CSE508 Network Security

Stony Brook University

2024-02-27

Symmetric Key Cryptography

Michalis Polychronakis

Stony Brook University

Cryptography

Goals

Confidentiality

Keep content secret from all but authorized entities

Integrity

Protect content from unauthorized alteration

Authentication

Confirm the identity of communicating entities or data

Non-repudiation

Prevent entities from denying previous commitments or actions

Basic Terminology

Plaintext: `top secret message`

Ciphertext: `eza dpncpe xpddlrp`

Cipher: algorithm for transforming plaintext to ciphertext *(encryption)* and back *(decryption)*

Key: (usually secret) information used in a cipher

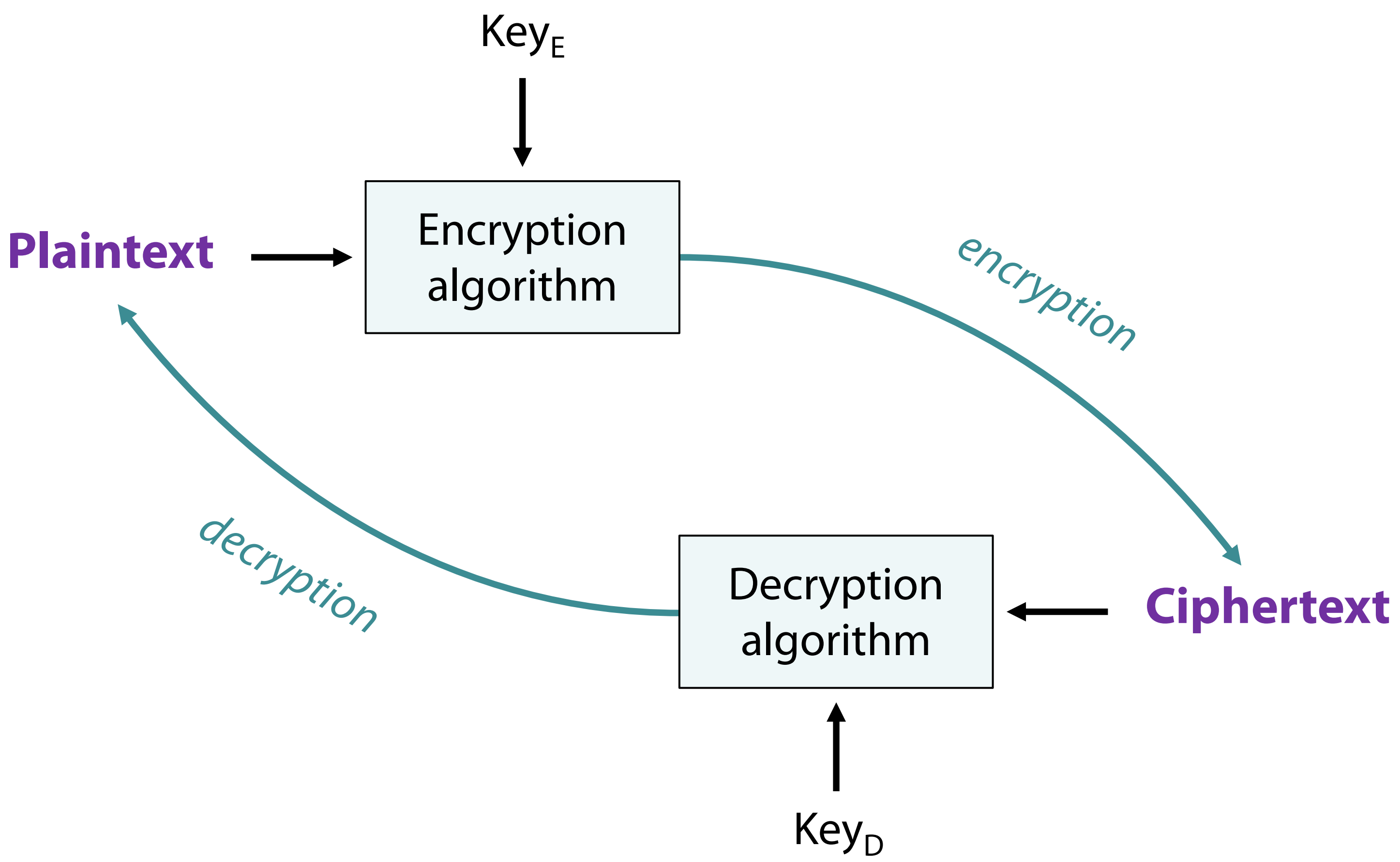
Known to sender, receiver, or both

Cryptanalysis (codebreaking): the study of methods for deciphering ciphertext without knowing the secret key

Cryptology: the broader field of "information hiding"

Cryptography, cryptanalysis, steganography, …

Plaintext vs. Ciphertext



Cryptosystem

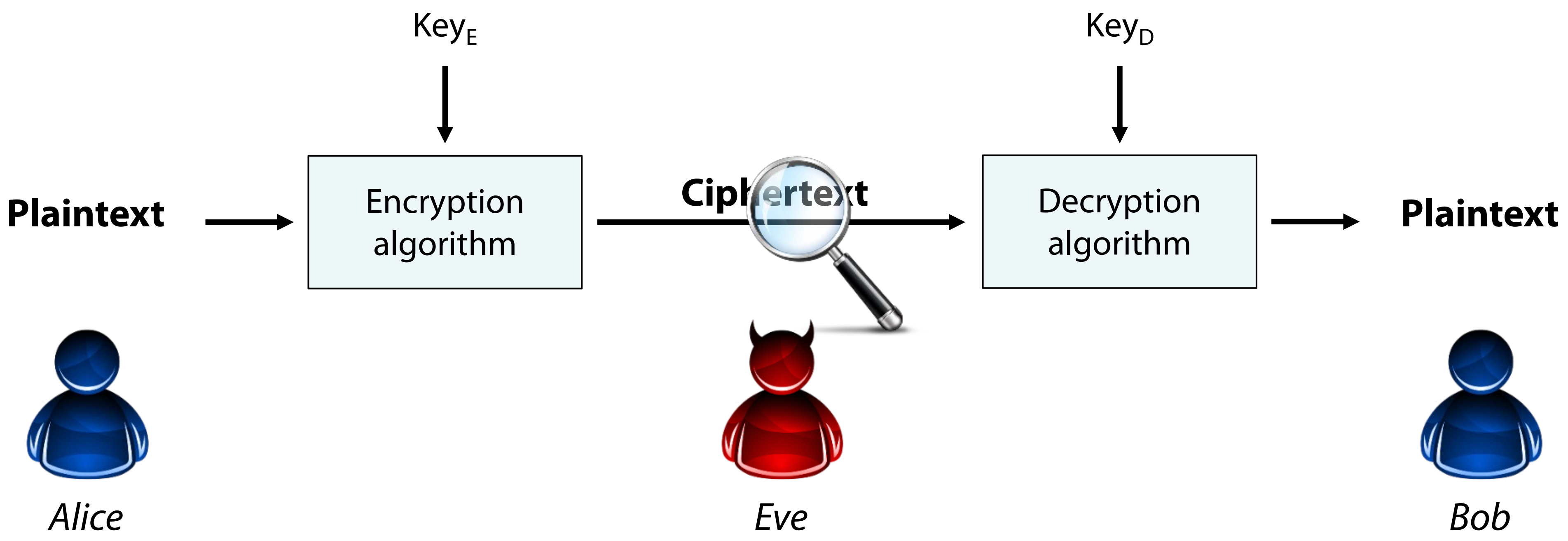
A suite of cryptographic algorithms that take a key and convert between plaintext and ciphertext

