Practical Cryptographic Systems

Symmetric Cryptography II

Instructor: Matthew Green

Housekeeping

- A1 out (notes: not Enigma!)
- Reading quiz/assignment coming on Weds!
  - Boneh/Shoup book readings
- Projects
  - I will put up an updated list on Github and we'll talk Weds about this

Numbers stations

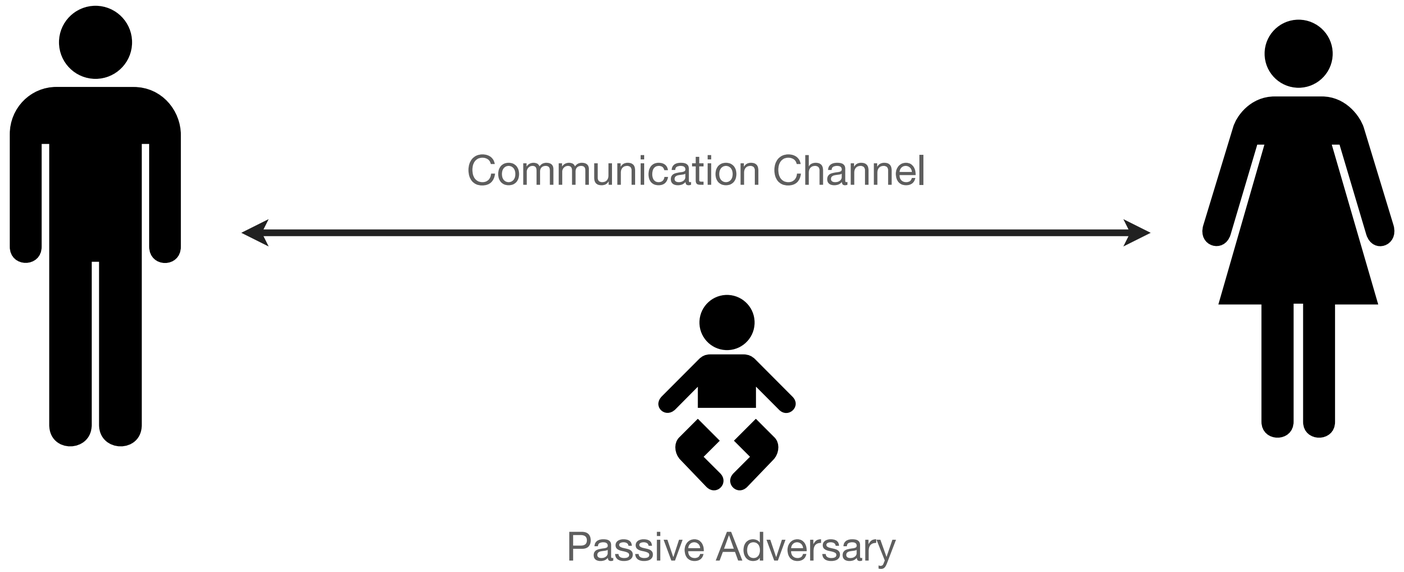
https://www.sigidwiki.com/wiki/Category:Numbers_Stations

