Communications Security for the Twenty-first Century: The Advanced Encryption Standard

Susan Landau

Editor's Note: This article is the second in a two-part series by the author in the *Notices*. The first article in the series is "Standing the Test of Time: The Data Encryption Standard", which appeared in the March 2000 *Notices*, pages 341–349.

Cryptography was once the domain of generals and small children, but the advent of the Information Age changed that. In the early 1970s the National Security Agency (NSA) and the National Bureau of Standards (NBS) realized that noncombatant adults needed to protect their sensitive, but unclassified, information. Though NSA is the usual government agency for building cryptosystems, the agency was unwilling to design a cryptosystem for public consumption.

Instead, NBS issued a public solicitation for a cryptographic algorithm. IBM responded. The company submitted a cryptosystem with a 56-bit key. (An assumption, first codified by Kerckhoffs in the nineteenth century, holds that security of a cryptosystem should rest entirely in the secrecy of the key and not in the secrecy of the algorithm. A conventional cryptosystem is considered secure when its work factor—the amount of time needed to decrypt—is about $2^{\text{key length}}$.) The new algorithm became the Data Encryption Standard (DES). In the first article of the two-part series, I described DES and the design principles behind "block-structured algorithms". The box in the present article briefly defines some technical terms that were introduced in my DES article; more detail about these definitions may be found in that article. In the present article I describe the mathematics and politics behind DES's successor: the Advanced Encryption Standard.

Susan Landau is an associate editor of the Notices *and is senior staff engineer at Sun Microsystems Inc. Her e-mail address is* `susan.landau@east.sun.com`.

A Twenty-Year Battle over Cryptography

Many in industry and academia were skeptical of DES. Concern centered on whether NSA had placed a "trapdoor" in the algorithm (a shortcut to decryption). There were also objections to DES's key length; critics believed that the relatively short key length had been chosen so that NSA could read DES-encrypted traffic.

During the next two decades there were frequent battles over cryptography. Using export controls and threats of other legal action, the U.S. government attempted to stop the spread of strong cryptography.[1] Seeking to build secure computer systems, industry found export controls on cryptography to be a major obstacle—though, to be sure, not the only one.

In the late 1970s several MIT faculty were told they would be violating laws on arms exports if they presented their research in public-key cryptography at a conference in Ithaca, New York. Foreign nationals would be present, and discussion of the cryptographic research in such a venue was viewed as export of military arms.

Several inventors of cryptographic devices found themselves silenced when secrecy orders, which forbid inventors from publicly discussing their work, were placed on their patent applications. NSA director Bobby Inman warned that if scientists did

[1] *"Strong cryptography" is a floating term, meaning whatever cryptography is invulnerable to present technology. In the 1970s, 56-bit DES was strong. Currently, 56 bits is viewed as insecure, and 70–90 bits is strong. For applications that will need to be secure well into the twenty-first century, strong is a key size of 128 bits or more.*

not exercise "prior restraint" on cryptographic publications, the NSA might seek legislation that would.

Tensions were high. Under lawyers' advice, the MIT professors presented their work, the now well-known RSA algorithm. The secrecy orders on the patent applications were lifted, and an American Council of Education study recommended a two-year experiment in NSA prepublication review of research in cryptography. Although voluntary, the reviewing experiment eased the situation, and the informal arrangement continues to this day. Over the years relations between the cryptographic research community and the NSA have grown amicable. But if cryptographic research went public, cryptographic products did not spread widely.

Export control prevented that. Cryptographic algorithms are intellectual ideas written on pieces of paper and are protected in the U.S. by the First Amendment. Cryptographic products are not; they are part of manufactured goods. Cryptographic systems, and the products that incorporate them, are subject to export controls. The United States is the undisputed leader in computer hardware and software. Cryptography is most useful when integrated into large systems, and controls on U.S. cryptographic products effectively prevent the spread of cryptography. Because U.S. manufacturers are reluctant to develop and maintain products with differing grades of cryptography for domestic and foreign consumption, export controls also effectively limit domestic use of strong cryptography. When strong encryption is available, products with shorter keys become difficult to sell; manufacturers are unwilling to risk overseas markets by supplying better products domestically.

The government's tactics sometimes led to absurdities. In 1994 a security engineer applied for a license to export a copy of the popular book *Applied Cryptography*. Export of the book was permitted under the First Amendment, of course, but an export license for the book's appendix, which contains source code for cryptographic algorithms on a floppy disk, was denied. In another case, a mathematics professor at the University of Illinois wanted to post cryptographic code on the Internet and sought an opinion from the Department of State, which said no. Both cases are wending their way through the courts.

Industry battles centered on bits: how many would the government allow in exported products? Under a 1992 agreement the magic number was 40 bits. DES is 56 bits. With narrow exceptions, products incorporating DES could not be exported.[2]

[2] *These numbers are for private-key cryptography, in which encryption and decryption use the same key. Public-key cryptography typically needs significantly more bits to achieve the same level of security as a private-key algorithm. Thus, for example, the 128 bits to be used in the Advanced Encryption Standard is viewed as providing the same level of security as a 2,560-bit RSA key [10].*

Some Terminology in the DES Article

A *block cipher* is a cryptosystem on a string of symbols that sequentially repeats an internal function, called a *round*. DES is a 16-round block cipher, with sixteen 48-bit subkeys that are generated from DES's 56-bit key. *Feistel ciphers* are block ciphers in which the $2t$-bit input is split in half as L_{i-1} and R_{i-1}; in the i^{th} round, the left half L_i is the right half of the previous round, $L_i := R_{i-1}$, while the right half R_i is a function of both previous halves, $R_i := L_{i-1} \oplus f(R_{i-1}, K_i)$. Here f is an arbitrary round function, and each round key K_i is derived from the key K. Feistel ciphers are provably invertible. Decryption is the algorithm in reverse, with subkeys used in the opposite order. (In order to make decryption a genuine inverse of encryption, the final round of a Feistel cipher switches the ciphertext to (R_r, L_r).) Variations of Feistel ciphers allow "unbalance"; e.g., the block may be split into more than two parts, with several of the pieces acted upon.