Main components

- Plaintext space: set $\boldsymbol{P}$ of possible plaintexts
- Ciphertext space: set $\boldsymbol{C}$ of possible ciphertexts
- Key space: set $\boldsymbol{K}$ of encryption/decryption keys
- Encryption algorithm: $\boldsymbol{E} : \boldsymbol{P} \times \boldsymbol{K} \rightarrow \boldsymbol{C}$
- Decryption algorithm: $\boldsymbol{D} : \boldsymbol{C} \times \boldsymbol{K} \rightarrow \boldsymbol{P}$

$$\forall p \in P, k \in K : D(E(p, k), k) = p$$

Basic Threat Model



Main Cryptographic Function Types

Hash functions: *no key*

- Input of arbitrary length is transformed to a fixed-length value
- One-way function: hard to reverse

Secret (symmetric) key functions: *one key*

- Shared secret key is used for both encryption and decryption

Public (asymmetric) key functions: *two keys*

- *Key pair:* public key is known, private key is always kept secret
- Encrypt with public key and decrypt with private key
- Encrypt with private key and decrypt with public key

Randomness

Cryptosystems rely on the ability to generate random numbers

- Both unpredictable and secret from any adversary

Random number generators (RNG) ➔ true randomness

- Transfer entropy from the analog world to the digital world: temperature, mouse movements, keyboard timings, network activity, I/O devices, …
- Obtaining true random values is usually expensive and slow ➔ impractical for direct use in cryptographic algorithms

Pseudorandom number generators (PRNG) ➔ pseudorandomness

- Deterministic algorithm that takes as input a few true random bits (the *seed*) and produces a larger amount of (artificially) random output

Pseudorandom Number Generators (PRNG)

Cryptographic (CSPRNG) vs. Non-Cryptographic

Non-crypto PRNGs focus on producing a uniform distribution (e.g., for games) but are insecure: their output may be predictable (!)

Key properties of CSPRNGs

Unpredictable: An attacker who does not know the seed cannot predict the output

Deterministic: running the algorithm with the same seed generates the same output

Linux default CSPRNG: `/dev/urandom`

Randomness mixing

Multiple sources of randomness are mixed in an *entropy pool* → resilience against potentially compromised individual entropy sources

The OS seeds the entropy pool at boot time

Then periodically updates it and re-seeds the CSPRNG with new randomness

https://blog.cloudflare.com/lavarand-in-production-the-nitty-gritty-technical-details/

Kerckhoffs's Principle

A cryptosystem should be secure even if everything about the system, except the key, is public knowledge

The security of the system must rest entirely on the secrecy of the key

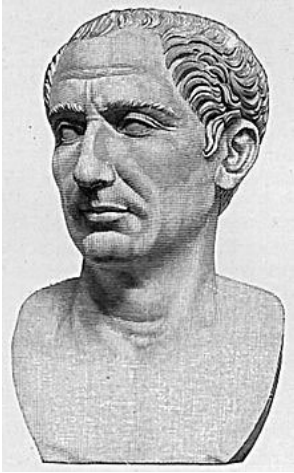
- Only brute force attacks should be possible (otherwise the algorithm is broken)

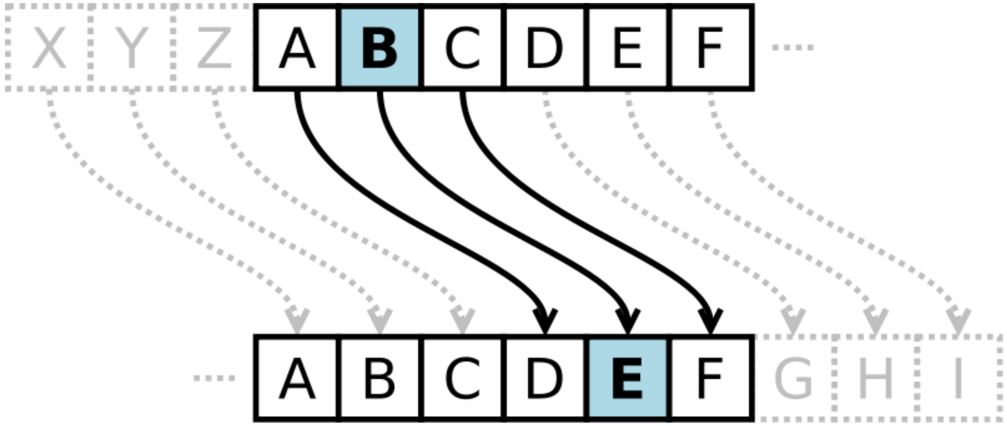
Contrast with *security by obscurity*: every implementation secret creates a potential failure point

- The internals of widely used secret algorithms will eventually become known (reverse engineered, leaked, stolen, …)
- Difficult to deploy a new algorithm if an old one is compromised

A public implementation enables (and encourages) scrutiny by experts

Caesar Cipher






```
Ciphertext: WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ
Plaintext:  the quick brown fox jumps over the lazy dog
```

Shift by x (e.g., ROT-13)

Monoalphabetic substitution

Shift Ciphers

Plaintext space: $\boldsymbol{P} = \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots, \mathbf{Z}\}$

Ciphertext space: $\boldsymbol{C} = \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots, \mathbf{Z}\}$

Key space: $\boldsymbol{K} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots, \mathbf{25}\}$