http://www.numbersoddities.nl/rusmilcw.html

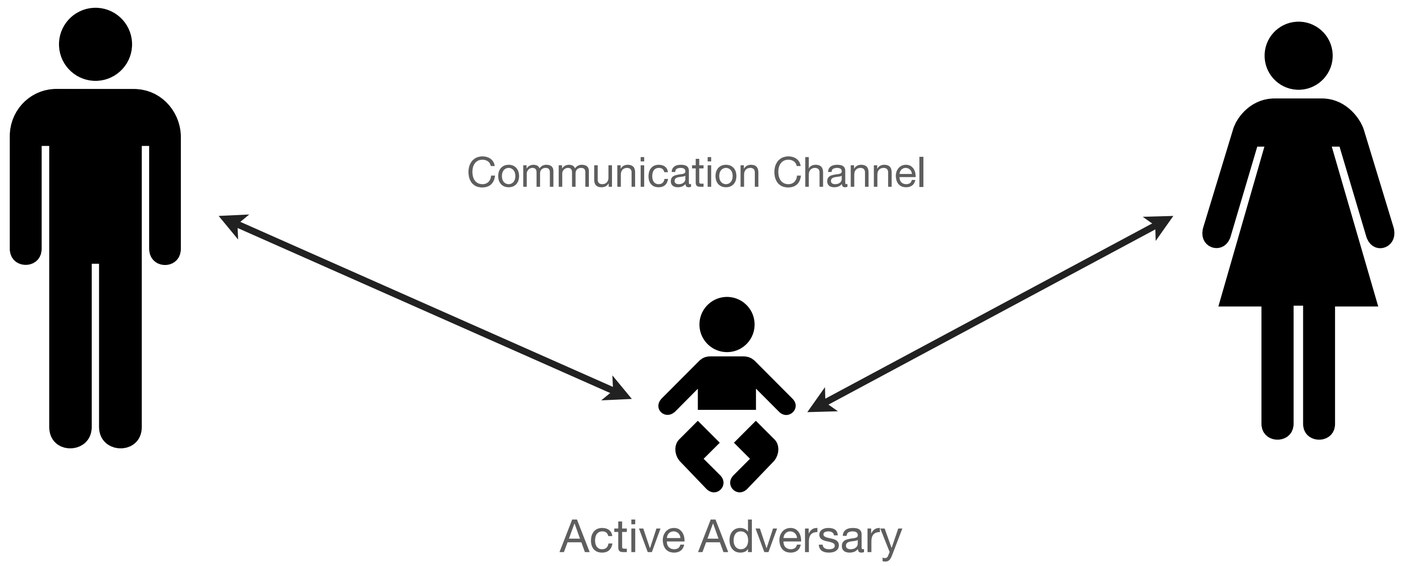
News

Review

Review



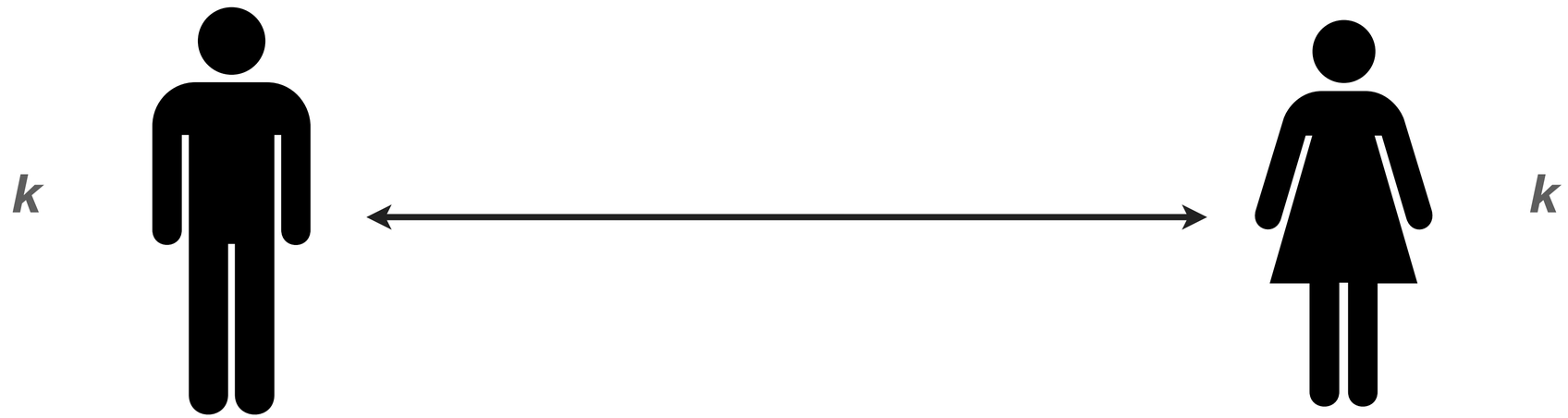
Review



Review: security properties

- Two basic properties we like to achieve:
  - Data confidentiality
  - Data authenticity (“integrity”)

Review: symmetric encryption



A cipher is a pair of algorithms: (E, D)

$$C \leftarrow E(K, M)$$
$$M \leftarrow D(K, C)$$

Not shown: the algorithm for generating keys. This is usually defined implicitly.

Also see Boneh-Shoup Section 2.2

A cipher is a pair of algorithms: (E, D)

ciphertext key plaintext

$$C \leftarrow E(K, M)$$

$$M \leftarrow D(K, C)$$

Also see Boneh-Shoup Section 2.2

A cipher is a pair of algorithms: (E, D)

$$C \leftarrow E(K, M)$$
$$M \leftarrow D(K, C)$$

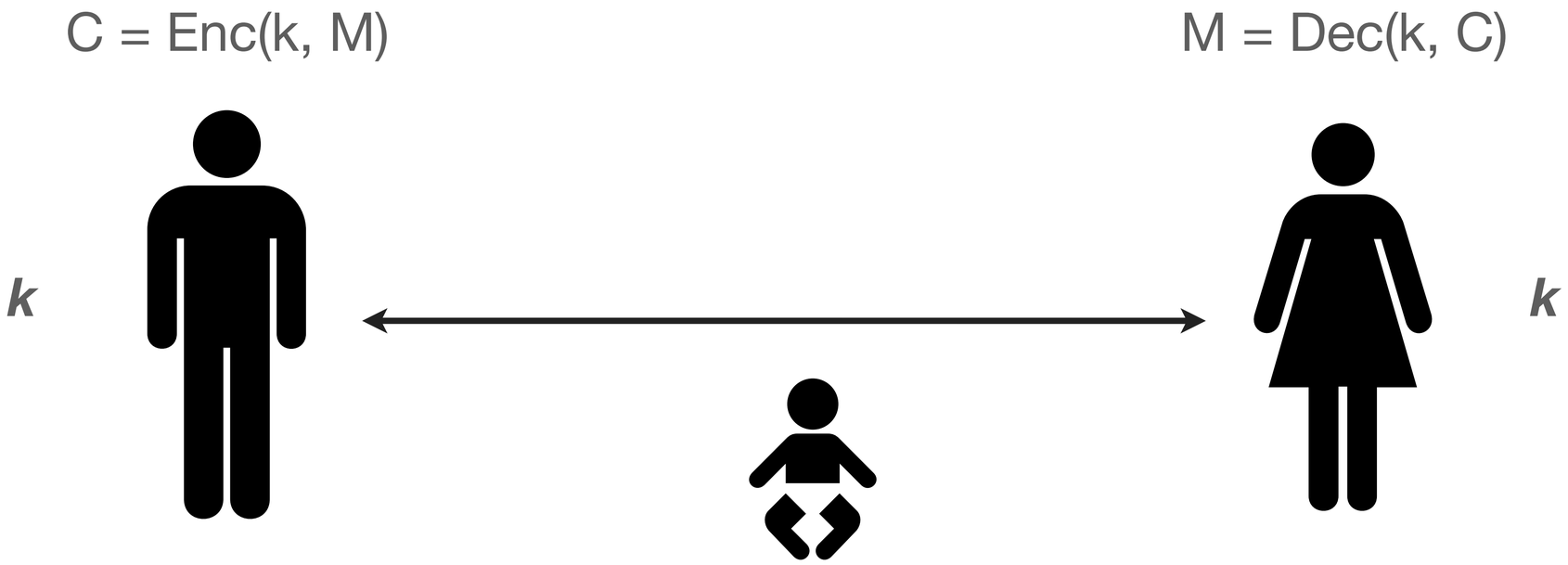
Ciphertext/Plaintext/Key “spaces”: $K \in \mathcal{K}, M \in \mathcal{M}, C \in \mathcal{C}$

Correctness: $\forall K, M : D(K, E(K, M)) = M$

Not shown: the algorithm for generating keys. This is usually defined implicitly.

Also see Boneh-Shoup Section 2.2

Review: symmetric encryption



What the adversary should know:

Enc, Dec, maybe some information about M
but nothing about k

Last time

- Classical ciphers: what’s wrong with them?

Last time

- Classical ciphers: what's wrong with them?
  - Small key size (e.g., shift cipher)
  - Leakage of plaintext patterns (e.g., substitution cipher)

Last time

- Classical ciphers: what's wrong with them?
  - Small key size (e.g., shift cipher)
  - Leakage of plaintext patterns (e.g., substitution cipher)
- One-time ciphers
  - "Perfectly" (information theoretically) secure
  - … if you use them correctly***

  *** *Key must be as long as all plaintext, perfectly random, never re-used*

Review: perfect secrecy

Definition 2.1 (perfect security). *Let $\mathcal{E} = (E, D)$ be a Shannon cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. Consider a probabilistic experiment in which the random variable $\mathbf{k}$ is uniformly distributed over $\mathcal{K}$. If for all $m_0, m_1 \in \mathcal{M}$, and all $c \in \mathcal{C}$, we have*

$$\Pr[E(\mathbf{k}, m_0) = c] = \Pr[E(\mathbf{k}, m_1) = c],$$

*then we say that $\mathcal{E}$ is a **perfectly secure** Shannon cipher.*

From Boneh-Shoup 0.6, p.8

What if you don’t use OTP correctly?

What if you don’t use OTP correctly?