S-boxes are functions that map n bits to m bits; they are look-up tables that introduce nonlinearity into block-structured algorithms.

There are several attack models, the most popular of which are:

- *ciphertext-only:* the adversary has access to the encrypted communications;
- *known-plaintext:* the adversary has some plaintext and its corresponding ciphertext;
- *chosen-plaintext:* the adversary chooses the plaintext to be encrypted;
- *chosen-ciphertext:* the adversary picks the ciphertext to be decrypted.

A 1996 National Research Council report on cryptography policy recommended an immediate loosening of export controls. No changes occurred until 1998, when a $250,000 special-purpose machine built by the Electronic Frontier Foundation cracked a DES-encrypted message in 56 hours. At that point U.S. export controls were relaxed to permit DES in exported products. In recent months export controls have been lifted even further, with no limit on number of bits.

A DES Replacement: The Advanced Encryption Standard

A DES replacement was overdue. In 1997 the National Institute of Standards and Technology (NIST, formerly the National Bureau of Standards) announced a competition for the algorithm's replacement and held public meetings to discuss the criteria for a proposed Advanced Encryption Standard (AES). Key length was most important. A 1996 ad hoc committee argued 90 bits was currently the minimum key length needed to provide data security for twenty years [1]. NIST sought that much security and more—encrypted files should remain confidential well after AES was retired. NIST settled on a minimum key length of 128 bits.

(Although triple-DES—three iterations of DES—had become popular, several factors led NIST not to adopt the algorithm as the successor to DES. Triple-DES had 48 rounds where 32 would probably do. Software implementations on many machines were too slow for digital video and other high-speed data. The 64-bit block size was found to have intrinsic vulnerabilities when used with anticipated twenty-first-century data rates. So triple-DES was not that attractive as a Federal Information Processing Standard. And since the American National Standards Institute was already developing a triple-DES standard, NIST could simply endorse that and spend its funding on developing a new algorithm with higher capabilities.)

Given the government's intransigence over exporting strong cryptography, NIST's proposal was surprising, and initial reaction was mistrustful. Would the process be coopted by the National Security Agency? Opening the AES process to non-U.S. citizens was crucial. Theorems do not establish a cryptographic algorithm's strength; surviving cryptanalytic attacks does. And cryptanalysis expertise is international—the most effective cryptanalytic attacks on DES have been made by Israeli and Japanese researchers. A review process limited to U.S. citizens would result in an untrusted Advanced Encryption Standard—and a potentially vulnerable one. NIST allowed foreign submissions and foreign viewing of the candidates. A foreign national who wanted software implementations of the candidates could have them. The person had to register with NIST and promise not to pass on the algorithms (even if obtained from another source). While within the U.S. export-control laws, in spirit this system formed a contrast to the export rules so recently enforced regarding DES.

After some public input, NIST settled on straightforward requirements: (i) the algorithm must implement private-key cryptography (also known as *symmetric* cryptography), (ii) the algorithm must be a block cipher, and (iii) the algorithm must work on 128-bit blocks and with three key sizes: 128, 192, and 256 bits. If selected, candidates would have to be available worldwide on a nonexclusive, royalty-free basis.

Evaluations would be on security, cost, and implementation flexibility. Since simplicity aids in understanding, implementing, and assessing the security of the candidates, design simplicity would also be a factor. The winner should work in a variety of venues, including 8-bit processors, high-performance data switches, high-definition television, voice and satellite communications, and smart cards (credit cards with a small processor). This last requirement presents serious complications. Diverse technologies have competing technical requirements; sauce for the goose may really cook the gander.

There would be several evaluation meetings to discuss the submissions; after the second, designers would be allowed "minor" tweaks to their submissions. After a year NIST would determine five finalists; after another, the winner or winners—NIST could pick more than one. NIST could also choose to modify the candidates in picking a winner: for example, changing the number of rounds in an algorithm.

The jump in key size over DES was impressive. Requiring the algorithm's *worldwide* availability on a nonexclusive, royalty-free basis acknowledged the role AES would play in international e-commerce.

NIST's biggest challenge was evaluating the candidates' strength. Cryptanalysis is a young science that lacks an overarching theory. Certifying a 128-bit symmetric key algorithm is a voyage into the unknown. NIST could use mathematical arguments and various measures (e.g., how much a candidate's output was indistinguishable from a random permutation) to establish an algorithm's security. But such approaches are only as strong as the imagined attack model. At the end one is left with statements of the form: "We tried, and algorithm $\mathcal{X}$ could not be attacked by methods $\mathcal{D}$, $\mathcal{L}$, or $\mathcal{S}$." Such an approach does not inspire confidence; linear cryptanalysis was not discovered until nearly twenty years after DES's unveiling. And then there is the potential of trapdoors. For years many feared NSA had put a trapdoor in DES. Now NIST had to ensure that an algorithm whose design it did not control did not have hidden aspects.

Complicating analysis is NIST's lack of expertise. Congress empowered NIST to develop civilian computer security and cryptography standards, but has not provided adequate funding for the job. Concern centered on NSA. Would the agency's evaluations be public? NIST would rely on a combination of public and its own studies for the first round of evaluations, and the lab would show its choice of candidates to NSA before making the list public. It was assumed by many that NSA would share serious concerns it had about an algorithm's security with NIST, but left unstated was what constituted a serious concern. Suppose NSA's analysis showed that an algorithm with a 128-bit key provided at most 120 bits of security; would that be sufficient for the algorithm to be considered secure? How about 110? Where would the line be drawn? NSA was not saying.

