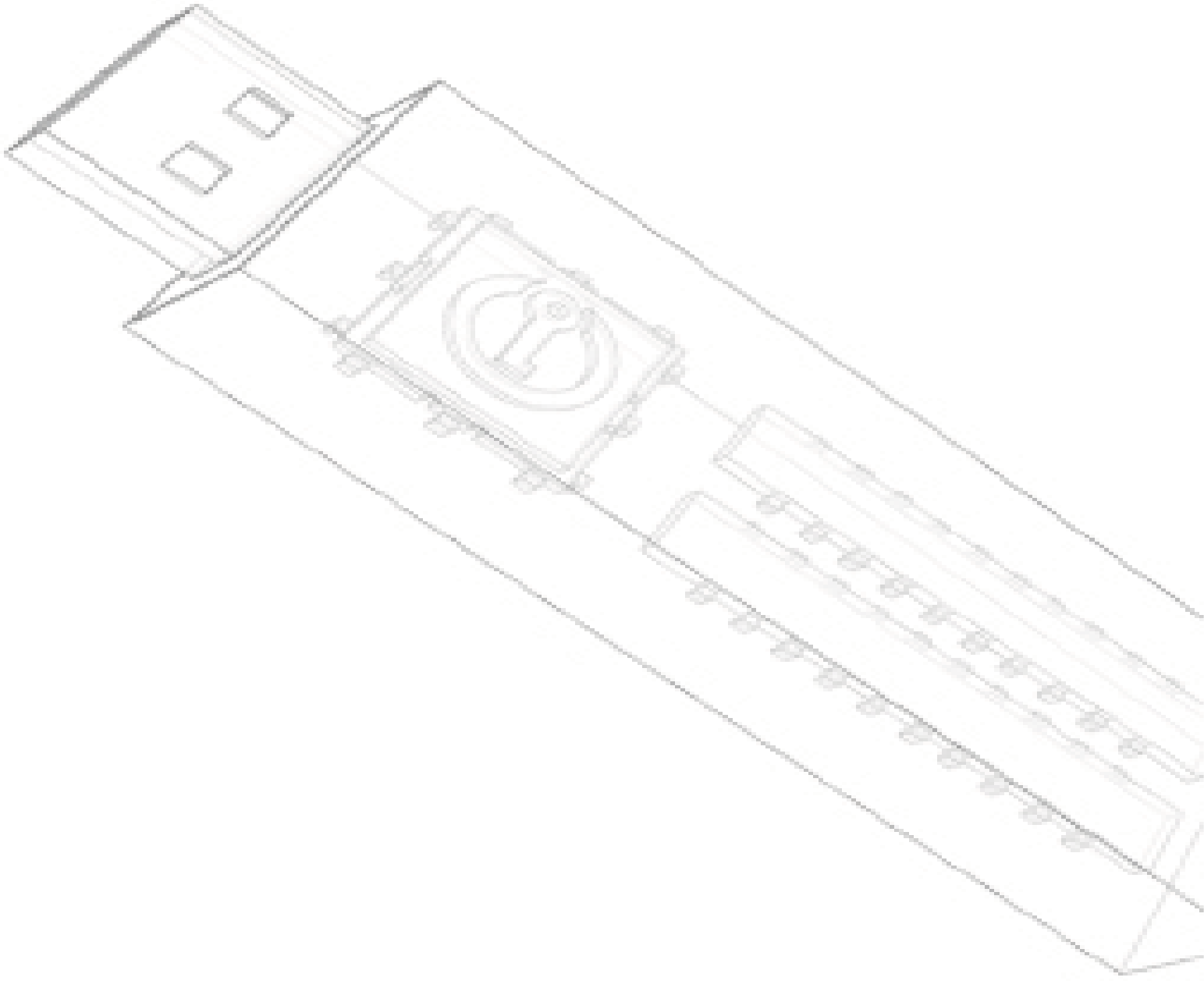
IronKey Data Encryption Methods

An IronKey Technical Brief
November 2007



Information
Depth: Technical

IRONKEY

Introduction

IronKey is dedicated to building the world's most secure flash drives. Our dedication is displayed in the attention to detail that our designers have brought to security regimen of the IronKey to ensure it is invulnerable to known attacks:

» The IronKey is provisioned at the factory with a 2,048-bit RSA Device Key Pair, which is used to authenticate the device and encrypt all communications between servers and the device. The key resides on the Cryptochip and is not accessible. This protects the IronKey against on-the-wire attacks or attempts to access or guess the device key.