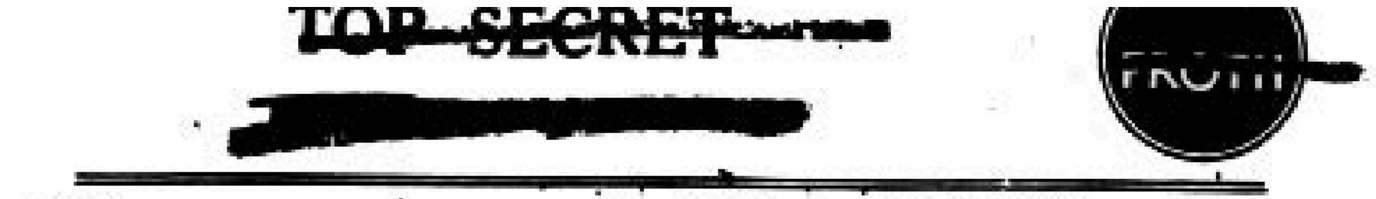
The U.S. Army's Signal Intelligence Service, the precursor to the National Security Agency, began a secret program in February 1943 later codenamed VENONA

The mission of this small program was to examine and exploit Soviet diplomatic communications but after the program began, the message traffic included espionage efforts as well.

Although it took almost two years before American cryptologists were able to break the KGB encryption, the information gained through these transactions provided U.S. leadership insight into Soviet intentions and treasonous activities of government employees until the program was canceled in 1980.

The VENONA files are most famous for exposing Julius (code named LIBERAL) and Ethel Rosenberg and help give indisputable evidence of their involvement with the Soviet spy ring.

What if you <u>don't</u> use OTP correctly?



~~TOP SECRET~~ [redacted]

FROM

USSR

[redacted]

Ref No: S/NBF/T500

Issued: [redacted]29/3/1954

Copy No: 20

"ALEKSANDR" TO BE WARNED NOT TO USE ANNA COLLOMS' ADDRESS.

From: NEW YORK

To: BUENOS AIRES

No.: -

29 September 1943

[Letter][a]no. 14.

Do not write any more to the address of An[n]a COLLOMS [i]. Warn ALEKSANDR[ii] about this. I shall send a new address in the very near future.

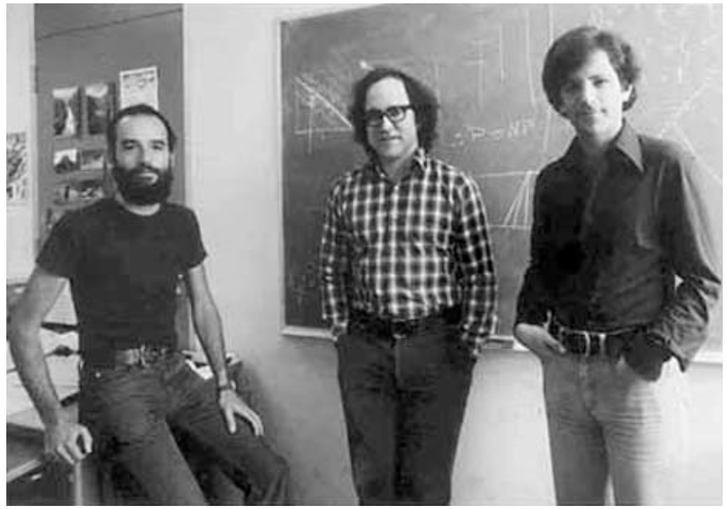
OTP pad re-use (board)

The 1970s

1972

1976
(1974)

U.K. GCHQ



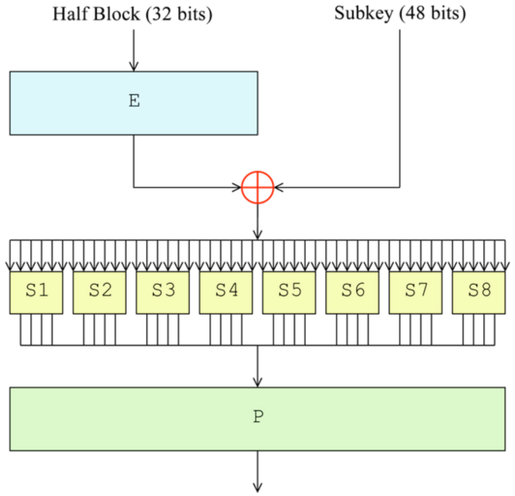
1977
(1973)

The Implications

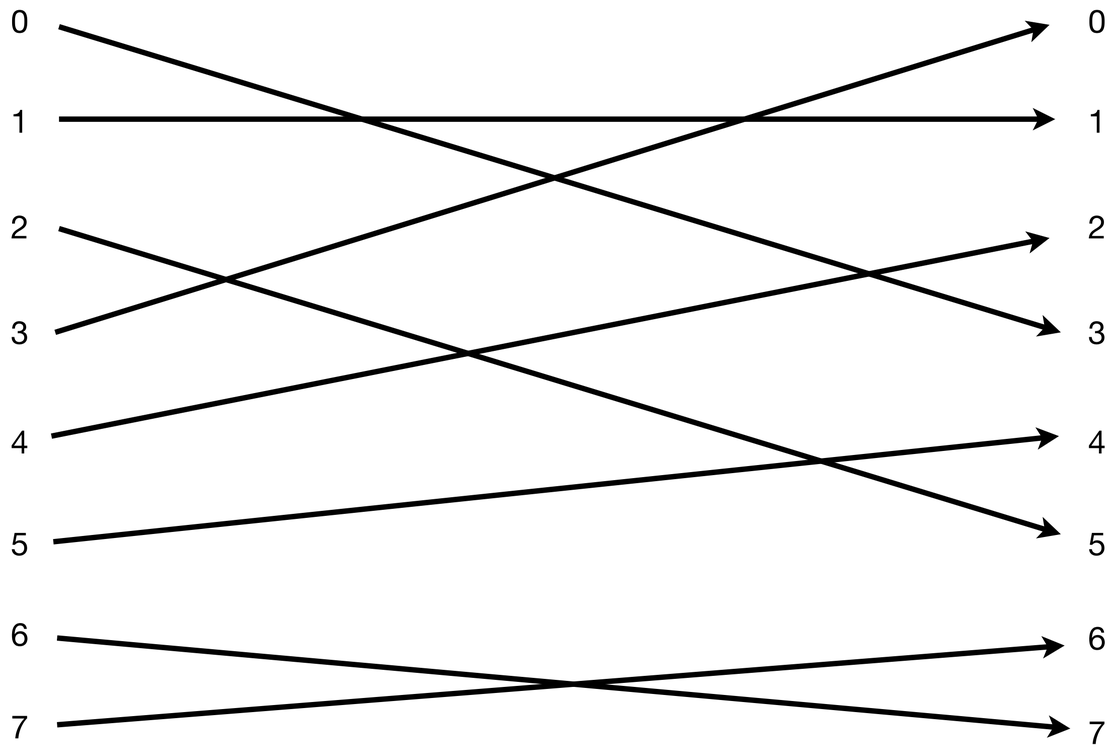
- Exponential increase in study & usage of cryptography in industry, academia
- Wide-scale deployment of cryptographic systems
- Provable Security
  - Cryptographic Systems can be reduced to some hard mathematical problem