The AES Candidates

AES candidates were due June, 15, 1998. Of the twenty-one submitted, fifteen met NIST's criteria: LOKI97 (Australia); Rijndael (Belgium); CAST-256 and DEAL (Canada); FROG (Costa Rica); DFC (France); Magenta (Germany); E2 (Japan); CRYPTON (Korea); Hasty Pudding Cipher (HPC), MARS, RC6, SAFER+, and Twofish (United States); and Serpent (United Kingdom, Israel, Norway). Except for the Hasty Pudding Cipher, all of the "U.S."-candidates included non-U.S. nationals on their design team.

In August 1999 NIST announced the five finalists: MARS, RC6, Rijndael, Serpent, and Twofish. These were widely accepted—along with some support for E2—as the "best" submissions, and NSA called these "appropriate choices," reported NIST. But there was also concern that the first year of public evaluation had concentrated on picking off the easy targets and that the remaining evaluation period was insufficient for a full evaluation of the finalists. The winner(s) will be determined in summer 2000.

In some cases the reasons for an algorithm's elimination were clear. DEAL was slow. HPC suffered from some weak keys (1 in 256 keys has 2^{30} equivalent keys—pairs of keys that give the same encryption). FROG was vulnerable to a variety of differential and linear cryptanalysis attacks (an analyst with sufficiently many targets could expect to recover the first key after $2^{56.7}$ steps). LOKI97 was similarly vulnerable to an attack using only 2^{56} chosen plaintexts; Magenta's key scheduling (the method by which key bits are furnished to the algorithm) made the algorithm very easy to break; indeed, the break happened shortly after the algorithm's presentation.

Other eliminations were based on the competition. In terms of speed, CAST was in the middle of the pack, but the algorithm had large ROM requirements. SAFER+ was slow. In design and speed, both algorithms were similar to Serpent, but Serpent beat SAFER+ on speed and CAST on versatility (good performance in a variety of environments). Serpent made the next round; CAST and SAFER+ did not. Although CRYPTON's performance was similar to that of Rijndael and Twofish, CRYPTON did less well and had a key scheduling weakness.[3] E2 was similar to Rijndael and Twofish but slower and was not implementable on low-end smart cards—E2 uses too much RAM, and it lost out. DFC was an odd case. Its framers had given proofs of the algorithm's security, but their model was sufficiently restrictive that the algorithm lacked a convincing case of security. With some weaknesses already apparent and too slow in certain tests, DFC was cut. The variety of systems—and the variety of attacks upon them—demonstrated the multidimensionality of cryptosystem design.

Cryptographic Design

Though there are many wrong ways to build a cryptosystem, there is no clear right way. To simplify implementation and a security analysis, a system should be easy to describe. It should run efficiently. It must be invertible, though extremely hard to invert without the key. It should have small key-size and memory requirements. To an extent that may be surprising to mathematicians, design decisions are technology driven: current processors determine which operations are fast and thus useful to incorporate into the algorithm. DES's S-boxes were designed to take advantage of 8-bit processors; AES's, 32-bit ones.

One school of thought in cryptosystem design lets the new technology strongly guide the choice of operations, thereby obtaining algorithmic complexity with high-speed performance. IDEA, an early 1990s cryptosystem, does this. IDEA's round function uses bitwise addition modulo 2, usually written $\oplus$ or XOR; addition modulo 2^{16}; and modified multiplication modulo $(2^{16} + 1)$, all of which run quickly on the 16-bit processors available at the time the algorithm was proposed.

NSA takes a different tack. Realizing that any widely deployed system will be implemented across a variety of hardware and software systems, the agency believes in "keep it simple" and prefers to use elementary primitives such as XOR and table lookup. As opposed to more complex operations such as floating-point arithmetic, these functions act the same way regardless of system architecture.

There are countless other tradeoffs. Perhaps the most fundamental contrast is between those algorithms that are simpler to verify but potentially less secure and those that are more complex and potentially more secure but more difficult to verify. In a block-structured cryptosystem, this particular issue plays out on the question of rounds: should there be many simple rounds or fewer, more complex ones? Even relatively simple cryptosystems can be secure when run for 32 rounds.

System designers typically begin with a set of capabilities; this may be the architectures or processors on which the algorithm will run and the set of operations that are standard to this set of machines (and thus can be done efficiently), or a set of performance constraints. The latter may include maximizing overall speed or minimizing size of memory or the time needed to change keys. This is an issue in Asynchronous Transfer Mode, a high-performance data-switching network technology.

While cryptosystem design should be a standardized procedure in much the way that bridge building is, the fact is that bridge building is much better understood. The purpose of a cryptosystem is to make decryption of messages extremely difficult without the key. The design of a cryptosystem has a dual objective: ensure cryptanalysis is difficult while making the security certifiable. The complexity of the two tasks and a lack of knowledge about how to achieve the second generally result in the design of cryptosystems done with

[3]*CRYPTON's developers submitted a "tweak" to the key scheduling during the first round of evaluations. But since CRYPTON was viewed as a weak candidate on other grounds, NIST ignored this modification.*

cryptanalysis well in mind, system certifiability much less so.

Some rules of thumb are standard. No output bit should be a linear function of the input bits; indeed, no linear function of the output bits should be a linear function of the input bits [3], [9]. This does not mean that linear functions cannot be part of a cryptosystem, but that the system must include nonlinearity. In block-structured algorithms the nonlinearity is frequently achieved by using look-up tables called S-boxes.

One can define the distance of a function to the set of linear functions, and the upper bound is achieved by a set of functions known as "bent functions". Bent functions are as distant as one can get from linear functions and were used in the design of the S-boxes for the AES submission CAST, for example.

