemen, Vincent Rijmen
Canada	CAST-256	Entrust Technologies, Inc.
	DEAL	Richard Outerbridge, Lars Knudsen
Costa Rica	FROG	TecApro Internacional S.A.
France	DFC	Centre National pour la Recherche Scientifique (CNRS)
Germany	MAGENTA	Deutsche Telekom AG
Japan	E2	Nippon Telegraph and Telephone Corporation (NTT)
Korea	CRYPTON	Future Systems, Inc.
USA	HPC	Rich Schroeppel
	MARS	IBM
	RC6	RSA Laboratories
	SAFER+	Cylink Corporation
	TWOFISH	Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson
UK, Israel, Norway	SERPENT	Ross Anderson, Eli Biham, Lars Knudsen

Table 1: Accepted AES Candidates

DFC (Decorrelated Fast Cipher) is a Feistel network with eight rounds.

round function. For 128-bit keys or 192-bit keys there are six rounds; for 256-bit keys there are eight rounds. After the final round, the two halves of the data word are not "unswapped," which introduces a slight asymmetry between encryption and decryption besides the order of the round keys. The key schedule expands the user key by repetition, XORs it with constant offset values, and encrypts it with DES in the Cipher Block Chaining mode under a fixed key.

The presenter, Richard Outerbridge, portrayed DEAL as a sensible evolution of the well-studied DES, surpassing the security of triple DES, and avoiding the weaknesses of DES and triple DES. The key schedule was chosen to avoid equivalent keys, related keys, and the complementation property. He emphasized that DEAL could be efficiently implemented on many platforms "almost overnight," because DES has already been extensively deployed. He acknowledged that DEAL is at least as slow as triple DES, especially in its key setup, so DEAL is not suited for constrained environments that require dynamic rekeying. He also acknowledged a recent attack due to Lucks [5].

DFC

DFC (Decorrelated Fast Cipher) is a Feistel network with eight rounds. The round function uses multiplication and addition modulo, reduction modulo, and a "confusion" permutation. This permutation uses addition modulo and the XOR operation with two fixed constants and another constant that is chosen from a table according to six of the data bits. Decryption is identical to encryption up to the order of the round keys.

The presenter, Serge Vaudenay, emphasized that the designers were concerned with using the recently developed technique of "decorrelation" to provide "provable security" against iterated attacks of order 2, according to a certain security model. If this could be achieved, it would imply resistance to several classes of attacks, including linear and differential ones; the designers' strategy was to tolerate imperfect decorrelation as long as it could be quantified. He proceeded to explain their particular assumptions and the security results they achieved, forecasting, for example, that exhaustive search of an 80-bit key would require at least several decades. The documentation also cited implementations of DFC on various platforms, claiming a speed rate greater than all commercial implementations of DES.

E2 ("Efficient Encryption") is a Feistel network with twelve rounds, preceded by an initial transformation and followed by a final transformation.

E2

E2 ("Efficient Encryption") is a Feistel network with twelve rounds, preceded by an initial transformation and followed by a final transformation. The initial transformation consists of key XOR, modular multiplication in 32-bit blocks with a round key, and a byte permutation; the final transformation is its inverse. The round function consists of a permutation

sandwiched between two keyed substitutions, followed by a byte rotation. The permutation is a linear transformation of data bytes; each keyed substitution consists of key XOR followed by the application of an 8x8 s-box to each byte. The construction of the s-box is based on the composition of a power function in $GF(2^8)$ and an affine function in $\mathbf{Z}/2^8\mathbf{Z}$. Decryption is identical to encryption up to the order of the round keys.

The presenter, Shiho Moriai, explained the rationale for the design, emphasizing the goals of security, efficiency, and flexibility. She claimed that two substitutions per round allow more speed for a given level of security than one substitution per round, and she spoke at some length about the construction and the properties of the s-box. It was constructed by mixing operations from two different groups, both to provide security against algebraic attacks and to convince the user that there are no trapdoors. She also claimed that the s-box could be efficiently implemented on many platforms, including those with 8-bit processors. She claimed that nine rounds of E2 would provide sufficient security against differential and linear attacks; the extra three rounds therefore constitute "insurance," along with the initial and final transformations, which are intended to resist new, as yet unknown, attacks.

FROG

Frog is an unconventional substitution-permutation network with eight rounds. The expanded key functions as an "interpreter" to sequentially process each byte of the data block. First, the byte is XORed with a byte of key material, and the result indexes another byte of key material. This byte in turn modifies three bytes of the data block: substituting for the original data byte, XORing with the following data byte, and XORing with a third data byte, which is also determined by key material. There is a complicated procedure for generating the large internal key from the user key. Decryption is the inverse of encryption.