» All communications between the IronKey and the host computer across the USB port are also encrypted using the RSA 2,048 bit Device Key Pair, minimizing the ability of sniffers to intercept information during transfer.

» IronKeys use a FIPS 140-2 compliant True Random Number Generator to generate an AES 128-bit CBC encryption key at the time the device is initialized by the owner. As a result, no one but the device's owner can possibly have this encryption key, and the key is invulnerable to typical attacks on pseudo-random number generators.

» The AES key is stored in hardware in the IronKey's Cryptochip, in a specially protected area of non volatile storage, shielded by layers of metal in the chip and further encrypted using a SHA-256 hash of the user's password. These keys are never stored in flash memory, on a local computer, or in a database. Because the Cryptochip will self-destruct if it detects any physical tampering by a thief or hacker, it is protected against physical attacks.

» The device destroys all data after 10 incorrect password attempts. The counter for this capability is stored in a specially-protected area of non-volatile storage, and cannot be reset by any means, including removing and replacing the device from the USB port.

» IronKey's patent-pending "self-destruct" methodology incorporates an exhaustive hardware erase of all flash and Cryptochip memory. This is not a simple clearing of file allocation tables, but a secure overwriting of data. This is done in hardware rather than via a software application for the ultimate protection.

It would take 149 trillion years to crack a 128-bit AES key.

- NIST

These features, among others, ensure that all of data and the keys stored on the device are invulnerable to physical access, RNG subversion, and power analysis techniques, as well as direct cryptanalytic, input based, state compromise, sniffing/on-the-wire, meet-in-the-middle, and replay attacks.

This technical brief explains the logic behind 128-bit CBC encryption as the standard for the IronKey. The major critical element that distinguishes IronKey's implementation of AES, and what allows us to use 128-bit encryption securely, is the mode used for encryption in combination with the hardware-based storage of the IronKey's AES encryption keys. The AES standard allows five encryption modes, Cipher Block Chaining (CBC), Electronic Code Book (ECB), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR)[1] . These modes were originally specified for the Federal Data Encryption Standard (DES) and are described in detail in FIPS Publication 81[2] Some manufacturers use the ECB mode of operation, which has a higher information leakage rate than CBC mode. Because it is most often implemented in software on USB devices, ECB mode is potentially less secure even with 256-bit keys than CBC mode with 128-bit keys generated and stored in hardware.

What kind of data encryption does the IronKey use?

IronKey uses the Advanced Encryption Standard (AES), which is a Federal Information Processing Standard (FIPS) that specifies the cryptographic algorithm for use by U.S. Government organizations and enterprises to protect sensitive information[3].

AES uses the Rijndael algorithm, which is a consistently good performer in hardware and software.

The AES algorithm is called Rijndael. The National Institute of Standards and Technology (NIST) went through a rigorous 4 year process of evaluating data encryption algorithms before settling on the Rijndael algorithm as the Advanced Encryption Standard.

NIST chose Rijndael as the AES based on its combination of security, performance, efficiency, ease of implementation and flexibility.

Specifically, Rijndael is consistently a very good performer in both hardware and software across a wide range of computing environments regardless of its use in feedback or non-feedback modes. Its key setup time is excellent, and its key agility is good. Rijndael's very low memory requirements make it very well suited for restricted-space environments, in which it also demonstrates excellent performance. Rijndael's operations are among the easiest to defend against power and timing attacks.

The AES Rijndael algorithm replaces the DES algorithm as the US Government's encryption standard.

What Key Sizes are Supported by the AES Algorithm?

The AES algorithm supports key sizes of 128, 192 and 256 bits.

How Long Would It Take To Crack an AES 128-Bit Key?

