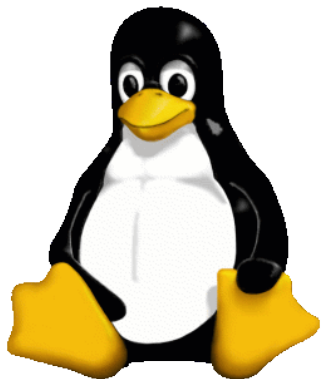


Introduction to Cryptography

Modes of Operation



The Linux Penguin

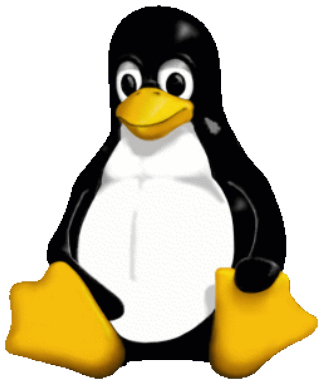


(Image by lewing@isc.tamu.edu and The GIMP; retrieved from

[https:](https://upload.wikimedia.org/wikipedia/commons/a/af/Tux.png)

[//upload.wikimedia.org/wikipedia/commons/a/af/Tux.png](https://upload.wikimedia.org/wikipedia/commons/a/af/Tux.png).)

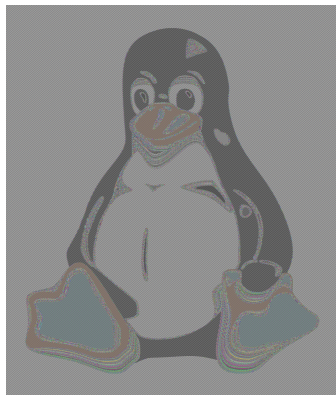
The Linux Penguin, Encrypted



(Image by lewing@isc.tamu.edu and The GIMP; retrieved from

[https:](https://upload.wikimedia.org/wikipedia/commons/a/af/Tux.png)

[//upload.wikimedia.org/wikipedia/commons/a/af/Tux.png](https://upload.wikimedia.org/wikipedia/commons/a/af/Tux.png).)



(Image by Filippo Valsorda; retrieved from

<https://blog.filippo.io/the-ecb-penguin/>.)

- Clearly, something went wrong
- Encrypted data is supposed to be random—why is the second image recognizable?
- AES was used incorrectly—every 16 bytes were encrypted separately
- We saw with stream ciphers that how they're used matters; the same is true for block ciphers
- What we need are *modes of operation*

More Details on the Failure

- The image was encoded in **PPM binary** format—3 bytes per pixel
- For constant color areas—most of this image—every 48 bytes, the cipher blocksize (16 bytes) and the pixel blocksize (3 bytes) align
- In other words, AES will see the same plaintext, which it's encrypting to the same ciphertext
- We need a way to make AES—or any block cipher; the same would have happened with DES every 24 bytes—encrypt differently even when the plaintext block is the same
- That is one of the goals of modes of operation

Bear in Mind...

In *The Mythical Man-Month*, Fred Brooks wrote this about programming. It's even more true about cryptography:

First, one must perform perfectly. The computer resembles the magic of legend in this respect, too. If one character, one pause, of the incantation is not strictly in proper form, the magic doesn't work

The little things *really* count in cryptography—and it's not always obvious why.

Do not invent your own crypto!

Modes of Operation

There are many modes of operation. Let's start with the five basic confidentiality modes.

- ECB Electronic Code Book
- CBC Cipher Block Chaining
- CFB Cipher Feedback
- OFB Output Feedback
- CTR Counter Mode