The presenter, Dianelos Georgoudis, emphasized that FROG was designed under a different paradigm than conventional ciphers. Because the key determines the computational process, that process is hidden from potential attacker, and the algorithm is difficult to model mathematically. The presenter claimed, for example, that FROG resists linear and differential attacks because the substitutions are initialized with effective random values that are hidden. The other important design principle was simplicity, which, he claimed makes trapdoors and obscure structural flaws unlikely. In fact, the presenter claimed that should FROG be found to resist current methods of attack even though it was not specifically designed to do so, then one would gain confidence that it would resist future attacks, whose nature we cannot now predict. He acknowledged and discussed a recent attack due to Wagner, Ferguson, and Schneier[7].

Frog is an unconventional substitution-permutation network with eight rounds.

HPC

HPC (Hasty Pudding Cipher) is a set of five subciphers, each covering a range of possible block sizes; the "medium" cipher applies to the 128-bit blocks mandated for the AES. In addition to the expansion of the user key into a lookup table, the cipher features an independent, secondary key, called the "spice," whose use and concealment are optional. The algorithm mixes these two types of key material with the data block in a complicated series of steps involving addition, subtraction, the XOR operation, fixed rotations, and data-dependent rotations. Decryption is the inverse of encryption.

The presenter, Rich Schroeppel, emphasized that HPC is an "omni-cipher"; in other words, it is flexible enough to handle variable spice size, any key size, and, especially, any block size. He said that the algorithm is "forward-looking" in that it runs best on 64-bit architectures, but, conversely, it is "smartcard hostile," and, also, "doesn't favor Pentium." He claimed that the algorithm is fast, but cited the disadvantages of the code length, the dynamic storage size, and the slow primary key setup. He acknowledged that the algorithm is inelegant and therefore hard to analyze, but nevertheless he claimed that HPC has good security.

HPC [...] is flexible enough to handle variable spice size, any key size, and, especially, any block size.

LOKI97

LOKI97 is a based on LOKI89 and LOKI91. It varies the Feistel structure in that, both before and after the round function is applied to half of the data block, key material is added to that half. Therefore, decryption requires corresponding key subtractions as well as the usual reordering of the round keys. The round function consists of a keyed permutation, a fixed expansion function, two s-boxes, one 13x8 and the other 11x8, a fixed permutation, another

LOKI97 [...] varies the Feistel structure in that, both before and after the round function is applied to half of the data block, key material is added to that half.

MAGENTA [...] is a Feistel network without "unswapping" after the final round.

expansion, this time by key material, followed by another application of the s-boxes. The s-boxes are given by cubing in GF(2^{13}) and GF(2^{11}).Decryption is similar but not identical to encryption.

The presenter, Jennifer Seberry, first mentioned some weaknesses of the predecessors to LOKI97, including attacks on reduced round versions, but claimed that the full round versions are secure. She then discussed the design goals: no simple relations, no bad keys, and resistance to linear and differential attacks. She explained the rationale behind the elements of the algorithm. The key features of the round function were the double substitution-permutation layer, the completeness property, and the hiding of the round function achieved by the extra key addition incorporated into the Feistel structure. She cited several advantageous properties of the s-boxes. She discussed a recent attack due to Rijmen and Knudsen [6] and suggested possible changes in the algorithm for dealing with it.

MAGENTA

In each round [of MARS], one fourth of the data word updates each of the other three fourths of the data word.

MAGENTA (Multifunctional Algorithm for General-purpose Encryption and Network Telecommunication Applications) is a Feistel network without "unswapping" after the final round. For 128-bit and 192-bit keys there are six rounds, and for 256-bit keys there are eight rounds. The round function acts on the bytes of the data concatenated with bytes of the round subkey. The building blocks are a fixed permutation of individual bytes, the XOR operation, and a shuffling of the bytes. The permutation is discrete exponentiation of a fixed primitive element in a given representation of GF(2^8). The round subkeys are simply disjoint 64-bit segments of the key. Because the subkeys are arranged symmetrically, decryption is almost identical to encryption, up to the swapping of the two halves of the data.

RC6 is a parameterized family of encryption ciphers that use a modified Feistel structure

The presenter, Michael Jacobson, Jr., explained the algorithm and its algebraic properties, emphasizing the simplicity of the design. Discrete exponentiation provides the property of confusion, and he cited the transparency of the technique as an advantage over the use of s-boxes. Diffusion is provided by the shuffle structure, which is based on the fast Fourier transform. He presented analysis of the avalanche properties, other statistical properties, and the linear and differential characteristics of the round function, claiming that there are no practical linear and differential attacks. He also claimed that the algorithm is efficient in both hardware and software; he acknowledged the existence of some weak keys.