In the late 1990s, specialized "DES Cracker" machines were built that could recover a DES key after a few hours. In other words, by trying possible key values, the hardware could determine which key was used to encrypt a message.

Assuming that one could build a machine that could recover a single DES key in a second (i.e., try 255 keys per second), then it would take that machine approximately 149 thousand-billion (149 trillion) years to crack a single 128-bit AES key. To put that into perspective, the universe is believed to be less than 20 billion years old.

The Critical Importance of Using the Right Mode of AES

NIST has set five encryption modes for AES. The two most popular are Cipher Block Chaining (CBC) and Electronic Code Book (ECB).

NIST has defined five modes of operation for AES and other FIPS-approved Ciphers:

- CBC (Cipher Block Chaining)
- ECB (Electronic Codebook)
- CFB (Cipher Feedback)
- OFB (Output Feedback)
- CTR (Counter)

IronKey uses the CBC mode of AES, which was originally designed for encrypting large blocks of data securely. This mode can be very difficult to implement in hardware for NAND flash USB flash drives, because it requires generating and storing random Initialization Vectors (IV) for blocks. Use of a randomly generated IV prevents generation of identical ciphertext from blocks which have identical data that spans the first block of the cipher algorithm's block size.

ECB mode is easier to implement, but was designed to encrypt small data blocks and leaks information when used on large data blocks.

Because of the complexity of implementing the CBC mode of AES in hardware, some USB flash drive competitors have implemented the ECB mode of AES. ECB is much easier to implement, because it does not require IVs to be calculated and stored. Thus ECB was only designed to encrypt small blocks of data. AES ECB mode leaks information when used to encrypt large blocks of data. This is because the same input data will result in identical output data. This also allows the AES encryption key to be more easily derived by an attacker. AES CBC mode using a shorter key, such as 128 bits, is far more secure for encrypting large blocks of data than using ECB mode with a larger key such as 256 bits.

Dr. Bart Preneel, an esteemed cryptographer, has published numerous publications and course notes regarding the correct way to use an AES block cipher . Below is his illustration of why ECB mode AES encryption is not secure for encrypting large blocks of data.
Figure 1 is the source plaintext, in this case an image.

Figure 1
Source Plaintext

Always use cipher block chaining (CBC) mode instead of the electronic code book (ECB) mode
- The Laws of Cryptoraphy