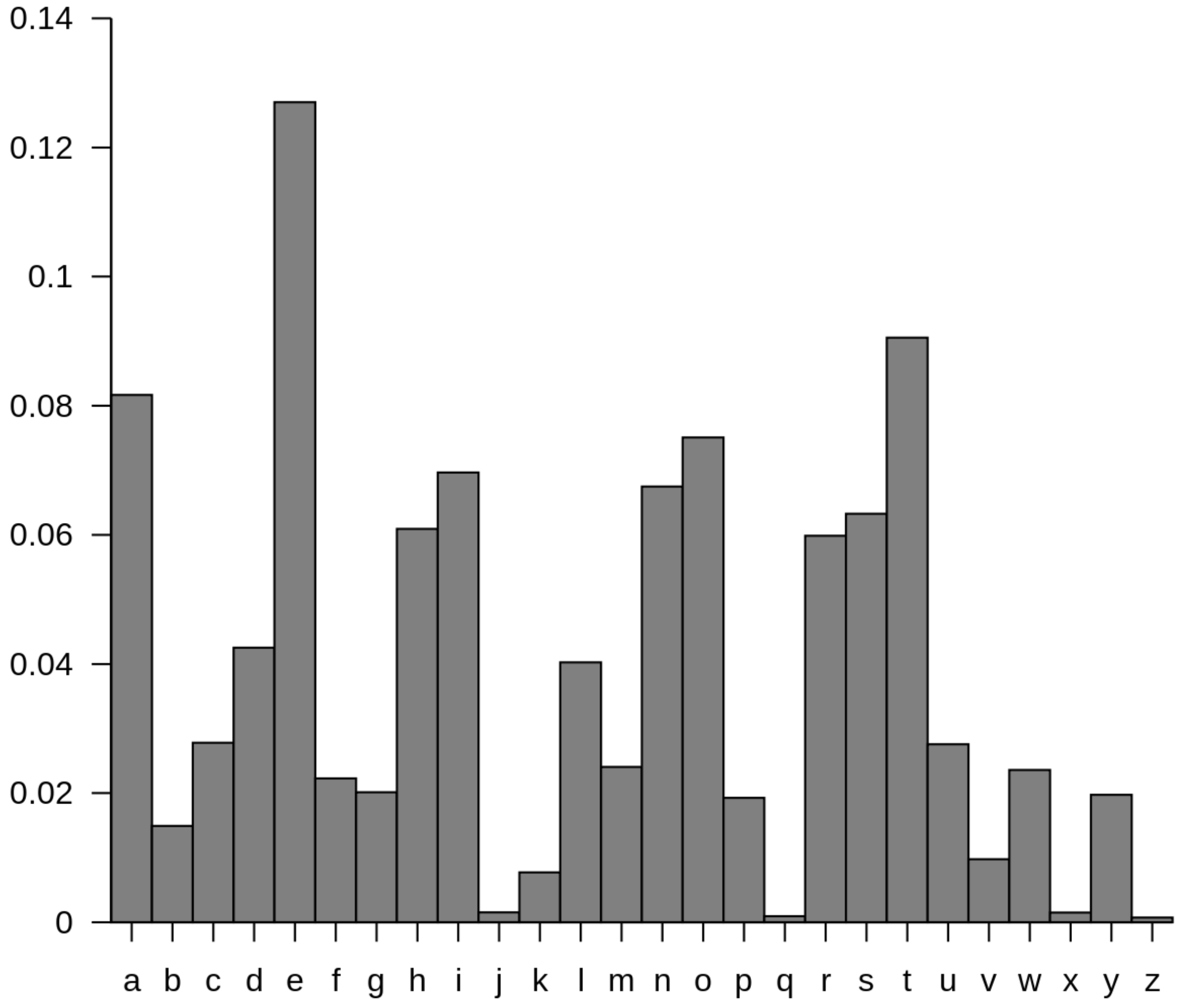
Encryption algorithm: $\boldsymbol{E(x, k) = (x + k)} \textbf{ mod 26}$