After the presentation, several attendees of the conference mounted attacks on MAGENTA based on the symmetry of the subkeys [3].

MARS

MARS is a cipher with thirty-two modified Feistel rounds structured as follows: key addition, eight rounds of "unkeyed forward mixing," eight rounds of "keyed forward transformation," eight rounds of "keyed backwards transformation," eight rounds of "unkeyed backwards mixing," and key subtraction. In each round, one fourth of the data word updates each of the other three fourths of the data word. The unkeyed rounds use two 8x32 s-boxes, addition, and the XOR operation. In addition to those elements, the keyed rounds use 32-bit key multiplication, data-dependent rotations, and key addition. Decryption is not identical to encryption, although it is similar in structure.

The presenter, Shai Halevi, explained the rationale for wrapping the keyed "cryptographic core" with unkeyed mixing: by providing good avalanche of the input bits, the unkeyed rounds are intended to hinder an attacker from stripping away the first and last rounds. He also claimed that this heterogeneous structure would prove resilient against new, as yet undiscovered, attacks. He cited the variety of operations, both known and new, used in the keyed rounds as another protection against future attacks. He discussed the round function of the keyed rounds in more detail, including an analysis of its linear and differential properties. He claimed that MARS offers high resistance to known attacks, better than triple DES, and runs faster than single DES in some implementations.

RC6

RC6 is a parameterized family of encryption ciphers that use a modified Feistel structure; under the parameters given for the AES submission, there are twenty rounds. The data block is partitioned into four 32-bit words. In each round, the second word updates the first word, while, in parallel, the fourth word updates the third word, after which the positions of the four words are rotated. The updating uses a quadratic transformation—requiring a 32-bit modular

multiplication and addition—the XOR operation, a data-dependent rotation, and key addition. There is also key addition before the first round and after the last round. The decryption routine is derived from the encryption routine by inverting each step.

The presenter, Ron Rivest, emphasized the algorithm's simplicity, speed, and security. He explained in seven steps how the designers of RC6 adapted RC5 to meet the AES submission requirements. An important improvement was to determine the amount of the data-dependent rotations, a main source of the overall security, by the quadratic function; this method is also efficient because 32-bit multiplication is well supported on modern processors. He presented implementation results supporting his claim that RC6 is perhaps the fastest of the candidate algorithms. He cited security analysis of the algorithm, including both its resistance to linear and differential attacks and the security of the key expansion.

RIJNDAEL

Rijndael is a substitution-linear transformation network with ten, twelve, or fourteen rounds, depending on the key size, and with block sizes of 128, 192, or 256 bits, independently specified. The data block is partitioned into a 4x4, 4x6, or 4x8 array of bytes. The round function consists of three parts: a non-linear layer, a linear mixing layer, and a key XOR layer. There is also key XOR before the first round. The non-linear layer is an 8x8 s-box applied to each byte. The s-box is constructed by considering the byte as an element of GF(2^8), finding its multiplicative inverse, then applying to the corresponding vector an affine transformation over GF(2). The linear layer consists of a shifting of the rows of the array and a mixing of the columns based on maximum distance separable codes. In the last round the column mixing is omitted.

The presenter, Joan Daemen, explained the elements of the cipher: for example, he cited the diffusion properties of the linear layer, and he claimed that the s-box would be difficult to model algebraically. Although he discussed its security against a variety of attacks, he focused on the advantages of the algorithm in its implementations. There is no algorithm setup; the key schedule is fast; the code is compact; there is extensive parallelism. Thus, the algorithm runs fast on a wide range of processors, plus, he claimed, it is very flexible in hardware. He particularly mentioned its suitability for smart cards, while acknowledging that executing the inverse cipher could be twice as slow as executing the cipher there.

Rijndael is a substitution-linear transformation network with ten, twelve, or fourteen rounds, depending on the key size, and with block sizes of 128, 192, or 256 bits, independently specified.

SAFER

SAFER+ is a substitution-linear transformation network based on the SAFER (Secure and Fast Encryption Routines) family of ciphers. There are eight, twelve, or sixteen rounds, depending on the key size, plus an output transformation after the final round. The round function consists of key-controlled substitution on the sixteen bytes of the data block followed by an invertible linear transformation on the entire data block. The substitution function acts on each individual byte with a combination of key addition, key XOR, and either a fixed permutation or its inverse. The permutation corresponds to discrete exponentiation of a fixed generator in the multiplicative group of integers modulo 257. The linear transformation is generated by a combination of the Pseudo-Hadamard Transform matrix and the "Armenian Shuffle" permutation. The decryption routine is derived from the encryption routine by inverting each step.