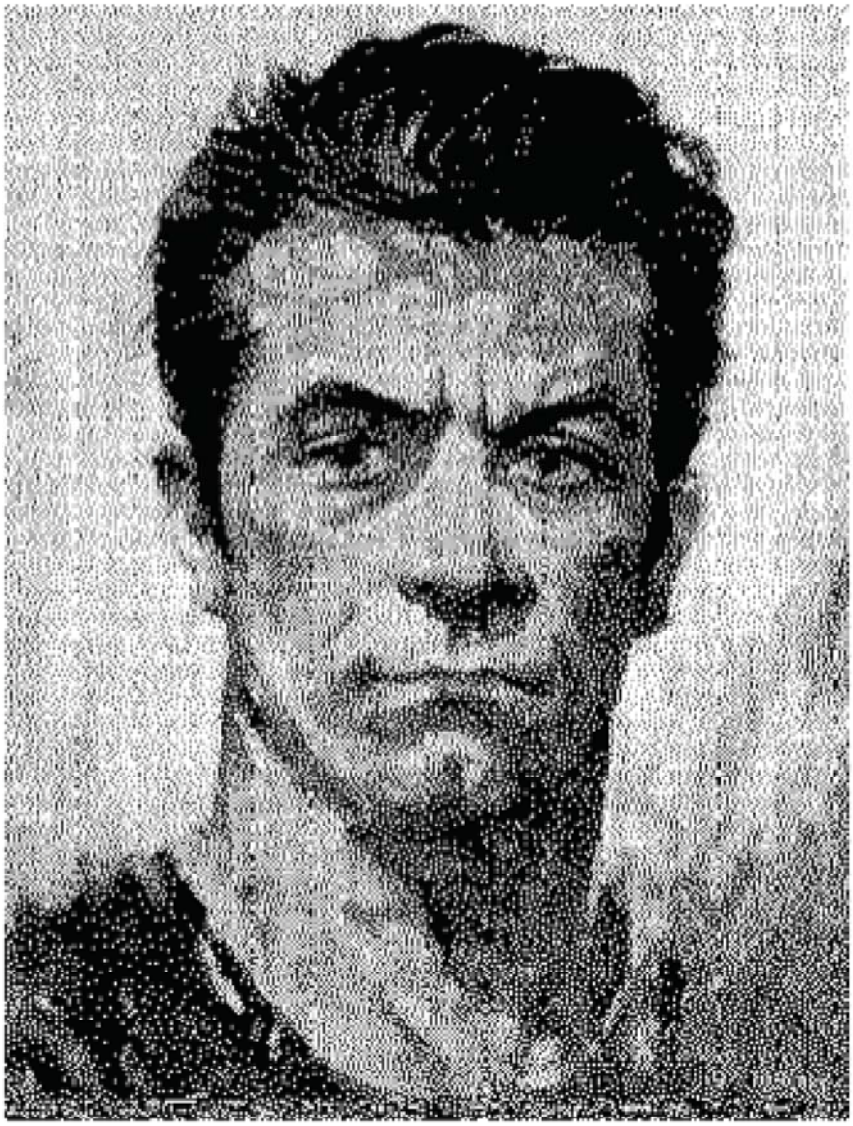


Figure 2 shows this image encrypted with AES ECB mode. It is clear that the image structure is visible and is highly vulnerable to attack. A larger key size does not make this more secure, because the same key is used on every block, there is no random IV used on blocks, and thus the blocks may chain entropy to each other.

A larger key size doesn't make AES ECB more secure, as the same key is used on every block with no random Initialization Vector.

