Polynomials in the Nation's Service: Using Algebra to Design the Advanced Encryption Standard

Susan Landau

1. INTRODUCTION. Cryptography, the science of transforming communications so that only the intended recipient can understand them, should be a mathematician's playground. Certain aspects of cryptography are indeed quite mathematical. Public-key cryptography, in which the encryption key is public but only the intended recipient holds the decryption key, is an excellent demonstration of this. Both Diffie-Hellman key exchange and the RSA encryption algorithm rely on elementary number theory, while elliptic curves power more advanced public-key systems [**21**], [**4**].

But while public key has captured mathematicians' attention, such cryptography is in fact a show horse, far too slow for most needs. Public key is typically used only for key exchange. Once a key is established, the workhorses of encryption, private- or symmetric-key cryptosystems, take over. While Boolean functions are the mainstay of private-key cryptosystems, until recently most private-key cryptosystems were an odd collection of tricks, lacking an overarching mathematical theory. That changed in 2001, with the U.S. government's choice of Rijndael[1] as the Advanced Encryption Standard. Polynomials provide Rijndael's structure and yield proofs of security. Cryptographic design may not yet fully be a science, but Rijndael's polynomials brought to cryptographic design "more matter, with less art" (*Hamlet*, act 2, scene 2, 97).

Rijndael is a "block-structured cryptosystem," encrypting 128-bit blocks of data using a 128-, 192-, or 256-bit key. Rijndael variously uses x^{-1}, $x^7 + x^6 + x^2 + x$, $x^7 + x^6 + x^5 + x^4 + 1$, $x^4 + 1$, $3x^3 + x^2 + x + 2$, and $x^8 + 1$ to provide cryptographic security. (Of course, x^{-1} is not strictly a polynomial, but in the finite field $GF(2^8)$ $x^{-1} = x^{254}$ and so we will consider it one.)

In this paper I will show how polynomials came to play a critical role in what may become the most widely-used algorithm of the new century. To set the stage, I will begin with a discussion of a decidedly nonalgebraic algorithm, the 1975 U.S. Data Encryption Standard (DES), which, aside from RC4 in web browsers and relatively insecure cable-TV signal encryption, is the most widely-used cryptosystem in the world.[2] I will concentrate on attacks on DES, showing how they shaped future ciphers, and explain the reasoning that led to Rijndael, and explain the role that each of Rijndael's polynomials play. I will end by discussing how the algebraic structure that promises security may also introduce vulnerabilities.

Cryptosystems consist of two pieces: the algorithm, or method, for encryption, and a secret piece of information, called the *key*. In the nineteenth century, Auguste Kerckhoffs observed that any cryptosystem used by more than a very small group of people will eventually leak the encryption technique. Thus the secrecy of a system must reside in the key.

[1] "Rijndael" is pronounced "Rhine Dahl" and is a combination of the names of the algorithm's two designers, Joan (pronounced Jo han) Daemen and Vincent Rijmen.

[2] Both DES and Rijndael were made into Federal Information Processing Standards, which means that the systems were approved for sale to the Federal government. The government's purchasing power causes many FIPS to become de facto commercial standards.

An algorithm is considered well designed if the expected workfactor to decode an encrypted message is approximately equal to the time it takes to search the key space. Thus one can expect that, if the key length is k bits, the time to decrypt is approximately 2^{k-1}. Addition of a single bit to key length doubles the size of the search space.

What is considered secure depends on mathematics and technology. Current processors do about a billion (or approximately 2^{30}) instructions a second; doing a billion encryptions or decryptions, which are more complex operations, is a matter of minutes or hours. Thus, no matter how good the mathematics behind it, a 30-bit cryptosystem is insecure. DES has a 56-bit key; doing a billion billion computations is approximately what it takes to break DES and is well within the capability of current technology. If we up the ante to a billion billion billion computations, or approximately 2^{90} computations, we are talking about a billion processors working for a billion seconds (approximately thirty years). With a 128-bit key, Rijndael is one step beyond this.

Attacks on cryptosystems vary widely. Attackers hope to determine the unencrypted form of the message from captured encrypted versions. From the point of view of the attacker, an even more successful attack is one that determines the key. Recent exploits, including the brute-force search of the keyspace that enabled decryption of encrypted Al Qaeda files [**9**] and the placement of an FBI "key-logging system" on the computer of an alleged loan shark [**41**], achieved exactly that.

In this paper I am interested in *mathematical* attacks on cryptosystems, not physical ones. I distinguish between *passive* attacks, in which the adversary monitors the communication channel, and *active* ones, in which the adversary may transmit messages in order to obtain information (e.g., the encrypted version of a particular message). Passive attacks are safer to mount, but typically they yield less information. However, if a cryptosystem can be shown to be secure against a *chosen-text* attack, in which the adversary chooses the plaintext to be encrypted or the plaintext to be encrypted depends on previously encrypted messages, then the cryptosystem is deemed to be quite secure. Chosen-text attacks are largely used to simplify analysis of cryptosystems, but because of such devices as "smart cards" (credit-card sized objects equipped with small processors), such attacks can actually occur.

I assume that the unencrypted message—the *plaintext*—is a string of bits that is to be transformed into an encrypted string, or *ciphertext*. Cryptosystems have to satisfy several rules. Given the key, they should be fast to compute, they must be invertible (though hard to invert without the key), and they must not use too much memory or key. The rules arise from a combination of theoretical and practical requirements. Invertibility is needed to enable the intended recipient to read the message. Low-power devices must be able to encrypt and decrypt; thus memory requirements must be low. Since the secret communication of the key bits is expensive, the key cannot be too large.

Techniques for encryption date almost from the beginning of human writing. Substitution—using one fixed set of symbols for another—and transposition—permuting a set of symbols—are the oldest as well as the simplest ways of transforming a block of letters. In 1500 B.C.E., a Mesopotamian scribe used substitution of cuneiform signs that had differing syllabic interpretations (much as "ghoti" can be an alternate spelling of "fish"[3]) to disguise a formula for pottery glazes.

Neither substitution nor transposition works well by itself. Frequency analysis, using the relative occurrence of letters, pairs, triples, and so forth, is a strong tool against

[3] This example is due to George Bernard Shaw. Pronounce "gh" as in "tough," "o" as in "women," and "ti" as in "nation."

both.[4] The fifteenth-century Arabic encyclopedia *Subh al-a 'sha* included a sophisticated discussion on using frequency distributions for cryptanalysis [**20**, p. 95]. Any message of reasonable length that is encrypted using a substitution or moderate-length transposition cipher (that is, short with respect to the length of the message) can be quickly deciphered by frequency analysis. Indeed, a trained cryptanalyst can break a simple single-alphabet substitution cipher given only twenty-five characters of ciphertext.

Despite this apparent weakness, substitution and transposition ciphers are too useful to be ignored. The information theorist Claude Shannon observed that the two fundamental techniques for encryption are confusion—obscuring the relationship between plaintext and ciphertext—and diffusion—spreading the change throughout the ciphertext. Substitution is the simplest type of confusion, and transposition the simplest method of diffusion.

Cryptanalysis can be viewed as an approximation problem: given ciphertext, determine the plaintext by approximating the decryption function. Linear functions of the input and key make poor choices for encryption functions because such systems can always be broken by solving small sets of linear equations. But because cryptographic functions must be invertible, must be fast to compute, and should have small key size and memory requirements, linear functions are irresistible. Simple operations such as XOR (exclusive or—also written $\oplus$—bitwise addition modulo 2), substitution, and permutation are ingredients the combination of which can produce a cryptosystem whose strength is greater than the sum of its parts. Typically such instructions are combined and used in an *iterative block cipher*, a cryptosystem that operates on a block of data and sequentially repeats a set of primitives; each repetition is a *round* of the function. DES is such a block cipher.

2. THE DATA ENCRYPTION STANDARD (DES). DES was developed in the mid-1970s after members of the National Security Agency (NSA) and the National Bureau of Standards (NBS, later renamed the National Institute of Standards and Technology, or NIST) recognized that the government's increasing use of computers for civilian data would require new privacy-protection technology. Although NSA was the traditional designer of Federal cryptosystems, the agency was reluctant to develop a system that would be available for public scrutiny, feeling that to do so would give outsiders insight into the agency's design philosophy.

NBS put out a request for proposals for an algorithm to encrypt 64-bit blocks of data. The short version of the story is that IBM responded and submitted the algorithm that was approved in 1977 as the Data Encryption Standard. The long version is somewhat more complicated and best left to another venue (see, for example, [**44**]). Most controversial was DES's key length; already in 1977, many considered this too short.

At the time of the NBS request, IBM was at work developing automated teller machines for Lloyds Bank, London. One member of the IBM group, Horst Feistel, had participated in the development of *IFF* ("Identify Friend or Foe") systems for the Air Force. Feistel had proposed a general scheme design for block-structured cryptosystems that enabled self-invertibility. Self-invertibility is important in cryptosystem design, for it enables a single object (a chip, a piece of software) to both encrypt and decrypt.

[4] The letter "e" appears in 13 percent of English text; the letters "t, r, n, i, o, a, s" are the next most frequent letters. Similarly, there is data on the frequency of various letters appearing at the beginning and end of words, etc.

DES specifications. Assume that there is a $2t$-bit block to be encrypted using r rounds of a block cipher. Split the block in half: left, L_0, and right, R_0; the two parts will be operated on independently. In each round, the right half R_i becomes the new left half L_{i+1}. The new right half R_{i+1} is a function of the old right and left halves, R_i and L_i respectively, and the portion of the key K_{i+1} used in that round. In round i:

$$\begin{aligned} L_i &\leftarrow R_{i-1}, \\ R_i &\leftarrow L_{i-1} \oplus f(R_{i-1}, K_i). \end{aligned}$$

Inversion is easy, since

$$\begin{aligned} R_{i-1} &= L_i, \\ L_{i-1} &= L_{i-1} \oplus f(R_{i-1}, K_i) \oplus f(R_{i-1}, K_i) \\ &= R_i \oplus f(L_i, K_i). \end{aligned}$$

That decryption is the algorithm run in reverse, with subkeys used in the opposite order, is mathematically elegant and practical. In order to make decryption a genuine inverse of encryption, the final round of a Feistel cipher switches the two halves.

DES is a 16-round Feistel cipher (see Figure 1). DES begins with $\mathcal{P}$, a permutation that efficiently distributes data, and ends with $\mathcal{P}^{-1}$. In between are the sixteen Feistel rounds. The permutations $\mathcal{P}$ and $\mathcal{P}^{-1}$ are not cryptographically important, but they are part of the DES standard. Implementations without them are not DES.

The DES round function is where the interesting cryptographic work occurs, so I will examine it in some detail. It consists of:

$$\begin{aligned} L_i &\leftarrow R_{i-1}, \\ R_i &\leftarrow L_{i-1} \oplus P\Big(S\big(E(R_{i-1}) \oplus K_i\big)\Big). \end{aligned}$$

The function E expands the 32-bit right half to 48 bits by repeating certain bits, S is the S-box function (to be described shortly), and P permutes the output from the S-boxes to achieve diffusion. In each round 48 bits of the key are selected according to a *key schedule*, which dictates the way key bits are supplied to the algorithm, ensuring that a different subset of key bits is used in each DES round.

Since $\mathcal{P}$, E, P, and $\mathcal{P}^{-1}$ are all linear functions, the nonlinearity—and cryptographically interesting aspect—of DES must come from the S-boxes. There are eight S-boxes, each mapping six bits to four. Writing the input bits as $b_1, \ldots, b_6$, one can view b_1 and b_6 as "instruction bits" operating on "data bits" b_2, b_3, b_4 and b_5. Each DES S-box is a 4×16 table, with instruction bits determining the row and data bits determining the column; the output is the table entry.