The presenter, James Massey, explained how SAFER+ is neither a Feistel cipher nor a substitution-permutation cipher, but rather a generalization of the latter, giving the designer more freedom to seek the best properties. SAFER+ replaces the "Hadamard Shuffle" from the original SAFER family with the "Armenian Shuffle"; he claimed that this resulted in faster diffusion and better resistance to differential attacks. Some other advantages he cited were the byte orientation, the scalability of the bytes, the lack of "suspicious-looking" tables, and the mixing of additive groups. He compared C implementations of SAFER+ and DES by the same programmers to argue that the former cipher was much faster on a Pentium platform. He also claimed that SAFER+ with its eight rounds is secure against linear and differential attacks with a margin of safety, acknowledging, however, that there is no proof of complete security.

SAFER+ is a substitution-linear transformation network based on the SAFER (Secure and Fast Encryption Routines) family of ciphers.

SERPENT

Serpent is a substitution-linear transformation network. It has thirty-two rounds, plus an initial and a final permutation to simplify an optimized implementation. The round function consists of key XOR,

Serpent is a substitution-linear transformation network.

thirty-two parallel applications of the same 4x4 s-box, and a linear transformation, except in the last round, when another key XOR replaces the linear transformation. The algorithm cycles through eight different s-boxes; thus, each of them is used in four rounds. The decryption routine is the derived from the encryption routine by inverting each step.

The presenter, Eli Biham, emphasized that the designers adopted an ultra-conservative philosophy with respect to security, because the AES will need to withstand advances in both engineering and cryptanalysis for many decades. Thus they chose to base Serpent on a combination of s-boxes and linear mappings, a familiar and well-studied combination from its use in DES, and they chose to use twice as many rounds as even their conservative security analysis dictated. In addition to summarizing this analysis, the presenter described how "bitslicing" could be used to implement the algorithm efficiently, so that it would run as fast as DES.

TWOFISH

Twofish is a slightly modified Feistel network with sixteen rounds. The round function acts on two 32-bit words with four key-dependent 8x8 s-boxes, followed by a fixed 4x4 maximum distance separable matrix over GF(2^8), a pseudo-Hadamard transform, and key addition.

The modification to the Feistel structure is the insertion of one-bit rotations before and after the results of the round function are XORed with the other two words of the data block. This introduces a slight asymmetry between encryption and decryption besides the order of the round subkeys.

The presenter, Bruce Schneier, explained how each element of the algorithm had to meet the test of "performance driven design." He explained how each element contributed to the security of the cipher, especially the key-dependent s-boxes. He claimed that these have an advantage over fixed s-boxes, which can be studied for weaknesses, although at the cost of longer setup times. He justified why Twofish's process for generating s-boxes, from two fixed permutations and key material, would not yield weak s-

Attacking Elliptic Curve Cryptosystems Using the Parallel Pollard rho Method

Adrian E. Escott
John C. Sager
Alexander P. L. Selkirk
BT Laboratories

Dimitrios Tsapakidis
NatWest

1. Introduction

Over the last year several of Certicom's easier elliptic curve challenges have been broken. The data from these completed challenges give a good indication of how difficult elliptic curve cryptosystems are to break in practice. In this paper we describe the successful attack on the ECCp-97 challenge and also provide data from trial runs on smaller elliptic curves.

2. Background

An elliptic curve cryptosystem works in a subgroup G of size n of the group formed by an elliptic curve (for more details, see Koblitz [6] or Menezes [7]). A user selects a generator P of G and randomly generates his private key d between 1 and n–1. These variables are used to calculate the public key $Q = d \cdot P$. Everything except d is publicly known. The cryptosystem can be broken by solving the elliptic curve discrete log problem (ECDLP), that is calculating d knowing only P, Q and G.

With n prime (which gives the best security), the best known method of solving the ECDLP is the parallelisation of the Pollard rho method [8] by van Oorschot and Wiener [11]. In the Pollard rho method the subgroup G is partitioned into 3 subsets S_1, S_2 and S_3 of roughly equal size. Two numbers a_0 and b_0 are randomly generated such that $1 \le a_0, b_0 \le n-1$. Starting with $X_0 = a_0 \cdot P + b_0 \cdot Q$, a sequence $\{ X_i \}$ is calculated using the relation

$$X_i = \begin{cases} P + X_{i-1} & X_{i-1} \in S_1 \\ 2 \cdot X_{i-1} & X_{i-1} \in S_2 \\ Q + X_{i-1} & X_{i-1} \in S_3 \end{cases}$$