Differential and Linear Cryptanalysis

The most serious attacks on block-structured algorithms to date are differential and linear cryptanalysis; both are described in greater detail in my DES article. Differential cryptanalysis is a chosen-plaintext attack that relies on the idea that a fixed input difference may, with high probability, generate a particular output difference. Key bits can be determined by encrypting pairs of plaintexts X, X' with prescribed bitwise difference $Z = X \oplus X'$ and by seeing which key bits are "suggested" by the output difference.

Linear cryptanalysis is a known-plaintext attack that works by finding linear relationships between plaintext, ciphertext, and key bits which reveal information about the key. Let $B[i]$ denote the i^{th} bit of an array B, and define

$$B[i_1, i_2, \dots, i_k] = B[i_1] \oplus B[i_2] \oplus \cdots \oplus B[i_k].$$

Let $\mathcal{P}, \mathcal{C}$, and $\mathcal{K}$ be the plaintext, ciphertext, and key respectively. Fundamentally, one is seeking relationships of the form:

$$\mathcal{P}[i_1, i_2, \dots, i_a] \oplus \mathcal{C}[j_1, j_2, \dots, j_b] \\ = \mathcal{K}[k_1, k_2, \dots, k_c].$$

There are some interesting correlations in the round function of DES. For example, the second input bit of the fifth S-box agrees with the XOR of all four output bits of that S-box with probability $\frac{12}{64} = 0.19$.

Let R_1 be DES after the first round, let R_{15} be DES after the fifteenth round, and let p be the probability that the equation

$$R_1[i_1, i_2, \dots, i_a] \oplus R_{15}[j_1, j_2, \dots, j_b] \\ = \mathcal{K}[k_1, k_2, \dots, k_c]$$

holds. There is a linear relationship of this type that holds with probability $1/2 - 1.19 \times 2^{-21}$ for random plaintexts and their associated ciphertexts. This relationship depends on the fifth S-box as follows: There are twelve key bits input to the fifth S-box in R_1 and R_{15}; guess these twelve subkey bits in rounds 1 and 15 (6 in round 1, 6 in round 15). If the guess is correct, the equation will be satisfied with probability p (if not, the probability will be much lower). For m plaintext-ciphertext pairs, count the number of plaintext-ciphertext pairs for which this equation holds. If the guess on the subkey bits is correct, the expected value of this sum will be pm or $(1-p)m$. If one performs this computation for more than $|p - 1/2|^{-2}$ pairs, there is a high likelihood of success. The attack uses 2^{43} plaintext-ciphertext pairs—fewer than exhaustive search, but certainly not fast.

In the case of DES, both differential and linear cryptanalysis are theoretical rather than practical attacks. Yet these are very powerful cryptanalytic techniques that cannot be ignored by system designers. To defend against differential cryptanalysis, one modifies the S-boxes until for every fixed nonzero input difference to DES there is no output difference that occurs with high probability. For linear cryptanalysis one performs a similar calculation; one ensures that no linear relationship of the output bits occurs with probability too far from 1/2. (The farther the probability is from 1/2, the greater the number of rounds needed to ensure resistance.)

What Does It Mean for an Algorithm to Be Secure?

Even determining so fundamental a notion as what it means for an algorithm to be secure is quite complicated. Exactly what is one protecting against? There are cryptosystems that are believed secure but whose implementation puts them at risk. Part of the complication with a strict mathematical analysis is ensuring that an appropriate model of security and certification is captured.

Cryptosystems are *Boolean functions*, i.e., functions mapping n bits to m bits. For simplicity, I begin with $m = 1$. Designers seek balance. Functions should spread their output evenly; e.g., functions that map to a single bit should have half the output be 0, half 1. Let hwt($\underline{s}$) count the number of nonzero components of a vector $\underline{s}$ (this is the Hamming weight of $\underline{s}$). One way to say f is balanced is that hwt(f) = 2^{n-1}.

The product of r distinct variables is an r^{th} order product. Every Boolean function $f(x_1, \dots, x_n)$ can be written uniquely as a mod 2 sum of distinct r^{th} order products, $0 \le r \le n$. The maximum order that occurs is called the *nonlinear order* of f. For example, $f(x_1, x_2, x_3) = x_1 + x_3 + x_2 x_3$ has nonlinear order 2.

One can require that a cryptographic function be r^{th}-order balanced, meaning that the output is statistically independent of any linear combination of r input variables (this is known as being *r^{th}-order correlation immune*). Guo-Zhen Xiao and

James Massey showed that having $f(x)$ be r^{th}-order correlation immune is equivalent to having $f(\underline{x})$ be statistically independent of any subset of at most r input variables. Balance can go only so far. It might be better if for any subset S of $\{0, \ldots, n-1\}$, $f|S$ (f fixed on S) were balanced. But the only functions that satisfy this are XOR and its complement.

Propagation of change is also presumably a good characteristic for a cryptosystem. Arthur Webster and Stafford Tavares defined the *strict avalanche criterion*, which says that with every change in an input bit, the output bit changes with probability 1/2. This idea generalizes: f satisfies the *propagation criterion of degree k* ($PC(k)$) if $f(x)$ changes with probability of 1/2 whenever i bits of x are complemented, $1 \le i \le k \le n$. An information-theoretic interpretation of $PC(k)$ is that if $1 \le \text{hwt}(\underline{s}) \le k$, then the mutual information between $f(\underline{x})$ and $f(\underline{x} \oplus \underline{s})$ is zero [9].

Since cryptosystems cannot be built solely from linear functions, a natural algebraic question to ask is whether higher-order polynomials are correlation immune. For small r there are many balanced quadratic functions that are r^{th}-order correlation immune. However,

Theorem *([9], p. 248): There are no quadratic functions of n variables that are r correlation immune for $\lfloor \frac{2n}{3} \rfloor \le r \le n$ and no balanced quadratic functions that are r^{th}-order correlation immune if $n-2 \le r \le n$. Here $\lfloor \cdot \rfloor$ denotes greatest integer.*

Let us now turn to Boolean functions that map n bits to m bits for general m. The *nonlinear order* of such a function is the maximum of the nonlinear orders of the m coordinate functions.