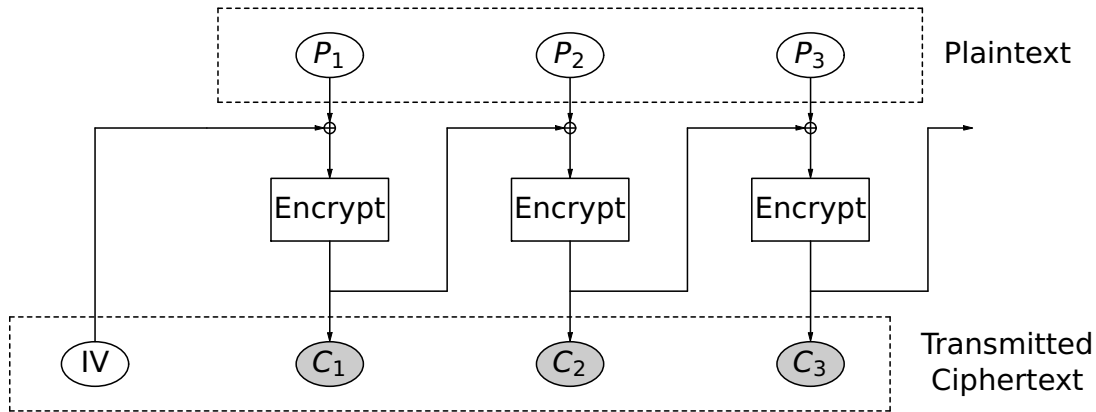
There are other modes for things like encrypting disks, encrypting keys, authenticating messages, and *format-preserving encryption*: map digits to digits, letters to letters, etc.

- In ECB mode, plaintext is encrypted 16 bytes at a time, as is
- This is what was done with the penguin image
- It's called a "code book" mode because, for any given key, it effectively is a code book: there's a constant mapping from plaintext blocks to ciphertext blocks
- This leaks information—the byte distribution of the *file* isn't flat if there are suitable plaintext repetitions
- Except for special purposes, *never* use ECB mode
- We write $E(k, P) \rightarrow C$ to denote "encryption of plaintext P with key k to produce ciphertext C "

CBC: Cipher Block Chaining

- CBC is probably the most popular mode of operation, though there are newer ones that should generally be used in its place
- It combines the output of the previous encryption block with the plaintext of the next block
- This means that identical plaintexts are encrypted differently!
- But: it requires an *initialization vector* (IV), an extra block of ciphertext

CBC: Cipher Block Chaining



$$E(k, P_i \oplus C_{i-1}) \rightarrow C_i$$
$$D(k, C_i) \oplus C_{i-1} \rightarrow P_i$$

Properties of CBC

- The ciphertext of each encrypted block depends on the IV and the plaintext of all preceding blocks.
- There is a dummy initial ciphertext block C_0 known as the *Initialization Vector* (IV); the receiver must know this value.
- Consider a 4-block message:

$$C_1 = E(k, P_1 \oplus IV)$$

$$C_2 = E(k, P_2 \oplus C_1)$$

$$C_3 = E(k, P_3 \oplus C_2)$$

$$C_4 = E(k, P_4 \oplus C_3)$$

If C_2 is damaged during transmission, what happens to the plaintext?

Error Propagation in CBC Mode

- Look at the decryption process, where C' is a corrupted version of C :

$$P_1 = D(k, C_1) \oplus IV$$

$$P_2 = D(k, C'_2) \oplus C_1$$

$$P_3 = D(k, C_3) \oplus C'_2$$

$$P_4 = D(k, C_4) \oplus C_3$$

- P_1 depends only on C_1 and IV , and is unaffected
- P_2 depends on C_2 and C_1 , and hence is corrupted
- P_3 depends on C_3 and C_2 , and is also corrupted. The enemy can control the change to P_3 .
- P_4 depends on C_4 and C_3 , and not C_2 ; it thus isn't affected.
- Conclusion: Two blocks change, one of them predictably
- If this matters, some form of integrity check must be added