Adrian E. Escott, JohnC. Sager, and Alexander P. L. Selkirk work in the Security Research Group at BT Laboratories. They can be contacted at adrian.escott@bt-sys.bt.co.uk, jcs@zoo.bt.co.uk, and alexander.selkirk@bt-sys.bt.co.uk respectively. Dimitrios Tsapakidis works for NatWest and can be contacted at dimitris@rabbit.co.uk.

for all $i \ge 1$. This sequence defines two futher sequences $\{ a_i \}$ and $\{ b_i \}$ where $X_i = a_i \cdot P + b_i \cdot Q$. If for some $i \ne j$ we have $X_i = X_j$ then

$$a_i \cdot P + b_i \cdot Q = a_j \cdot P + b_j \cdot Q$$

Rearranging this and substituting for $Q = d \cdot P$ gives

$$d = \frac{a_i - a_j}{b_i - b_j} \mod n$$

Hence if $b_i - b_j \not\equiv 0 \pmod n$, d can be calculated. To find $X_i = X_j$ Pollard suggests calculating $(X_i, a_i, b_i, X_{2i}, a_{2i}, b_{2i})$ until $X_i = X_{2i}$ as this requires no storage.

In van Oorschot and Wiener's method, a set of distinguished points D of G is selected. Each client calculates a sequence $\{ X_i \}$ until it finds an $X_i \in D$. This X_i and its associated a_i and b_i are submitted to a central server and the client starts again from a new starting point. The server stores all the submitted points until a point X_i is received twice and the private key is calculated. Assuming each processor is the same speed, the expected overall running time T of the algorithm satisfies

$$T = \left(\frac{\sqrt{\frac{\pi n}{2}}}{m} + \frac{1}{\theta} \right) t$$

where m is the number of processors used, θ is the proportion of points in D and t is the time taken to calculate the next X_i. This method requires the server to store $O(\theta \sqrt{\frac{\pi n}{2}})$ distinguished points.

In November 1997 Certicom announced a series of elliptic curve challenges [3]. The challenges are for three types of curve: over $\mathbb{F}p$, with p a large prime, over $\mathbb{F}_{2^n}$ and Koblitz curves over $\mathbb{F}_{2^n}$. The group size ranges from 79 to 358 bits. Table 1 gives the data on the challenges completed so far. The attack on ECCK-95 used a method similar to that given in Gallant et al. [4]. The attack on ECCp-79 used Pollard's original iteration function, while the other attacks used an iteration function with more iterators.

3. Implementation for the ECCp-97 challenge

The parallel Pollard rho algorithm has several characteristics which make it a very natural problem to

The data from these completed challenges give a good indication of how difficult elliptic curve cryptosystems are to break in practice.

We found that good, regularly updated statistics available from web pages were an important factor in promoting interest in the problem and thus recruiting CPU resources.

distribute over many machines. Firstly each client repeats a small loop of code and requires virtually no data to run. This keeps the size of the distributed programs very small—the Windows NT/95 client for the ECCp-97 challenge was only 40Kbytes in size. Secondly each client runs independently of the rest of the system and the only communication needed is to pass data from the clients to the server. Finally the server can use the submitted data to reconstruct the point and check it is valid. This prevents invalid results being stored by the server.

During the ECCp-97 challenge, clients inside the BT firewall generally communicated directly with the server using a TCP/IP connection. If the server was down, some clients communicated with a proxy while others stored the result until they found another point. Clients outside the firewall either communicated via TCP/IP with a further proxy, also outside the firewall, or mailed the results to an email address. Both the external and email proxies periodically passed the results to the server. We found that good, regularly updated statistics available from web pages were an important factor in promoting interest in the problem and thus recruiting CPU resources. Each client could also choose to join a particular group. This improved the recruitment rate even further by encouraging groups to compete for the maximum contribution rate. One group even created its own statistics page. Roughly two out of every three machines that ever submitted a point kept running until the end.

Over 1200 machines from at least 16 countries took part in the ECCp-97 challenge.

The server ran on a twin processor Pentium II 300 Mhz machine running NT. It used only a fraction of the processing power of the machine with distinguished points arriving roughly every 3 seconds. A 2 Gbytes partition of the machine's hard disk was used for storing the distinguished points in a hash table. The clients ran with low priority and could store results locally for resubmission if they failed to get a connection to the server. A distinguished point had a particular 30 bits of its x co-ordinate all zero. This was chosen as it was efficient to check for distinguished points and to balance the storage requirements of the server against the time taken for the slow clients to find a distinguished point. We used an iteration function with 16 different iterators, rather than Pollard's original function as this is more efficient (see section 4.1). The code was written specifically for the problem because this was found to be far quicker than using a general purpose maths library. We also increased the iteration rate of the code by making each client run several paths in parallel. This allowed us to reduce the number of modular inversions (the slowest operation in an elliptic curve addition) performed as it is possible to invert x numbers using $3x-3$ multiplications and 1 inversion.