Figure 2 shows S-box 3. Note that instruction bits have four possible values $(00, 01, 10, 11)$, data bits sixteen $(0000, 0001, \ldots, 1111)$. Although table entries are written $0, \ldots, 15$, that is simply shorthand for $0000, \ldots, 1111$. Also note that each row of the table has all possible entries (which therefore appear exactly once each). This is no accident.

Other than the fact that each row of the table includes all values between 0 to 15 inclusive, the entries in S-box 3 appear more or less random; the crucial word is "appear" (more on that later).

Attacks on DES. There were many objections to DES when it was proposed as a Federal Information Standard. Almost all were about key length, but some researchers

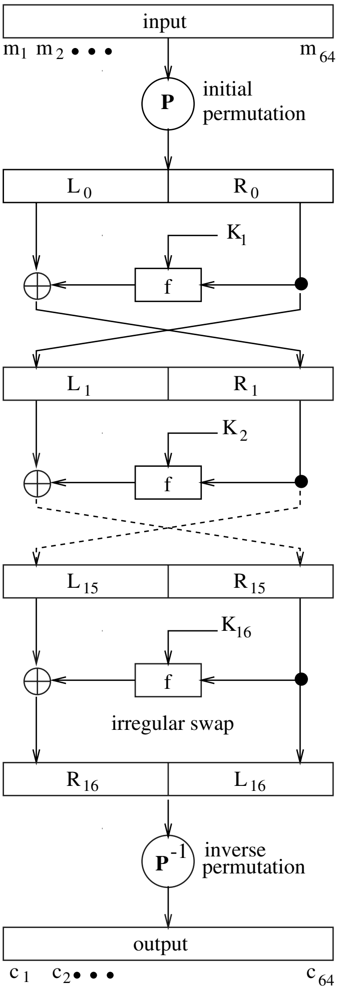

Figure 1. Data Encryption Standard.

$$\begin{pmatrix} 10 & 0 & 9 & 14 & 6 & 3 & 15 & 5 & 1 & 13 & 12 & 7 & 11 & 4 & 2 & 8 \\ 13 & 7 & 0 & 9 & 3 & 4 & 6 & 10 & 2 & 8 & 5 & 14 & 12 & 11 & 15 & 1 \\ 13 & 6 & 4 & 9 & 8 & 15 & 3 & 0 & 11 & 1 & 2 & 12 & 5 & 10 & 14 & 7 \\ 1 & 10 & 13 & 0 & 6 & 9 & 8 & 7 & 4 & 15 & 14 & 3 & 11 & 5 & 2 & 12 \end{pmatrix}$$

Figure 2. S-box 3.

expressed concern about the DES S-boxes. Did they have any "trapdoors," secret information that would enable those who knew them to decrypt more easily, even without a key? Little information about the S-boxes was forthcoming until the early 1990s, when two Israeli researchers, Eli Biham and Adi Shamir, discovered *differential cryptanalysis*, an attack that exploits the nonlinearity of DES.

Using a chosen-plaintext attack, Biham and Shamir sought to find information about the key bits being used in a particular DES-encryption session. Let P_1 and P_2 be two 64-bit plaintexts. DES's nonlinearity means that

$$\mathrm{DES}(P_1 \oplus P_2) \neq \mathrm{DES}(P_1) \oplus \mathrm{DES}(P_2).$$

Because DES's only nonlinearity lies in the S-boxes, the difference between the left- and right-hand sides of the equation must come from the S-box actions. For each S-box create a "difference distribution table," a table of the distribution of all input XORs (there are sixty-four of these) and output XORs (there are sixteen of these) pairs. The entries in the table are the number of pairs with particular input and output differences. Differential cryptanalysis exploits the variance in the output differences.

Take S-box S_3. Figure 3 shows the first few rows of the difference distribution table [**3**] for S_3; the reader can fill in the rest of the table using the description of S-box S_3 given in Figure 2. Though the entries of the table are in decimal, notation along the rows and columns (denoting the S-box entries) is in hexadecimal. (Hexadecimal is standard notation for numbers expressed in base 16. The hexadecimal digits are: $0, 1, \dots, 9, A, B, \dots, F$.)

Input	Output XOR															
XOR	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	4	2	12	0	14	0	4	8	2	6	10
2	0	0	0	2	0	2	0	8	0	4	12	10	4	6	8	8
3	8	6	10	4	8	6	0	6	4	4	0	0	0	4	2	2
4	0	0	0	4	0	2	4	2	0	12	8	4	6	8	10	4
5	6	2	4	8	6	10	6	2	2	8	2	0	2	0	4	2
6	0	10	6	6	10	0	4	12	2	4	0	0	6	4	0	0
7	2	0	0	4	4	4	4	2	10	4	4	8	4	4	4	6
8	0	0	0	10	0	4	4	6	0	6	6	6	6	0	8	8
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
3F	2	6	4	0	0	10	8	2	2	8	6	4	6	2	0	4

Figure 3. Difference distribution for S-box 3.

To be sure, there are some patterns. By symmetry, all entries in the table are even. The sum of all elements in a row is 64. What is more interesting is the lack of patterns: the average value of an entry is 4, but the entries exhibit wide variance.

Consider the row where the input difference of P_1 and P_2 to S_3 is 8. The output difference has no chance of being 0, 1, 2, 4, 8, or D, $4/64$ chance of being 5 or 6, $6/64$ chance for each of 7, 9, A, B, or C, $8/64$ chance for each of E or F, and $10/64$ chance of being 3.

Now consider the input difference 1 in S_3 with output difference D. There are two pairs of inputs that satisfy this particular pair of input/output XORs. These pairs must be duals: (P_1, P_2) and (P_2, P_1). And indeed $(1C, 1D)(= (011100, 011101))$ and $(1D, 1C)$ are the only pairs that have input difference 1 and output difference D in S-box 3.

Now I do not actually know the input to the S-box; I only know the input to the algorithm. For a moment, I will confine my analysis to a single round of DES. Of course, I know the input difference to the round rather than to the S-boxes, but the linearity of the remaining round operations—E, $\oplus$, and P—means that I can calculate the one from the other.

I will first focus on a single S-box, say S_3. Suppose the input difference $\Delta P = 1$, and we try $P_1 = A$, then $P_2 = B$. The linearity means that $\Delta P = 1 = E(A) \oplus E(B) = (E(A) \oplus K_i) \oplus (E(B) \oplus K_i)$. Thus A and B are possible input candidates for an input/output pair for the round. Let ΔC be the XOR of the output of this round. I can work backwards from the difference distribution table to determine which key bits would give the output ΔC.

This does not determine the key bits completely. But I can start with another input/output pair, and this one, too, will also identify potential key bits. As Biham and Shamir observed, the right key must occur as a possibility for both input/output pairs.

The values ΔP and ΔC impose restrictions on the key bits of the DES round; we say that the pair $(\Delta P, \Delta C)$ *suggests* subkey values compatible with these restrictions. Some pairs suggest many key bits, some none, some a few. This is the crux of differential cryptanalysis.

For each suggested key value, the corresponding entry in a frequency table is increased. Differential cryptanalysis succeeds if the correct subkey is suggested more often than other values. Now the wrong subkey may also be suggested as well as the right subkey. Experimental work with DES shows that to have a 50 percent chance of getting the right subkey, computing with between 20 and 40 "right" pairs (pairs $(\Delta P, \Delta C)$ with conditional probability $P(\Delta C \mid \Delta P) \neq 0$) is sufficient.

It is time for the analysis to go from one round to many and from one S-box to eight. This takes us to the second crucial idea in differential cryptanalysis: characteristics. *Characteristics* are differences in plaintext pairs that have a high probability of causing certain differences in ciphertext pairs. A trivial characteristic is input $\Delta P = 0 = \Delta C$ (that is, begin and end with the same string). This occurs, of course, with probability 1. A more interesting one-round characteristic has 0 as the input difference to seven S-boxes, while the input to the remaining S-box is nonzero and is chosen by picking one whose output difference distribution has the roughest distribution, preferably with one large value. (Since several of the input bits to this remaining box also affect two neighboring S-boxes, these must be zero.) One high-probability way to do this is:

$$\begin{array}{lrl} S_1: & C \to E & \text{with probability } 14/64, \\ S_2, \dots, S_8: & 00 \to 0 & \text{with probability } 1. \end{array}$$

One can now concatenate the two one-round characteristics described in the previous paragraph to get a two-round characteristic that has probability $14/64$. Indeed, one can put these together to have a three-round characteristic with probability $(14/64)^2 \approx .05$ [**3**, p. 26].

An *iterative characteristic* is one that can be concatenated with itself. Biham and Shamir developed what they believe is an optimal set. Their differential cryptanalysis attack on DES is [**3**, p. 21]:

1. Pick an appropriate input difference ΔP.
2. Create an appropriate number of plaintext pairs with this ΔP, encrypt with DES, and store the ciphertext pairs.
3. For each pair, from the plaintext ΔP and the ciphertext pair determine the expected output difference of as many S boxes in the last round as possible.
4. For each possible key value, count the number of pairs that result with the expected output change using the value in the last DES round.
5. The right key value is the one suggested by all the key pairs.

Biham and Shamir found a 13-round characteristic that requires encryption of 2^{47} chosen plaintexts. The algorithm finds 48 bits of the key used in round 16 and then determines the other 8 bits by exhaustive search (a relatively fast process in this case). The time to perform this attack is essentially bounded by the 2^{47} encryptions.

Biham and Shamir tried variations on DES—no permutation P, reordering the S-boxes—and each variation produced a weaker algorithm than the original. IBM nodded in agreement. After the Biham-Shamir attack, a member of the IBM team, Don Coppersmith, described how the IBM group had designed DES to prevent differential cryptanalysis [**6**]. While the S-boxes were generated randomly, potential candidates were checked against a set of criteria designed to thwart differential cryptanalysis [**7**], [**18**], [**45**].

Call an S-box *active* if the input difference is nonzero, and *inactive* otherwise. Differential cryptanalysis works particularly well if exactly one S-box is active. So the IBM rules required [**6**]:

- If two inputs to an S-box differ in exactly one bit, their outputs should differ in at least two bits.
- If two inputs to an S-box differ exactly in the middle two bits, their outputs must differ by at least two bits.
- If two inputs to an S-box differ in their first two bits and agree on their last two, the two outputs must differ.
- For any nonzero 6-bit difference between inputs, no more than eight of the thirty-two pairs of inputs exhibiting that difference may result in the same output difference.

There were several more rules. No cryptographic algorithm should be a linear function. But neither the IBM researchers—nor anyone else at the time—had considered the natural generalization that *no linear combination* of the output bits should be *a linear function* of the input bits. In 1993 Mitsuru Matsui used this for a *linear-cryptanalysis* attack on DES [**27**].[5]

Linear cryptanalysis seeks linear relations between the input and output bits of DES. For example, the parity of the first, second, third, fourth, and sixth input bits of the third S-box agrees with the parity of all of the output bits 38 of 64 times. If the S-box were truly random, you would expect that agreement half the time. That little difference—6 more than the expected 32—tips the balance, and probabilistically reveals information about the key bits.

Linear cryptanalysis works by chaining together relations such as the one just mentioned. More formally, let $B[i]$ denote the ith bit of an array B of any length, and define

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

Linear cryptanalysis is based on the tendency of some equations

$$L_i[i_1, i_2, \dots, i_a] \oplus R_i[j_1, j_2, \dots, j_b] = K_i[k_1, k_2, \dots, k_c]$$

to hold with probability bounded away from $1/2$. Another way to say this is is that the probability that equations hold—or don't hold—is strictly less than $1/2$.

