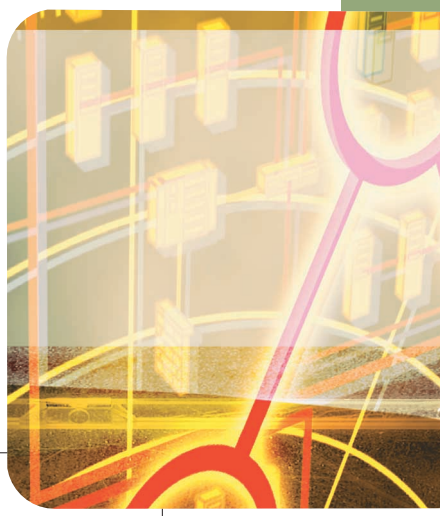
Selecting the Advanced Encryption Standard

The US National Institute of Standards and Technology selected the Advanced Encryption Standard, a new standard symmetric key encryption algorithm, from 15 qualifying algorithms. NIST has also made efforts to update and extend their standard cryptographic modes of operation.



William E. Burr
National Institute of Standards and Technology

The Advanced Encryption Standard (AES) is the new standard encryption algorithm that will replace the widely used Data Encryption Standard (DES). When the National Institute of Standards and Technology (NIST) set out to develop a new encryption standard in 1997, it set in motion a chain of activities that promises to build a foundation for stronger and better cryptographic standards for the 21st century, which is vital in this era of e-commerce and e-government.

Replacing the DES

In 1973, NIST's predecessor organization, the National Bureau of Standards (NBS), invited interested parties to submit candidate block encryption algorithms for a Federal Information Processing Standard (FIPS), a standard for use by federal agencies. IBM offered the only credible candidate, which, after some modification, was adopted in 1977 as the Data Encryption Standard, FIPS 46.[1] The DES's introduction marked the first public discussion and standardization of strong encryption, which previously had been primarily the province of governments and military security agencies.

In its early days, the DES had its controversies: During the evaluation process, the original IBM submission's key size was reduced from 128 bits to 56 bits. IBM made some modifications to the DES substitution tables (called S-boxes) that are used repeatedly in the encryption process to make complex transformations of data. Martin Hellman and others suggested that the 56-bit key size was too small, and that because no rationale was ever given for the design of the S-boxes, it was difficult to evaluate the security they provided.[2] Hellman claimed there was even a possibility that a back door might have been introduced that let the US National Security Agency (NSA) illicitly decrypt messages. However, in 1991, when Eli Biham and Adi Shamir discovered differential cryptanalysis, which was previously unknown in the open literature, it turned out that those changes to the S-boxes made the final version considerably more resistant to attack than the original version. The general belief (never explicitly confirmed) is that the NSA told IBM enough to help them make DES resistant to such attacks.[3,4]

The DES has proven itself over the years. It has become the standard to which all block ciphers are compared. However, by the mid 1990s, it was clear that the DES's 56-bit key was no longer big enough to prevent attacks mounted on contemporary computers, which were thousands of times more powerful than those available when the DES was standardized. During the original DES selection, there was discussion of using DES more than once with different keys to extend its effective key size, and a standard for Triple DES (TDES) that encrypts each block three times with DES, using two or three different keys, was introduced with FIPS 46-3 in 1999; TDES should be secure for many years to come.[5] In 1977, NBS believed that encryption was best done in hardware and DES was designed for hardware implementation. DES is poorly suited to implementation in software on general purpose computers that operate most efficiently on words of 16 to 64 bits because it operates internally on 4 and 6 bits at once and requires "shuffling" of individual bits, which is slow on most common computers, but fast in hardware. Moreover, the TDES needs three times the computation of DES. Un-

Exclusive-OR

The exclusive-OR (XOR) operation is a simple binary operation widely used in digital logic and cryptography. Let "⊕" represent XOR, then: 0 ⊕ 0 = 0, 1 ⊕ 0 = 1, 0 ⊕ 1 = 1, and 1 ⊕ 1 = 0. An example of a multibit XOR operation is: 1101 ⊕ 0110 = 1011, in contrast to addition, where 1101 + 0110 = 0011. You can think of XOR as bit-by-bit addition without the carry. XOR operations are faster than addition because addition requires time for carries to ripple down the results. XOR has useful properties for cryptography: $a \oplus a = 0$ and $a \oplus b \oplus a = b$. If R is a cryptographically generated random string, P is a plaintext string, and C a ciphertext string, then we can generate ciphertext as $C = R \oplus P$, and we can recover the plaintext $P = R \oplus C$.

fortunately, software encryption is very important today. It was therefore time to consider a new encryption standard that was both more secure than the TDES and more efficient in software.

Consequently, NIST decided it needed a new federal standard, which it dubbed the Advanced Encryption Standard. Because the US Federal Government needs secure communications with international parties and preferred to use ordinary commercial products for its information technology needs, the AES had to be suitable for general international commercial use. Additionally, it was important that NIST select the AES by a process that was perceived to be fair and open, preventing any public suspicion that the standard contained a hidden back door.

Selection criteria

NIST conducted an open competition to select the AES. The basic requirements were that it be a block cipher, such as DES or TDES, and be at least as secure as TDES but more efficient in software. In April 1997, NIST held an open public workshop about the selection criteria. That September, NIST issued a call for candidate block cipher algorithms that

- would be unclassified and publicly disclosed;
- offered key sizes of 128, 192, and 256 bits;
- had a 128-bit block size; and
- would be available royalty free anywhere in the world.

