# CS 114:

## Network Security

Lecture 3 - Secret Key Cryptography

Prof. Daniel Votipka

Spring 2023

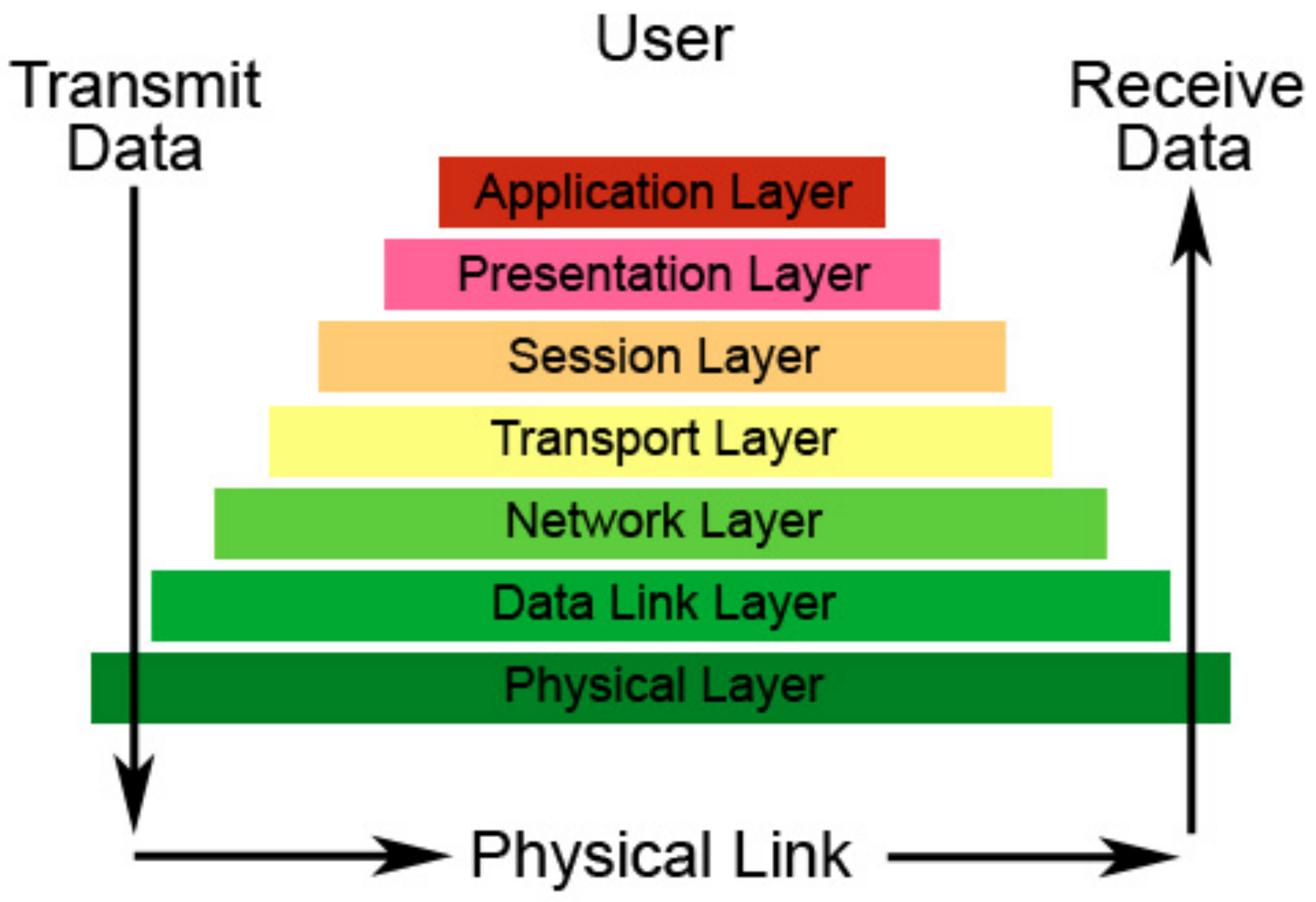
(some slides courtesy of Prof. Micah Sherr)