I must take a brief digression. In the context of Boolean functions, the equation:

$$a \oplus b = 0$$

is a linear equation and is equivalent to $a = b$. The equation

$$a \oplus b = 1$$

[5] In 1985 Adi Shamir observed some interesting correlations between bits in S-boxes [**43**]; these can be expressed by Matsui's linear approximations.

is an affine equation and is equivalent to $a \neq b$. Yet these two formulations are, in some sense, two ways of representing the same relationship—"NOT $a \oplus b = 0$" has the same meaning as "$a \oplus b = 1$." Because of the essential equivalence between linear and affine functions, I will freely go back and forth between the two types of functions, using whichever formulation is most convenient.

Matsui observed that a particular linear relationship

$$L_1[i_1, i_2, \ldots, i_a] \oplus R_{15}[j_1, j_2, \ldots, j_b] = K[k_1, k_2, \ldots, k_c]$$

holds with probability $p = 1/2 - 1.19 \times 2^{-21}$ for random plaintexts and their associated ciphertexts. For m plaintext-ciphertext pairs, one counts 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 correct subkey guesses. The attack uses 2^{43} plaintext-ciphertext pairs. (For more details on differential or linear cryptanalysis, see [**3**] or [**27**], respectively.)

In theory, differential and linear cryptanalysis represented serious attacks on DES, both breaking the algorithm faster than a brute-force search of the key space. In practice, these attacks were not serious threats to the algorithm. The 2^{47} and 2^{43} DES encryptions needed by differential and linear cryptanalysis, respectively, each take sufficiently much time that neither attack was faster than a simple brute-force search of the 56-bit key space.

Chip speeds were increasing, chip prices were decreasing; DES's security could not last. In the summer of 1998 DES met its match. A specially-built two-hundred-and-fifty thousand dollar computer, the DES-Cracker, decrypted a DES-encoded message in fifty-six hours using a simple brute force search of the key space. (By contrast, a 1994 linear cryptanalysis attack using twelve networked computers took fifty days to break a DES-encoded message.) Six months later, 100,000 networked PCs and the DES-cracker dropped the time to twenty-two hours. DES was no longer an effective long-term security solution.

But I am running ahead of our story. Even if linear and differential cryptanalysis were not practical attacks on DES, they were mathematically important. Any future cryptosystem would have to take them into account.

Designing block ciphers. The near-simultaneous development of DES and public-key cryptography in the mid 1970s sparked great interest in cryptography, which previously had been almost exclusively the domain of intelligence agencies. In 1981 the first public cryptographic research meeting in Santa Barbara attracted fewer than fifty researchers. By the late 1990s, there were a number of annual meetings (including CRYPTO, Eurocrypt, Asiacrypt, and Fast Software Encryption) and a large international community of researchers. The research had many strands.

Some found their way into Rijndael. Willi Meier and Othmar Staffelbach suggested that certain nonlinearity measures used by mathematicians would be appropriate for cryptographic system design [**28**]. From these ideas, Josef Pieprzyk proposed algebraic methods for constructing nonlinear functions [**37**], [**38**]. Kaisa Nyberg explored S-boxes and applied some of Pieprzyk's ideas in S-box design [**34**]. Joan Daemen studied round functions from the point of view of differential and linear cryptanalysis and proposed a new paradigm, the wide-trail approach [**10**]. With others, he used wide trail and Nyberg's S-boxes in the cryptosystem SHARK [**40**]. Thomas Jakobsen and Lars Knudsen found an "interpolation attack" against simple algebraic ciphers, such as those used in SHARK [**19**]. Two SHARK designers, Daemen and Vincent Rijmen, countered with the cryptosystem Square [**11**]. Knudsen broke Square using a different attack method [**11**]. Rijndael rose from Square's ashes. I will trace how these threads were woven into Rijndael.

Abstract approaches to S-box design. Meier and Staffelbach observed that certain nonlinearity properties (in particular, distance of nonlinear functions to affine functions) were preserved under affine transformations [**28**]. I need some formalism to explain what that means.

Boolean functions are maps from $\{0, 1\}^n$ to $\{0, 1\}$ or, more generally, to $\{0, 1\}^m$. The *parity function*, a mapping from $\{0, 1\}^n$ to $\{0, 1\}$, is an example of the first type of Boolean function; it computes the sum modulo 2 of the inputs: $\text{parity}(x_1, x_2, \dots, x_n) = x_1 \oplus x_2 \oplus \dots \oplus x_n$. Complementation of an *n-bit* Boolean vector of 0s and 1s, in which all 0s are changed to 1s, all 1s to 0s, is an example of a Boolean function from $\{0, 1\}^n$ to $\{0, 1\}^n$.

Definition. The *Hamming distance* $d(u, v)$ between two n-bit Boolean vectors u and v is the number of bits at which the two vectors differ. Let f and g be two Boolean functions from $\{0, 1\}^n$ to $\{0, 1\}^m$. Then the *Hamming distance* $d(f, g)$ is the number of input values at which f and g differ.

For example, the distance between the n-bit vector of all 0s and the n-bit vector of all 1s is n. If $f : Z_2 \times Z_2 \times Z_2 \to Z_2 \times Z_2 \times Z_2$ is given by $f((x_1, x_2, x_3)) = (x_1 + 1, x_2 + 1, x_3 + 1)$, with all sums taken modulo 2, then the Hamming distance between $f(x)$ and the identity map on $Z_2 \times Z_2 \times Z_2$ is 8, since these two functions differ at all inputs.

Let $\mathcal{F}_n$ denote the set of all Boolean functions of n variables taking values in Z_2. Let $\mathcal{L}_n$ signify the set of all linear functions from binary strings of length n into Z_2, and consider a Boolean function f in $\mathcal{F}_n$. Pieprzyk suggested the following means for quantifying the nonlinearity of f.

Definition. The *nonlinearity* $\mathcal{N}_f$ is the Hamming distance between f and the set of all linear functions $\mathcal{L}_n$ in $\mathcal{F}_n$, that is [**38**]:

$$\mathcal{N}_f = d(f, \mathcal{L}_n) = \min_{\alpha \in \mathcal{L}_n} d(f, \alpha).$$

The simplest view of cryptographic functions is that they are maps from $\{0, 1\}^n$ to $\{0, 1\}^m$. A richer viewpoint is to view them as functions from $GF(2^n)$, the finite field (also called the Galois field) of 2^n elements, to $GF(2^m)$. Then in constructing cryptographic functions, one one can build upon the underlying algebraic structure already present in $GF(2^n)$.

The field $GF(2^n)$ is an extension of degree n over Z_2, the finite field of two elements. Thus it can be written as $Z_2[y]/(r(y))$, where $r(y)$ is an irreducible polynomial of degree n over Z_2. It is not hard to show—but I will not—that for every positive value of n there is a field of 2^n elements and that this field is unique up to isomorphism. (This is also true for every prime power p^n.) Note that a finite field $GF(2^n)$ is a vector space of dimension n over Z_2.

Typically one denotes elements in $GF(2^n)$ as polynomials, that is, as elements in $Z_2[y]/(r(y))$, where $r(y)$ is a generator for the ideal in this quotient ring representation. For example, $y^3 + y + 1$ is irreducible over Z_2; elements in $Z_2[y]/(y^3 + y + 1)$ are polynomials in y of degree 2 or less with coefficients in Z_2.

As there is only one irreducible polynomial of degree 2 over Z_2, there is a single way to denote a field of 2^2 elements in the form $Z_2[y]/(r(y))$, namely, as $Z_2[y]/(y^2+y+1)$. There are two irreducible polynomials of degree 3 over Z_2 : x^3+x+1 and x^3+x^2+1. Thus there are two ways to denote the field of 2^3 elements as polynomial extensions of Z_2: $Z_2[z]/(z^3+z+1)$ and $Z_2[y]/(y^3+y^2+1)$. Isomorphisms between the two fields are not difficult to find. Since the polynomial z^3+z+1 factors into $(z+y^2+y)(z+y^2+1)(z+y+1)$ in $Z_2[z, y]/(y^3+y+1)$, the isomorphisms betweeen $Z_2[z]/(z^3+z+1)$ and $Z_2[y]/(y^3+y^2+1)$ are

$$\begin{aligned}
f_1: \quad a_2z^2+a_1z+a_0 \mapsto \quad & a_2(y^2+y)^2+a_1(y^2+y)+a_0 \\
&= a_2(y+1)+a_1(y^2+y)+a_0 \\
&= a_1y^2+(a_2+a_1)y+(a_2+a_0), \\
f_2: \quad a_2z^2+a_1z+a_0 \mapsto \quad & a_2(y^2+1)^2+a_1(y^2+1)+a_0 \\
&= (a_2+a_1)y^2+a_2y+(a_1+a_0), \\
f_3: \quad a_2z^2+a_1z+a_0 \mapsto \quad & a_2(y+1)^2+a_1(y+1)+a_0 \\
&= a_2y^2+a_1y+(a_2+a_1+a_0),
\end{aligned}$$

where a_2, a_1, and a_0 are in Z_2.

In constructing cryptographic functions, Pieprzyk adopted the perspective of $GF(2^n)$ and viewed maps $f=(f_1, f_2, \dots, f_n)$ as permutations of $GF(2^n)$, where the maps work separately on the n components of $GF(2^n)$ (the components are the coefficients of an element written as a polynomial in $GF(2^n) \approx Z_2[y]/(r(y))$, in which $r(y)$ is some specified irreducible polynomial of degree n over Z_2). Pieprzyk suggested the following:

Definition. If $f=(f_1, f_2, \dots, f_n)$ is a permutation of $GF(2^n)$, then the nonlinearity of f is defined by

$$N_f = \min_i(\mathcal{N}_{f_i}, \mathcal{N}_{f_i^{-1}}),$$

where, in an abuse of notation, f^{-1} is defined as $(f_1^{-1}, f_2^{-1}, \dots, f_n^{-1})$.

The viewpoint of $GF(2^n)$ led Pieprzyk to propose two functions—x^3 and, more generally, x^{2^k+1} for $k>2$—as possible cryptographic functions. This presents a difficulty: the definition of nonlinearity means the function is determined component-wise, but how would one easily determine the nonlinearity measure of exponentiation, which does not operate component-wise? Pieprzyk turned to the trace function. Let α be an element in K, a finite extension of a finite field F.

Definition. The *trace* $\mathrm{Tr}_{K/F}(\alpha)$ of an element α in K relative to F is the sum of the conjugates of α with respect to K.

Thus $\mathrm{Tr}_{K/F}(\alpha)$ is the sum of the members of the orbit of α under the Galois group of K over F. Consider the field extension $K=Z_2[y]/(y^3+y^2+1)$ over $F=Z_2$. Now x^3+x^2+1 factors into $(x-y^2)(x-y)(x-(y^2+y+1))$ in $K[x]$. Thus the conjugates of y in K are y^2+y+1 and y^2. That means that $\mathrm{Tr}_{K/F}(y^2)=(y^2)+(y)+(y^2+y+1)=1$. That is exactly what we should expect, since y^2 is a root of y^3+y^2+1 and the constant term of a minimal polynomial is the sum of the roots of the polynomial.

Using the trace, Pieprzyk showed that all components of the functions x^3 and x^{2^k+1} in $GF(2^n)$, with n odd, achieve maximum nonlinearity; this is independent of the basis chosen [**38**]. But Piepryzk's definition of nonlinearity was not quite what was needed.

Matsui's linear cryptanalysis attack on DES had showed that it was important that *no linear combination* of the output bits of a cryptographic function be a linear combination of the input bits. Kaisa Nyberg formalized this, providing a more general condition for nonlinearity than Pieprzyk's.