Perhaps the most surprising requirement was the 128-bit block size. Before the AES competition, nearly all block cipher designs followed the DES's lead with a 64-bit block size. Block cipher encryption algorithms are used for several purposes, including message authentication. Authentication applications that use an n-size block are generally subject to *collisions* (when two different messages have the same hash value), which become increasingly likely after $2^{n/2}$ authenticator fields (usually called *tags*) are available. Moreover, in many modes of using block ciphers, a $2^{n/2}$ term determines the security bounds, and the mode becomes less secure or insecure after $2^{n/2}$ ciphertexts are available to an attacker. Thus, enlarging the block size is at least as important to the AES's increased security as enlarging the key size is. The AES is significantly stronger than the TDES in many applications, not because of its larger key sizes—in which the differences in strength might be of little practical importance—but because the AES block size is twice as large.

Qualifying candidates

At the first AES Candidate Conference in August 1998, NIST solicited public comments on 15 qualifying AES candidate algorithms. At the second conference, held in Rome in March 1999, NIST analyzed the candidates; a few months later, it announced five finalist candidate algorithms.

MARS. IBM's algorithm, MARS, uses innovative techniques including multiplication, data-dependent rotations, and conventional S-boxes. NIST felt MARS was worthy because it was fast on machines with 32-bit multiplier hardware, and it uses several separate techniques that a cryptologist must defeat to break the cipher. MARS's primary disadvantage was that it required a fast multiplier and ran slowly on lightweight platforms such as 8-bit embedded microprocessors.

RC6. Submitted by RSA Data Systems, RC6 is a simple, elegant algorithm that relies on multiplication and data-dependent rotations but does not have S-boxes. RC6 is fast on platforms that have a 32-bit multiplier (it only requires a squaring circuit), but not as efficient on simpler platforms without hardware support for multiplication (or squaring). Opinions differed about whether RC6's simplicity and comparative ease of analysis were a source of comfort or worry.

Rijndael. Joan Daemen and Vincent Rijmen's algorithm Rijndael uses 8-bit S-boxes as its source of nonlinearity and simple XOR (exclusive-or) operations (it does not need addition). Because of its simple operations and byte orientation, Rijndael is fast in hardware and on 8-, 32-, and 64-bit microprocessors. Rijndael was the only finalist that adjusts the number of rounds needed for each key size, improving performance for the 128-bit size. Like RC6, opinions differed on whether Rijndael's simplicity was a comfort or a worry.

Serpent. Produced by Ross Anderson, Eli Biham, and Lars Knudsen, Serpent (like Rijndael) uses only simple XOR type operations and eight 4-bit S-boxes, with an architecture that allows 32 simultaneous applications of each S-box. Although Serpent is fast in hardware, the de-

signers adopted a conservative security margin with several rounds (in comparison to its competitors) that hurt its software performance. Like Rijndael and RC6, Serpent's simple structure sparked either comfort or worry.

Twofish. A team of American companies and academics produced Twofish, which generates its S-boxes from the key and uses several innovative techniques, including key set-up options that can be optimized for different performance environments. Some criticized Twofish for being too complex to analyze in the short AES timeframe, whereas others felt that its complexity provided an extra margin of safety. Twofish was overall a good performer on most platforms, but not a leader.

MARS, Twofish, and RC6 were all considered "American" submissions; Rijndael's designers were Belgian and Serpent's designers were from the UK, Israel, and Denmark. There was much verbal speculation that NIST would never select a "foreign designed" algorithm—there was even speculation, particularly in Europe, that the competition had been "fixed" to select MARS, which was, like the DES, an IBM design.

Selecting the AES

Following the public review period, NIST spent much of the summer of 2000 reviewing the data, comparing the five finalists, and then selecting the winner. Criteria included

- General security (the most important consideration)
- Performance
- Algorithm characteristics, including intellectual property

The final selection process and the reasons for the eventual winner's selection are comprehensively documented in James Nechvatal and his colleagues' "Report on the Development of the Advanced Encryption Standard (AES)."[6]

Security

A strong symmetric key encryption algorithm such as the AES has one basic security goal—that the best attack against it should be *key exhaustion* (trying every possible key until you find one that works). If key exhaustion is the best attack, then key size determines the symmetric key algorithm's strength. To find an n-bit key, it is, on average, necessary to try 2^{n-1} keys, but if you make n sufficiently large, this becomes wildly impractical. We know breaking 56-bit DES by key exhaustion is practical, based on demonstrations in 1997 and 1998.[7] Certainly, large organizations with substantial resources can break algorithms by key exhaustion for keys larger than 56-bits today, although most cryptographers believe 80 bits will be secure for a few more years.

This is why the AES uses 128-, 192-, or 256-bit keys. Moore's law, a heuristic that has held roughly true for several decades, observes that progress in semiconductor devices doubles the computational power available for a constant cost approximately every 18 months. If Moore's law continues to hold, and key exhaustion by conventional means is the best attack, then the 128-bit AES should still be as secure as an 80-bit algorithm is today until 2066, and the 256-bit AES should be as secure as an 80-bit algorithm is now for several centuries. However, different technologies such as quantum computing could change the rules of the game, and who can predict how long Moore's law will hold? Nevertheless, Moore's law is the best estimator we have.

How do we ensure that the best attack on an algorithm is key exhaustion? There is no way to prove it.