Data Encryption Standard

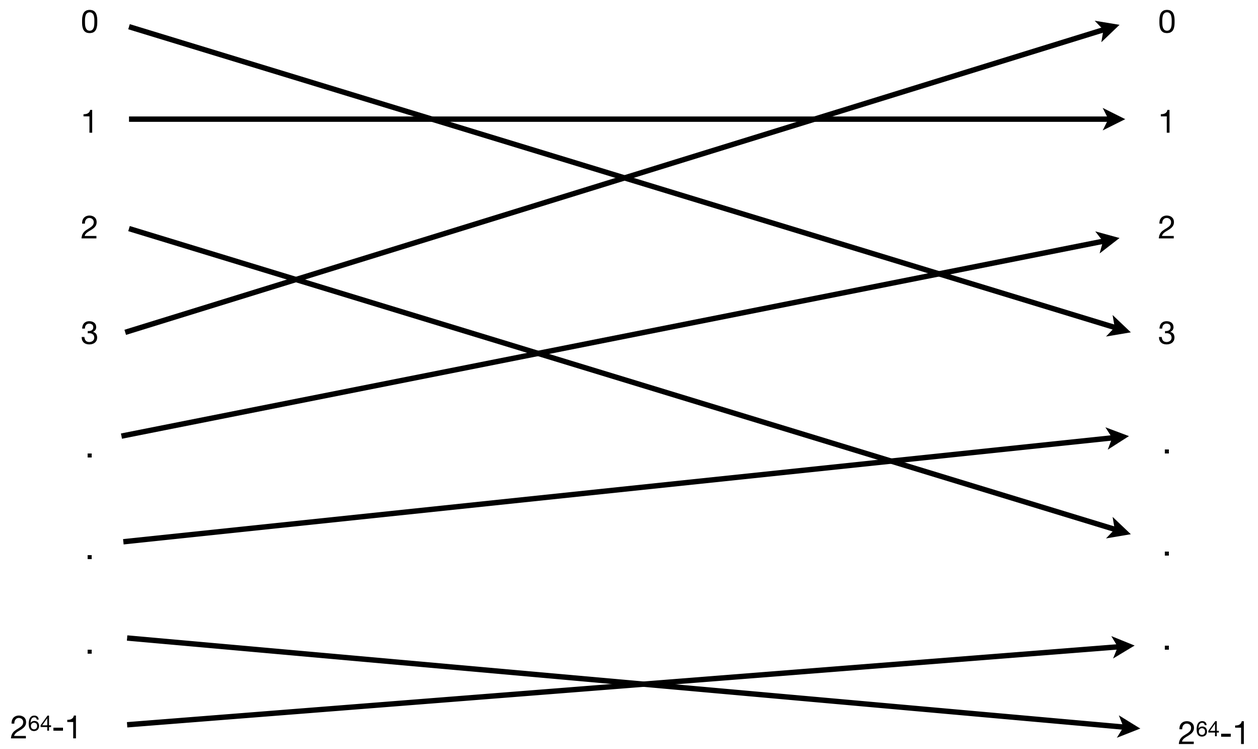
- Commercial-grade Block Cipher
  - 64-bit block size
  - 56 bit key (+ 8 bits parity)
  - “Feistel Network” Construction



Permutation



Permutation



Ciphers / Permutation Families

- Can’t have just one permutation
  - Alice & Bob know the permutation
    Adversary should not
  - Permutation is “random” (ish)
  - For a 64-bit input block, how many possible permutations are there?

Ciphers / Permutation Families

- Can’t have just one permutation
  - Alice & Bob know the permutation
    Adversary should not
  - Permutation is “random” (ish)
  - For a 64-bit input block, how many possible permutations are there?
  - How does this compare to DES key length?

Block Cipher

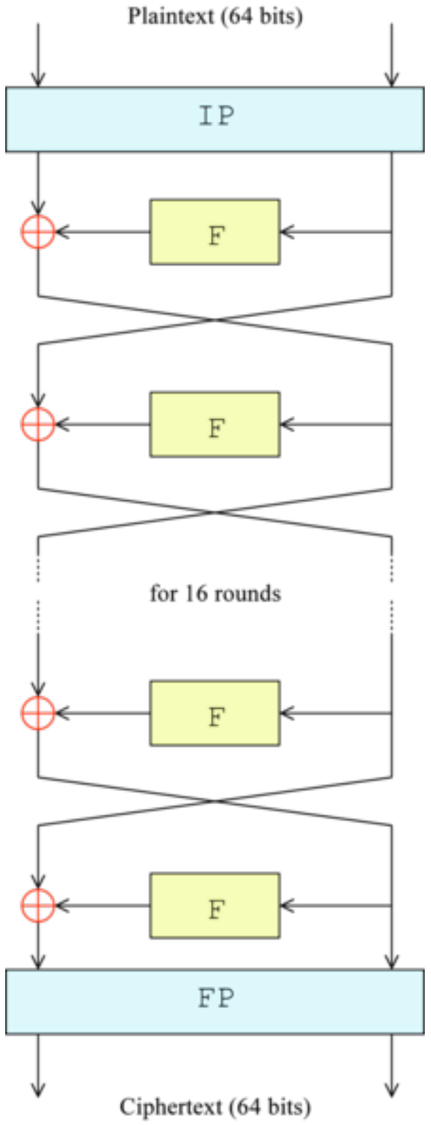
- Block cipher is a family of permutations
  - Indexed by a key (DES = 56 bit key)
  - “Pseudo-random”

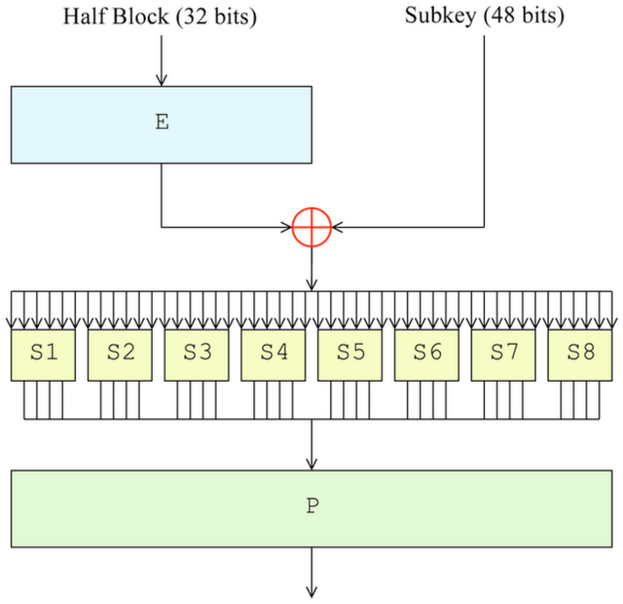
Block Cipher

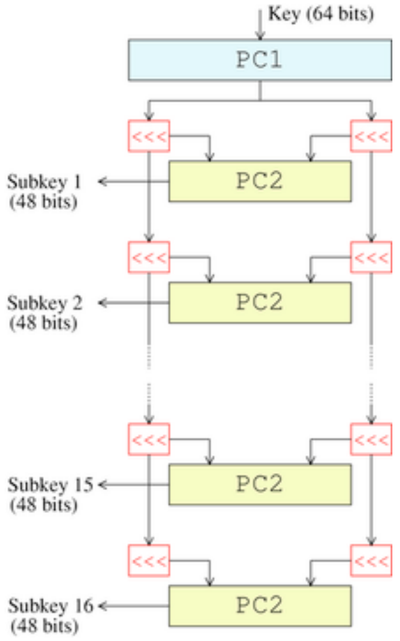
- Block cipher is a family of permutations
  - Indexed by a key (DES = 56 bit key)
  - Ideally: “Pseudo-random permutation (PRP)”