Definition. Let $f : F^n \to F$ be a function. Then the *Hamming distance* $\mathcal{N}(f)$ of f from the set of affine functions is [**33**]:

$$\mathcal{N}(f) = \min_{u \in F^n, v \in F} \left|\left\{x \in F^n \mid f(x) \neq u^T x + v\right\}\right|.$$

This can extended to $f : F^n \to F^m$ by

$$\mathcal{N}(f) = \min_{u \in F^n, w \in F^m, v \in F, u \neq 0, w \neq 0} \left|\left\{x \in F^n \mid w^t f(x) \neq u^T x + v\right\}\right|.$$

Nyberg's definition of nonlinearity is preserved under the taking of inverses [**33**]:

Proposition 1. *If $f : F^n \to F^n$ is a permutation, then $\mathcal{N}(f^{-1}) = \mathcal{N}(f)$.*

DES used one way to construct S-boxes: randomly, subject to some preconditions. Nyberg sought a more systematic approach to S-box construction. S-boxes should be highly nonlinear. There were other important criteria. Every Boolean function $f(x_1, \ldots, x_n)$ can be written uniquely as a sum modulo 2 of distinct rth order products, where $0 \leq r \leq n$. The maximum order that occurs is called the *nonlinear order* of f. For example, $f(x_1, x_2, x_3) = x_1 + x_3 + x_2x_3$ has nonlinear order 2. If the S-box had low nonlinear order, then the cryptographic function would not be sufficiently algebraically complex (recall that the only nonlinear piece of the round function is the S-box). Nyberg also suggested that the S-boxes should have efficient construction and computability [**33**].

The way to thwart differential cryptanalysis is to design the system so that the differences from the round functions do not "pile up." Nyberg proposed:

Definition. Let G_1 and G_2 be finite Abelian groups. A mapping $f : G_1 \to G_2$ is *differentially δ-uniform* if for all nonzero α in G_1 and for all β in G_2

$$\left|\left\{z \in G_1 \mid f(z + \alpha) - f(z) = \beta\right\}\right| \leq \delta.$$

From the point of view of differential cryptanalysis, it would be best if a mapping f from n-bit strings to n-bit strings were differentially c-uniform, for a small constant c. But that is rather much to expect. Nyberg showed that Pieprzyk's polynomial functions had good differential properties [**33**].[6]

Theorem 2. *Let $f(x) = x^{2^k+1}$ be a polynomial in $GF(2^n)$, and let $s = \gcd(k, n)$. Then $f(x)$ is differentially 2^s-uniform.*

Pieprzyk's functions also do well in regard to distance from linear functions. One should be able to compose a function f with linear maps without affecting the function's distance from linear functions. It is well known that all linear transformations

[6] Thomas Beth and Cunsheng Ding proved related results [**2**].

from K into F can be written as maps $\beta \mapsto \mathrm{Tr}_{K/F}(\beta\alpha)$ for some fixed α in F (see, for example, [**25**, p. 52]). (When the fields involved are clear, $\mathrm{Tr}_{K/F}$ is often abbreviated Tr.) Nyberg showed that the following is true in [**33**]:

Proposition 3. *Let $f(x) = x^{2^k+1}$ be a permutation of $GF(2^n)$, where $\gcd(k,n) = s$. Then for nonzero ω in $GF(2^n)$ the Hamming distance of $\mathrm{Tr}(\omega f(x))$ from the set of linear Boolean functions is $2^{n-1} - 2^{((n+s)/2)-1}$ whenever $\omega \neq 0$.*

Unfortunately Pieprzyk's x^{2^k+1} polynomials fail another of Nyberg's criteria. The following is well known [**5**, p. 97]:

Theorem 4. *Let η be a nonzero element of $GF(2^n)$, and let $f(x) = x^e$ be a permutation of $GF(2^n)$. Then*

$$\deg\left(\mathrm{Tr}(\eta x^e)\right) = w(e),$$

where $w(e)$ is the Hamming weight of e (the number of nonzero bits in e).

The Hamming weight of $2^k + 1 = 10\ldots01$ is 2. Thus Pieprzyk's polynomials do not make the cut. But perhaps they can be transformed? Recall that inverse functions preserve properties of nonlinearity. Nyberg observed [**33**]:

Proposition 5. *If $f : G_1 \to G_2$ is a differentially δ-uniform bijection, then so is f^{-1}.*

If $f(x) = x^{2^k+1}$ with $\gcd(k,n) = 1$, then $f^{-1}(x) = x^e$ with $e = (2^{k(n+1)-1})/(2^{2k} - 1)$. This function has good Hamming weight, $(n+1)/2$. Nyberg investigated an even better function: $f(x) = x^{-1}$. Now $f(0)$ is undefined, but in cryptography there are freedoms that are lacking in mathematics. If at certain values a function is undefined, a cryptographer can strike out on her own and define the function value at such points in whatever way is convenient. For example, in the case at hand one can define $f(0) = 0$. That is what Nyberg did.

With the aid of Propositions 1 and 3, it is easy to see that the distance of f from linear functions is at least $2^{n-1} - 2^{n/2}$. And Proposition 4 shows that the degree of f is $w(2^n - 2) = n - 1$.

As Nyberg demonstrated in [**33**], the map $x \mapsto x^{-1}$ is also differentially 4-uniform. Let F be a field and consider the equation

$$(x+\alpha)^{-1} - x^{-1} = \beta,$$

where x is not equal to 0 or $-\alpha$. This equation can be rewritten as

$$\beta x^2 + \alpha\beta x + \alpha = 0$$

which, since F is a field, has at most two solutions. The equation can have two more solutions if $x = -\alpha$ or $x = 0$. Thus Nyberg's inverse function is differentially 4-uniform. Inverses can be computed efficiently using the Euclidean algorithm.[7] This inverse function satisfies all of Nyberg's requirements for a round function.

The wide-trail strategy. Meanwhile very different threads were forming the warp of Rijndael. S-boxes do not provide diffusion; other steps of the round function must

[7] This is not, however, used in the Rijndael's S-boxes, which rely on table look-up rather than computing inverses.

do that. Joan Daemen, a Ph.D. student at the Katholieke Universiteit in Leuven, Belgium, had proposed a new model for understanding round functions. Daemen's idea, a natural one in the context of differential and linear cryptanalysis, was to analyze the round function pieces—the S-box substitution and the linear maps—separately, and in particular, to ensure that a cryptanalytic attack cannot "bypass" the nonlinear aspects of the algorithm. Daemen named this approach the *wide-trail* strategy. I will sketch its main ideas, concentrating on its role in introducing polynomials into Rijndael. A fuller discussion of the wide-trail strategy appears in [**14**] and [**15**].

In the wide-trail strategy, the round transformation ρ is considered as two separate parts γ and λ [**11**]:

- γ is a nonlinear transformation that manipulates bits locally (where locally means that an output bit depends on a small set of input bits, and nearby output bits depend upon neighboring input bits);
- λ is a linear mixing transformation with high diffusion; λ can be identified with an $nm \times nm$ matrix M: $\lambda(a) = b$ if and only if $M \cdot a = b$.

Each round consists of three parts: first γ, then λ, and finally a key addition $\sigma[k^{(r)}]$, where $k^{(r)}$ is the round subkey for the rth round (see Figure 4).

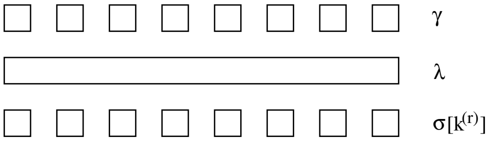

Figure 4. The wide-trail strategy: bricklaying.

These parts are analyzed separately. Let $\mathcal{E} = \rho_r \circ \rho_{r-1} \circ \cdots \circ \rho_1$ be an r-round Boolean mapping on n-bit vectors to n-bit vectors. A *differential trail* is a series of $r+1$ difference patterns: $a', d_1, \dots, d_{r-1}, b'$. The *selection pattern* of a Boolean vector is the set of places at which the vector has a "1." A *linear trail* U over $\mathcal{E}$ is a sequence of $r+1$ selection patterns, $U = (u_0, u_1, \dots, u_r)$. In the wide-trail approach, all S-boxes are identical.

The wide-trail strategy is to operate as follows:

- Choose an S-box for which both the maximum probability of any differential trail and the maximum correlation of linear combinations of input and output bits are as small as possible.
- Choose the linear transformations so that there are no trails with few active S-boxes.

The latter is the "wideness" of the wide-trail strategy. The strategy is designed to spread diffusion broadly, to develop diffusion according to mathematical principles, and to perform diffusion in a mathematically analyzable way. This contrasts with DES diffusion, which is statistically analyzable but lacks any apparent structure.

In DES, diffusion occurs at the bit level. Daemen and Rijmen proposed a different model. They ignored the diffusion that occurs within the S-boxes and concentrated on the diffusion provided by λ, the "linear-mixing" part of the transformation.

Call equal-sized subsets of bits *bundles*, and let a *bricklayer function* be a mapping that is decomposable into several Boolean functions operating independently on bundles. Each S-box will be an m-bit-to-m-bit box, and there will be n S-boxes (thus the cipher works on an nm-bit block). Analyzing diffusion in terms of subsets of bits may not be intuitive, but by using the bundle formalism, Daemen and Rijmen constructed functions with provably good diffusion. From now on, mappings will be discussed in terms of bundles.

As in the terminology used for S-boxes, a bundle is *active* if it has nonzero input difference. In both differential and linear cryptanalysis, one is interested in the bits that are "on." Define the *parity of a Boolean vector* $\mathbf{b}$ to be $\mathbf{e}^T\mathbf{b}$, where $\mathbf{e}$ is a "selection vector" consisting of zeros and ones ($\mathbf{e}$ selects which bits of $\mathbf{b}$ to XOR together). Following Daemen and Rijmen, the *bundle weight* of a configuration (a configuration registers the settings of all the bits at a particular point) is the number of active bundles. Applied to a difference pattern a', $w_b(a')$ is the number of active bundles in a'; applied to a selection pattern v, $w_b(v)$ is the number of active bundles in v. Daemen and Rijmen introduced the following definitions for quantifying the diffusion in λ, the linear-mixing part of the round:

Definition. The *differential branch number* of a transformation $\theta : \{0, 1\}^n \to \{0, 1\}^n$ is

$$\mathcal{B}_d(\theta) = \min_{a \neq b} \left\{ w_b(a \oplus b) + w_b\big(\theta(a) \oplus \theta(b)\big) \right\},$$

where w_b is the number of active bundles.

Definition. The *linear branch number* of a linear transformation $\lambda : \{0, 1\}^n \to \{0, 1\}^n$ is

$$\mathcal{B}_l(\lambda) = \min_{\alpha \neq 0} \left\{ w_b(\alpha) + w_b(M^T\alpha) \right\},$$

where α is the bundle partition and M is the matrix such that $\lambda(x) = M \cdot x$.

Daemen and Rijmen observed the following properties of branch numbers:

1. the branch number of a permutation and its inverse are equal;
2. branch numbers are unaffected by key addition; and
3. a bricklayer permutation operating on individual bundles does not change active bundles into inactive ones, or vice versa.

Thus if a cipher $\mathcal{E}$ consists of two transformations θ_1 and θ_2, with θ_2 being a bricklayer function operating on bundles, the branch numbers of $\theta_2 \circ \theta_1$ and θ_1 are equal.

For the cryptanalyst analysis is easiest when just a single S-box is active. Thus the designer of a cryptographic algorithm seeks to avoid worst-case diffusion, which occurs with a single active bundle. Clearly, the best that can be done from a designer's point of view is for the upper bound for branch numbers $\mathcal{B}$ to be $n + 1$, where n is the number of bundles (each bundle will have m bits). An invertible linear mapping that achieves this is called *optimal*. Maximal distance separable codes provide a way of constructing such optimal linear maps.