Decryption algorithm: $\boldsymbol{D(x, k) = (x - k)} \textbf{ mod 26}$

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Caesar Cipher: $k = 3$

Easy to break using frequency analysis



Distribution of letters in a typical sample of English language text

https://en.wikipedia.org/wiki/Frequency_analysis

Vigenère Cipher

Plaintext: ATTACKATDAWN

Key: **LEMON**LEMONLE

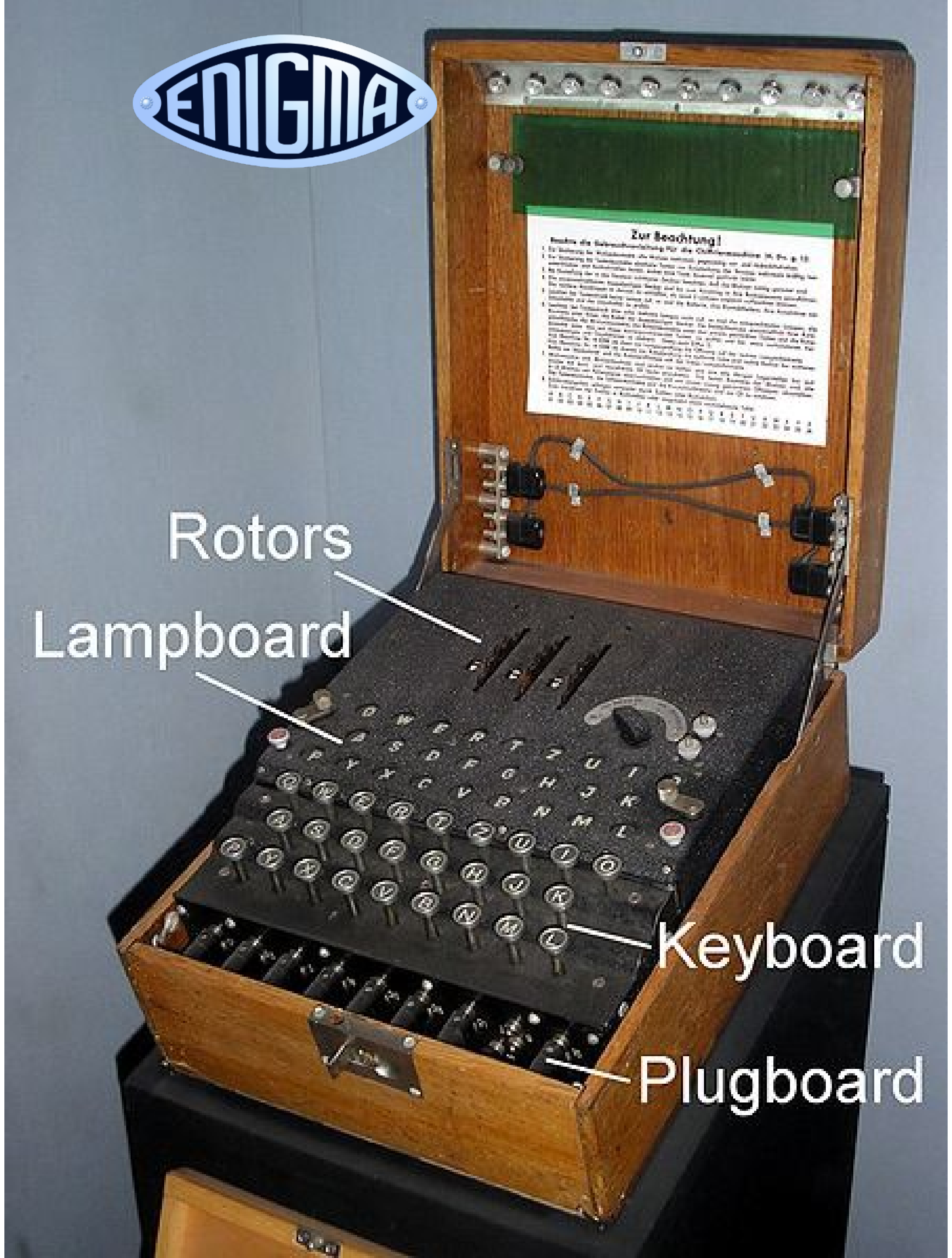
Ciphertext: LXFOPVEFRNHR

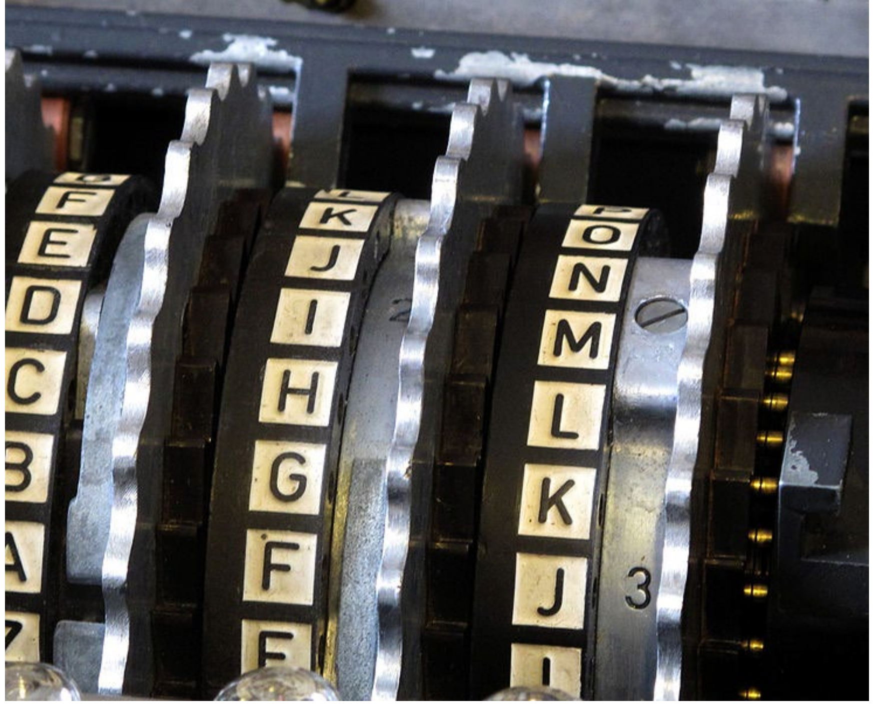
Polyalphabetic substitution

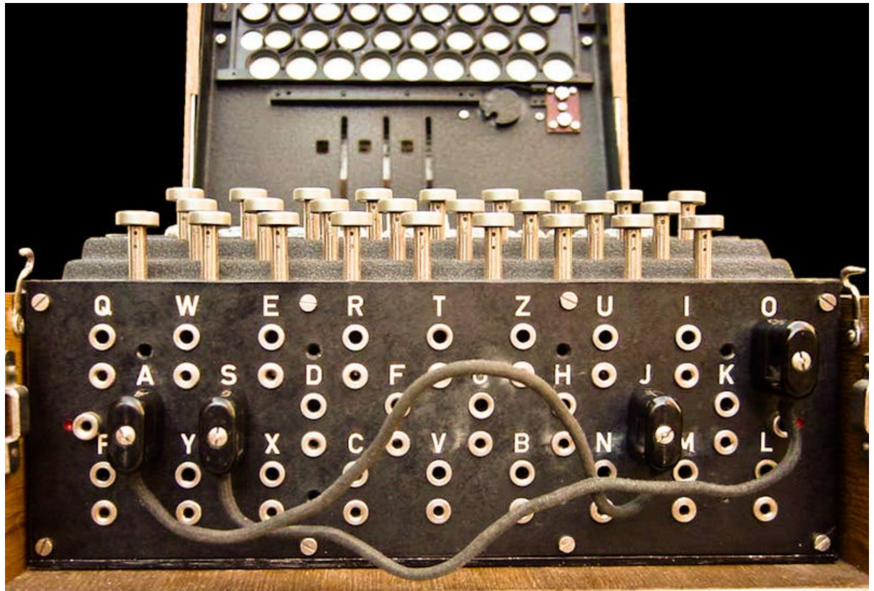
Successive shift ciphers with different shift values depending on a key

Defeats simple frequency analysis, but still easily breakable

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y


ENIGMA
Rotors
Lampboard
Keyboard
Plugboard




Q W E R T Z U I O

Properties of a Good Cryptosystem

Given a ciphertext, an adversary should not be able to recover the original message

- Enumerating all possible keys must be infeasible
- There should be no way to produce plaintext from ciphertext without the key

The ciphertext must be indistinguishable from true random values

- Given a ciphertext, the probability of any possible plaintext being the one that is encrypted should be the same

Cryptographic algorithms should be computationally efficient

- Most practical uses require fast encryption, decryption, hashing
- There are exceptions: deliberately slow password-based key derivation functions for hindering brute force/dictionary attacks *(future lecture)*

Basic Attack Models

Known Ciphertext: attackers have access to only a set of ciphertexts

In practice, some information about the plaintext might be available: language, character distribution, protocol fields, type of content, …

Brute force frequency analysis, probable word analysis, informed guessing, …

Known Plaintext: attackers have access to both the plaintext and its corresponding ciphertext

Passive attacker: obtains at least one pair of plaintext and ciphertext

Even partial mappings can be enough

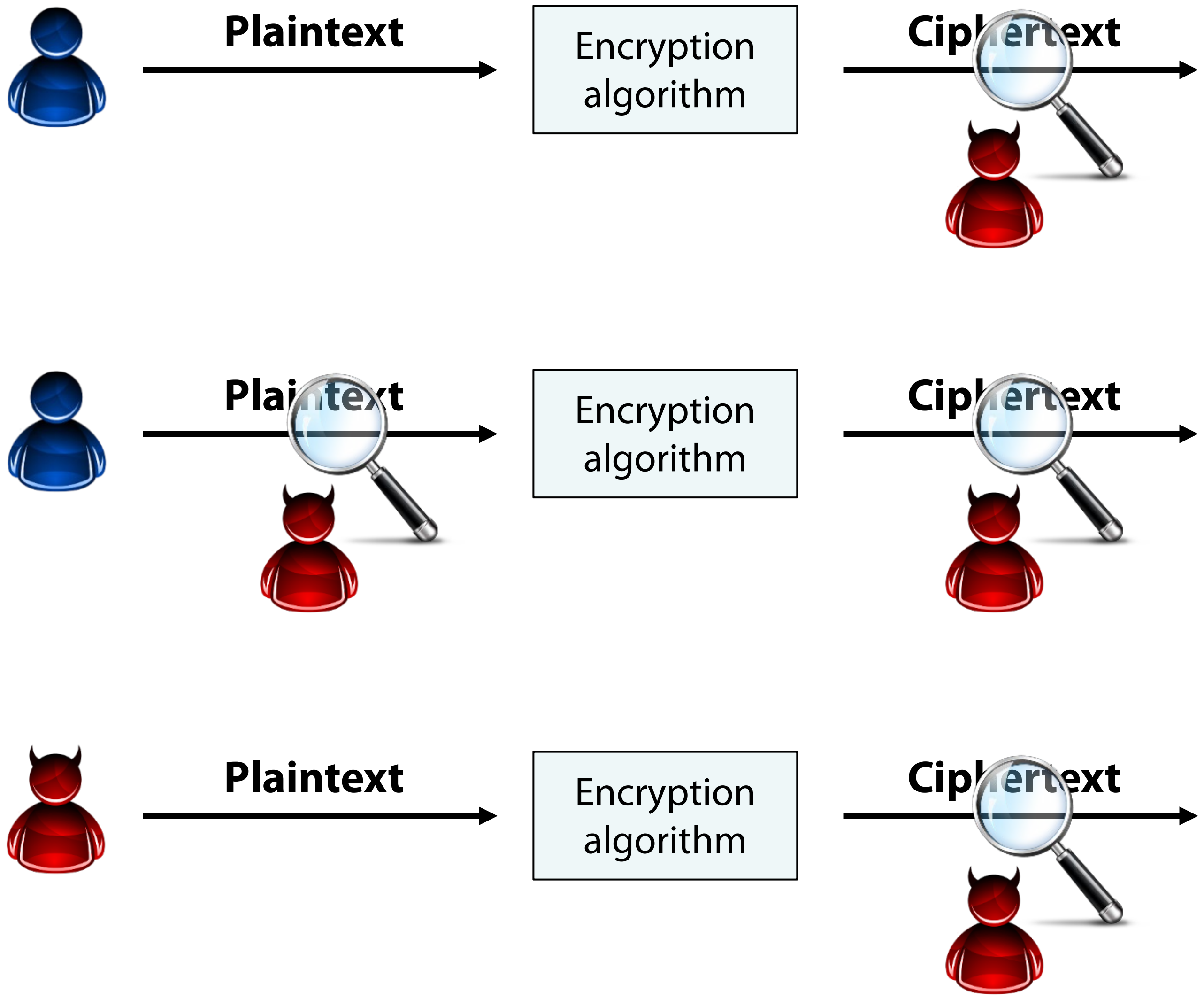
Chosen Plaintext: attackers can obtain the ciphertexts of arbitrary plaintexts of their own choosing