    (i.e., attacker who does not know the key can’t determine whether you’re using a random permutation, or a PRP)

DES: 64-bit Block, 56-bit Key

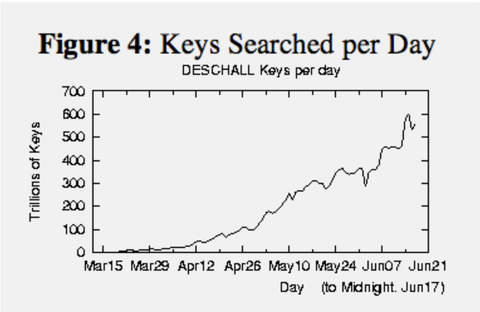


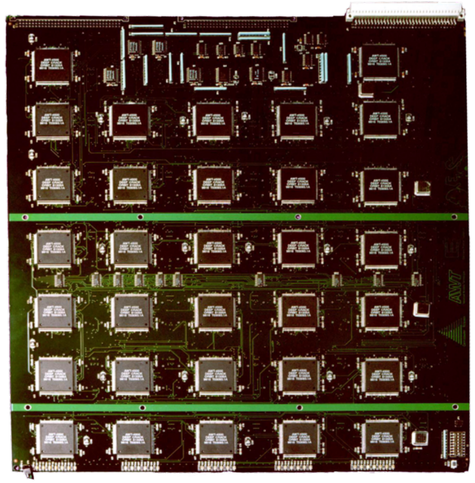




DES

- Some “clever” attacks on DES
  - However: practical weakness = 56 bit key size
  - Practical solution: 3DES (also deprecated)





U.S. Data-Scrambling Code Cracked With Homemade Equipment

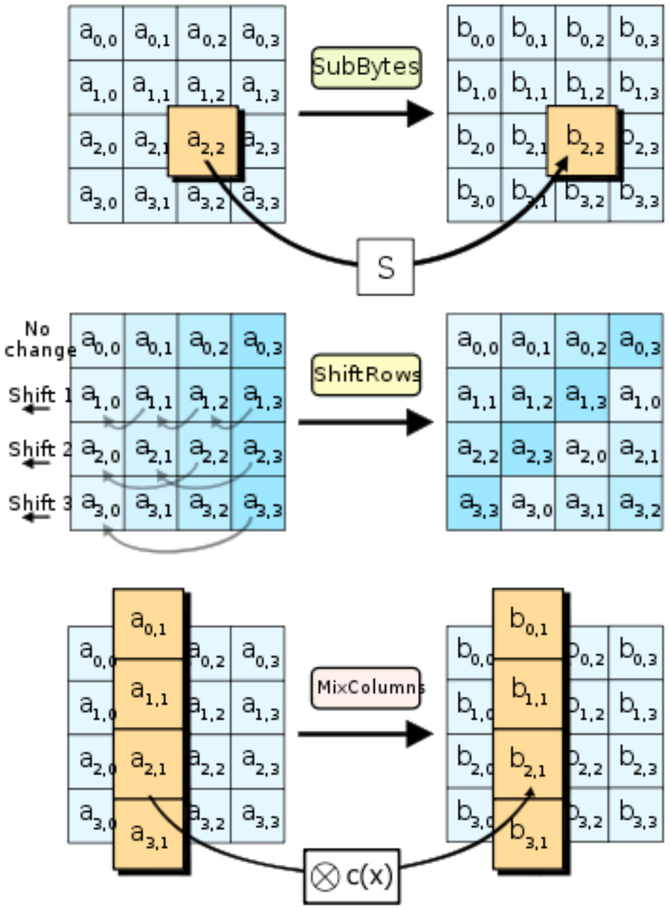
By JOHN MARKOFF

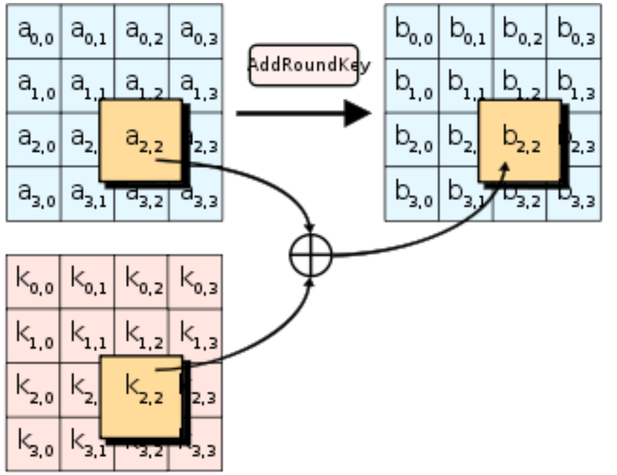
SAN FRANCISCO -- In a 1990s variant of a John Henry-style competition between man and machine, researchers using a homemade supercomputer have cracked the government's standard data-scrambling code in record time -- and have done it by out-calculating a team that had harnessed thousands of computers, including some of the world's most powerful.

AES

- NIST open competition (around 1998). Requirements:
  - Fast in software & hardware
  - Larger block size (128 bit)
  - Longer keys (128/192/256-bit)
  - Reviewed by academics & NSA
- 5 finalists:
  - MARS, RC6, Rijndael, Serpent, and Twofish

AES: 128-bit Block, 128/192/256-bit Key





Using Block Ciphers

- ECB is not semantically secure, hence we use a “mode of operation”
  - e.g., CBC, CTR, CFB, OFB (and others)
- These provide:
  - Security for multi-block messages
  - Randomization (through an Initialization Vector)

Using Block Ciphers

- Obvious (bad) idea:
  - Take every consecutive chunk of plaintext
  - Pass it into the block cipher
  - Concatenate all the output blocks
  - This is called “Electronic Codebook Mode” (ECB)

Using Block Ciphers

- Obvious (bad) idea:
  - Take every consecutive chunk of plaintext
  - Pass it into the block cipher
  - Concatenate all the output blocks
  - This is called “Electronic Codebook Mode” (ECB)
- **What’s the problem with this?**