Definition. A k-dimensional subspace of the vector space $GF(2^m)^n$ that satisfies the condition that any pair of distinct vectors of the subspace have Hamming distance at least d, and d is the largest number with such a property, is a *linear* (n, k, d)*-code* over $GF(2^m)$.

Codes with $d = n - k + 1$ are *maximal distance separable codes*; Reed-Solomon codes are a well-known example. These can have length up to $q + 1$, where q is the size of the finite field over which the codes are defined. A linear code C of length n and dimension k can be represented by a $k \times n$ generator matrix G. Then C is formed by the subspace of dimension k spanned by the rows of G. There are various representations for C; namely, if T is a $k \times k$ matrix of full rank, then TG is also a generator matrix for C. The matrix $G_e = TG$ of the type $[I \mid B]$, where I is the $k \times k$ identity matrix and B is a $k \times (n - k)$ matrix, is called the *echelon form* of G. Daemen and Rijmen proved [**40**, p. 103]:

Theorem 6. *Let C be a $(2n, n, n + 1)$-code over $GF(2^m)$, and let $G_e = [I|B]$ be the generator matrix for C in echelon form. Then C defines an optimal invertible linear mapping $\gamma : GF(2^m)^n \to GF(2^m)^n$ via*

$$\gamma : X \mapsto Y = BX.$$

This gives a straightforward way to create a bricklayer map. “Bricks” can be n m-bit-to-m-bit S-boxes, while the diffusion layer is a linear map corresponding to a $(2n, n, n + 1)$-code over $GF(2^m)$. Rijmen, Daemen, and colleagues from Katholieke Universiteit built the first instantiation of the wide-trail method: SHARK.

Putting the pieces together: SHARK and Square. SHARK was a 6-round cipher operating on 64-bit blocks that combined the wide-trail linear diffusion approach with Nyberg’s S-boxes [**40**]. SHARK(n, m, r) consisted of r rounds, with n m-bit S-boxes. Each round consisted of three steps:

- one added in the key;
- one transformed the bundles;
- one was a linear map on the bundles (designed as indicated earlier).

The algorithm ended with one extra key addition and an inverse diffusion operation. (The inverse diffusion was employed to simplify inversion.)

But not all was well. The simple function $f(x) = x^{-1}$ presented a problem. Jakobsen and Knudsen found an attack on the simple function [**19**]; the attack was not actually applicable to SHARK itself, which was already using a more complex S-box function than x^{-1}. I follow [**19**], beginning my explanation of the Jakobsen-Knudsen attack with SHARK$(1, m, r)$, SHARK with a single S-box.

With a single S-box, the linear mapping is multiplication by a constant. In each round, the key is added, and the result is inverted and then multiplied by an element in $GF(2^m)$. Regardless of the number r of rounds the ciphertext C can be expressed as a simple fraction of the plaintext P,

$$C = \frac{P \oplus a}{bP \oplus c},$$

with key-dependent constants a, b, and c. Using only three plaintext/ciphertext pairs, one can determine a, b, and c. The key is not revealed; one is able to encrypt and decrypt without it.

With two S-boxes (SHARK$(2, m, r)$), the situation gets more complicated. One views the input plaintext P DES-fashion as $P = P_L P_R$ (P_L is the left half of the plaintext, P_R is the right half). Let L_i be the left half after i rounds, R_i the right half.

Then:

$$L_i = p_i(P_L, P_R)\big/q_i(P_L, P_R),$$
$$R_i = r_i(P_L, P_R)\big/s_i(P_L, P_R),$$

with p_i, q_i, r_i, and s_i in $GF(2^{32})[P_L, P_R]$.

Now we know that

$$L_i = \frac{a_1}{L_{i-1} \oplus k_{iL}} \oplus \frac{a_2}{R_{i-1} \oplus k_{iR}},$$

where k_{iL} and k_{iR} are the keys for the left and right halves, respectively, of round i, and a_1 and a_2 are constants. This simplifies to

$$L_i = \frac{p_i(P_L, P_R)}{(L_{i-1} \oplus k_{iL})(R_{i-1} \oplus k_{iR})}.$$

If e_i signifies the degree of q_i, then it is clear that $e_i \leq 2e_{i-1}$. Since $e_1 = 1$, this shows that $e_i \leq 2^{i-1}$. The degrees of p_i, r_i, and s_i are similarly bounded by 2^{i-1}.

After r rounds, the number of coefficients in p_r and q_r is $(2^{r-1} + 1) + (2^{r-1} + 1) = 2(2^{r-1} + 1)$. Note that knowing these pairs will also determine r_i and s_i. Thus computing $2(2^{r-1} + 1)$ plaintext/ciphertext pairs would break SHARK$(2, m, r)$ with a simplified S-box. Since the value proposed for r was 6, this attack is quite feasible.

There is nothing special about the choice of 2 in the preceeding argument. If one attempts the same attack on SHARK(n, m, r), a similar argument shows that $2(n^{r-2} + 1)^n$ plaintext/ciphertext pairs are needed to determine the coefficients of the interpolation polynomials. (Actually, since SHARK uses an "inverse" diffusion layer, there are only $r - 1$ diffusions and thus only $2(n^{r-2} + 1)$ plaintext/ciphertext pairs are required.) This number is *independent* of the size of the S-box.

The weakness in the cryptosystem is that the S-box function can be expressed as a rational function of low degree, and thus coefficients can be determined from a small number of plaintext/ciphertext pairs. A more complex S-box function is necessary. To paraphrase Einstein, S-boxes "should be as simple as possible, but no simpler."

Daemen and Rijmen wanted to preserve the properties for which Nyberg had chosen x^{-1} (high nonlinearity, high nonlinear order, resistance against differential cryptanalysis, efficient construction and computability). An obvious choice was to take x^{-1} and compose it with a simple function that would add sufficient algebraic complexity. Daemen and Rijmen chose an affine function.

In the composition, each input byte b is first viewed as an element of

$$GF(2^8) \simeq Z_2[x]/(x^8 + x^4 + x^3 + x + 1);$$

the mapping is the natural embedding $b = b_7b_6 \dots b_0 \mapsto b_7x^7 + b_6x^6 + \cdots + b_0 = b(x)$. The first mapping sends the polynomial $b(x)$ to $(b(x))^{-1}$ in the field $Z_2[x]/(x^8 + x^4 + x^3 + 1)$ (with $b(x) = 0$ being sent to 0). The resulting byte is then mapped to $(x^7 + x^6 + x^3 + x^2) + (x^7 + x^6 + x^5 + x^4 + 1)b(x)$ and reduced modulo $(x^8 + 1)$.

Daemen and Rijmen picked $x^8 + 1$ as the modulus polynomial because it was simple—as simple as one could get. The affine multiplier $x^7 + x^6 + x^3 + x^2$ is the simplest among those polynomials coprime to $x^8 + 1$ [**12**].

The resulting S-box has no fixed points (S-box$(a) = a$) and no "opposite" fixed points (S-box$(a) = \overline{a}$, the complement of a). Although there are no known attacks against S-boxes with fixed points or opposite fixed points, the more simplicities and potential points of attack one can eliminate in a cryptographic algorithm, the better.

SHARK was defined with a simple S-box, but an efficient implementation on a PC would require large lookup tables. In their next cipher, Daemen and Rijmen traded S-box complexity for a more complex array layout. Now diffusion had to proceed "up" and "across," but this extra step was compensated for by simpler S-boxes. This system, Square, a 128-bit block cipher with 128-bit key, placed the data into a 4×4 array of bytes.

There was a third change: a simpler implementation of row diffusion. In SHARK diffusion was accomplished by way of a Reed-Solomon maximal distance separable code defined by the echelon form B of the generation matrix of a $(2n, n, n+1)$-code. From an implementation viewpoint, the SHARK diffusion matrix was less than optimal: matrix entries were in $GF(2^8)$ and each row of the matrix had different entries. Multiplication time would be faster if the coefficients were from a smaller field, while storage would be simplified if the matrix rows were the same. To accomplish either of these is impossible if the matrix is to be of full rank. But through the use of circulant matrices Daemen and Rijmen found a way to come close to meeting both of these objectives. A *circulant matrix* is one where each row is a circular shift one over of the previous row: $b_{ij} = b_{0,j-i(\bmod n)}$. Combinatorics furnishes the connection [**26**]:

Theorem 7. *An (n, k, d)-code C with generator matrix $G_e = [I \mid B]$ is maximal distance separable if and only if each square submatrix of B is nonsingular.*

A probabilistic analysis shows that there are many 4×4 maximal distance separable circulant matrices; Daemen and Rijmen choose a matrix C with "small" coefficients in $GF(2^8)$ (entries are in hexadecimal), namely,

$$C = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}.$$

Bytes of the row are viewed as coefficients of a polynomial in $GF(2^8)[x]/(x^4+1)$. The column mapping is multiplication in $GF(2^8)$ by the polynomial $3x^3 + x^2 + x + 2$ modulo $x^4 + 1$. For example, the first row of C corresponds to the polynomial $2x^3 + 3x^2 + x + 1$.

Square has four simple steps in each round:

- one transforms the rows;
- one transforms the bytes;
- one transposes the array;
- one adds in the key.

Square begins with an inverse row transformation (or alternatively, the first round of Square omits the row transformation). The byte transformation is Nyberg's x^{-1} function followed by the aforementioned affine map.

Knudsen discovered an attack on the 6-round version of Square. I follow [**11**] but present only a simplified version of Knudsen's attack on five rounds of Square. The attack focusses on the byte-oriented nature of the algorithm.

Let Λ be a set of 256 states that are all different in specified bytes and are the same in the remaining bytes; call these *active* and *inactive* bytes, respectively. Consider how Square's mappings affect active and inactive bytes. The S-box transformation does not alter the configuration of which bytes are active and which are inactive. The transposition mapping effects a change in the status of bytes, but the number of active bytes remains the same. In contrast, the row mapping affects the number of active bytes: what was a single active byte becomes an active row.

Let λ index the active bytes, that is, for x and y in Λ

$$\begin{cases} x_{ij} \neq y_{ij} & \text{for } (i, j) \in \lambda, \\ x_{ij} = y_{ij} & \text{for } (i, j) \notin \lambda. \end{cases}$$

Now let us construct a set Λ in which only a single byte is active. Note that this byte cycles through all possible $2^8 = 256$ values. Since the first round of Square does not include the row diffusion, at the end of the first round only one byte is active. During the second round of Square, the row operation transforms this to an active row; the transpostion operation transforms this into an active column. Now there are four bytes that cycle through all 256 values while the remaining twelve bytes are constant. During the third round, each of the bytes in the active column is transformed into an active row; by the end of the third round, all bytes are active and cycle through all values.

We call Λ *balanced* if its entries range over all possible values. At the end of the third round, the output bytes are balanced over the Λ-set. Let a^l_{ij} be the bytes at the input of the fourth round, let b^l_{ij} be the bytes at the output of the fourth round, and let θ be the row operation. At the output of the fourth round, we find:

$$\begin{aligned} \oplus_{l\in\Lambda} b^l_{ij} &= \oplus(2 \cdot a^l_{ij} \oplus 3 \cdot a^l_{i+1,j} \oplus 1 \cdot a^l_{i+2,j} \oplus 1 \cdot a^l_{i+3,j}) \\ &= (2 \oplus_l a^l_{ij}) \oplus (3 \cdot \oplus_l a^l_{i+1,j}) \oplus_l (1 \cdot \oplus a^l_{i+2,j}) \oplus (1 \cdot \oplus_l a^l_{i+3,j}) \\ &= 0 + 0 + 0 + 0 \\ &= 0. \end{aligned}$$

Thus the bytes at the output of the fourth round are balanced. However, the next step, the S-box operation of the fifth round, destroys that balance.