Active attacker: has access to an *encryption oracle*

Known Ciphertext

Known Plaintext

Chosen Plaintext



Computational Difficulty

Modern cryptography: seek formal guarantees about the "strength" of encryption schemes

- Codes, secret writing, and other older encryption schemes were ad hoc and eventually broken

Information-theoretic security

- Unbreakable even with unlimited computing power: *there is simply not enough information*
- Not possible if the key is shorter than the message size ➔ impractical for most uses

Computational security

- Can be broken with enough computation, but *not in a reasonable amount of time*
- Rely on *computationally hard* problems: easy to compute but hard to invert in *polynomial* time (integer factorization, discrete logarithm, …)
- Assume *computationally limited adversaries* ➔ frustrate exhaustive enumeration

One-time Pad

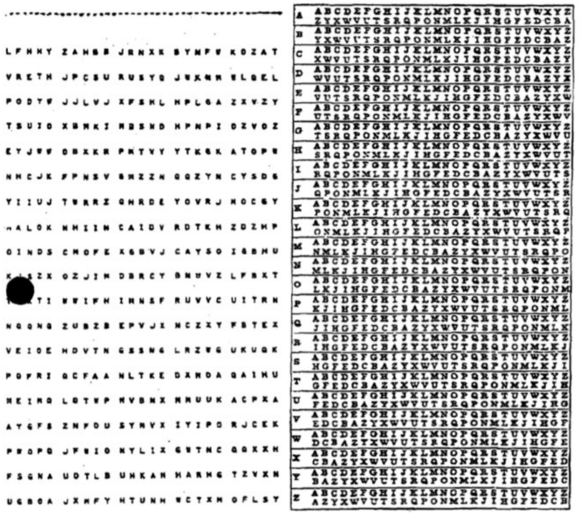
XOR plaintext with a *keystream*

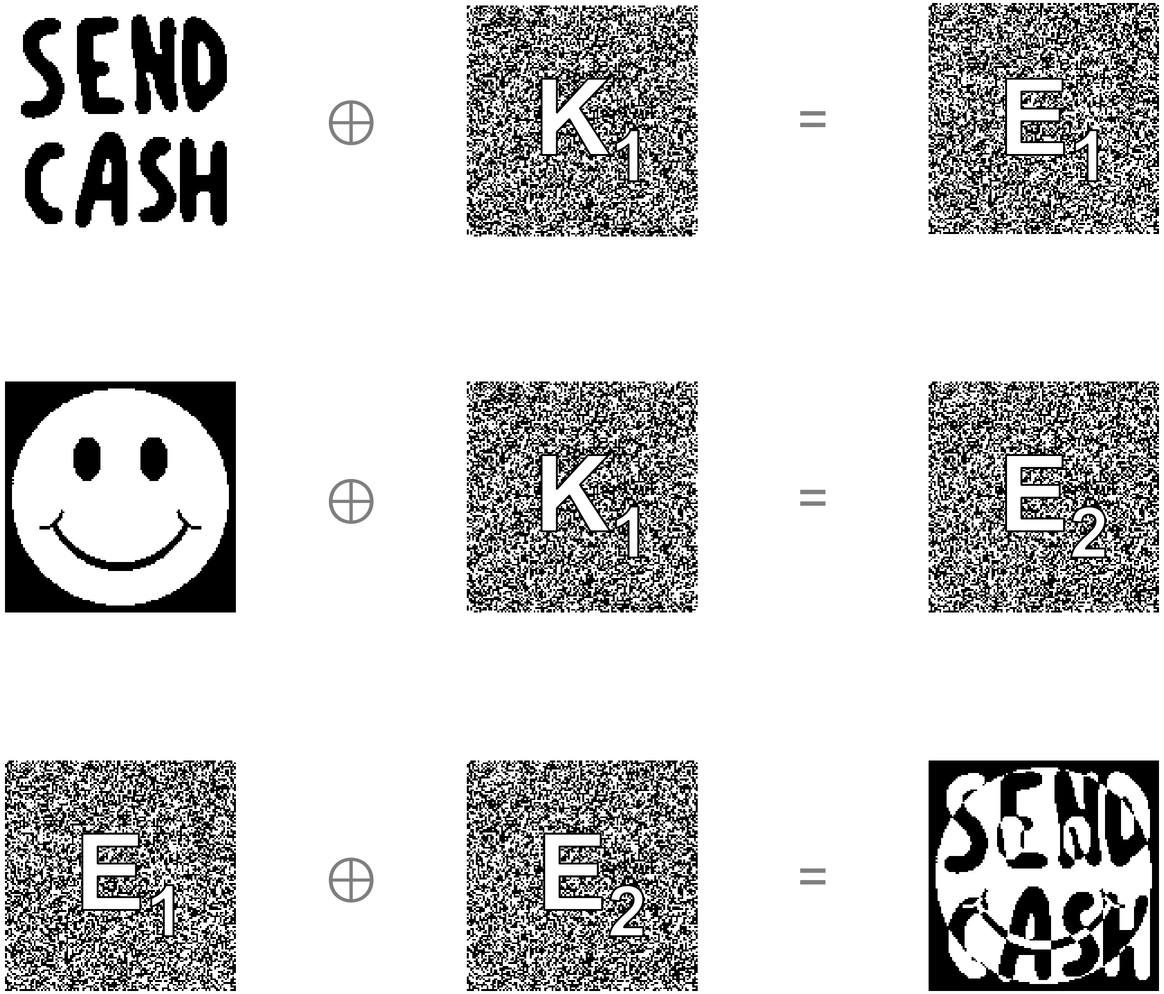
- 1882: Frank Miller [Bellovin '11]
- 1917: Vernam/Mauborgne cipher

Information-theoretically secure against ciphertext-only attacks (Shannon 1949)

The keystream must be

- Truly random
- As long as the plaintext
- Kept completely secret
- Used only once (!)