Figure 2.
Encrypted With AES in ECB Mode

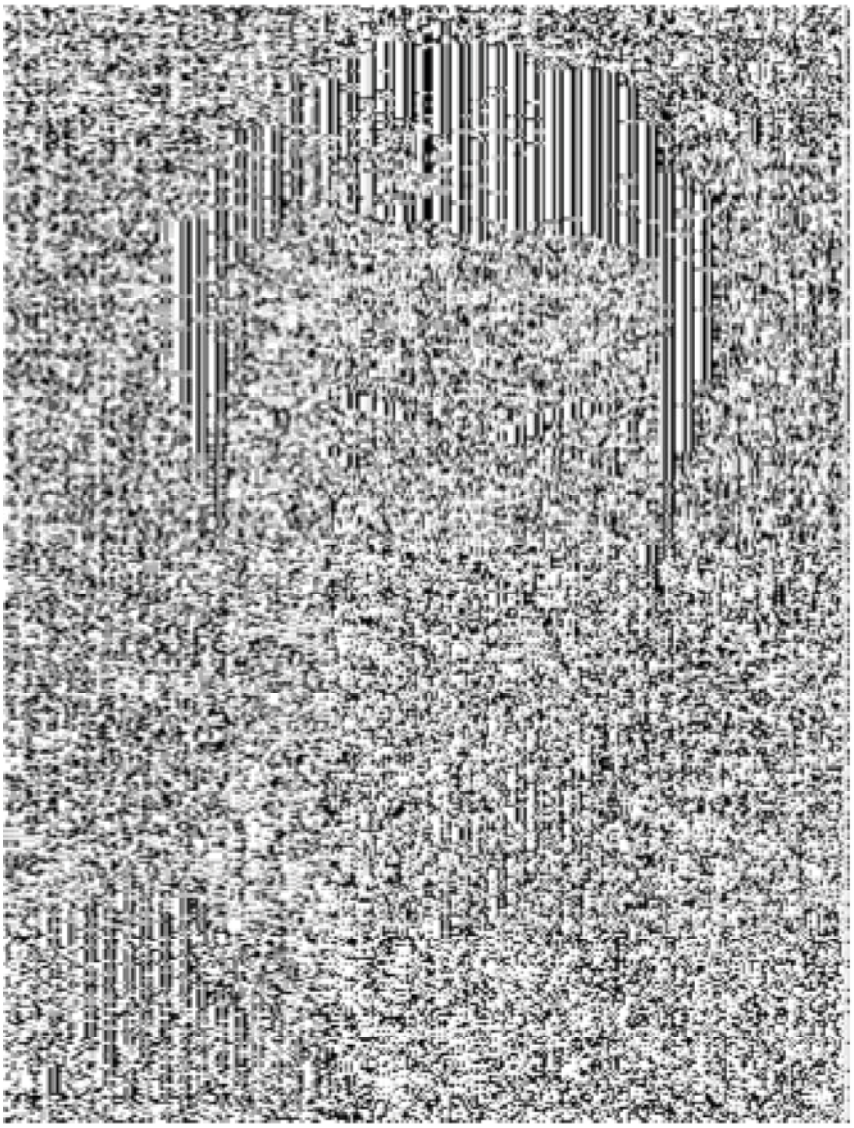
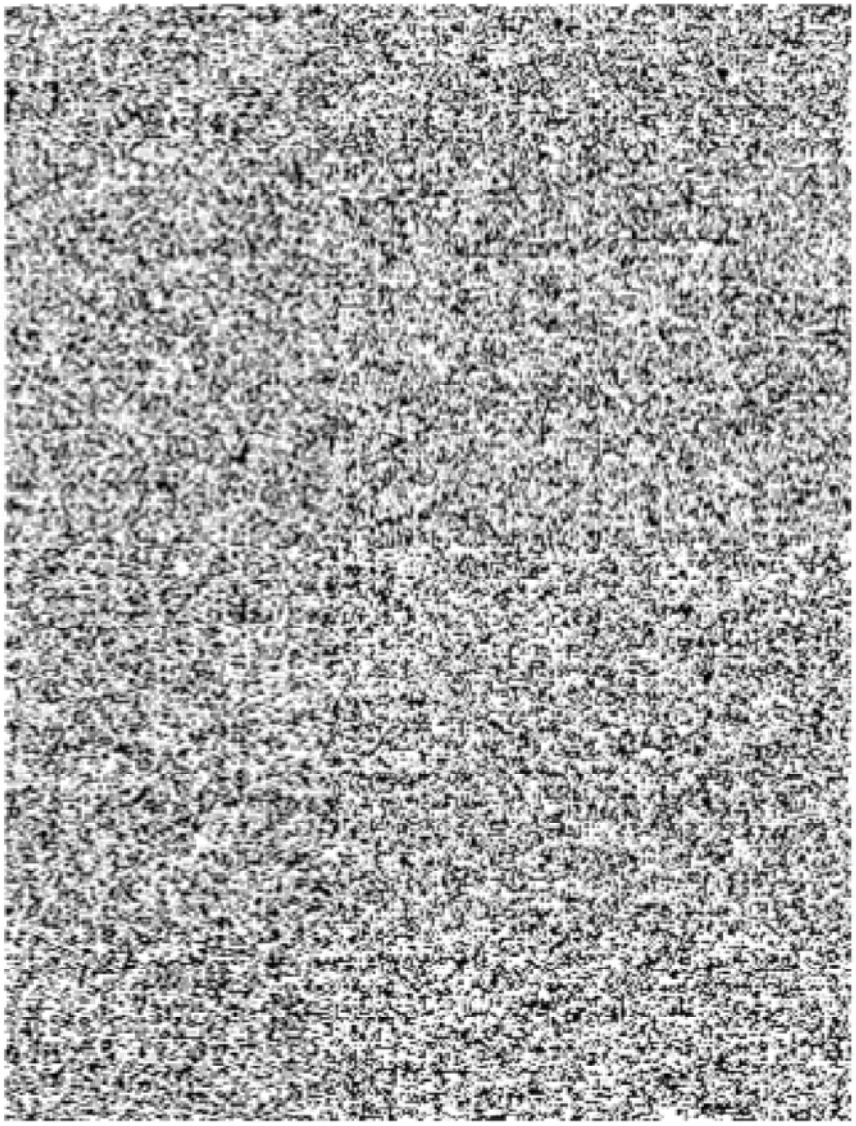


Figure 3
Encrypted With AES in CBC Mode



Cracking IronKey's 128-bit AES CBC mode isn't feasible. So attacks on the IronKey will focus on getting access to the encryption key.