Cutting and Pasting CBC Messages

- Consider the encrypted message

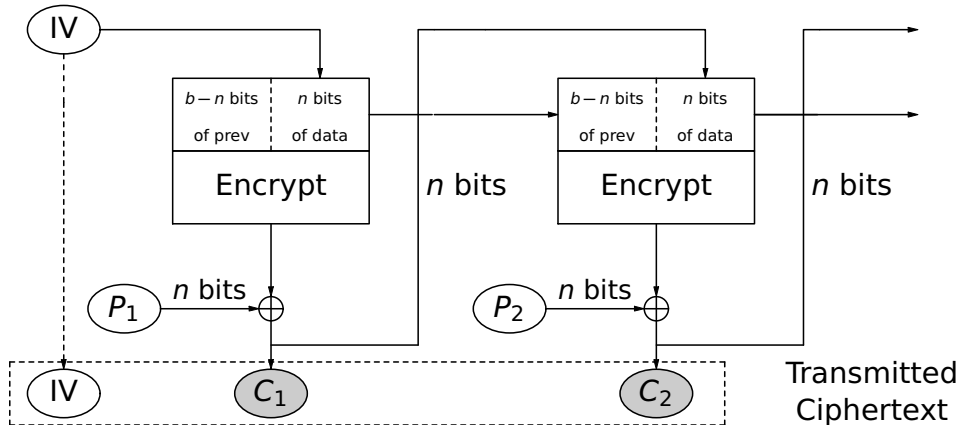
$$IV, C_1, C_2, C_3, C_4, C_5$$

- The shortened message IV, C_1, C_2, C_3, C_4 appears valid
- The truncated message C_2, C_3, C_4, C_5 is valid: C_2 acts as the IV. It decrypts to P_3, P_4, P_5 .
- Even C_2, C_3, C_4 is valid, and will decrypt properly to P_3, P_4 .
- Any subset of a CBC message will decrypt cleanly.
- If we snip out blocks, leaving IV, C_1, C_4, C_5 , we only corrupt one block of plaintext.
- Conclusion: if you want message integrity, you have to do it yourself.

- The IV provides *semantic security*: identical messages have different ciphertexts
- Common message start is also hidden
- The IV *must* be unpredictable by the enemy
- Good strategy: hash (or HMAC) a counter initialized from a random value to produce the IVs
- Note: since the IV is transmitted unencrypted, the other side need not know this key

- CBC mode requires the input to be a multiple of the cipher's block size.
- Must (somehow) handle other lengths
- Usual strategy: (securely) transmit explicit input length
- Option: *Cipher-Text Stealing* (see RFC 2040). Does not increase message size

n -bit Cipher Feedback



$$P_i \oplus E(k, C_{i-1}) \rightarrow C_i$$
$$E(k, C_{i-1}) \oplus C_i \rightarrow P_i$$

Properties of Cipher Feedback Mode

- Underlying block cipher used only in encryption mode
- Feedback path actually incorporates a shift register: shift old cipher input left by n bits; insert first n bits of previous ciphertext output
- 8-bit CFB is good for asynchronous terminal traffic — but requires one encryption for each *byte* of plaintext
- It's one way to make a stream cipher out of a block cipher
- Errors propagate while bad data is in the shift register — 17 bytes for CFB₈ when using AES.
- Copes gracefully with deletion of n -bit unit
- Interesting uses: CFB₁ (for HDLC links), CFB₈, CFB₁₂₈
- IV selected the same way as in CBC mode

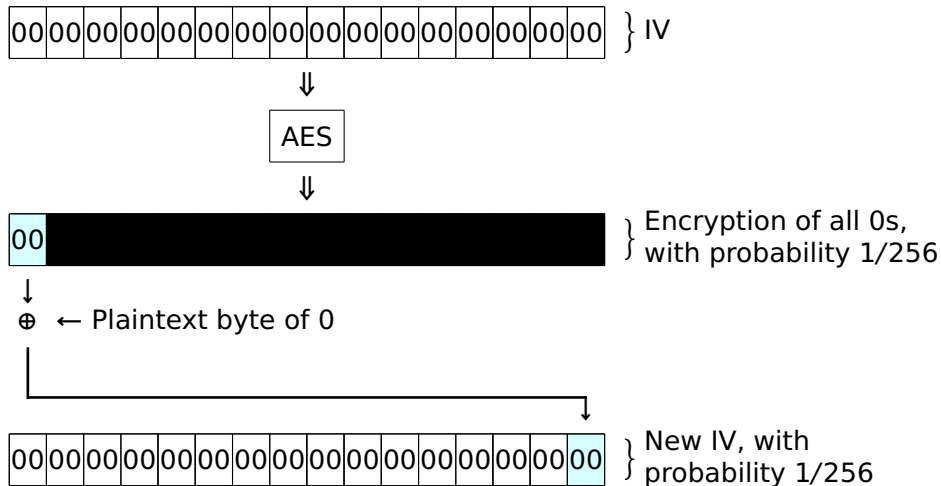
A Weakness of CFB₈

- Suppose the IV is all zeroes (a bad idea. . .)
- Suppose you're encrypting all zeroes (plausible)
- With probability $1/256$, the ciphertext will be unchanged! Why?