As in all other rounds, an output byte in the fourth round is a function of an intermediate state and the key byte. We can guess a key value for the fourth key and from this calculate the intermediate states in the fourth round. If the value of this byte is not balanced over Λ, then the guessed key byte was incorrect. We can try doing this for all possible key bytes. All but one should be wrong. Knudsen implemented this attack and found that two Λ-sets were sufficient to guess the key with an overwhelming probability of success. The attack takes 2^9 time using 2^9 plaintexts.

In the attack of five rounds of Square, we assume a value for four bytes of the fifth round key. As before, wrong key assumptions are determined by verifying that the bytes in the fifth round are not balanced. This attack needs 2^{40} key values to be checked.

Knudsen showed how to extend this to an attack on six rounds that took 2^{72} steps. Square needed a better *security margin*, a good ratio between the number of rounds for which there is a known attack and the actual number of rounds of the algorithm. Daemen and Rijmen recommended that Square be implemented with at least eight rounds.

3. THE ADVANCED ENCRYPTION STANDARD. Activity was simultaneously proceeding on another front. Although the U.S. Government had long resisted the idea, by the mid-1990s it was clear that DES needed replacing. In 1997 the National Institute of Standards and Technology (NIST) announced a competition for an Advanced Encryption Standard (AES). The process by which DES had become a Federal encryption standard had been less than transparent, and there had been significant public distrust of DES. This time NIST wanted an open process.

NIST sought public comment on proposed requirements for the Advanced Encryption Standard. Cryptographers responded. NIST listened, modifying technical and legal requirements. In January 1997 NIST announced a competition for a symmetric-key algorithm running on 128-bit blocks of data using 128-, 192-, or 256-bit keys. Designers would be required to give up proprietary rights; the algorithm would be internationally available without royalties.

Cryptographers from around the world submitted candidates; fifteen met NIST's requirements. NIST organized conferences in California and Rome. Attacks were presented. NIST encouraged public discussion on its AES Web pages (aes.nist.gov). After nineteen months of evaluation, NIST announced five finalists: MARS, RC6, Rijndael, Serpent, and Twofish. For the most part, the public community agreed with NIST's choices. So did the NSA, which advised NIST that its choices were "appropriate."

There was another year of study. In October 2000, NIST announced that Rijndael was "the best overall algorithm for the AES ... [the algorithm's] combination of security, performance, efficiency, implementability, and flexibility make it an appropriate selection for the AES." After an additional year of evaluation, in November 2001 the Department of Commerce officially declared Rijndael the Advanced Encryption Standard.

The very public evaluation effort has led to widespread acceptance of AES. The cryptography community is a rather contentious lot, but it has been virtually unanimous in its praise of NIST's AES effort and the choice of Rijndael as the Advanced Encryption Standard. This is high praise indeed.

Rijndael Specifications. Daemen and Rijmen said their goal was an algorithm that would be resistant to known attacks, would exhibit speed and compactness on a variety of platforms, and would have a simple design. The latter is one way Rijndael stood out from the competition.

I will present the 128-bit key Rijndael; the other versions differ only in number of rounds and data array size. The algorithm should look *very* familiar.

In the 128-bit key version, data are placed in a 4×4 array and the algorithm has ten rounds. As required by NIST, Rijndael operates on a 128-bit block of data. The algorithm divides the block into sixteen 8-bit bytes and all operations occur on the bytes of the array. Each Rijndael round consists of four operations:

- one transforms the bytes;
- one tranforms the rows;
- one transforms the columns;
- one adds in the key.

Rijndael begins with a key addition, and the last round of Rijndael omits the column transformation. The byte substitution is the same as Square's and is the only nonlinear aspect of Rijndael. In the row mixing, the ith row is circularly shifted $i - 1$ columns to the right. The column mixing is identical to Square's. The last operation is an XOR of the key bits with the elements of the array. The 128-bit key is put into a linear array of 4-byte words (exactly like the 4×4 block array of the data in Square).

Key bytes are XORed with the data in the obvious way. Although I have not emphasized key schedules in my discussion, proper key schedules are critical in algorithm design. Rijndael's consists of the original 128-bit key, followed by pieces 128 bits long that have several simple transformations applied to them. Each 128-bit section consists of four 4-byte words with each word being the XOR of the preceeding 4-byte word and, except for the first word in the 4-byte section, the corresponding word in the previous piece. For the first word in the 4-byte section, the word is first rotated one byte to the left, then the bytes are transformed by the Rijndael S-box. Finally a round-dependent constant (see [**12**] for details) is XORed with these bytes.

Symmetry is broadly employed in Rijndael's design, but symmetry does not appear in the key schedule. That is to avoid any chance of *equivalent keys*, a pair of keys giving the same encryption.

Is Rijndael secure? The choice of Rijndael did not please everyone. Several well-known cryptographers objected to the algorithm's algebraic simplicity, fearing this would enable attacks. In their submission to NIST, Daemen and Rijmen had observed that the action of the S-box on the bytes b_{ij} could be written:

$$S(b_{ij}) = \lambda_8 + \Sigma_{d=0}^{7}\lambda_d b_{ij}^{255-2^d}$$

for certain constants $\lambda_0, \ldots, \lambda_8$ [**12**]. That simplicity looked enticing to cryptanalysts.

In comments to NIST, Richard Schroeppel expressed concerns about the byte-structure of Rijndael and the fact that the algorithm was mostly linear. But Schroeppel's biggest worry was that the only nonlinear aspect of Rijndael was x^{-1}, which he felt was too simple given that the underlying field was of characteristic 2 [**42**]. By moving the constant λ_8 into the key addition step, Daemen and Rijmen's equation becomes

$$S(b_{ij}) = \Sigma_{d=0}^{7}\lambda_d b_{ij}^{-2d},$$

where the arithmetic is in $GF[2^8]$.

Other steps of Rijndael have simpler algebraic expressions. Niels Ferguson, Doug Whiting, and Schroeppel proposed the following method of attack on Rijndael [**17**]. Let k_{ij}^r be the key at the (i, j)th position in round r and, as before, let the arithmetic be in $GF[2^8]$. Then:

$$b_{ij}^{r+1} = k_{ij}^{(r)} + \Sigma_{e_r\in\varepsilon, d_r\in\mathcal{D}}\lambda_{i,e_r,d_r}(b_{e_r,e_r+j}^{(r)})^{-2^{d_r}},$$

with $\varepsilon = \{0, 1, 2, 3\}$ and $D = \{0, \ldots, 7\}$. Note that this can be rewritten as

$$b_{ij}^{r+1} = k_{ij}^{(r)} + \Sigma_{e_r\in\varepsilon, d_r\in\mathcal{D}}\frac{\lambda_{i,e_r,d_r}}{(b_{e_r,e_r+j}^{(r)})^{2^{d_r}}}.$$

Consider what happens with 128-bit Rijndael, that is, when $r = 10$. In general, each round of the computation is likely to double the number of terms, running into "intermediate value swell." But characteristic 2 changes that. Raising elements to a power of 2 in a field of characteristic 2 causes intermediate terms to disappear. Thus

$$b_{ij}^{(3)} = k_{ij}^{(2)} + \Sigma_{e_2\in\varepsilon, d_2\in\mathcal{D}}\frac{\lambda_{i,e_2,d_2}}{\left(k_{e_2,e_2+j} + \Sigma_{e_1\in\varepsilon, d_1\in\mathcal{D}}\frac{\lambda_{e_2,e_1,d_1}}{(b_{e_1,e_1+e_2+j})^{2^{d_1}}}\right)^{2^{d_2}}}.$$

Another way to put this is:

$$b_{ij}^{(3)} = k_{ij}^{(2)} + \Sigma_{e_2 \in \varepsilon, d_2 \in \mathcal{D}} \frac{\lambda_{i,e_2,d_2}}{(k_{e_2,e_2+j}^{(1)})^{2^{d_2}} + \Sigma_{e_1 \in \varepsilon, d_1 \in \mathcal{D}} \frac{(\lambda_{e_2,e_1,d_1})^{2^{d_2}}}{(b_{e_1,e_1+e_2+j}^{(1)})^{2^{d_1+d_2}}}}.$$

Cut to essentials and replace expanded key bytes with K, constants with C, and subscripts and exponents with asterisks, understanding that these K^* and C_* represent different values. This will give a sense of the formulas. After six rounds, the expression takes the form

$$b_{ij}^{(6)} = K^* + \Sigma_{e_5 \in \varepsilon, d_5 \in \mathcal{D}} \frac{C_*}{K^* + \Sigma_{e_4 \in \varepsilon d_4 \in \mathcal{D}} \frac{C_*}{K^* + \cdots}}$$

After six rounds, there are 2^{35} terms of the form $C_*/(K^* + p_*^*)$. After ten rounds of Rijndael, there are 2^{50} such terms. If we could efficiently solve these 2^{50} equations, we could break Rijndael. There are points to note, however. The first is that the equations are an oversimplification, since the C_*, K^*, and p_*^* represent different values each time they appear. The key word here is *efficiently*. Ferguson et al. claim that a symbolic equation program that worked in $O(n^7)$ steps, where n is the number of terms, would enable an attack on Rijndael faster than a brute-force search of the key space. But this is a strong claim for a proposed attack with so many simplifying assumptions built into it.

Sean Murphy and Matt Robshaw of Royal Holloway College took a different tack to the same general approach [**31**]. In [**29**], they pulled Rijndael apart, but unlike the king's horses and the king's men, Murphy and Robshaw were able to put the algorithm back together again. The revision showed a possible direction for attack.

Since the same constant is added to each byte in the S-box, while the rest of the round function consists of row shifts, multiplication by a matrix in $GF(2^8)$, and key addition, Murphy and Robshaw suggested that Rijndael's round operations could be regrouped by putting the addition, appropriately modified, into the key-addition step. They argued that this arrangement was more "natural": with it, the S-box linear diffusion step can be part of the linear-diffusion layer.

Murphy and Robshaw considered the entire linear-diffusion layer as a $GF(2)$-linear map, given by a 128×128 matrix $\mathcal{M}$ over $GF(2)$. They noticed two surprising things about $\mathcal{M}$. First, its characteristic polynomial, $c(x) = (x+1)^{128} = x^{128} + 1$, and its minimal polynomial, $m(x) = (x+1)^{15}$, were both remarkably simple. Second, $m(x)$ was of small degree. They noted that sixteen iterations of $\mathcal{M}$ give the identity permutation.

Yes, replied Daemen and Rijmen, one can play such games with the linear does the fact that $\mathcal{M}^{16} = I$ really mean? If you combine key addition with itself, you get back the identity. So what? We opted for simplicity, Daemen and Rijmen continued, and so the fact that $\mathcal{M}^{16} = I$ is no surprise. That does not mean that Rijndael is insecure. One could view DES through the same type of glasses and discover, for example, that by reordering the output bits of the S-boxes and appropriately modifying the round permutation P, one still gets DES, but now the diffusion layer will give the identity after only twelve applications [**39**], rather than the sixty one gets with the usual version of P. Daemen and Rijmen argued that, while looking at the separate pieces of a cryptographic round function is a useful and important exercise, the round function as a whole is greater than the sum of its parts.

The volley returned to Murphy and Robshaw [**30**]. Their main point remained "whether the cryptanalyst can find a more novel way to combine the rich structure in the diffusion layer of Rijndael with the highly structured inverse map" [**29**]. Two years later Murphy and Robshaw proposed one [**31**].