Getting at an IronKey's AES keys is prevented by generating the keys with a hardware-based True Random Number Generator on the device.

The Importance of Strong Keys

Since cracking data encrypted with IronKey's 128-bit AES CBC mode encryption is not feasible, most attacks will focus on getting access to the encryption key itself. The IronKey prevents these types of attacks by generating the AES key with a hardware-based Random Number Generator on the device. The AES key is never imported or exported from the device onto a PC host.

IronKey's hardware encryption is always on and cannot be disabled or tampered with.

The IronKey stores the AES key in a tamper-resistant Cryptochip, which is designed to withstand power analysis attacks, and to self-destruct from physical and electrical attacks. Furthermore, the key itself is encrypted using a SHA256 hash of the user's password.

Preventing Brute Force Password Guessing

Since attacking the AES encrypted data on an IronKey is unfeasible, and access to the key is protected by the Cryptochip, the most feasible attack is a password guessing attack. The IronKey prevents these attacks by implementing password verification and unlocking in hardware. If an attacker enters the password incorrectly 10 times, the CryptoChip initiates a self

destruct process that eliminates access to the AES key, and also erases all the encrypted data with a low-level NAND flash hardware erase.

Conclusion

IronKey's approach to data encryption using 128-bit CBC encryption is tied closely to the physical limits of hardware-based encryption on a USB device as well as the overall security regime underlying the device. When looked at from that perspective, IronKey hardware-based encryption, combined with an AES 128-bit CBC key, provides the best combination of strength and performance for IronKey device security.

Find more information about IronKey online at: *www.ironkey.com*

IronKey, Inc.
5150 El Camino Real, Suite C31
Los Altos, CA 94022 USA
+1 (650) 492-4055
info@ironkey.com

---

[1] To understand the concepts behind these modes of operation in detail, see Schneir, Bruce. Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd Edition, (Boston: John Wiley, 1996).

[2] "FIPS Publication 81 - Des Modes of Operation" (Washington, DC: US Government Printing Office, December, 1980). See also http://csrc.nist.gov/publications/fips/fips81/fips81.htm

[3] "FIPS Publication 197 – "Announcing the Advanced Encryption Standard." (Washington, DC: US Government Printing Office, November, 2001). See also http://csrc.nist.gov/publications/fips/fips197/fips197.htm. Also see other publications listed in the bibliography at the end of this technical brief.

Bibliography

AES

"AES Questions and Answers." (Washington, DC: US GPO, November, 2001). See also http://www.nist.gov/public_affairs/releases/aesq&a.htm.

Dworkin, Morris., "Recommendations for Block Cipher Modes of Operation." (Washington, DC: NIST Special Publication 800-28A, November, 2001).

"FIPS Publication 197 – "Announcing the Advanced Encryption Standard." (Washington, DC: US GPO, November, 2001). See also http://csrc.nist.gov/publications/fips/fips197/fips197.htm.

"FIPS Publication 81 - DES Modes of Operation" (Washington, DC: US Government Printing Office, December, 1980). See also http://csrc.nist.gov/publications/fips/fips81/fips81.htm.

Soto, Juan Jr., "Randomness Testing of the AES Candidate Algorithms." (Washington, DC: NIST, 2001). See also http://www-08.nist.gov/archive/aes/round1/r1-rand.pdf.

AES CBC vs. ECB Encryption

Schneir, Bruce. Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd Edition. (Boston, John Wiley, 1996)

Bart Praneel's Home Page: http://homes.esat.kuleuven.be/~preneel/.

http://www.cs.rice.edu/~dwallach/courses/comp527_f2004/week02-2-crypto-intro.pdf
http://crypto.stanford.edu/cs155-spring06/04-crypto.pdf.

Wagner, Neal R., The Laws of Cryptography. (San Antonio, TX: Neal Wagner, 2003). See also http://www.cs.utsa.edu/~wagner/lawsbookcolor/laws.pdf.