The finite field $GF(2^n)$ can be identified additively with n bits in a noncanonical way, and then the function $f(x) = x^{-1}$ with $f(0) = 0$ is a Boolean function from n bits to n bits. Kaisa Nyberg [8] studied this function and other power-type functions. She showed that this f has nonlinear order $n-1$ if $n > 1$, the nonlinear order being independent of the way that $GF(2^n)$ is identified with n bits. She showed that this function has good cryptographic properties. But it must be combined with other cryptographic functions, since it has a compact representation and thus is subject to an "interpolation attack". Otherwise, it might be possible to express the cryptosystem as a rational function of low degree, and coefficients could be determined from a small number of plaintext/ciphertext pairs.

S-boxes, which form the basis of block-structured algorithms, are considerably more complex than the simple Boolean functions just described. There has been progress in this direction. Nyberg observed that a round function should have:

1. high nonlinearity, i.e., large distance from linear functions;
2. high nonlinear order, i.e., the degrees of the output bit functions are large;
3. resistance against differential cryptanalysis; and
4. efficient construction and computability.

In addition, all linear combinations of the output bits should have high nonlinearity. Combining the specific functions Nyberg studied with various affine transformations, one achieves these goals.

A full theory of S-box design does not exist. I suspect that NSA, which has been in the cryptosystem-design business for at least a generation longer than the public cryptographers, did not have a theory of S-box design twenty years ago—or DES would have been built resistant to linear cryptanalysis. Such a theory would also have to account for interactions between the S-boxes. In their differential cryptanalysis attack on DES described in my DES article, Eli Biham and Adi Shamir demonstrated that S-box interaction played a role in the strength of the algorithm; Mitsuru Matsui showed this was true also in resistance (or lack thereof) to linear cryptanalysis.

No mathematical theory accounts for attacks that are "out of the box". Paul Kocher recently successfully broke a number of "secure" algorithms using timing and power-analysis attacks. Using a chosen-ciphertext attack, Kocher timed decryption to determine which operations were being used. This revealed which decryption key bits were a "1". Thus Kocher found the decryption key. Using this approach, Kocher determined the exponents used in the Diffie-Hellman key-exchange algorithm and factored the modulus used for the RSA algorithm [6]. Power-analysis attacks rely on the remarkably effective observation that the power consumed during encryption and decryption depends on the operation being performed and the data being processed. Kocher has produced startling pictures in which the 16 rounds of DES are clearly visible from the graph of power consumption of a smart card during DES encryption. Kocher's attacks, which rely on the physical aspects of the implementation, had not been part of any previous model considered by cryptographers.

Different cooks add their own ingredients to a cryptographic brew. The five NIST finalists, for example, include an "extended" Feistel network (MARS), two standard Feistel networks (RC6, Twofish), one substitution-permutation network (Serpent), and an algorithm that relies on finite-field operations to construct the S-box (Rijndael). MARS and RC6 use multiplication to perform diffusion, but MARS multiplies key words by data words, while RC6 multiplies words that are formed from a combination of key and data. Twofish uses "key-dependent" S-boxes that are constructed on the fly. In any given round, Serpent implements one S-box in parallel—32 copies of it. No other finalist, or candidate, does that.

The AES Finalists

I will briefly describe each of the finalists, explaining the design principles behind them. Full descriptions, including implementation details, are available through the NIST Web site.[4] For this article I will confine myself to the implementation with a 128-bit key.

MARS

In MARS, IBM designers used the well-established Feistel network, the reasonable idea that multiplication provides good diffusion properties, the fact that all modern processors support multiplication of 32-bit numbers, and their intuition that an algorithm in which the top and bottom rounds of a cipher employ functions different from the middle ones is better resistant to differential and linear cryptanalysis.

MARS breaks the 128-bit input block into four 32-bit words. MARS uses a 32-round unbalanced Feistel network: in each round one data word and some key words modify the remaining data words. The algorithm begins by adding key words to the four data words. The key is then ignored for the next eight rounds. In each of these rounds MARS uses two S-boxes with one distinguished word. This word determines the indices for the S-boxes that will modify the other three data words, called "target words".

MARS's S-boxes were built using SHA-1[5] applied to some fixed constants, specifically, expansions of the fractional parts of e and π. To assure users the algorithm has no trapdoors, algorithm designers are careful to explain their choice of fixed parameters. Input parameters were varied until the generated S-boxes had good differential and linear properties. This is reminiscent of the design procedure for DES, though the construction procedure is more explicit—and more public.

Denote the S-boxes by $S0$ and $S1$ and the four bytes of the distinguished word by $b0, b1, b2$, and $b3$. The first target word is XORed with $S0[b0]$ and added to $S1[b1]$, the second is added to $S0[b2]$, and the third is XORed with $S1[b3]$. The distinguished word is rotated 24 bits. All the words are then rotated before the next round. To thwart easy differential attacks, after four of the rounds (the first, second, fifth, and sixth) certain data words are added to others.

Symmetry in a cryptosystem (symmetry within a round function, the same round functions at the beginning and end of the algorithm) simplifies the system's architecture and its security analysis. But to thwart various attacks, MARS's first and last eight mixing rounds differ from the sixteen core rounds. And to thwart differential and linear cryptanalysis attacks, MARS's designers chose to make the rounds at the beginning and end of their algorithm asymmetric: the last eight rounds of MARS process words in a different order from the first eight. Similarly, to provide resistance to chosen-ciphertext attacks, data words in the middle sixteen rounds of the algorithm are processed in a different order in the first eight rounds of the core from that in the last eight.