ECB Mode

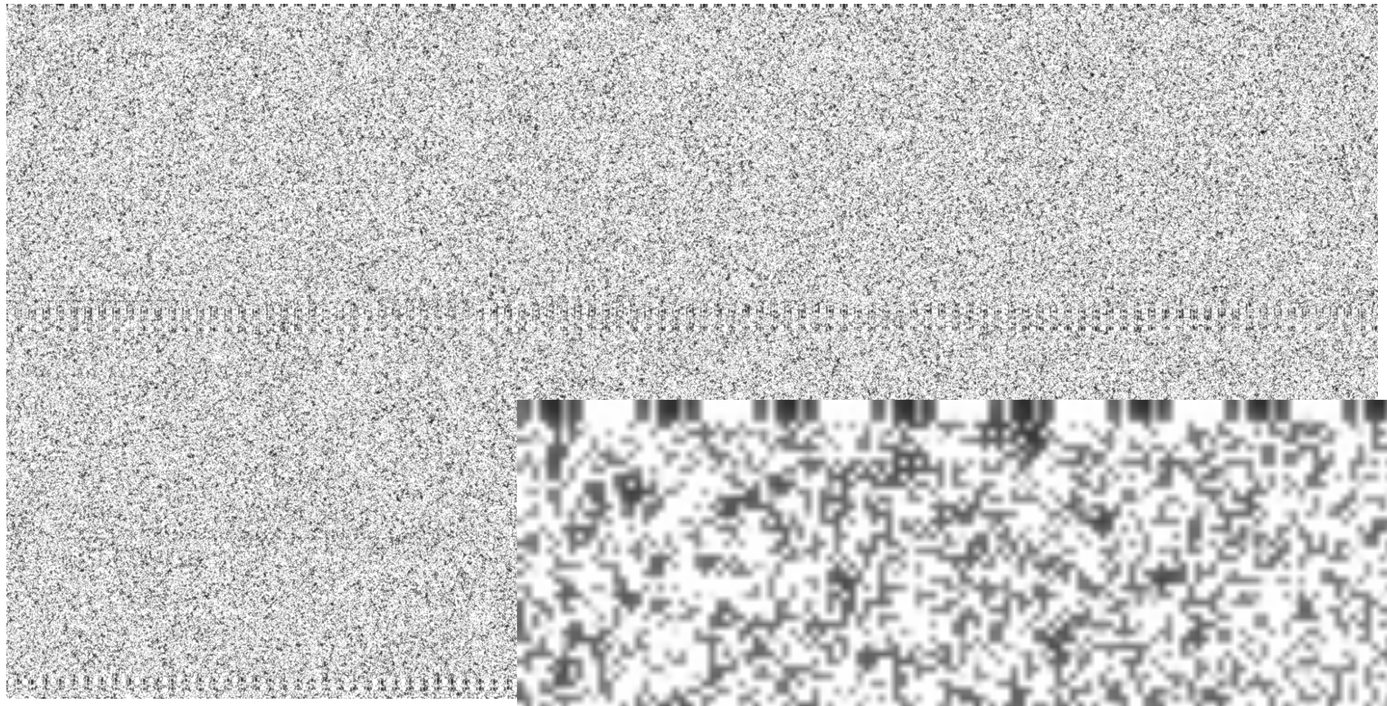
- Problem #1: ECB is deterministic

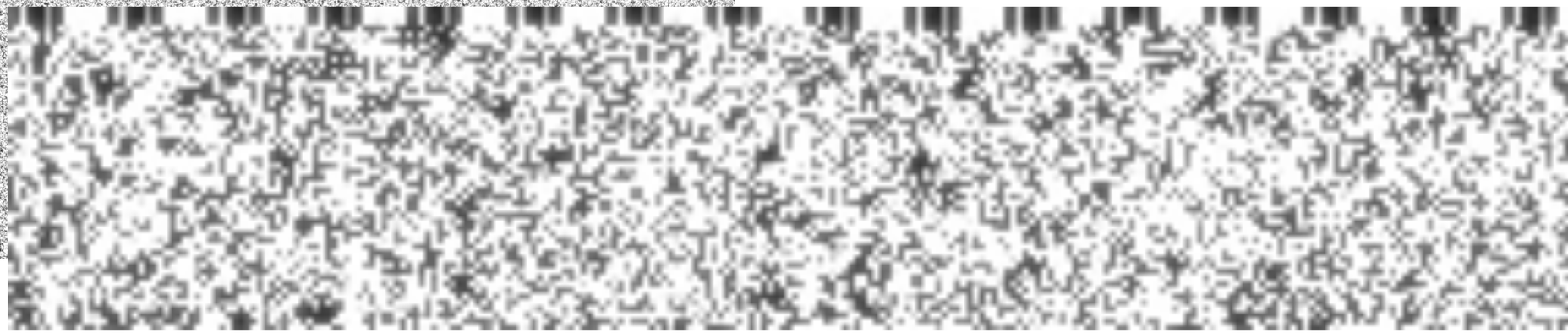
E(Attack Monster)

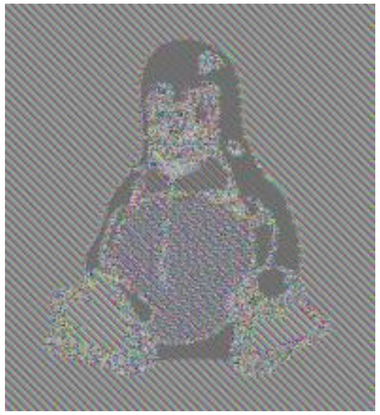
E(Monster Attacks)

ECB Mode

- Problem #2: Leakage of plaintext patterns







Security of Encryption

- Semantic Security
  - Due to Goldwasser & Micali (1980s)
  - Informally: An encryption scheme is secure if adversary who sees ciphertext “learns as much” as adversary who doesn’t see ciphertext.

•Even if adversary can request chosen plaintexts

  - How do we state this formally?

Semantic security

- Semantic Security (IND-CPA)