How do we ensure that the best attack on an algorithm is key exhaustion? There is no way to prove it. Rather, cryptologists study the algorithm and attempt to find *shortcut attacks*—methods that find the key with less than 2^{n-1} operations. Any attack that requires less than 2^{n-1} operations is considered a "break" of the algorithm, even though it might be totally impractical to carry out the attack. Therefore, breaking the 256-bit version of an algorithm is in some sense easier than breaking the 128-bit version because far more operations are permitted to do so. For example, suppose a particular attack found the key of the 256-bit version of an algorithm with 2^{200} operations but required 2^{140} operations to find the key for the 128-bit version. In that case, the 256-bit version would be considered broken whereas the 128-bit version would be considered unbroken, even though the attack on the 256-bit version would be wildly impractical and the 256-bit version would remain far stronger than the 128-bit version.

Many well-known methods for attacking algorithms require large amounts of either known or chosen plaintext or ciphertext. That is, the analyst is assumed to have either a large amount of plaintext–ciphertext pairs available or the ability to encrypt or decrypt chosen values and obtain the results. You can probably convince yourself that any attack that requires 2^{100} chosen or known plaintexts is not likely ever to be a practical concern in a real application.

Nearly all symmetric key algorithms consist of several repetitions of a function called the *round function*. Analysts typically work on simplified versions of an algorithm, usually with fewer invocations of the round func-

tion, and attempt to find a technique that results in a shortcut attack for the simplified versions. They then try to extend the attack to the full algorithm with the full number of rounds. Algorithm designers typically estimate the number of rounds needed to defeat known attacks and then add more rounds as a safety factor. NIST might have adjusted the number of rounds but chose to evaluate each algorithm with the number submitted by the algorithms' designers.

If no shortcut attacks are found after intensive study, then the algorithm is considered secure. The longer and more intensively researchers study algorithms without finding a shortcut attack, the more confidence there is in the algorithm. Cryptographic researchers did not report any shortcut attacks against the finalist algorithms during the AES selection process. However, participants in the AES process and other researchers developed several shortcut attacks against reduced round variants of various candidates. NIST considered a suggestion that their selection panel might attempt to compare the ratio of rounds in the full candidate algorithms to the number of rounds in the simplified breakable versions to measure the algorithms' safety margin. In the end, this did not seem to be meaningful, if for no other reason than this measure did not account for the differences in the resources (such as plaintext required for the attacks), and because it wasn't clear that such a margin would apply to novel attacks, which were apparently necessary to break the complete algorithms.

During the selection process, NIST could not distinguish between the security that the finalist algorithms offered. All were apparently well designed to resist known attack methods. For example, the strongest attack found against 128-bit Rijndael during the selection process worked against seven (of 10) rounds of the Rijndael and

During the selection process, NIST could not distinguish between the security that the finalist algorithms offered.

required more than 2^{127} pairs, 2^{101} bytes of memory, and 2^{120} operations.

The NIST selection panel was left with arguments about the simplicity or complexity of candidate algorithms. The comparative simplicity of some of them made analysis more straightforward, which could be construed as an advantage (because we can analyze this, we know how strong it is) or as a disadvantage (because this is simple, a new attack is more likely to be able to break it). Other algorithms were more complex, which also could be construed as an advantage (even if you break *a*, you still have to get through *b* and *c* to break the entire algorithm) or a disadvantage (this is too complex to analyze, but there might be something wrong that an attacker will eventually find). These arguments, while interesting, were also inconclusive.

After analysis, the candidates appeared to be secure against all known attacks, and NIST did not feel that it could make a decision based on security considerations with the available analytical tools.

Performance

Performance has many dimensions, although initial discussion of the candidates focused on their performance on 32-bit Pentium processors. All finalist algorithms offered better performance than the TDES when implemented in software, but they had significant differences in performance. NIST considered a wealth of data regarding the candidates' performance on many platforms, including Pentiums, Itaniums, RISC processors, 8-bit embedded microprocessors, digital signal processors (DSPs), field programmable gate arrays (FPGAs), and application-specific integrated circuits (ASICs). All the candidates performed at least moderately well on Pentiums and similar platforms, but the weakest software candidate, Serpent, was arguably the fastest in hardware.

Intellectual property

One goal was for the AES to be available royalty free worldwide. This might seem a strange goal for a US Department of Commerce agency such as NIST, but decision was widely applauded in the cryptographic community. This objective also seemed reasonable because the DES's patents had expired and NIST had many unpatented algorithms to choose from.

Intellectual property did not factor into the final selection, though, because NIST required the candidates' designers to put their intellectual property in the public domain if it chose their algorithm. Additionally, NIST did not find that the finalists infringed on other patents. (This does not mean that such patents do not exist, however, or that patent holders might not dispute this statement. Moreover, there is always the possibility that an AES selection might have infringed on a patent that was issued subsequent to the selection.)

Intellectual property considerations factored into a key decision, though: should NIST select only one AES algorithm or more than one? There was a strong argument for more, so that all our eggs would not be in one basket should a new analytical technique break the winner. However, a straw poll at the third AES conference ran strongly against multiple algorithms. Three arguments against multiple algorithms were that

Hash functions

Cryptographic hash functions are the utility tools of cryptography. A hash function does not have a key. It generates a short, fixed-size message digest from a long, variable-size message. Every bit of the digest is a complex function of every bit of the message, and it is impractical to find two different messages with the same message digest value. Hashes are used in digital signatures (we actually sign the message digest, rather than the message, with a private key operation) as integrity checks, for message authentication codes, and in random number generation, key derivation, and a host of other applications. Prior to the AES selection, the only FIPS approved hash algorithm had been SHA-1, with a 160-bit message digest.