Tufts UNIVERSITY

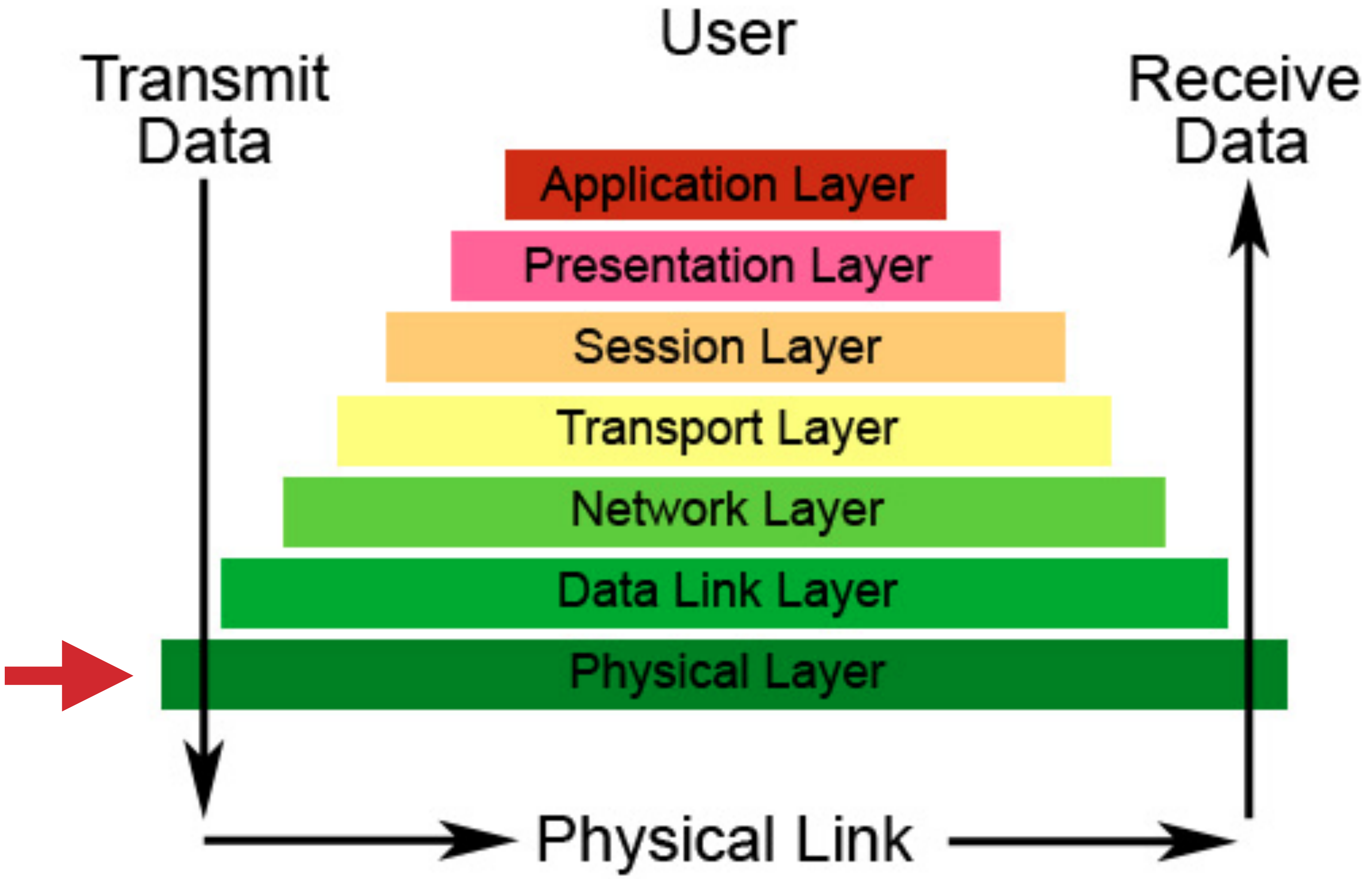
# Administrivia

- Homework 0 due Jan. 26th (today) at 11:59pm
- Homework 1, part 1 due Feb. 2nd at 11:59pm
- Make sure you read the socket programming HowTo (the required reading for today's lecture)