$M_1, ..., M_n$

$E(k, M_1), ..., E(k, M_n)$

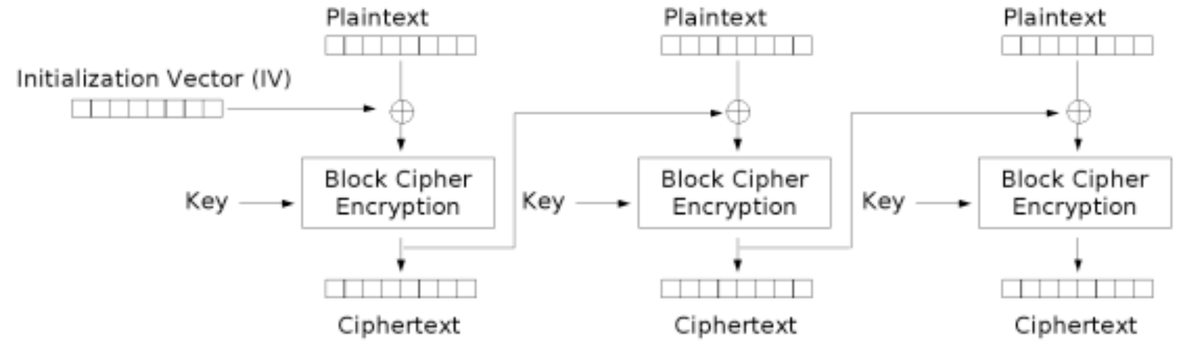
M_0, M_1

$E(M_b)$

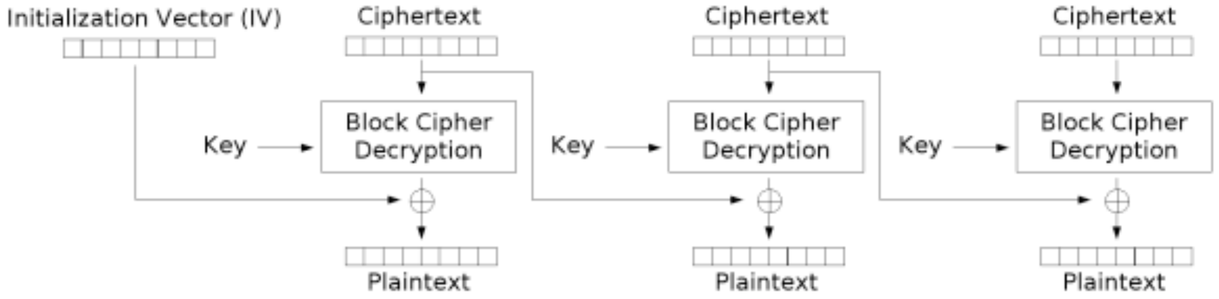
Using Block Ciphers

- ECB is not semantically secure, hence we use a “mode of operation”
  - e.g., CBC, CTR, CFB, OFB (and others)
- These provide:
  - Security for multi-block messages
  - Randomization (through an Initialization Vector)

CBC Mode



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Security of CBC

- Is CBC a secure encryption scheme?
  - Yes, assuming a secure block cipher (and a passive adversary)
  - Correct (random) IV generation
  - Can prove this under assumption that
    block cipher = Pseudo-Random Permutation (PRP)
- Bellare, Desai, Jokipii & Rogaway (2000)
  - Easy to use wrong...
    - Most important: use a unique & random IV!

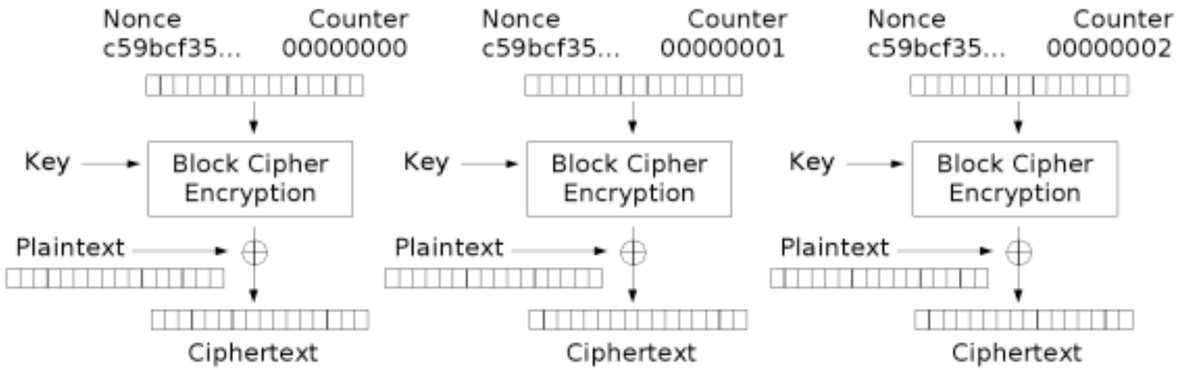
The size of the frame of data to be encrypted or decrypted (i.e. how often a new CBC chain is started) depends on the particular application, and is defined for each in the corresponding format specific books of this specification. Unless otherwise specified, the Initialization Vector used at the beginning of a CBC encryption or decryption chain is a constant, $iv_{0,}$ which is:

$0BA0F8DDFEA61FB3D8DF9F566A050F78_{16}$

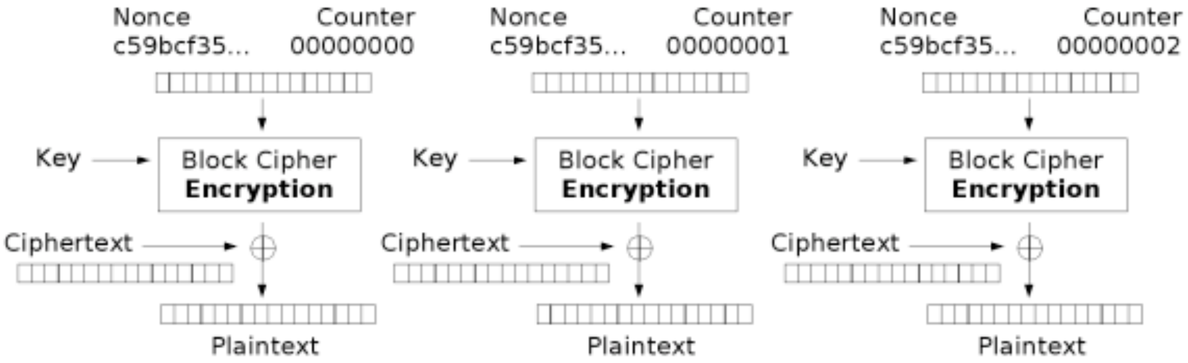
Advanced Access Content System (AACS)

Introduction and Common Cryptographic Elements

CTR Mode



Counter (CTR) mode encryption



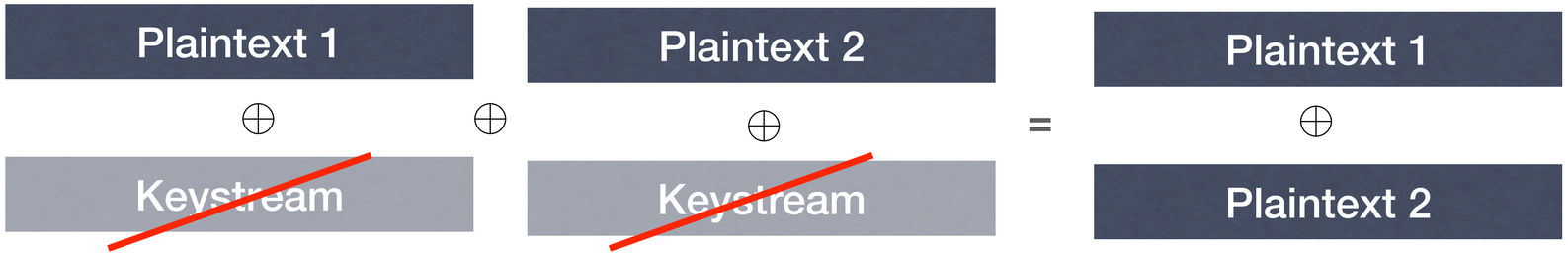
Counter (CTR) mode decryption

CTR (intuition)