- multiple AES algorithms would confound interoperability;
- hardware vendors feared that they would be forced to implement all the AES winners on their chips, costing precious real estate; and
- the more AES algorithms, the greater the chance that at least one of them would have intellectual property problems, forcing vendors to license a patent.

Based on this, NIST decided to pick one algorithm.

The winner

On 2 October 2000, less than four years after the selection process began in 1997, NIST announced that the Rijndael algorithm was the new AES. NIST primarily selected it because of its strong performance on nearly all platforms and its ease of implementation in hardware. It had been the most popular choice in straw polls at the final AES conference and was favorably received overall by the international cryptographic community. (In fact, the selection of a non-US design surprised many in the community and probably increased the international acceptance of the AES.) On 6 December 2001, after a comment period and formal approval by the US Secretary of Commerce, NIST officially announced its adoption of the AES.[8]

Extending other algorithms

Although the selection of Rijndael and the publication of the AES were major milestones, they marked the beginning of another sort. Now that truly strong symmetric key encryption is available, we must make correspondingly strong algorithms of other types available because encryption algorithms are rarely used alone. For example, consider sending an email message encrypted under the AES. You could do this by encrypting the actual AES encryption key with the recipient's public key, which is found in a public key certificate protected by a digital signature. Four different algorithms are used for this encryption:

- A public key transport algorithm transports the actual encryption key to the receiver
- A public key signature algorithm authenticates the message sender and validates the certificate that provides the public key for transporting the encryption key
- A hash algorithm is also used in the signatures
- The AES encryption algorithm encrypts the message.

Like a chain, if any of those cryptographic links are weak, then the entire process is vulnerable.

After selecting the AES, NIST standardized an augmented suite of secure hash functions in FIPS 180-2,[9] the Secure Hash Standard, in August 2002. FIPS 180-2 specifies the old standby 160-bit SHA-1 hash function and three new hash functions: SHA-256, SHA-384, and SHA-512, with 256-, 384- and 512-bit message digests (see the "Hash functions" sidebar).

NIST is now revising FIPS 186-2[10], the Digital Signature Standard, which specifies the digital signature algorithm (DSA) and adopts the RSA (named for its creators Rivest, Shamir, and Adleman)[11] signature algorithm specified in ANSI X9.31[12] and the elliptic curve digital signature algorithm (ECDSA) specified in ANSI X9.62.[13] These algorithms each create digital signatures using different mathematical methods. Each provides several allowable key sizes that supply varying levels of strength. However, all were written to work specifically with SHA-1; DSA was defined only for key sizes up to 1,024 bits. This revision, FIPS 180-3, will extend key sizes for DSA and specify appropriate hashes for larger keys.

NIST has also begun preparing a key management standard that will encompass public key techniques for key agreement or transport, and for other topics such as specific special methods for "wrapping" (encrypting) symmetric keys with other symmetric keys.

NIST recognized the need for a whole suite of algorithms whose strength was comparable to 128-, 196-, and 256-bit AES. For example, SHA-1 is a 160-bit hash, which for many applications is about as strong as an 80-bit symmetric key algorithm. NIST therefore has begun to extend and upgrade its other standards so that across-the-board choices exist comparable in strength to the AES. Table 1 summarizes the estimated strength of the current and planned NIST algorithms.

With the exception of TDES, the assumption for the

Block ciphers

The AES is a symmetric key block encryption algorithm. Symmetric key algorithms are algorithms that use the same key to perform an operation and its inverse. For example, they use the same key to encrypt and decrypt a message, or they use the same key to create and verify a Message Authentication Code (MAC).

Block ciphers operate on fixed blocks of data at a time. They encrypt data and are building blocks for other functions such as message authentication, pseudorandom number generation, or hashing. Figure 1 in the main text (next page) illustrates the basic electronic codebook (ECB) mode of block cipher operations, which occurs under the *symmetric key*'s control, so called because the same key is used to both encrypt and decrypt data. An *m*-bit block of *plaintext* data and an *n*-bit key are input to the encryption algorithm, which generates an *m*-bit block of *ciphertext* output. The same key and the ciphertext block are input to the decryption algorithm to output the plaintext again.

The secrets are the plaintext and the key. Although military algorithms are traditionally kept secret, this is infeasible for open commercial use, and everyone in the field knows the AES algorithm. Security depends only on the key's secrecy. The goal of a strong symmetric key encryption algorithm is that there is no way to decrypt the data except by knowledge of the key and no better way to find that key than key exhaustion. Put plainly, given infinite resources, key exhaustion always works. But, on average, key exhaustion takes 2^{n-1} operations, and if n is large enough, all the combined computer power in the world cannot accomplish the task in a human lifetime or even for centuries.

The AES block size is 128 bits (before the AES, 64 bits was common) and the key sizes can be either 128, 192, or 256 bits. Every key defines a different codebook, mapping each plaintext value to a unique ciphertext value. We assume that an adversary can obtain some plaintext–ciphertext pairs (he or she might know the contents of some encrypted messages). The block size must, therefore, be large enough that an adversary cannot accumulate enough pairs to learn any appreciable fraction of the total codebook. In practice, we prevent this by using large blocks and by changing keys often enough that only a small fraction of the codebook is ever used. Obviously this constraint on key life imposed by a 128-bit block cipher such as the AES is much less of a limitation than for a 64-bit block cipher.

Table 1. Key sizes for equivalent strengths of common algorithm types.