SEND CASH
⊕
K_1
=
E_1
⊕
K_1
=
E_2
E_1
⊕
E_2
=
SEND CASH

One-time Pad

Plaintext space:	*all n-bit sequences*
Ciphertext space:	*all n-bit sequences*
Key space:	*all n-bit sequences*
Encryption algorithm:	$E(x, k) = x \oplus k$
Decryption algorithm:	$D(x, k) = x \oplus k$

Advantages

Easy to compute: simple XOR operation (bit by bit)

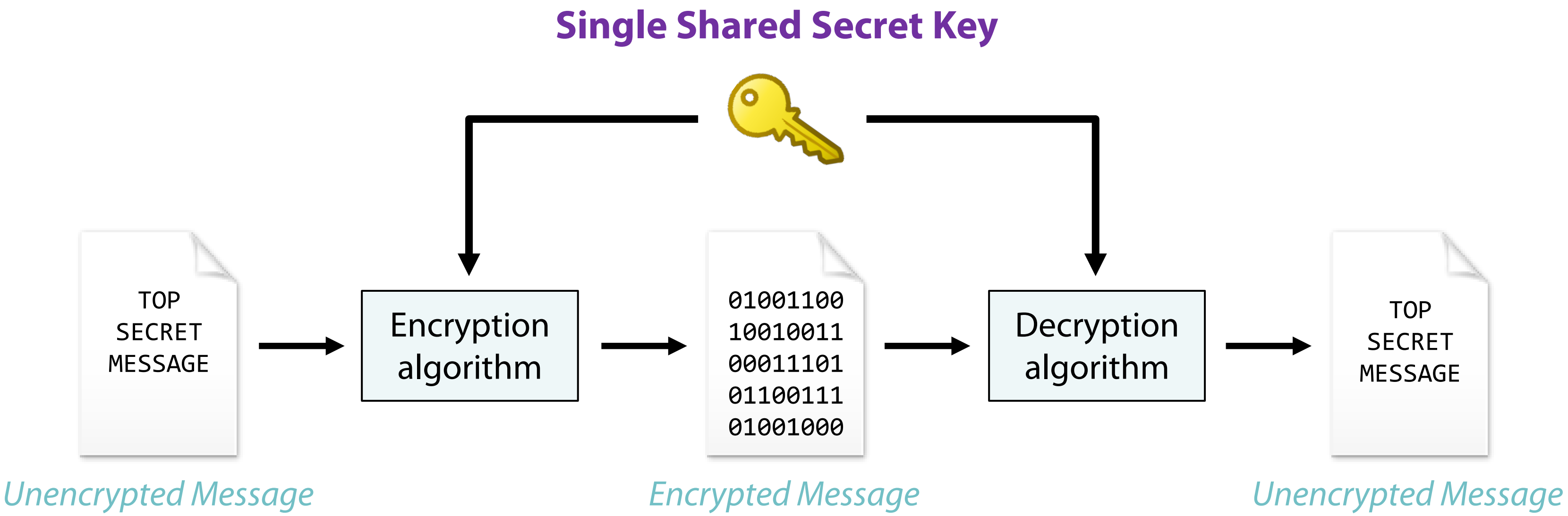
Impossible to break: trying all keys simply yields all possible plaintexts

Disadvantages

Key size: must be as long as the plaintext

Key distribution: how can the sender provide the key to the receiver securely?

Symmetric Key Cryptography



Pros: Fast
Short keys
Well known
Simple key generation

Cons: Secrecy of keys
Number of keys
Management of keys
$n(n-1)/2$ keys needed for n parties

Secret Key Generation

Cryptographic keys should be randomly generated

They have to be completely unpredictable – three main ways:

Random

Symmetric key: use a CSPRNG to generate n bits for an n-bit key

Asymmetric key pair: CSPRNG-generated seed input to a *key generation algorithm*

Based on a password

Password-based Key Derivation Function (PBKDF): transforms a password into a key

Key agreement protocol

Series of messages exchanged between two or more parties that ends with the establishment of a shared key

Protecting Secret Keys

Secret keys must be kept protected, yet available to be used

Generate a key from a user-supplied password (PBKDF)

- Benefit: no key is stored at all – exists only in the user's brain (or post-it note)
- Drawback: users pick weak passwords → an attacker who captures an encrypted message can attempt to decrypt it by guessing/brute-forcing the password (future lecture)

Key wrapping: encrypting the key using a second key

- Now we need to protect the second key…
- Practical compromise: generate the second key from a password using a PBKDF
- Example: key phrase to protect SSH private keys

Specialized hardware

- Benefit: keys remain safe even if the host system is fully compromised
- Drawback: slow, expensive, inconvenient
- Smartcards, security tokens: used in addition to (or in place of) a password
- Hardware security modules (HSM): generate and store keys; perform cryptographic operations

Block Ciphers

Process one block at a time

Combination of substitution and transposition (permutation) techniques

Examples:

DES (Data Encryption Standard), AES (Advanced Encryption Standard) – replaced DES, …

Stream Ciphers

Process one bit or byte at a time

Plaintext is combined (XOR) with a *pseudorandom* keystream *(this is NOT the same as an one-time pad)*

Synchronous vs. asynchronous (self-synchronizing)

Examples:

RC4, any block cipher in OFB or CTR mode, …

Block Ciphers

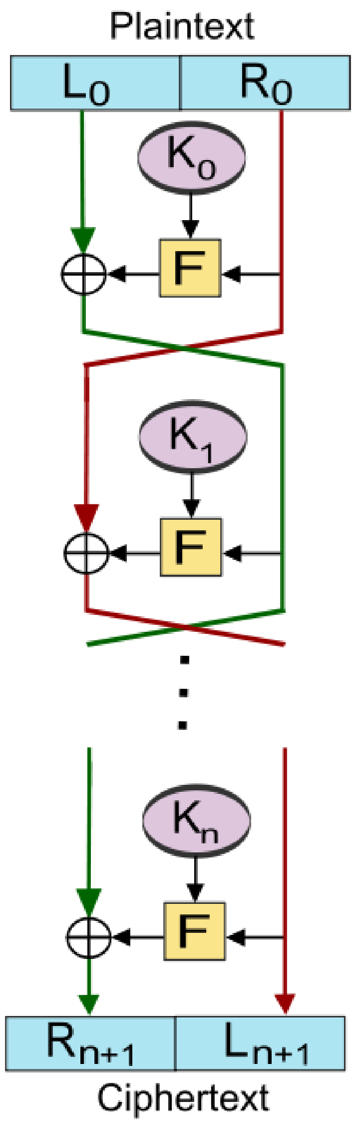
Multiple rounds of substitution, permutation, …

Confusion: each character of the ciphertext should depend on several parts of the key

Diffusion: changing a plaintext character should result in several changed ciphertext characters

	DES	AES
Key length	56 bits	128, 192, 256 bits
Block size	64 bits	128 bits
Rounds	16	10, 12, 14
Construction	Substitution, permutation	Substitution, permutation, mixing, addition
Developed	1977	1998
Status	Broken	OK (for now)

DES rounds



Modes of Operation

Direct use of block ciphers is not very useful

- Enemy can build a "code book" of plaintext/ciphertext equivalents
- Message length should be multiple of the cipher block size

How to repeatedly apply a block cipher to securely encrypt/decrypt inputs of arbitrary length?