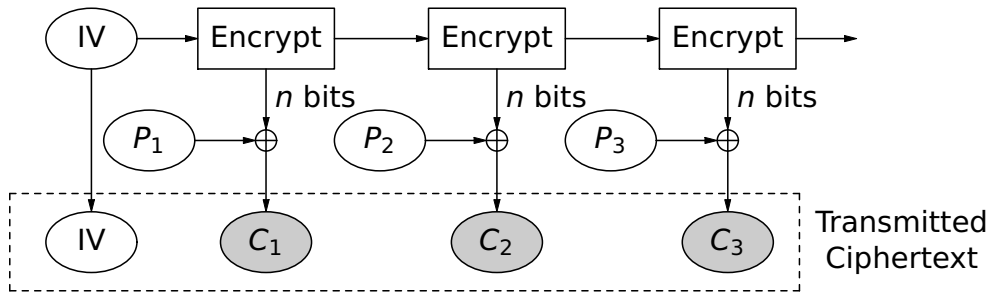
A Weakness of CFB₈

- Suppose the IV is all zeroes (a bad idea. . .)
- Suppose you're encrypting all zeroes (plausible)
- With probability $1/256$, the ciphertext will be unchanged! Why?
- By the properties of AES, all 256 values for a single byte of ciphertext are equally probable
- In this case, though the plaintext—16 bytes of 0—is constant, the keys can change
- A major vulnerability in Windows due to such an error was recently found and patched

CFB₈ with All-Zero IV



n -bit Output Feedback



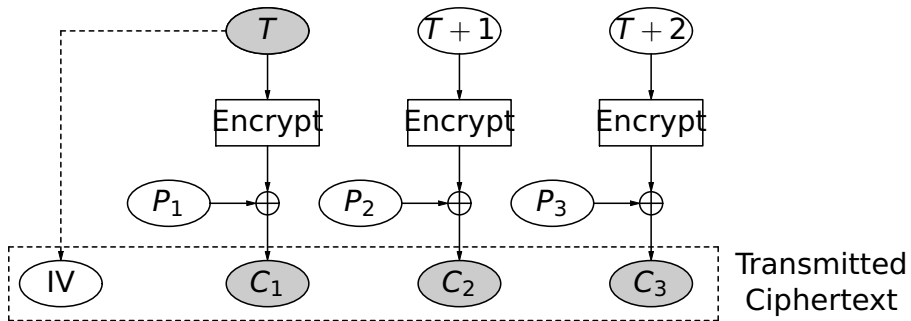
Properties of Output Feedback Mode

- No error propagation
- Active attacker can make controlled changes to plaintext
- OFB is another form of *stream cipher*
- Useful when random errors but not error propagation can be tolerated (think fax machines), but there's no chance of bit loss, so resynchronization isn't important
- Cipher output blocks for a message can be generated before the plaintext is ready
- Also: OFB is a secure random number generator
- Never reuse the same key/IV pair with OFB. Why?

Properties of Output Feedback Mode

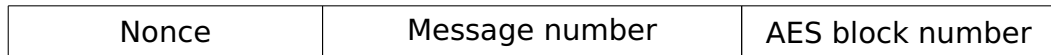
- No error propagation
- Active attacker can make controlled changes to plaintext
- OFB is another form of *stream cipher*
- Useful when random errors but not error propagation can be tolerated (think fax machines), but there's no chance of bit loss, so resynchronization isn't important
- Cipher output blocks for a message can be generated before the plaintext is ready
- Also: OFB is a secure random number generator
- Never reuse the same key/IV pair with OFB. Why?
- This is a stream cipher—just as with RC4, an attacker can play one ciphertext against another

Counter Mode



Properties of Counter Mode

- Another form of stream cipher
- Frequently split the counter into three sections: per-session random nonce, message number, and block number within the message



- Active attacker can make controlled changes to plaintext
- Highly parallelizable; no linkage between stages, hence useful for very high speed traffic
- Vital that counter never repeat for any given key—why?
- Again, this is a stream cipher

Which Mode for What Task?