SYMMETRIC KEY (BITS)	SYMMETRIC KEY ALGORITHM	HASH ALGORITHM	PUBLIC KEY ALGORITHM KEY SIZE (BITS)		
			DISCRETE LOG (PUBLIC/PRIVATE)[1]	RSA	ELLIPTIC CURVES[2]
80	SKIPJACK	SHA-1	1024/160	1024	160
112	TDES		2048/244	2048	224
128	AES-128	SHA-256	3072/256	3072	256
192	AES-192	SHA-384	7680/384	7680	384
256	AES-256	SHA-512	15360/512	15360	512

1. Includes digital signature algorithm (DSA) and Diffie-Hellman key agreement algorithm.
2. Includes elliptic curve digital signature algorithm (EC-DSA) and elliptic curve Diffie-Hellman (ECDSA) key agreement algorithm.

symmetric key algorithms in Table 1 is that the best attack is key exhaustion. As mentioned earlier, a key exhaustion attack on an n-bit key will, on average, require 2^{n-1} operations. Because each TDES encryption requires three DES encryptions, each with a 56-bit key, the total key is 168 bits long; however, a "meet in the middle" attack finds a TDES key in 2^{112} operations.[14] A similar attack makes key exhaustion on double DES nearly as efficient as on DES. A computer needs a large memory to carry out the attack, but Table 1 is based only on computational costs.

A hash algorithm's strength depends on how it is used. For example, a digital signature scheme based on a hash algorithm is vulnerable to forgeries if there are collisions present. For an n-bit hash, you can expect to find a collision after $2^{n/2}$ messages are hashed. Therefore, an n-bit hash for a digital signature scheme has equivalent strength to an $n/2$-bit symmetric key algorithm, as Table 1 shows. Avoiding collisions might not be important for other hash functions; for example, in message authentication or pseudorandom number generation, the hash algorithms in Table 1 might correspond to larger symmetric key sizes.

For public key algorithms, estimating the computation needed to discover keys is based on the best known current methods for factoring and discrete log computation, and on an estimate of future progress in factoring or

discrete logarithm techniques. The most effective known attacks on these algorithms also have large memory requirements, but the estimates in Table 1 do not account for memory requirements.

Modes of operation

Block ciphers are used to protect data in certain basic patterns called *modes of operation*. After NIST introduced the DES in 1977, it produced *FIPS 81, DES Modes of Operation*, which defines four encryption modes for the DES algorithm: the electronic codebook (ECB) mode, the cipher block chaining (CBC) mode, the cipher feedback (CFB) mode, and the output feedback (OFB) mode.[15] These four modes, informally known as the "NIST modes," have been widely adopted by users of cryptography around the world. Although these modes are defined in FIPS 81 specifically for the 64-bit block and 56-bit key of DES, they can be extended to other block and key sizes.

Of the modes, two include two full block cipher modes that directly encrypt entire blocks of data at a time:

- *ECB mode,* as Figures 1 illustrates, is the mode in which a block of plaintext is simply encrypted directly to form the ciphertext, and is the basic operation from which all other modes are built. (See the "Block Ciphers" sidebar for an ECB mode explanation.) However, when ECB mode encrypts data, the same plaintext input results in the same ciphertext output, so patterns in the input are revealed to an eavesdropper. Therefore, ECB mode is insecure for many applications.
- *CBC mode*, illustrated in Figure 2, is the mode in which the ciphertext of block *i* is XORed with the plaintext of block *i*+1 before it is encrypted. The first block is XORed with an initialization vector (IV) that need not be kept secret. CBC mode hides data patterns in the ciphertext and is more secure than ECB mode. One problem with CBC mode, though, is that it is strictly block serial, so block *i* must be encrypted before block *i*+1, limiting its ability to perform multiple operations in parallel. A single bit error in a communications channel usually results in two full blocks of corrupt plaintext.

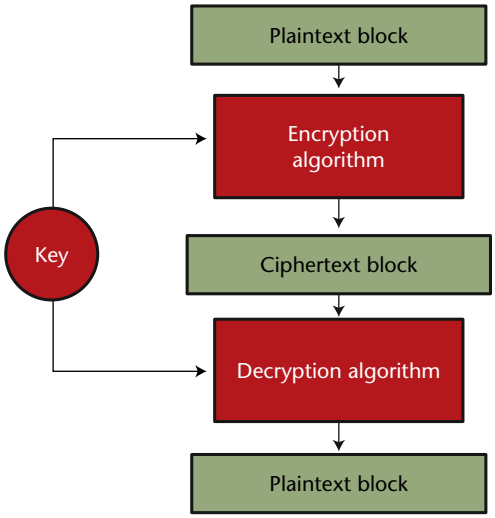


Figure 1. Block cipher encryption in Electronic Code Book (ECB) mode.

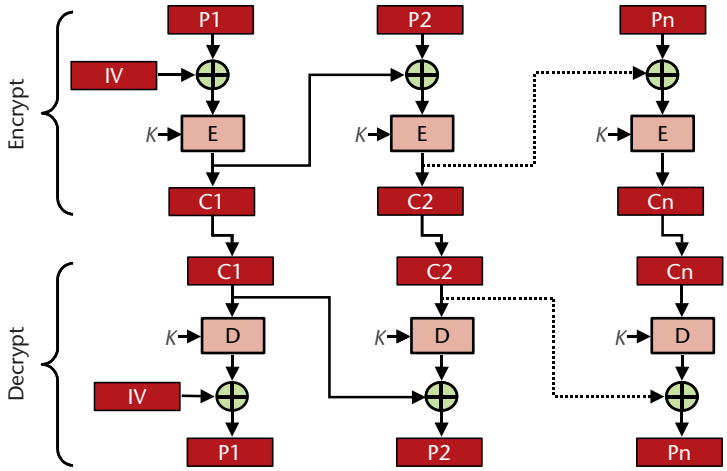


Figure 2. Block cipher encryption in Cipher Block Chaining (CBC) mode.