Challenge	Group	Date	Group OPs
ECCp-79	INRIA	Dec 6 97	1.4×10^{12}
ECC2-79	INRIA	Dec 16 97	1.7×10^{12}
ECCp-89	INRIA/BT	Jan 12 98	3.0×10^{13}
ECC2-89	INRIA	Feb 9 98	1.8×10^{13}
ECCp-97	BT/INRIA	Mar 18 98	2.0×10^{14}
ECC2K-95	INRIA	May 21 98	2.2×10^{13}

Table 1: *Data on completed challenges*

Over 1200 machines from at least 16 countries took part in the ECCp-97 challenge. The answer was found in 53 days after 186,364 distinguished points had been submitted. This was about 45% of the expected number of distinguished points and roughly translates to 200,000,000,000,000 elliptic curve additions. The peak rate was 5,000,000,000,000 iterations/day, which would have given an expected time of between 85 and 90 days. A 600 MHz 22164 Alpha achieved a rate of 440,000 iterations per second which was about 1/130th of the peak rate. A Pentium II 300MHz gave 125,000 iterations/second, roughly 1/460th of the peak rate. The difference in speed/MHz was primarily due to the 64 bit word size on an Alpha. Roughly 70% of the work was done by BT machines with external machine contributing the rest. The largest external teams were INRIA and Digital who both contributed approximately 8% of the total work.

4. Trial Runs

In this section we consider the number of iterations needed to solve the ECDLP with different iteration

functions in various circumstances. The iteration function suggested by Pollard [8] had three iterators, namely add P, add Q or double the current X. We compare this iteration function against other functions with different iterators. We label an iterator

Iteration function	Curve 1	Curve 2
Expected	2079059	3161683
Original	2749374	4266805
AB8	2212875	3458876
AB16	2179500	3209512
AB32	2128170	3245083
AB7D1	2240148	3425943
AB15D1	2150450	3205342
AB31D1	2131877	3234529
A8	2206687	3390323
A16	2197391	3302869
A32	2125701	3180408
A7D1	2275606	3404742
A15D1	2153128	3278840
A31D1	2153716	3252593

Table 2: *Average number of iterations to solve ECDLP with different iteration functions*

AB if it is of the form add $a.P + b.Q$, A if it is of the form add $a.P$ and D if it is double the current X. Numbers are appended to show the number of iterators of each form in an iteration function. For example the iteration function AB8 has 8 iterators of the form add $a.P + b.Q$, whereas A14D2 has 14 iterators of the form add $a.P$ and 2 double iterators. Unless otherwise stated, the tests consists of 1000 runs on two curves over $\mathbb{F}p$ with group sizes of 42 and 43 bits respectively.

Multiple	0.5	0.75	1.0	1.25	1.5	1.75	2.0	2.25	2.5
Completed	16.7	33.7	51.7	68.3	80.6	89.0	94.5	97.6	99.0
Expected	17.8	35.7	54.4	70.7	82.9	91.0	95.7	98.1	99.3

Table 3: *The percentage ECDLP completed in less than the given multiple of the expected number of iterations*

4.1 Widening the iteration function

Test runs using Pollard's original iteration function needed more than the expected number of iterations to find the solution. An explanation of this is that the theoretical analysis assumes the iteration function is a random function, whereas an iteration with only three options is not very random. This suggest that having more than three iterators in the iteration function may be better as it produces a more random function. This idea is supported by Blackburn and Murphy [1], and Teske [10] who have considered single processor Pollard rho. Table 2 gives the number of iterations needed to solve the ECDLP using several iteration functions. The expected number of iterations for each curve is calculated using $\sqrt{\pi \cdot n/2}$ where n is the group size. The original iteration function is clearly the worst. As the number of iterators increase, the average number of iterations decreases, but the choice of iterators does not make a great deal of difference.

Table 3 compares the expected percentage of ECDLPs solved in less than the given multiple of the expected number of iterations and the observed percentage from 5000 runs over each curve using the iteration function A15D1. The following calculation for the expected percentage is taken from [11]. Let X be the random variable representing the number of elements needed before a duplication. Then for large n and $k = \sqrt{n}$ we have

$$\Pr(X > k) \approx e^{-k^2/(2n)}$$

The results show that solving the ECDLP using the parallel Pollard rho method is very slightly harder in practice than in theory.

Test runs using Pollard's original iteration function needed more than the expected number of iterations to find the solution.