Standard modes:

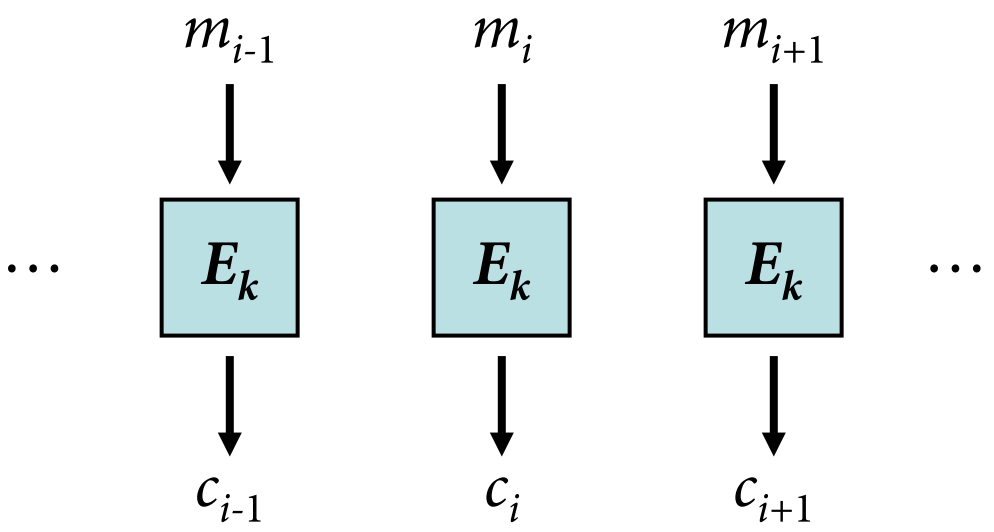
- **ECB:** Electronic Code Book
- **CBC:** Cipher Block Chaining
- **CFB:** Cipher Feedback
- **OFB:** Output Feedback
- **CTR:** Counter
- **GCM:** Galois/Counter Mode ("authenticated encryption" – next lecture)

Use this (unless there are specific important reasons not to)

ECB: Electronic Code Book Mode

Direct use of the block cipher

- Each block is encrypted independently ➔ parallelizable
- No chaining, no error propagation



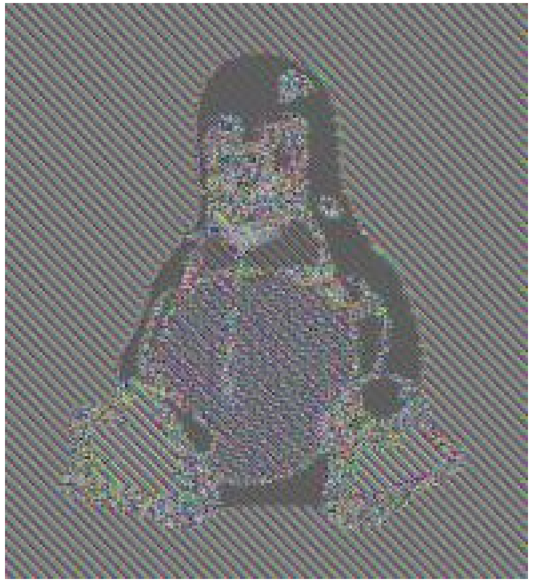
Problem: if $m_i = m_j$ then $c_i = c_j$

ECB: Electronic Code Book Mode

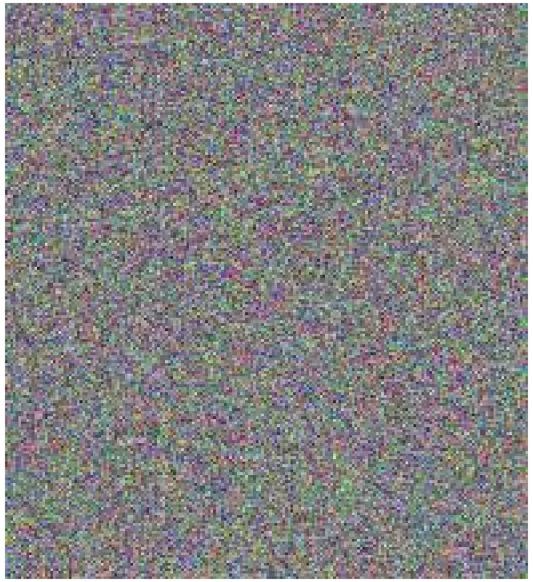
Data patterns may remain visible

Susceptible to replay attacks, block insertion/deletion

Plaintext



ECB Mode Encryption

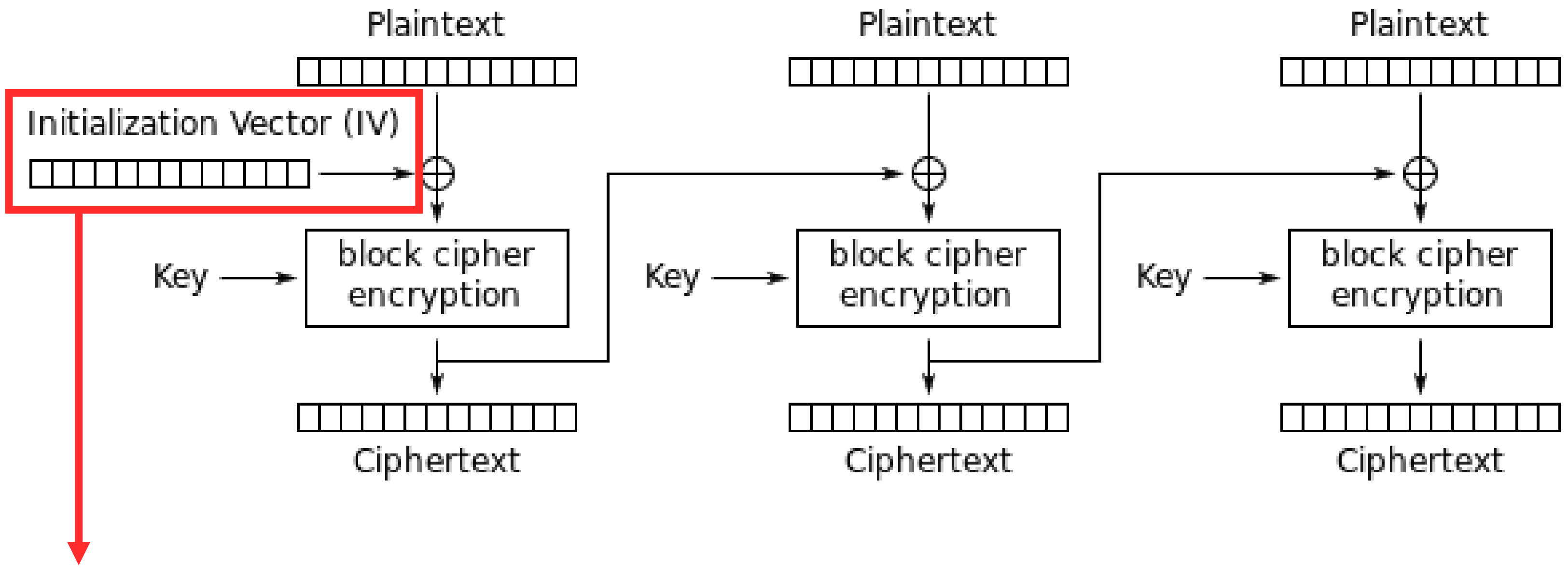


CBC/Other Modes

CBC: Cipher Block Chaining Mode

Each plaintext block is XOR'ed with the previous ciphertext block before being encrypted ➔ obscures any output patterns