FIPS 81 also includes two *stream cipher* modes, in which the block cipher generates a unique pseudorandom *keystream* that is determined by a key, an IV, and, in some cases, the plaintext itself. The sender then XORs the keystream with the plaintext to encrypt it, and the receiver then XORs the keystream with the ciphertext to decrypt it. The IV need not be kept secret but must be a *nonce*, a value that is never repeated during a cryptographic session period or until the key is changed. Only the block cipher encryption operation is needed; block cipher decryption is not used.

Using stream ciphers has its pitfalls; in particular, the keystream must not be repeated or plaintext could be revealed, and it is sometimes possible for an attacker to invert predictable bits in the plaintext. In many cases, if bits are added to or lost from the ciphertext during transmission, cryptographic synchronization is lost. Using stream ciphers with noncryptographic checksums in protocols is perilous and could let an active attacker decrypt plaintext without learning the key. For example, an application that uses a stream cipher insecurely is the IEEE 802.11 WEP protocol, which protects wireless Ethernet. WEP falls victim to such a checksum attack and also more or less guarantees that cipher streams repeat after a short period of operation.

You might ask, if stream ciphers come with all these

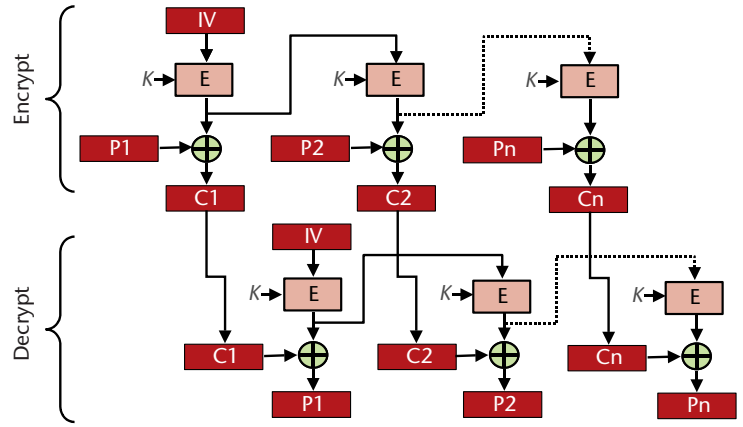


Figure 3. Stream cipher encryption in Output Feedback (OFB) mode.

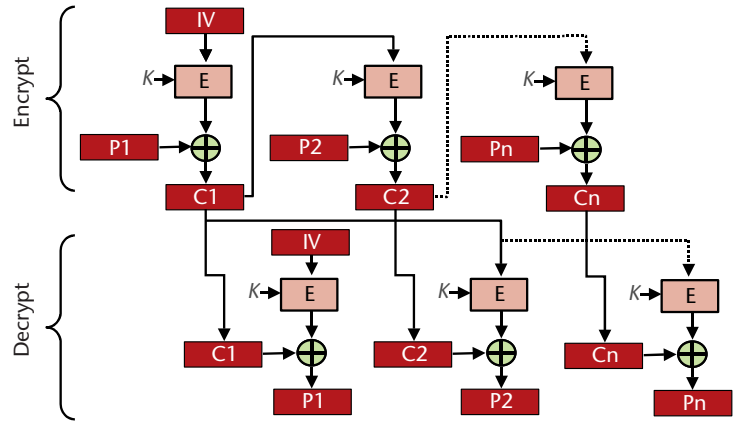


Figure 4. Stream cipher encryption in Cipher Feedback (CFB) mode.

pitfalls and perils, why do we use them? We use stream ciphers to protect data communications channels because the keystream can be generated in parallel with the rest of the channel processing. Therefore, stream ciphers only need to introduce one bit-time of additional latency to encrypt, plus one bit-time to decrypt. CBC or EBC mode, on the other hand, must accumulate a full block of data before encrypting or decrypting it, and also adds the time required for the encryption and decryption operations to the channel latency. This delay is intolerable in many communications applications.

The two stream cipher modes of FIPS 81 are

- *OFB mode.* As Figure 3 illustrates, in this mode, the encryption operations output is fed back into the input and used to generate a keystream, which is then XORed with the data to generate the ciphertext. Unlike many modes, ciphertext bit transmission errors are not expanded in the received plaintext, but if bits are lost or inserted in the ciphertext, cryptographic synchronization is lost.
- *CFB mode.* As Figure 4 illustrates, this is another stream cipher mode in which the block encryption output is XORed as a keystream with the plaintext, but the feedback term to the encryption algorithm is the ciphertext itself. If only one ciphertext bit at a time is fed back through a shift register into the block encryption operation, this mode can be made self-synchronizing. CFB transmission errors in the ciphertext are expanded in the plaintext output. In some cases, this error expansion can be a security advantage because it prevents an adversary from selectively inverting only one bit of plaintext.

NIST knew that it would have to revise the AES's modes of operation to account for the block and key size differences from DES. Moreover, a major complaint about the existing modes (except ECB) was that they all involved feedback or chaining, making it impossible to improve their performance with parallel encryption operations. On the other hand, ECB is almost always insecure if it is used to encrypt a lot of data directly. NIST then planned to produce a new, updated modes of operation standard that was general enough to work with any block cipher and included a counter mode.