4.2 Exploiting the inverse point

A method of improving the parallel Pollard method for a special class of curve is given in [4]. The improvement relies on performing the search on equivalence classes of points. A similar method can be employed on all curves by pairing a point with its inverse as a collision between a point and its inverse can be used to calculate the solution. To exploit this the iteration function must map the point X to the distinguished point D and the point X^{-1} to the distinguished point D^{-1}. This is done by any iteration

One disadvantage of the parallel Pollard rho method over the ordinary Pollard rho is the need to store the distinguished points at the server.

function containing only iterators that either add A to X and A^{-1} to X^{-1} or double the current X. Unfortunately with such an iteration function it is possible to enter an endless cycle by adding A at one iteration and A^{-1} at the next (longer endless cycles are also possible). Using an iteration function whose alternatives are only add $a \cdot P$ or double, it is easy to detect these cycles. Firstly no small useless cycle will contain a double, hence we only need check between two consecutive doubles. Furthermore between two doubles b_i stays fixed, so a cycle can be detected by comparing the values of a_i between two doubles. If a double iterator occurs reasonably frequently then the number of comparisons can be kept quite small. Once a cycle has been found it is exited by summing the points in the cycle. This ensures that the cycle is exited the same way every time and no collision is lost.

Iteration Function	Curve 1 Average	Curve 1 Waste	Curve2 Average	Curve2 Waste
Expected	1470117	N/A	2235647	N/A
A30D2	1478027	0.049	2259059	0.048
A60D4	1471242	0.012	2335387	0.012
A120D8	1466369	0.003	2291104	0.003
A28D4	1499838	0.042	2295493	0.041
A56D8	1496993	0.010	2274332	0.010
A112D16	1515831	0.003	2318027	0.003

Table 4: *Data from some runs exploiting the inverse point*

Table 4 examines runs that exploit the inverse point to get the solution. The expected is now $\sqrt{\pi \cdot n/4}$ as the search for collisions is a set of size $\frac{n+1}{2}$. Waste gives the percentage of the total iterations that were used to exit small cycles. For all the iteration functions the average number of iterations is close to the expected and the wasted iterations are less than 1% of the total number. All of the considered iteration functions have many doubles as this keeps the number of checks for small cycles down, but they are still all slower than an iteration function that does not exploit the inverse point. The A30D2, A60D4 and A120D8 iteration functions are approximately 6% slower whereas the others are roughly 3% slower. The code exploiting the inverse point has not yet been optimized so these figures could be improved.

4.3 Limited Memory

One disadvantage of the parallel Pollard rho method over the ordinary Pollard rho is the need to store the distinguished points at the server. The expected number of distinguished point to be stored is $\theta\sqrt{\pi \cdot n/4}$. Unfortunately the time taken for a client to find a distiguished point is $1/\theta$ and there is a limit to how long this can be made. This limit means we may not have enough memory to store every submitted distinguished point.

Table 5 contains data on runs where the number of stored distinguished points was limited. For these runs an A120D8 iteration function that exploits the inverse point was used. The number of distinguished stored was the given percentage of $\theta\sqrt{\pi \cdot n/4}$. The strategy for storing distinguished points was to keep only the most recently submitted ones. The table also gives the average and maximum number of iterations needed to solve the ECDLP. Being limited to storing 100% of the expected number of distinguished points does not have a drastic effect on the number of iterations needed. Limits of 80% and 60% are also not bad, although they are both progressively worse. At a limit of 40% the average has increased by a factor of roughly 1.75, while at a limit of 20% it has more than tripled and the maximum is much larger. From this it seems that to avoid increasing the number of iterations too badly, there should be enough memory available to store at least 50% of the expected number of distinguished points.

Percent	Curve 1 Average	Curve 1 Maximum	Curve 2 Average	Curve 2 Maximum
Unlimited	1466369	4675776	2291104	6754966
100	1618058	6582426	2389451	11124628
80	1711006	10848701	2632454	16738398
60	1975739	11333792	3107986	19320783
40	2553553	18830905	4005890	30760581
20	4857479	38453949	7078161	89301210

Table 5: *Data from runs with limited memory*

Log	Curve 1 Average	Curve 1 Multiple	Curve 2 Average	Curve 2 Multiple
1	1479752	1.01	2252691	1.01
2	2241090	1.52	3387977	1.52
3	2799904	1.90	4177655	1.87
4	3269120	2.22	4866307	2.18
5	3654872	2.49	5491087	2.46

Table 6: *Data from runs solving several discrete logs over the same curve*

4.4 Multiple Logarithms

The parallel Pollard rho method can be used to calculate several logarithms over the same curve, that is find d_1, d_2, … where $Q_1 = d_1 \cdot P$, $Q_2 = d_2 \cdot P$, etc. The method has also been suggested by Silverman and Stapleton [9]. An iteration function is chosen that is independent of all the Q_is, that is one of the form AxDy where x and y are integers. Then the standard parallel Pollard rho method is used to solve for the first private key d_1, except the distinguished points are kept. Next the parallel Pollard rho method is used to solve for d_2, but checking for collisions with all the previously found distinguished points (including those found when solving for d_1). This gives two possible types of collisions, both of which can be used to calculate d_2. Further d_i can be calculated in the same way.