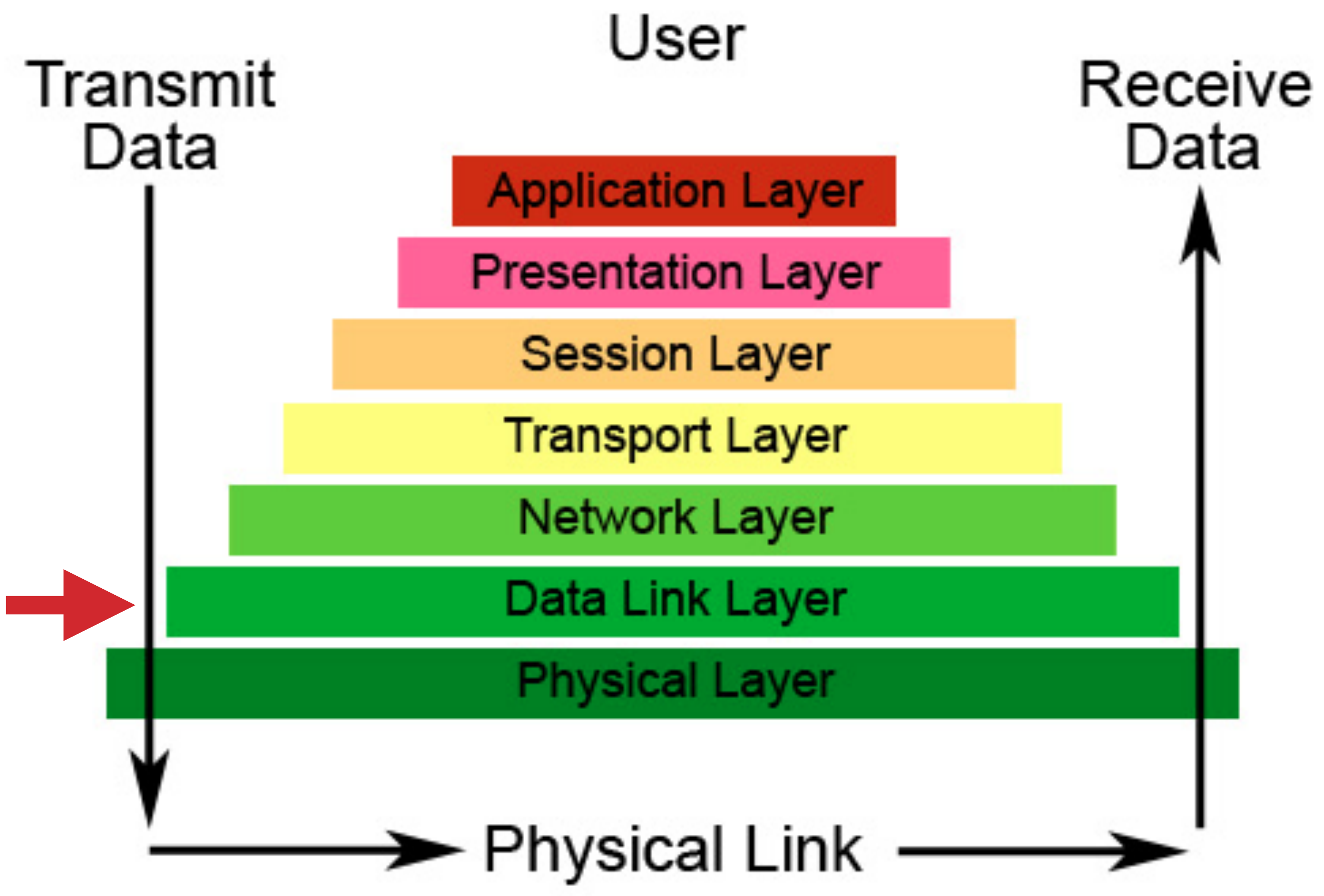
# The Seven Layers of OSI