Figure 5 shows a new mode, the *counter mode* (CTR), that many AES selection process participants requested who had high performance applications. In CTR mode, like CFB and OFB modes, a counter is encrypted to generate a keystream, which is then XORed with the plaintext to generate ciphertext. The counter values used with a given key must be nonces because the keystream, like any keystream, must never repeat. A property of CTR mode, shown in Figure 5, and different than CBC, CFB and OFB modes, is that there is no feedback or chaining; therefore you can perform several encryptions in parallel—a significant advantage in high-performance applications. Like OFB mode, ciphertext transmission errors are not expended, but an adversary can invert selected bits in the plaintext.

To accommodate the AES and to add a parallelizable mode, NIST Special Publication 800-38A[16] defines modes of operation for data encryption and decryption using any block cipher algorithm. It includes four confidentiality modes of FIPS 81: ECB, CBC, CFB, and OFB, along with the CTR mode. NIST issued a special publication rather than a FIPS in the near term because the organization continues to add new modes, refine existing modes, and add additional advice and guidance on their use. The refinements and additional advice prevent insecure mode use. For example, with CBC mode,

plaintext must be padded out to a full block before it is encrypted. NIST SP 800-38A gives an example of such a padding scheme, but does not require that specific one. Recent work has shown that some commonly used CBC padding schemes, when used in certain protocols, can be exploited to allow decryption of chosen blocks of ciphertext, independent of the encryption algorithm, without knowledge of the key.[17] Recent work in encryption modes and attacks that exploit the interactions between the encryption modes and the communications protocols they are imbedded in necessitate continued refinement of the specifications for even the existing modes.

New modes

Researchers have proposed more than a dozen new modes to NIST (see http://csrc.nist.gov/encryption/modes/proposedmodes). Among the more interesting of these are three that can be parallelized and that combine encryption, authentication, and integrity protection for little more than the cost of encryption. However, selecting a standard mode is made difficult by intellectual property issues, because these modes' inventors have applied for patents (some of which are perhaps conflicting) although none have been issued. NIST, like most standards organizations, allows the use of patented methods in standards, provided reasonable, nondiscriminatory licensing is available. However, cryptographic standards that use patented technology are generally an inflammatory issue in the wider cryptographic and protocol standards communities. In particular, patent licensing problems are often seen to frustrate open-source implementations, such as Open SSL, which have been a great catalyst in the evolution of network protocol and security standards.

The modes of operation work might be less glamorous but it's more fundamental than the selection of the AES itself. The AES block cipher could be broken, or other block ciphers that are much faster might be invented. In time, new block ciphers could be created with better properties than the AES. But the modes of operation, whose security is predicated on the security of the block cipher they use, can be used with any block cipher, and therefore are arguably more important than block cipher algorithm standards.

The past few years have been a period of great progress in the analysis of cryptographic modes and protocols, and new protocols come with proofs of some of their security properties (assuming the underlying block cipher's security). Cryptologists have discovered many attacks against NIST's original four modes as they have been applied in various protocols. Often these attacks involve interactions between the mode and some other protocol element, for example, a noncryptographic checksum or a padding scheme. Some attacks are easy to carry out in particular cases, some are difficult but still practical, and others are probably impractical in most cases. The emerging trend is to develop cryptographic mode specifications that are more resistant to such attacks. NIST expects the modes of operations effort to continue for the foreseeable future, encompass new modes, identify problems with existing modes, and give better advice on applying them safely.

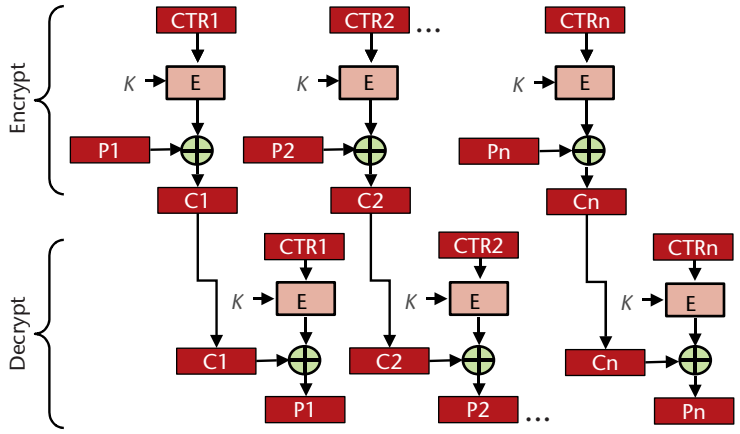


Figure 5. Stream cipher encryption in Counter (CTR) mode.

As a practical matter, the widespread adoption of a new encryption standard takes a long time. First the code must be incorporated into cryptographic toolkits. NIST believes this has generally happened with the AES; indeed, Rijndael's code appeared in several toolkits before NIST made its final selection. In many cases, it is necessary to extend or expand protocol standards to use the new algorithm; for example, an IETF RFC defines the cipher suites for use in the Transport Layer Security (TLS) Standard,[18] and no fewer than three Internet drafts deal with the intricacies of using the AES with the S/MIME secure E-mail Standard.[19–21] Next, implementations of the algorithm must work their way into actual products, and, finally, in many cases, a sufficient quantity of the AES-enabled product must ship before the algorithm sees any real use. For example, if you have an AES-enabled S/MIME client, it does little good until many or most S/MIME users also have AES-enabled clients.