This core of MARS is even more complex. Again, one data word modifies the three others, but this time through a combination of fixed rotation, multiplications, XOR, data-dependent rotations (rotations whose amount is determined by the data), and S-box lookups. MARS uses multiplication to prevent differential cryptanalysis attacks. It employs data-dependent rotations to thwart differential and linear attacks.

MARS's key schedule includes a key expansion via a linear transformation, a Feistel network to stir key bits, a test to eliminate any strings with ten consecutive 1's or 0's, and an XOR. The key schedule ensures the lowest two key bits in any key word involved in a multiplication are 1's. While not as complicated as the encryption routine itself, the MARS key schedule is nonetheless intricate. NIST commented about MARS that its "complexity makes analysis difficult in a restricted timeframe" [7]. This complexity may work against choosing MARS as the Advanced Encryption Standard.

RC6

RC6, a 20-round Feistel cipher out of RSA Security Inc., is much simpler. Whitfield Diffie, the co-inventor of public-key cryptography, has remarked that Ron Rivest, one of the principal designers of RC6 and co-inventor of the public-key algorithm RSA, is the "master of the too-good-to-be-true algorithm." This may be, but RC6's precursor, RC5, was introduced several years ago and has held up well so far. The fact that RC6 is similar to RC5 means that the public vetting of RC5 is likely to partially apply to RC6, thus simplifying certification of this AES submission.

Since the simpler algorithm sheds light on its descendant, I begin with RC5, which uses just three operations: XOR, addition, and rotation. The algorithm permits the user to set block size, $w = 16$, 32, or 64 bits; number of rounds, $0 \leq r \leq 255$; and number of key bytes, $0 \leq b \leq 255$. Let K_i be the key for the i^{th} round, whose construction will not be described (I will instead explain the RC6 schedule). RC5 with r rounds is

```
$A := A + K_0$
$B := B + K_1$
**for** $i := 1$ **to** $r$ **do** : {
        $A := ((A \oplus B) <<< B) + K_{2i}$
        $B := ((B \oplus A) <<< A) + K_{2i+1}$
}.
```

[4]http://aes.nist.gov/.

[5]*The Secure Hash Algorithm, SHA-1, is the algorithm used in the Federal Information Processing Standard SHS (Secure Hash Standard). A cryptographic* hash function *is a many-to-one mapping that provides a compact representation of an input; it has many applications, including providing an integrity check for a message.*

Here $a <<< b$ means rotate the w-bit word a to the left by the amount given by the least significant $\log w$ bits of b.

Instead of two registers of 32-bit words of RC5, RC6 operates on four registers (A, B, C, D) of 32-bit words. RC6 treats the four words in pairs, $(A, B), (C, D)$, and permutes the pairs in the last step of each round by

$$(A, B, C, D) := (B, C, D, A),$$

thus mixing them. As in RC5, data-dependent rotations provide much of the cryptographic complexity of RC6:

```
$B := B + K_0$
$D := D + K_1$
**for** $i := 1$ **to** $r$ **do** : {
        $t := (B \times (2B + 1)) <<< 5$
        $u := (D \times (2D + 1)) <<< 5$
        $A := ((A \oplus t) <<< u) + K_{2i}$
        $C := ((C \oplus u) <<< t) + K_{2i+1}$
        $(A, B, C, D) := (B, C, D, A)$
}.
```

And that, aside from "pre-whitening" and "post-whitening" steps and some minor modifications of the RC5 key schedule, is RC6. (*Whitening* is a simple idea: XOR key material with the input (or output) to a block algorithm. This is to prevent attackers from acquiring plaintext-ciphertext pairs.) Although at first RC6 does not appear to be a Feistel cipher, observe that during a round, the only action on blocks B and D are rotations to blocks A and C. Thus one can model $L := (B, D)$, $R := (A, C)$, and then indeed RC6 is a standard Feistel network.

RC6's key schedule is simple and similar to RC5's. Let $P = B7E15163$ and $Q = 9E3779B9$ be the hexadecimal representations of $e - 2$ and $\frac{1}{2}(\sqrt{5} - 1)$ respectively. Copy the key into an array W of w 32-bit words, adding extra 0's to the last word to fill it out as needed. Then the subkeys are generated as follows:

```
$K_0 := P$
**for** $i := 1$ **to** $2r + 3$ **do**
    $K_i := (K_{i-1} + Q) \mod 2^{32}$
$i := j := A := B := 0$
$v := 3 \times \max(w, 2r + 4)$
**for** $s := 1$ **to** $v$ **do**: {
    $A := K_i := (K_i + A + B) <<< 3$
    $B := W_j := (W_j + A + B) <<< (A + B)$
    $i := (i + 1) \mod 2(r + 1)$
    $j := (j + 1) \mod w$
}.
```

RC6's strength lies in the resistance to differential and linear cryptanalysis provided by the data-dependent rotations and in the diffusion provided by the quadratic function $f(x) = x(2x + 1)$.

Twofish

Twofish, proposed by Counterpane Systems, a U.S.-based cryptographic consulting firm, is a 16-round Feistel network with two modifications. One is a one-bit rotation before and after the data enter the round function proper. The other alteration is key-based S-boxes. Twofish's designers believe dynamically varying S-boxes enhance security.

Key-dependent S-boxes are unusual. DES's S-boxes were explicitly constructed to resist differential cryptanalysis; randomly constructed S-boxes are vulnerable to differential cryptanalysis attacks. The vulnerability is exploited through the examination of many plaintext-ciphertext pairs (in the case of 56-bit DES, on the order of 2^{47} chosen texts). While in any particular instantiation of Twofish some key-dependent S-boxes may be weak, the fact that the S-boxes are dynamically constructed complicates differential and linear cryptanalysis attacks.