Two aspects of Rijndael add to the complexity of analyzing the algorithm: the non-linear S-box operation and the S-box addition, which is linear over $GF(2)$; all other steps are linear over $GF(2^8)$. Murphy and Robshaw saw a way to embed AES in a larger system that, except for the inversion, would have all steps linear over $GF(2^8)$.

View the state space **A** of the 128-bit version of AES as a vector space over $GF(2^8)$. Define two sets: **B**, a cipher with a 128-byte message and key space that is a 128-dimensional vector space over $GF(2^8)$, and $\mathbf{B_A}$, a subset of **B** that corresponds to **A**. The set **B** will be the state space of the *Big Encryption System*, or BES.

Let F be a field, and let $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1})$ be a vector in F^n (F will be $GF(2^8)$ and n will be 8 and 64 for AES and BES, respectively). With 0 being mapped to 0, inversion is done componentwise: $\mathbf{a}^{-1} = (a_0^{-1}, a_1^{-1}, \ldots, a_{n-1}^{-1})$. For any a in F, define the *vector conjugate* $\tilde{\mathbf{a}}$ to be $(a^{2^0}, a^{2^1}, a^{2^2}, \ldots, a^{2^7})$. This gives a natural *vector conjugate mapping* ϕ from F^n to F^{8n}: namely, for $n = 1$ we let

$$\phi(a) = \tilde{\mathbf{a}} = (a^{2^0}, a^{2^1}, a^{2^2}, \ldots, a^{2^7}),$$

and extend this definition in the obvious way by setting

$$\phi(\mathbf{a}) = \big(\phi(a_0), \phi(a_1), \ldots, \phi(a_{n-1})\big)$$

for $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1})$ in F^n. Because the underlying field has characteristic 2,

$$\phi(\mathbf{a} + \mathbf{a}') = \phi(\mathbf{a}) + \phi(\mathbf{a}'),$$
$$\phi(\mathbf{a}^{-1}) = \phi(\mathbf{a})^{-1}.$$

The set $\mathbf{B_A}$ is defined so that $\mathbf{B_A} = \phi(\mathbf{A})$, a subset of **B**, is the *AES subset of BES*. Murphy and Robshaw view the state vector in AES as a column vector

$$\mathbf{a} = (a_{00}, a_{10}, \ldots, a_{30}, a_{01}, \ldots, a_{33})^T.$$

Similarly the state vector

$$\mathbf{b} = (b_{000}, \ldots, b_{007}, b_{100}, \ldots, b_{107}, \ldots, b_{777})^T$$

is written as a column vector. Then BES is defined so that the diagram in Figure 5 commutes:

$$\begin{tikzcd}
 & \mathbf{A} \arrow[r, "\phi"] \arrow[d] & \mathbf{B_A} \arrow[d] & \\
k \arrow[r] & \boxed{\text{AES}} \arrow[d] & \boxed{\text{BES}} \arrow[d] & \phi(k) \arrow[l] \\
 & \mathbf{A} & \mathbf{B_A} \arrow[l, "\phi^{-1}"'] &
\end{tikzcd}$$

Figure 5. The relationship between AES and BES.

Using the same technique as in [**29**], the BES S-box is limited to inversion and all linear-diffusion steps are combined into one step. The addition step of the S-box, suitably modified, becomes part of key addition. The BES S-box is inversion performed componentwise (with 0 mapped to 0).

The BES linear-diffusion step is a natural combination of the three pieces of AES linear diffusion: the multiplication step of the S-box, the row operation, and the column multiplication. Because of space constraints, I do not include details, but note that the only complex piece of this step is the column multiplication. Since the BES states are: $a_{00}^{2^0} (= a_{00}), a_{00}^{2^1}, \ldots, a_{00}^{2^7}, a_{01}^{2^0}, a_{01}^{2^1}, \ldots, a_{33}^{2^7}$, the column multiplication in BES is done by using powers of the circulant matrix generated by 2, 3, 1, and 1 (elements in $GF(2^8)$) to build a 128×128 matrix over $GF(2^8)$.

The AES S-box addition is a $GF(2)$-linear operation, so there is no natural way to represent it as a matrix multiplication. Murphy and Robshaw honed in on the same equation that Ferguson et al. had found useful:[8]

$$f(a) = \Sigma_{k=0}^{7} \lambda_k a^{2^k}$$

for a in F with $\lambda_0, \lambda_1, \ldots, \lambda_7 = 5, 9, F9, 25, F4, 1, B5, 8F$, respectively (coefficients are in hexadecimal) [**15**, p. 212], [**31**, p. 7].

The conjugates are in BES, so it is a simple matter to determine the matrix that replicates the AES addition action in BES. If $L_B = (\lambda_i^{2^j})$ then as Murphy and Robshaw observed [**31**, p. 8], "the entire set of $GF(2)$-linear operations in AES [can be represented] with a 128×128 matrix in the BES Lin_B," a block diagonal matrix with sixteen identical blocks L_B.

Murphy and Robshaw's key observation is that BES's round function can be written as [**31**, p. 8]:

$$\mathbf{b} \mapsto M_B \mathbf{b}^{-1} + k_{B_i},$$

where k_{B_i} is the ith round subkey for BES. Thus, a round of AES is "simply componentwise inversion and an affine transformation with respect to the *same* field $F = GF(2^8)$" [**31**, p. 9].

Murphy and Robshaw noted a number of "interesting" facts about BES:

- The linear diffusion matrix M_B of BES is sparse, with minimal polynomial $(x+1)^{15}$. This means that M_B is, in some sense, no more complicated than the equivalent matrix for AES. The matrix M_B can be put into Jordan canonical form $R_B = P_B^{-1} M_B P_B$, where R_B has 112 rows with two ones and sixteen rows with a single one (all other entries are zeros). But the most important issue is that the properties of M_B are properties over $GF(2^8)$, not over $GF(2)$.
- This new "version" of AES did not show particular vulnerabilities to differential or linear cryptanalysis, but an interesting parity relationship was exposed.

  Define a *parity equation* as a row vector $\mathbf{e}^T$. The matrix M_B fixes a subspace of $\mathbf{B}$ of dimension 16; the set of parity equations fixed by M_B forms a 16-dimensional vector subspace over $GF(2^8)$.

  Suppose the vector $\mathbf{p}$ has parity $p_e = \mathbf{e}^T \cdot \mathbf{p}$, so $t \cdot \mathbf{p}$ has parity tp_e for any t in F. Murphy and Robshaw found related subkeys k_i and tk_i such that the encryptions under BES of p_e and tp_e are c_e and tc_e, respectively. This holds with probability 1. Thus with the right set of related keys $\mathbf{k}_0, \ldots, \mathbf{k}_{10}$ and $t\mathbf{k}_0, \ldots, t\mathbf{k}_{10}$, if the plaintext

[8] The constant term has been folded into the key addition.

difference parity of $\mathbf{e}^T(\mathbf{p}_0 + \mathbf{p}_1)$ is p_e and of $\mathbf{e}^T(t\mathbf{p}_0 + t\mathbf{p}_1)$ is tp_e, then the ciphertext difference parities are c_e and tc_e, respectively.

  This is not quite as disturbing as it sounds. As Murphy and Robshaw note, these observations do not actually hold when the specifics of Rijndael's key schedule are taken into account. But they are interesting—and surprising.

- The encryption algorithm "preserves" algebraic curves. If one thinks about it, this is no surprise. Cryptographic functions are polynomial functions, so of course curves are preserved. What is interesting is that the curves are simple, that is, their equations do not involve many terms.

  What this tells us is that recovering an AES key is equivalent to solving a system of extremely sparse multivariate quadratic equations by expressing BES (and thus AES) as such a system. Murphy and Robshaw's analysis shows that, modulo the simplifying assumption that neither the key bytes nor the plaintext bytes are 0 (these assumptions are true for 53 percent of encryptions and 85 percent of 128-bit keys[9]), an AES encryption consists of an overdetermined multivariate system of 2,688 equations over $GF(2^8)$, of which 1,280 are sparse quadratics; the remainder are linear. The sparseness of the quadratics is what is important here. There are also a large number of equations that arise from the key schedule.

Before the reader jumps to her favorite symbolic computation system in an effort to crack an AES-encrypted message, she should note some facts:

- No current symbolic computation system can handle such a large system of equations. The algorithm by Faugere, Gianni, Lazard, and Mora for Grobner bases is the current best technique for such type of simplifications [**16**]. But in practice Grobner-basis techniques cannot currently solve systems with more than fifteen variables [**8**, p. 392].
- The general problem of solving large systems of simultaneous equations of degree greater than one is $\mathcal{NP}$-complete. Thus solving the general problem is polynomial-time equivalent to a large number of problems for which no polynomial-time solution is known (including the traveling salesman problem, satisfiability of Boolean formulas, etc.). Is this version of equation solving as hard as the general one? No one knows. In 2000, Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir showed how to solve randomly-generated systems of polynomial equations in expected subexponential time when the number of equations m is greater than ϵn^2, where $0 < \epsilon < 1/2$ and n is the number of variables. There is also strong evidence that randomly-generated sets of equations can be solved in subexponential time when $m > Bn$ for a number B that grows slowly with n.

  But—as the first sentence in the previous paragraph should suggest to the reader—perturb an easily solved discrete computational problem and one may end up with a computationally infeasible one instead. For example, solving $x^k = a$ in $GF(p^n)$ for a prime p can be done in $O(k \log^4 q)$ steps [**1**, p. 161]. Solving $x^k \equiv a (\mathrm{mod}\, pq)$ for distinct unknown primes p and q whose product pq is known is much harder (it is equivalent to decrypting a message that is encrypted under the RSA system).

Does Rijndael's algebraic formulation make the algorithm easier to crack than other cryptosystems not designed this way? I believe not. Rijndael's structure means that the algorithm lends itself to the type of analysis proposed by Ferguson et al., but we could attempt this analysis on other algorithms.

[9] If the assumption is false, some of the equations are incorrect.

A natural one to examine is DES. Here the randomness of the S-boxes and the P permutation works against us. The Boolean expressions for the S-boxes are complicated: on average each input bit is used at least sixteen times in the expression for the output bits [**22**]. The full DES formula has $16^{16} (= 2^{64})$ terms, and there are no "neat" structures to exploit.

Skipjack, the 80-bit encryption algorithm developed in the early 1990s by NSA for use in "Clipper" chips, is perhaps a better comparison. Skipjack has an 8-bit-to-8-bit S-box, and thus is closer in structure to Rijndael than DES is. Skipjack's S-boxes have a nice, regular structure. Discovering the complexity of Skipjack's S-box expressions might lend insight into both Skipjack and Rijndael.

The design goals for Rijndael. For Rijndael's designers, security was the primary concern, efficiency a close second. But Daemen and Rijmen sought simplicity—simplicity of specification and simplicity of analysis [**15**, p. 65]. Not every cryptographer sees simplicity as an important goal—two AES finalists, MARS and Twofish, have far more complex designs. (Some observers felt that this complexity was part of the reason the two algorithms were not chosen as the Advanced Encryption Standard, as their round functions were simply too difficult to analyze fully.) The wide-trail strategy is a simple paradigm to protect against differential and linear cryptanalysis. Through the various algorithms that employed the strategy—SHARK, Square, and Rijndael—Daemen and Rijmen used simple primitives. Thus Square's step that proved to be not quite good enough, the transposition, was a natural and simple way to implement column mixing. In Rijndael, the next instantiation of wide trail, this step was replaced by row shifting, an only-slightly-more complex function that appears to have achieved its goal.

Simplicity begets other criteria, including symmetry. Rijndael exhibits symmetry across the rounds, and within them.