An "AES Core" is a design for implementing the AES algorithm in hardware on an application specific integrated circuit or a field programmable gate array. ASICs and FPGAs are widely used to build electronic products. Specialized logic design companies (and some universities) develop such core designs and then sell them (there also some freeware cores) to companies developing electronic products, saving the developers from designing

their own complex logic designs for widely used functions, such as the AES encryption. A March 2003 Google search on "AES core" revealed 18 available AES core designs from the US (eight), the UK (three), Canada (one), Singapore (one), India (one), Australia (one), Austria (one), India (one), and Yugoslavia (one).

The NIST Cryptographic Module Validation Program (CMVP) validates cryptographic modules and tests algorithm implementations. The AES validation tests have only been available since March 2002, but by March 2003, 59 AES implementations were already validated.

The current draft of the IEEE 802.11i addresses the numerous security problems with the very successful 802.11 wireless Ethernet standards and now specifies two different AES-based encryption modes, both of which have been submitted to NIST as new modes of operation. One of these two modes, the Counter with CBC MAC (CCM) mode, has been selected as the "mandatory to implement" mode for IEEE 802.11i.[16] □

References

1. Federal Information Processing Standard 46, *Data Encryption Standard*, Nat'l Bureau of Standards, 1977.
2. M. Hellman,"DES Will Be Totally Insecure Within Ten Years," *IEEE Spectrum*, vol. 16, no. 17, July 1979, pp. 32–39.
3. E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," *Advances in Cryptology– Proc. CRYPTO '90*, Springer-Verlag, 1991, pp. 2–21.
4. E. Biham and A. Shamir, "Differential Cryptanalysis of the Full 16-Round DES," *Advances in Cryptology—Proc. CRYPTO '92*, Springer-Verlag, 1992, pp. 487–496.
5. *Federal Information Processing Standard 46-3, Data Encryption Standard,* US Nat'l Inst. Standards and Technology, 1999, http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.
6. J. Nechvatal et al., "Report on the Development of the Advanced Encryption Standard (AES)," *J. Research US Nat'l Inst. Standards and Technology*, vol. 106, no. 3, 2001, pp. 511–576, http://csrc.nist.gov/encryption/aes/round2/r2report.pdf.
7. "Cracking DES, Secrets of Encryption Research," *Wiretap Politics and Chip Design*, Electronic Frontier Foundation, 1998.
8. *Federal Information Processing Standard 197, The Advanced Encryption Standard*, Nat'l Inst. Standards and Technology, 2001; http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.
9. Federal Information Processing Standard 180-2, *Secure Hash Standard*, Nat'l Inst. Standards and Technology, 2001; http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf.
10. Federal Information Processing Standard 186-2, *Digital Signature Standard (DSS)*, Nat'l Inst. Standards and Technology, 2000; http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf.
11. R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Comm. ACM*, vol. 21, no. 2, Feb. 1978, pp. 120-126.
12. *ANSI X9.31-1998, Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)*, American Nat. Standards Inst., 1998.
13. *ANSI X9.62-1998, Public Key Cryptography for the Financial Services Industry: The EllipticCurve Digital Signature Algorithm (ECDSA)*, American Nat. Standards Inst., 1998.
14. R.C. Merkle and M. Hellman, "On the Security of Multiple Encryption," *Comm. ACM*, vol. 24, no.7, 1981, pp. 465–467.
15. *Federal Information Processing Standard 81, DES Modes of Operation*, Nat'l Bureau of Standards, 1977; www.itl.nist.gov/fipspubs/fip81.htm.
16. M. Dworkin, "Recommendation for Block Cipher Modes of Operation," NIST Special Publication 800-38A, Nat. Inst. Standards and Technology, 2001; http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.
17. S. Vaudenay, "Security Flaws Introduced by CBC Padding Applications to SSL, IPSEC, WTLS…," *DSC Tech. Report 200150*, http://lasecwww.epfl.ch.
18. P. Chow, "AES Cipher Suites for TLS," Internet Eng. Task Force, RFC3268: June 2002; www.ietf.org/html.charters/tls-charter.html.
19. J. Schaad, Use of the AES Algorithm in CMS, Internet Eng. Task Force Internet Draft, Nov. 2002; work in progress.
20. J. Schaad, "Wrapping an HMAC Key with a Triple-DES Key or an AES Key," Internet Eng. Task Force Internet Draft, Jan. 2002; work in progress.
21. J. Schaad and R. Housley, "AES Key Wrap Algorithm," Internet Eng. Task Force, RFC 3349, Sept. 2002; www.ietf.org/rfc/rfc3349.txt.
22. R. Housley, D. Whiting, and N. Ferguson, "Counter with CBC-MAC (CCM), AES Mode of Operation," proposed cryptographic mode of operation submitted to the US Nat. Inst. Standards and Technology, http://csrc.nist.gov/encryption/modes/proposedmodes/ccm/ccm.pdf.

William E. (Bill) Burr *is the manager of the NIST Security Technology Group, which is responsible for Federal Information Processing Standards for Cryptography, a member of the Advanced Encryption Standard team, and chairman of the Federal Public Key Infrastructure Technical Working Group. He earned a BEE from the Ohio State University. His interests include research measuring of computer instruction set architecture. At NIST, he worked to develop standards for computer peripheral interfaces, high speed networks, and security. He was the chairman of the X3T9.2 standards committee that developed the Small Computer Systems Interface (SCSI), and has worked on standards for PKI and encryption as a member of the NIST Computer Security Division.*