- CTR uses the cipher to expand a short cipher key (K) into a long string of pseudorandom output bits
  - This is called a “**keystream**”
  - We then XOR the keystream with a long message (or many messages)
  - This turns a block cipher into a “stream cipher”

Security of CTR

- Yes, assuming secure block cipher (PRP)
- However, counter range must never be re-used



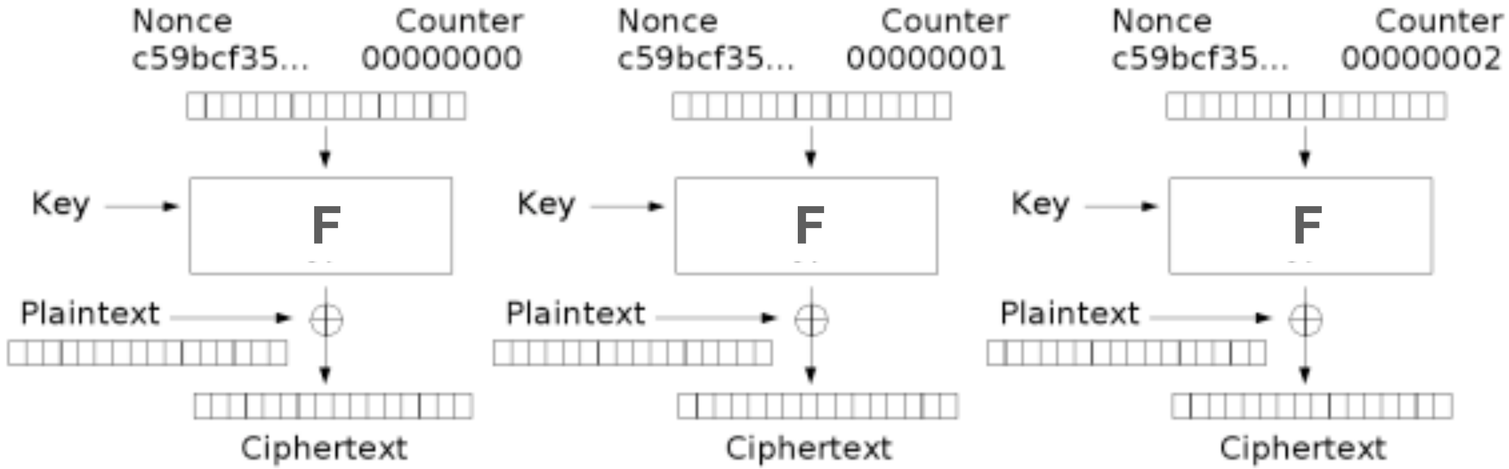
- Similar example: MS Word 2003
  - (they used RC4, but same problem)

CTR with other functions

- We've been assuming a cipher that is invertible (a permutation/block cipher)
- **Observation:** CTR never uses the "Decipher" (invert) algorithm of the cipher
  - So what if we don't use a block cipher at all?

CTR with other functions

- We've been assuming a cipher that is invertible (a permutation/block cipher)
- **Observation:** CTR never uses the "Decipher" (invert) algorithm of the cipher
  - So what if we don't use a block cipher at all?



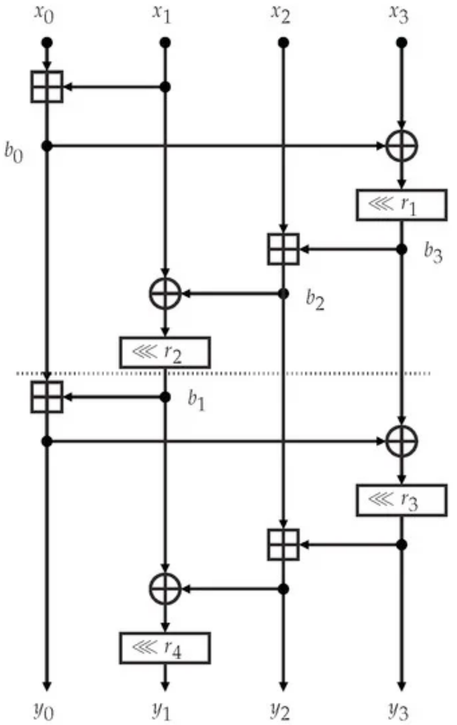
Counter (CTR) mode encryption

ChaCha20

- An example is the ChaCha20 stream cipher
  - Invented by Daniel J. Bernstein (DJB)
- ChaCha20 has one function:

$$F : \{0,1\}^{256} \times \{0,1\}^{512} \rightarrow \{0,1\}^{512}$$

key — counter (nonce) — keystream



- Note: ChaCha is <u>not</u> a permutation! It is not invertible and may output the same value on two different inputs.
- <u>ChaCha20 is faster in software than AES, unless you have hardware support</u>

ChaCha20

```c
#define ROTL(a,b) (((a) << (b)) | ((a) >> (32 - (b))))
#define QR(a, b, c, d) (            \
    a += b,  d ^= a,  d = ROTL(d,16),   \
    c += d,  b ^= c,  b = ROTL(b,12),   \
    a += b,  d ^= a,  d = ROTL(d, 8),   \
    c += d,  b ^= c,  b = ROTL(b, 7))
#define ROUNDS 20

void chacha_block(uint32_t out[16], uint32_t const in[16])
{
    int i;
    uint32_t x[16];

    for (i = 0; i < 16; ++i)
        x[i] = in[i];
    // 10 loops × 2 rounds/loop = 20 rounds
    for (i = 0; i < ROUNDS; i += 2) {
        // Odd round
        QR(x[0], x[4], x[ 8], x[12]); // column 0
        QR(x[1], x[5], x[ 9], x[13]); // column 1
        QR(x[2], x[6], x[10], x[14]); // column 2
        QR(x[3], x[7], x[11], x[15]); // column 3
        // Even round
        QR(x[0], x[5], x[10], x[15]); // diagonal 1 (main diagonal)
        QR(x[1], x[6], x[11], x[12]); // diagonal 2
        QR(x[2], x[7], x[ 8], x[13]); // diagonal 3
        QR(x[3], x[4], x[ 9], x[14]); // diagonal 4
    }
    for (i = 0; i < 16; ++i)
        out[i] = x[i] + in[i];
}
```

Point of order

- Proofs of security:
  - We don’t know how to prove that DES or AES (resp. ChaCha20) are secure block ciphers (resp. Secure pseudorandom function)
  - But if we assume that the block ciphers are secure PRPs then:

•We can prove that CBC & CTR & OFB & CFB etc.
are secure encryption modes against a passive adversary

http://www.cs.ucdavis.edu/~rogaway/papers/sym-enc-abstract.html

Malleability

- The ability to modify a ciphertext
  - Such that the plaintext is meaningfully altered
  - CTR Mode (bad)
  - CBC Mode (somewhat bad)

Hash Functions

- Convert variable-length string to small “tag”
  - Hash tables
  - Signatures
  - Software checksums
  - MAC functions (HMAC)
  - Encryption (OAEP)