Table 6 gives the number of iterations needed to solve the given number of logarithms and this number expressed as a multiple of the expected number. The results show that several logarithms over a curve are not significantly more secure than just one.

Conclusions

The running time of practical applications of parallel Pollard rho fit with the theoretical predictions. In light of the speed up from exploiting the inverse point, the expected number of iterations to solve the ECDLP should be considered to be $\sqrt{\pi \cdot n/4}$.

We feel that the current infrastructure could deal with the ECCp-109 challenge. If code for ECCp-109 that exploits the inverse point is less than 20% slower than the ECCp-97 code, then 12,000 Pentium Pro 200MHz machines would be enough to solve the challenge in a year. To put this figure in context distributed.net [5] claim to have processing power equivalent to at least 50,000 Pentium Pro 200MHz machines.

If code for ECCp-109 that exploits the inverse point is less than 20% slower than the ECCp-97 code, then 12,000 Pentium Pro 200MHz machines would be enough to solve the challenge in a year.

Acknowledgements

The authors would like to thank Robert Harley of INRIA for his help and co-operation during the challenges. We would also like to thank everyone who contributed time on their machines (for a list from the ECCp-97 challenge, see our submission [2]).

References

[1] Simon R. Blackburn and Sean Murphy. The number of partitions in pollard rho. Private Communication.

[2] BT's ECCp-97 challenge submission to Certicom. March 1997. http://www.labs.bt.com/projects/security/crackers/p-97.txt.

[3] Certicom. ECC challenges. http://www.certicom.com/chal/index.htm, 1997.

[4] Robert Gallant, Robert Lambert, and Scott Vanstone. Improving the parallelized pollard lambda search on anomalous binary curves. *Mathematics of Computation*, to appear.

[5] Distributed.Net: http://www.distributed.net.

[6] N. Koblitz. *A Course in Number Theory and Cryptography*. Springer-Verlag, 2nd edition, 1994.

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

[8] J.M. Pollard. Monte carlo methods for index computation (mod p). *Mathematics of Computation*, vol. 32 (no. 143): pp. 918-924, July 1978.

[9] B. Silverman and J. Stapleton. Contribution to ANSI X9F1, December 1997.

[10] E. Teske. Speeding up pollard's rho method for computing discrete logarithms. In *Proceedings of Algorithmic Number Theory Seminar ANTS-III*, number 1423 in Lecture Notes in Computer Science, pages 541-554. Springer-Verlag, 1998.

[11] Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptography*, to appear.

The 1999 RSA Data Security Conference

Our conference this year occurs during one of the most exciting periods in the history of the technology industry. Today, advances in global electronic business and electronic commerce are making news, fueling economic growth, and gaining the attention of industry and government leaders all around the world.

One of the vital elements underpinning this growth and excitement is cryptography and security—subjects the RSA Conference not only explores, but also defines. As always, the RSA Conference will bring attendees face-to-face with the most respected researchers, developers and visionaries in cryptography, as well as some of the technology industry's most inspired thinkers and doers. From theory to product development, from philosophy to politics, this conference puts data security in the spotlight and puts its attendees in the forefront.

Of particular interest to *CryptoBytes* readers is the RSA DES Challenge III, which will be launched at 9:00 AM (PST) on January 18, 1999, at the start of the conference. The goal of each challenge is not only to recover the secret key used to DES-encrypt a plain-text message, but to do so faster than previous winners in the series.

As before, a cash prize will be awarded for the first correct entry received. The amount of the prize will be based on how quickly the key is recovered. The DES key search machine that successfully completed RSA's previous challenge this past summer (see Paul Kocher's article in this issue) will be featured at the conference and will attempt the new challenge.

In this issue:

- ***Breaking DES***
- ***On the Transition from DES to AES***
- ***First Advanced Encryption Standard (AES) Candidate Conference***
- ***Attacking Elliptic Curve Cryptosystems Using the Parallel Pollard rho Method***

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

RSA Laboratories
A Division of RSA Data Security

2955 Campus Drive, Suite 400
San Mateo, CA 94403-2507

Tel 650/295-7600
Fax 650/295-7803

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