The round function begins with input whitening. The 128-bit input is broken into four 32-bit words W_0, W_1, W_2, W_3 (numbering from left to right), W_1 is rotated eight bits to the left, and W_0 and W_1 go through the four S-boxes, $S_0, \ldots, S_3$. The four S-boxes are distinct; all are 8-bit to 8-bit bijective and key dependent. They are constructed from permutations that satisfy good differential and linear properties. The S-box operation is followed by matrix multiplication, addition modulo 2^{32}, and addition of key bits. W_0 and W_1 are XORed to a whitened W_2 and W_3 respectively, then rotated one bit (to the right for W_0, to the left for W_1). The bit rotations are put in to thwart an attack that relies on the byte alignment of the S-boxes and matrix multiplications. These words become the two left words for the next round, while their old whitened selves become the two right words for the next round.

The matrix multiplication diffuses bits; it ensures that all single-byte input differences have unique output differences and that any single-byte input change produces an output Hamming difference of at least eight bits. The addition modulo 2^{32} combines the two words W_0 and W_1 for further mixing. The operations are $T := W_0 + W_1 \mod 2^{32}$, $W_1 := W_0 + 2W_1 \mod 2^{32}$, and $W_0 := T$.

Twofish's key schedule is relatively straightforward. But NIST warned that Twofish's overall complexity "has drawn some concern" ([7], p. 53).

Serpent

Serpent, created by three cryptographers from the United Kingdom, Israel, and Denmark, is a conservative design. There are 32 rounds—a high number—each of which consists of XORing the key and the intermediate data, a pass through S-boxes, and a linear function that combines fixed rotations and XOR (in the last round, the linear function is replaced by a key-mixing operation). While the rules used to generate the linear transformation appear ad hoc—with a $<<< 7$ here, a $\oplus$ there—they function as advertised: the linear transformation increases avalanche. After three rounds, each plaintext bit affects all data bits.

Each round of Serpent has 32 identical S-boxes (each a 4-bit to 4-bit one) applied in parallel. And herein lies the cleverness of Serpent. The bits are operated upon independently, and a 32-bit processor neatly works on the 128-bit data segment. The fact that each round uses 32 identical S-boxes means that the action of the S-boxes on bits $0, 1, 2, 3$ is identical to the action on bits $4, 5, 6, 7$; bits $8, 9, 10, 11$; etc. So bits 0,1,2,3 are fed to the first input of the processor and operated upon through a series of Boolean operations, while simultaneously bits 4, 5, 6, 7 are fed to the second input of the processor and operated upon by the *same set of operations*, etc. The result is bits 0, 4, ..., 124 of the output.

Now the processor can compute the next set of outputs. The inputs have already been fed in. Again, since the S-boxes are replicated, the same operation is done on all 32 bits of the processor. The outputs are bits 1, 5, ..., 125 of the output. This process is repeated twice more, and thus all 128 bits of output are computed. This is followed by the linear transformation that diffuses the results of the bits. During these operations the processor is used to its fullest.

Each S-box is used four times, in rounds $i, i+8, i+16, i+24$ of Serpent, for $0 \le i \le 7$. The S-boxes are created from DES's via a simple program that builds a 32×16 matrix consisting of the entries of the 8 DES S-boxes. (Each row of this matrix can be viewed as a 4-bit to 4-bit S-box.) The program swaps rows in this matrix around until it has 8 S-boxes that satisfy certain differential and linear characteristics.

Serpent's key schedule is simple. The algorithm uses 132 32-bit words of keying material. The initial key is padded to 256 bits, and an intermediate version is rewritten as $K := w_{-8}w_{-7}\dots w_{-1}$ (eight 32-bit words). Then $w_i := (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \phi \oplus i) <<< 11$ and:

$$\begin{array}{lcl}
\{k_0, k_1, k_2, k_3\} & := & S_3(w_0, w_1, w_2, w_3) \\
\{k_4, k_5, k_6, k_7\} & := & S_2(w_4, w_5, w_6, w_7) \\
\{k_8, k_9, k_{10}, k_{11}\} & := & S_1(w_8, w_9, w_{10}, w_{11}) \\
 & \dots & \\
\{k_{124}, k_{125}, k_{126}, k_{127}\} & := & S_4(w_{124}, w_{125}, w_{126}, w_{127}) \\
\{k_{128}, k_{129}, k_{130}, k_{131}\} & := & S_3(w_{128}, w_{129}, w_{130}, w_{131}),
\end{array}$$

where $\phi = 9E3779B9$ in hexadecimal is $\frac{1}{2}(\sqrt{5}-1)$, and the k_i are the keys used in the round function.

Serpent's security is based on a high number of rounds, which provides strong resistance to differential and linear cryptanalysis.

Rijndael

Rijndael, developed by two Belgian cryptographers, relies more directly on algebraic constructs than do the other algorithms. Let $GF(2^8)$ be defined by the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$, and then view the 128 bits = 16 bytes as elements of the field. The data are placed in a 4×4 array of elements of $GF(2^8)$.

Rijndael has ten rounds, each consisting of four operations: ByteSub, ShiftRow, MixColumn, and AddRoundKey (the last round skips the MixColumn operation). Let elements in the array be indexed beginning with 0. ByteSub has two steps: (i) each array element is replaced by its multiplicative inverse in $GF(2^8)$ (0 is mapped to itself), and (ii) the array undergoes a fixed affine transformation over $GF(2^8)$. Then ShiftRow cyclicly shifts the elements of the i^{th} row of the array i elements to the right. In MixColumn the columns of the array are considered as polynomials over $GF(2^8)$ (the column $A_i = (a_{0,i}, a_{1,i}, a_{2,i}, a_{3,i})$ is viewed as the polynomial $a_{3,i}x^3 + a_{2,i}x^2 + a_{1,i}x + a_{0,i}$, for example) and multiplied modulo $x^4 + 1$ by $03x^3 + 01x^2 + 01x + 02$ to give elements of a new 4×4 array B (thus, $b_{0,i}$ is the zeroth-degree term in the product of $a_{3,i}x^3 + a_{2,i}x^2 + a_{1,i}x + a_{0,i}$ with $03x^3 + 01x^2 + 01x + 02$ modulo $x^4 + 1$, $b_{1,i}$ is the coefficient of the "x" term, etc.). MixColumn diffuses the bits of each array element through its column. RoundKey is an XOR of the key (given by the key schedule) with the elements of the array.