- General file or packet encryption: CBC.
 - 👉 Input must be padded to multiple of cipher block size
- Risk of byte or bit deletion: CFB_8 or CFB_1
- Bit stream; noisy line and error propagation is undesirable: OFB
- Very high-speed data: CTR
- In most situations, an integrity check is needed

Integrity Checks

- Actually, integrity checks are almost always needed
- Frequently, attacks on integrity can be used to attack confidentiality
- Usual solution: use separate integrity check along with encryption
- Simple solutions don't work
- Choices: CMAC (a block cipher mode of operation), HMAC, combined modes of operation, lesser-known schemes
- Note well: always do the integrity check over the ciphertext, not the plaintext

Combined Modes of Operation


- There are some modes of operation that provide confidentiality and integrity
- Why not just use HMAC? Multiple keys, and multiple passes over the data
- 👉 In many situations, memory bandwidth, not CPU speed, is the limiting factor to system performance

Galois Counter Mode (GCM)

- Permits an optional plaintext header that is authenticated but not encrypted
- Note that the length of the two sections is included in the hash, to prevent extension or truncation attacks
- Encryption/decryption is counter mode-like, so caveats about key and IV reuse apply
- The integrity check is sequential, but the encryption and decryption, which are more expensive, can be done in parallel

 GCM is the preferred mode of operation for most purposes—but again, note that it's counter-like, so watch out for key and IV reuse

Moore's Law and Public Key Cryptography

- For RSA, doubling the modulus length increases encryption time by 4× and decryption time by 8×
-  For smart card use, remember that digital signatures involve decrypting, i.e., the more expensive operation
- Attack time against RSA is based on factoring algorithms, not brute force: there are far too many possible primes for brute force to be ever be possible
- (Density of prime numbers up to N approximately $N/\ln N$)
- For number field sieve, complexity of factoring is approximately proportional to

$$e^{\left(\sqrt[3]{\frac{64}{9} + o(1)} (\ln n)^{1/3} (\ln \ln n)^{2/3}\right)}$$

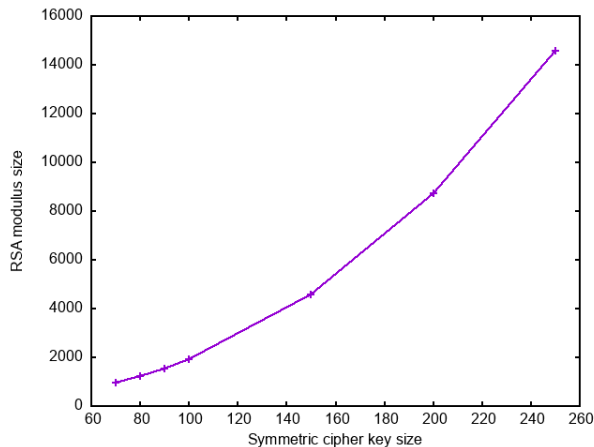
- Sub-linear, but space complexity goes up much faster
- There is a paper design for a \$10M machine (TWIRL) to factor a single 1024-bit number in one year

Rough Table of Key Length Equivalences

<i>Symmetric Key Size (bits)</i>	<i>RSA or DH Modulus Size (bits)</i>
70	947
80	1228
90	1553
100	1926
150	4575
200	8719
250	14596

Equivalent elliptic curve moduli are much shorter.
(Numbers by Orman and Hoffman, [RFC 3766](#))

Public versus Symmetric Key Sizes



- Remember that we need 128-bit *strength*—256-bit keys are for defense against massively parallel quantum computers
- But even a single quantum computer like that would attack RSA much more easily
- The interesting equivalence is for 128-bit keys

Current Key Length Recommendations

- For most purposes, 128-bit symmetric keys are sufficient—unless you're afraid of massively parallel quantum computers while the data is still sensitive
- For data that must be protected for many years, use 256 bits
- 1024-bit public keys are obsolete; new keys should probably be 2048 bits or longer
- Keys for certificate authorities (stay tuned) should be at least 3072 bits and perhaps 4096 bits
- As always, consider your adversary's powers
- “Attacks always get better; they never get worse”

Questions?



(Great egret eating a small fish, Central Park, June 23, 2020)