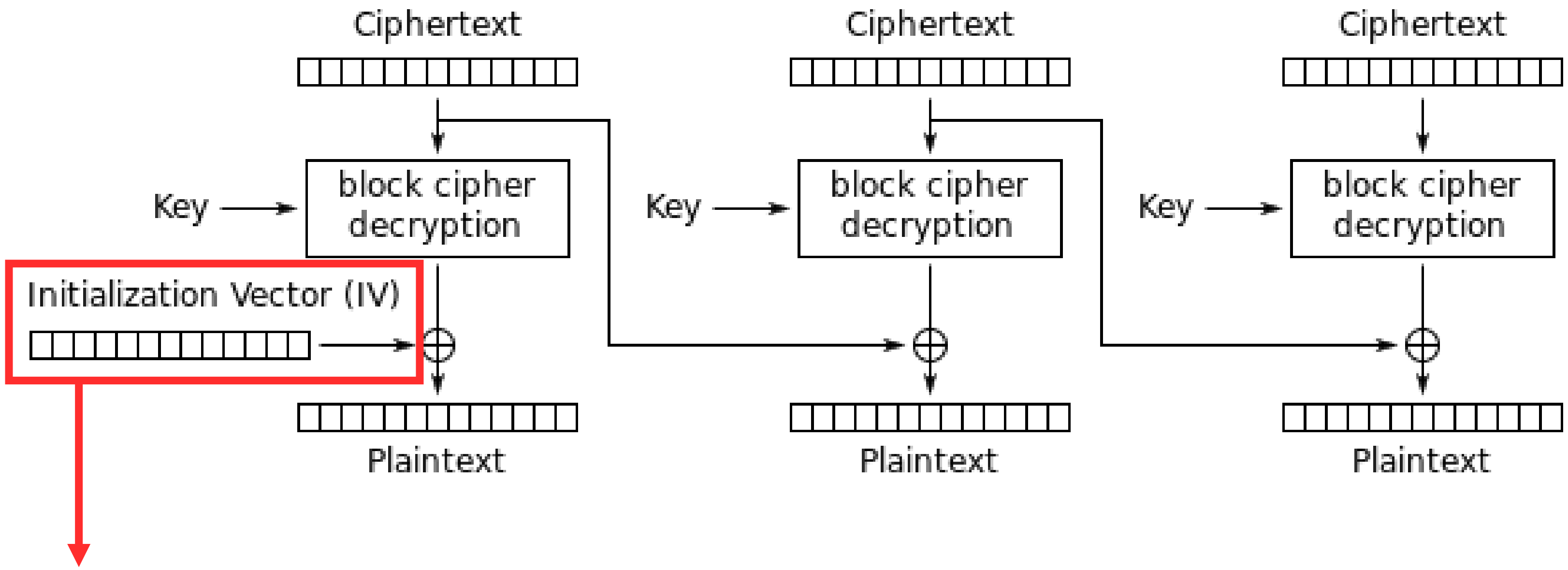
Sequential process (non-parallelizable)



Ensures that no messages have the same beginning ➔ **Must be random! Must never be reused!**

CBC: Decryption

An error in a transmitted ciphertext block also affects its following block (but not the subsequent ones)

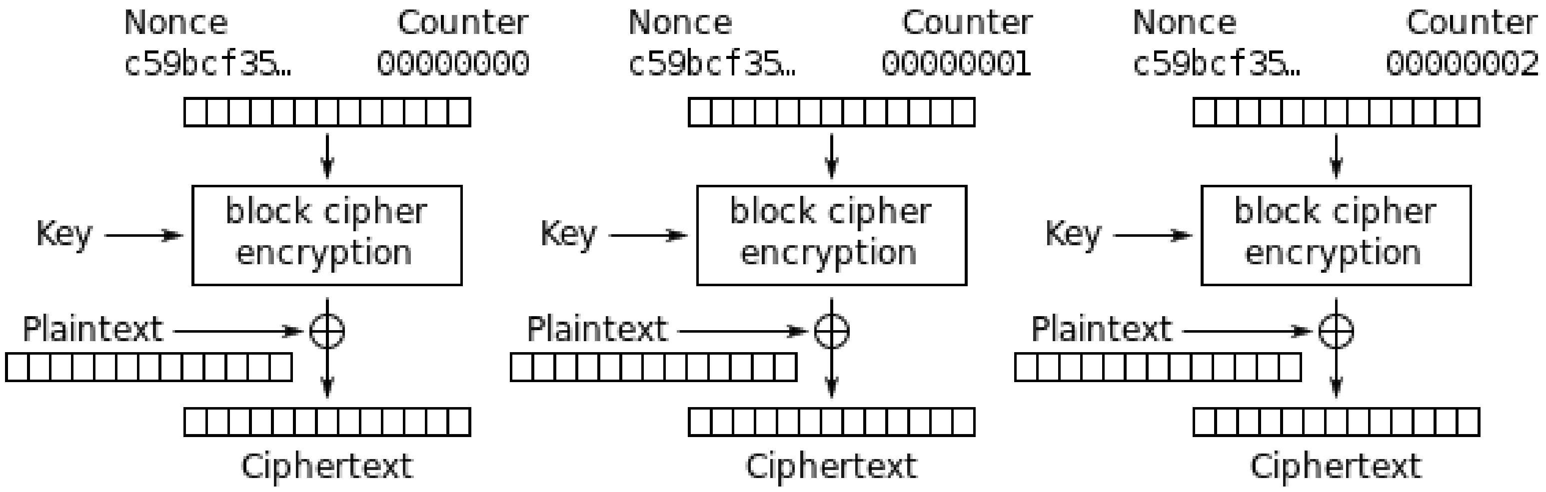


Both parties must use the same IV: can be transmitted along with the message (doesn't have to be secret)

CTR: Counter Mode

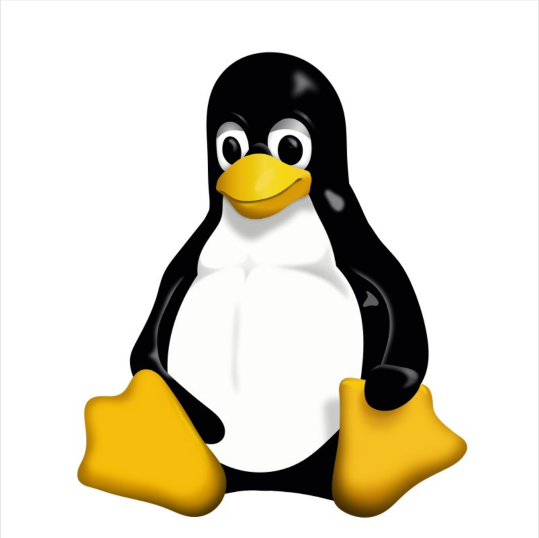
Turns a block cipher into a stream cipher:

- Essentially an approximation of one-time pad
- The pseudo-random keystream is generated by encrypting successive values of a counter combined with a nonce (IV)



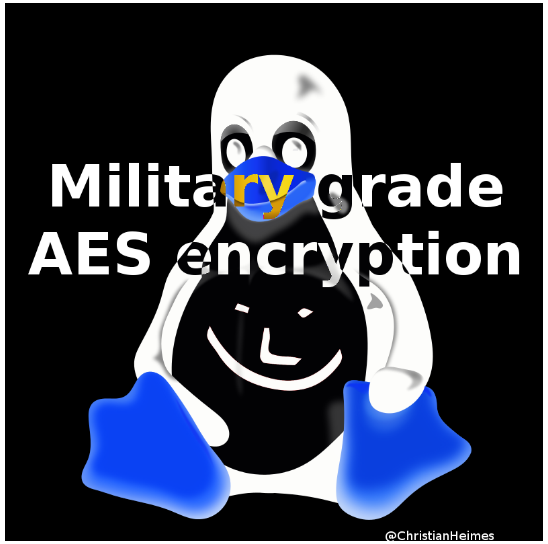
Counter (CTR) mode encryption

Seriously, never reuse the IV!



⊕

=



Both encrypted with AES-CTR/AES-GCM using the same key and IV