Daemen and Rijmen had an unstated goal in their algorithm design: transparency. The two designers wanted no suspicion of trapdoors. The polynomial defining the field is the first entry in Lidl and Niederreiter's table of irreducible polynomials of $GF(2^8)$ [**25**]. In using a public list to pick their parameters, Rijndael's designers showed there was nothing up their sleeves. The simplicity of $x^8 + 1$ and $x^4 + 1$ is another demonstration of the lack of hidden parameters.

Daemen and Rijmen wanted column mixing to be invertible, to be linear in $GF(2)$, to diffuse well, and to be fast on 8-bit processors. Polynomial multiplication satisfies invertibility, linearity in $GF(2)$, and is simple to describe, so it was a natural choice. The simpler the multiplicative coefficients are—recall that though the polynomial coefficients appear to be integers they are actually elements in $GF(2^8)$—the faster they are on 8-bit processors. Accordingly, $3x^3 + x^2 + x + 2$ works well. While $x^7 + x^6 + x^2 + x$, $x^7 + x^6 + x^5 + x^4 + 1$, and $3x^3 + x^2 + x + 2$ do not have the elegance of $x^8 + 1$ and $x^4 + 1$, the explanations for selecting those polynomials—"simplest among those polynomials coprime to the modulus" and "linear in $GF(2)$, diffuse well, fast on 8-bit processors"—makes clear that there is nothing hiding there either.

Rijndael's designers achieved efficiency. The algorithm was significantly faster than all other finalists in key set-up, and was in the middle of the pack for encryption. Decryption is slower because the key set-up takes longer, but the slowdown is not excessive. And NSA's analysis of hardware implementations showed that Rijndael was reasonable in its layout size and had excellent "throughput" (the amount of work done in a given period) [**46**].

Did the designers achieve security? Only time will give us an answer to that question. It is worth remembering that, despite many fears and statements to the contrary, DES proved to be a robust and secure cryptosystem. Rijndael was vetted in an open and international forum, with input from at least one national-security agency.[10] The U.S. government believes in AES: in June 2003, the U.S. government approved the use of 128-bit AES for the protection of all documents classified SECRET, and 192- and 256-bit AES for documents classified at the TOP SECRET level [**32**]. But when DES was put forth, neither the industrial and academic community nor NSA had anticipated linear cryptanalysis. Might there be a "linear cryptanalysis" lurking in AES's future? AES has been built with a significant margin of safety. The 128-bit version of AES does not need to be 128 bits secure; 125, 120, or even 112 would suffice. Check in after another half-century, and we will know whether AES has stood the test of time.

ACKNOWLEDGMENTS. This paper is adopted from an invited MAA lecture at the Joint Mathematics Meetings held in San Diego in January 2002. Some of the material originally appeared in "Standing the Test of Time: The Data Encryption Standard," *Notices of the American Mathematical Society*, March 2000, pp. 341–349, by the same author.

I appreciate the help of Bart Preneel, John Cremona, Whitfield Diffie, Neil Immerman, Hilarie Orman, Michael Quisquater, and Ann Trenk who commented on an earlier version of this paper. I am particularly grateful to Vincent Rijmen for his trenchant comments. I am also very grateful for the careful reading and valuable editorial comments this paper received from Dan Velleman.

REFERENCES

1. E. Bach and J. Shallit, *Algorithmic Number Theory: Volume 1, Efficient Algorithms*, MIT Press, Boston, 1996.
2. T. Beth and C. Ding, On almost perfect nonlinear permutations, in *Advances in Cryptology: Eurocrypt '93*, Springer-Verlag, Berlin, 1993, pp. 65–76.
3. E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, New York, 1993.
4. I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, Cambridge, 1999.
5. C. Carlet, *Codes de Reed-Mueller, codes de Kerdock et de Preparata*, Ph.D. thesis, publication of LITP, Institut Blaise Pascal, Université Paris 6, 1990.
6. D. Coppersmith, The Data Encryption Standard (DES) and its strength against attacks, in *IBM Journal of Research and Development* **30** (1993) 243–250.
7. ———, personal communication.
8. N. Courtois, A. Klimov, J. Patarin, and A. Shamir, Efficient algorithms for solving overdefined systems of multivariate polynomial equations, in *Advances in Cryptology: Eurocrypt '00*, B. Preneel, ed., Springer-Verlag, Berlin, 2000, pp. 392–407.
9. A. Cullinson, A computer in Kabul yields a chilling array of Al Qaeda memos, *Wall Street Journal* (December 31, 2001) A1–A3.
10. J. Daemen, *Cipher and Hash Function Design Strategies Based on Linear and Differential Cryptanalysis*, Ph.D. thesis, Katholieke Universiteit, Leuven, Belgium, 1995.
11. J. Daemen, L. Knudsen, and V. Rijmen, The Block Cipher Square, in *Fast Software Encryption*, E. Biham ed., LNCS 1267, Springer-Verlag, Berlin, 1997.
12. J. Daemen and V. Rijmen, *AES Proposal: Rijndael*; available at http://csrc.nist.gov/encryption/aes/rijndael.
13. ———, Answer to "New Observations on Rijndael," in *NIST Second Round Comment*, NIST AES website (csrc.nist.gov/encryption/aes), 2000.
14. ———, The wide trail design strategy, in *Cryptography and Coding*, 8th IMA International Conference, B. Honary, ed., Springer-Verlag 2001, Berlin, pp. 222–238.
15. ———, *The Design of Rijndael: AES—the Advanced Encryption Standard*, Springer-Verlag, Berlin, 2002.

[10] It is probably safe to assume that there was actually input from two. Had GCHQ, the British equivalent of NSA, discovered a serious flaw in Rijndael, NIST would have been informed.

16. J. Faugere, P. Gianni, D. Lazard, and T. Mora, Efficient computations of zerodimensional Groebner bases by changes of ordering, *J. Symb. Comput.* **16** (1993) 329–344.
17. N. Ferguson, R. Schroeppel, and D. Whiting, A simple algebraic representation of Rijndael, in *Selected Areas of Cryptography 2001*, S. Vaudenay and A. Youssef, eds., Springer-Verlag, Berlin, 2001, pp. 103–111.
18. E. Grossman, personal communication.
19. T. Jakobsen and L. Knudsen, Attacks on block ciphers of low algebraic degree, *J. Cryptology* **14** (2001) 197–210.
20. D. Kahn, *The Codebreakers*, Scribner, New York, 1996.
21. N. Koblitz, *Algebraic Aspects of Cryptography*, Springer-Verlag, Berlin, 1998.
22. M. Kwan, *Reducing the Gate Count of Bitslice DES*, Cryptology ePrint Archive, Report 2000/051, 2000; available at http://eprint.iacr.org.
23. S. Landau, Standing the test of time: the Data Encryption Standard, *Notices Amer. Math. Soc.* **47** (2000) 341–349.
24. ———, Communications security for the twenty-first century: the Advanced Encryption Standard, *Notices Amer. Math. Soc.* **47** (2000) 450–459.
25. R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications*, Cambridge University Press, Cambridge, 1986.
26. F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*, North-Holland., Amsterdam, 1978.
27. M. Matsui, Linear cryptanalysis method for DES cipher, in *Advances in Cryptology: Eurocrypt '93*, T. Helleseth, ed., Springer-Verlag, Berlin, 1994, pp. 386–397.
28. W. Meier and O. Staffelbach, Nonlinearity criteria for cryptographic functions, in *Advances in Cryptology: Eurocrypt '89*, J.-J Quisquater and J. Vandewalle, eds., Springer-Verlag, Berlin, 1989.
29. S. Murphy and M. Robshaw, New observations on Rijndael, in *NIST Second Round Comment*, NIST AES website (csrc.nist.gov/encryption/aes), 2000.
30. ———, Further comments on the structure of Rijndael, in *NIST Second Round Comment*, NIST AES website (csrc.nist.gov/encryption/aes), 2000.
31. ———, Essential algebraic structure within the AES, *Advances in Cryptology: CRYPTO '02*, Moti Yung, ed., Springer-Verlag, Berlin, 2002, pp. 1–16.
32. National Security Agency, Committee on National Security Systems, *National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information*, CNSS Policy No. 15, Fact Sheet 1, June 2003.
33. K. Nyberg, On the construction of highly nonlinear permutations, in *Advances in Cryptology: Eurocrypt '92*, R. Rueppel, ed., Springer-Verlag, Berlin, 1993, pp. 92–98.
34. ———, Differentially uniform mappings for cryptography, in *Advances in Cryptology: Eurocrypt '93*, T. Helleseth, ed., Springer-Verlag, Berlin, 1994, pp. 53–64.
35. ———, S-boxes and round functions with controllable linearity and differential uniformity, in *Fast Software Encryption: Second International Workshop 1994*, B. Preneel, ed., Springer-Verlag, Berlin, 1995, pp. 111–130.
36. K. Nyberg and L. Knudsen, Provable security against a differential attack, *J. Cryptology* **8** (1995) 27–38.
37. J. Pieprzyk, Nonlinearity of exponent permutations, in *Advances in Cryptology: Eurocrypt '89*, J.-J Quisquater and J. Vandewalle, eds., Springer-Verlag, Berlin, 1990, pp. 89–92.
38. ———, On Bent Permutations, Technical Report CS91/11, Department of Computer Science, University of New South Wales; presented at International Conference on Finite Fields, Coding Theory, and Advances in Communications and Computing, Las Vegas, 1991.
39. V. Rijmen, personal communication.
40. V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. De Win, The cipher SHARK, in *Fast Software Encryption: Third International Workshop*, D. Gollman, ed., Springer-Verlag, Berlin, 1996, pp. 99–112.
41. United States v. Nicodemo S. Scarfo, et al., Criminal Action No. 00-404 (NHP), United States District Court, District of New Jersey.
42. R. Schroeppel, Second Round Comments to NIST, in *NIST Second Round Comment*, NIST AES website (csrc.nist.gov/encryption/aes), 2000.
43. A. Shamir, On the security of DES, in *Advances in Cryptology: CRYPTO '85*, Hugh Williams, ed., Springer-Verlag, Berlin, 1985, pp. 280–281.
44. M. Smid and D. Branstad, The Data Encryption Standard: past and future, in *Contemporary Cryptology*, G. Simmons, ed., IEEE Press, New York, 1991.
45. W. Tuchman, personal communication.
46. B. Weeks, M. Bean, T. Rozylowicz, and C. Ficke, *Hardware Performance Simulation of Round 2 Advanced Encryption Standard Algorithms* (May 15, 2000); available at http://csrc.nist.gov/encryption/aes/round2/r2anlsys.htm.

47. G. Xiao and J. Massey, A spectral characterization of correlation-immune combining functions, *IEEE Trans. Inform. Theory* **34** (1988) 569–571.

SUSAN LANDAU (B.A. Princeton, M.S. Cornell, Ph.D. M.I.T.) is Senior Staff Engineer at Sun Microsystems Laboratories. Her mathematical interests include symbolic computation and algebraic algorithms: she has discovered several polynomial-time algorithms for problems whose previous best solutions took exponential time. Prior to her employment at Sun, Landau had been a faculty member at the University of Massachusetts and Wesleyan University, and she held visiting positions at Yale, Cornell, and the Mathematical Sciences Research Institute at Berkeley. Landau also spent many summers teaching at the Hampshire College Summer Studies in Mathematics, a mathematics program for high-ability high school students. Landau and Whitfield Diffie have written *Privacy on the Line: The Politics of Wiretapping and Encryption* (MIT Press, 1998).
Sun Microsystems Inc., MS UBUR02-311, P.O. Box 4002, Burlington, MA 01803-0902
susan.landau@sun.com