Rijndael admits many possibilities for parallelism: in the ByteSub and RoundKey operations the bytes can be operated on independently, and in the Shiftrow and MixColumn operations the rows and columns respectively can be independently manipulated.

The S-box (ByteSub) was designed for resistance to differential and linear cryptanalysis. It is invertible, and as Nyberg has shown it minimizes correlation between linear combinations of input bits and linear combinations of the output bits. MixColumn increases diffusion. Let x be a vector, and let A be a linear transformation. Define the *branch number* of a linear transformation as:

$$\min_{x \neq 0} \mathrm{hwt}(\underline{x}) + \mathrm{hwt}(\underline{A(x)}).$$

Since MixColumn works on columns independently, if a state has a single nonzero byte, the output can have at most four nonzero bytes. Hence the maximum branch number is 5. The polynomial $03x^3 + 01x^2 + 01x + 02$ achieves this maximum.

The key schedule for Rijndael is a simple expansion using XOR and cyclic shift.

Choosing a Winner

The finalists have varying strengths and weaknesses. RC6 and Rijndael have simple definitions; MARS and, to a lesser extent, Twofish have designs that complicate analysis. Serpent is slow on virtually all platforms, but its security looks good. In particular, the algorithm has a large "security margin", that is, a high number of rounds relative to differential and linear attacks that are successful on reduced-round versions of the algorithm. RC6 has a low security margin; MARS a large one.

The successful candidates are not perfect. All have serious problems in smart cards, where the attacker has limited access to the card's performance during encryption. Because of their use of multiplication and rotation, both MARS and RC6 are vulnerable to timing attacks. So is Twofish, although less so. But a differential power-analysis attack exhibited far more serious problems. Taking power samples of the whitening process from 100 independent block encryptions, a rogue smart-card implementation leaked all 128 bits of Twofish's key [2]. This was not due to a peculiarity of Twofish—all the other round-one AES candidates were equally vulnerable to this attack.

There are ways around such penetrabilities, but these come at a cost of time and space, neither of which is in great supply in smart cards. Perhaps smart cards are not an appropriate venue for the same algorithm expected to secure international e-commerce. A special-purpose algorithm might serve better.

When NBS put forth DES in 1975, the electronic world was in a fledgling state. Few anticipated the phenomenal growth of the Internet and e-commerce. AES is an ambitious undertaking. It is only two decades since the public community in cryptography numbered more than a handful of researchers.[6] NIST's venture is predicated on the idea that in the twenty-odd years since the sanctification of DES and the birth of public-key cryptography, cryptographic expertise outside the spy agencies has grown to the point that an algorithm to protect international commerce and communications can be developed by the public sector. AES is an interesting experiment—and a strong endorsement of the public expertise that has developed in so brief a time.

Acknowledgments

I have greatly benefited from discussions with Whitfield Diffie and Bart Preneel. Miles Smid, formerly of NIST, graciously answered many questions. The discussion of Boolean functions relies heavily on Chapter 5 of Preneel's thesis [9]. I also appreciate the help I received from Eli Biham, Don Coppersmith, Lars Knudsen, Vincent Rimjen, Bruce Schneier, and Yiqun Lisa Yin, who read and commented on an earlier version.

References

[1] Matt Blaze, Whitfield Diffie, Ronald Rivest, Bruce Schneier, Tsutomo Shimomura, Eric Thompson, Michael Wiener, Minimal key lengths for symmetric ciphers to provide adequate commercial security: A report by an ad hoc group of cryptographers and computer scientists. Available at `http://www.crypto.com/papers/keylength.txt/`.

[2] Suresh Chari, Charanjit Jutla, Josyula Rao, and Pankaj Rohatgi, A cautionary note regarding evaluation of AES candidates on smart-cards, The Second Advanced Encryption Standard Candidate Conference, March 22–23, 1999.

[3] Don Coppersmith, The Data Encryption Standard (DES) and its strength against attacks, *IBM J. Res. Develop.* **38** (1994), 243–250.

[4] Joan Daemen, Cipher and hash function strategies based on linear and differential cryptanalysis, Ph.D. thesis, Katholieke Universiteit Leuven, March 1995.

[5] Whitfield Diffie and Susan Landau, *Privacy on the Line: The Politics of Wiretapping and Encryption*, MIT Press, Cambridge, MA, 1998.

[6] Paul Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, *Advances in Cryptology, Proceedings Crypto '96,* LCNS, vol. 1109 (Neal Koblitz, ed.), Springer-Verlag, Berlin, New York, 1996, pp. 104–113.

[7] James Nechvatal, Elaine Barker, Donna Dodson, Morris Dworkin, James Foti, and Edward Roback, Status report on the first round of the development of the Advanced Encryption Standard, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology (to appear, *NIST J. Res.*), 1999.

[8] Kaisa Nyberg, Differentially uniform mappings for cryptography, *Advances in Cryptology—EUROCRYPT '93*, Springer-Verlag, Berlin, New York, 1994.

[9] Bart Preneel, Analysis and design of cryptographic hash functions, Ph.D. thesis, Katholieke Universiteit Leuven, January 1993.

[10] Michael Wiener, personal communication.

[6] *The first open meeting on cryptographic research occurred in Santa Barbara in 1981 and was attended by fewer than fifty people. This annual meeting, "CRYPTO", now draws upwards of five hundred attendees and is one of several international meetings and numerous workshops on cryptography.*