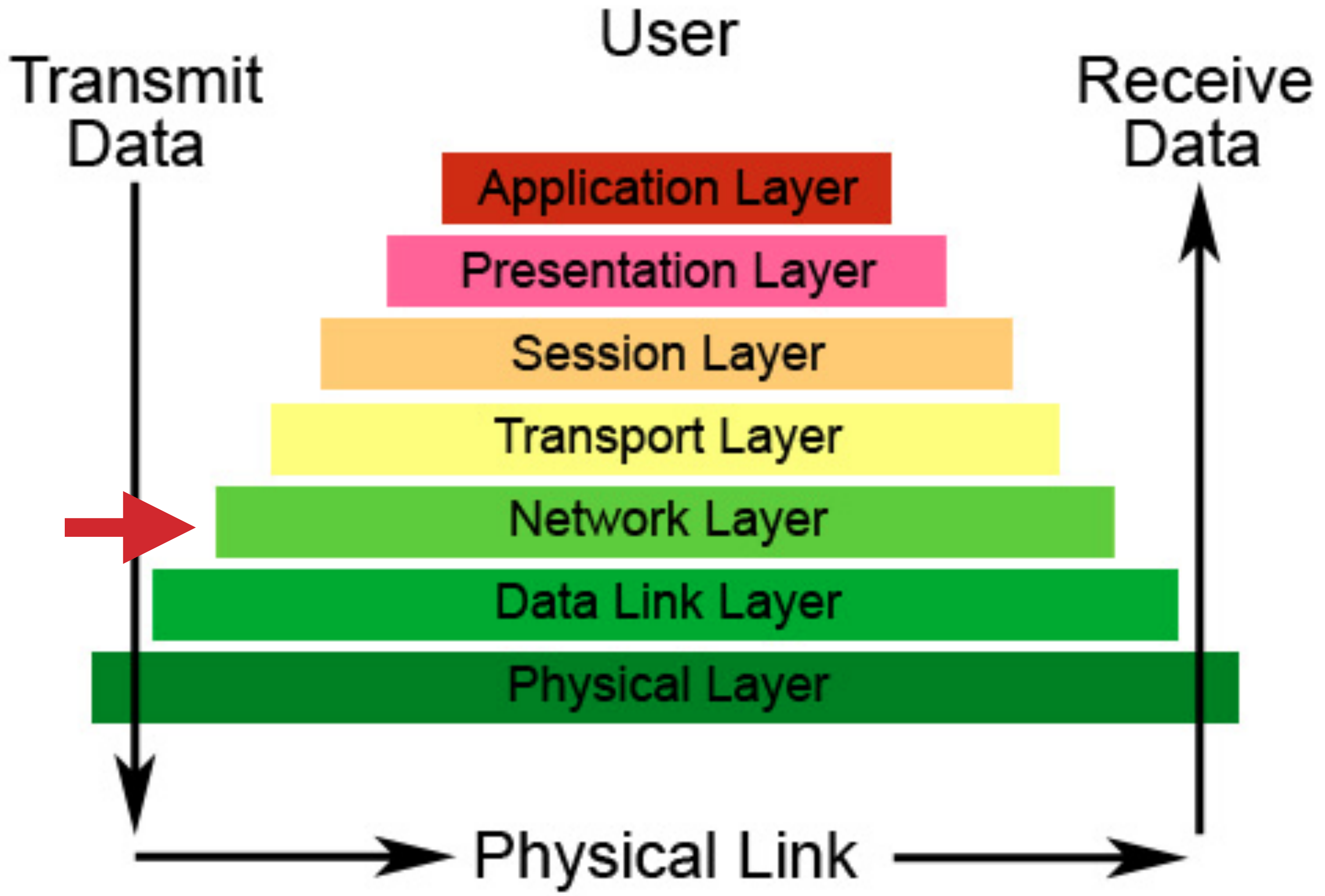
# The Seven Layers of OSI



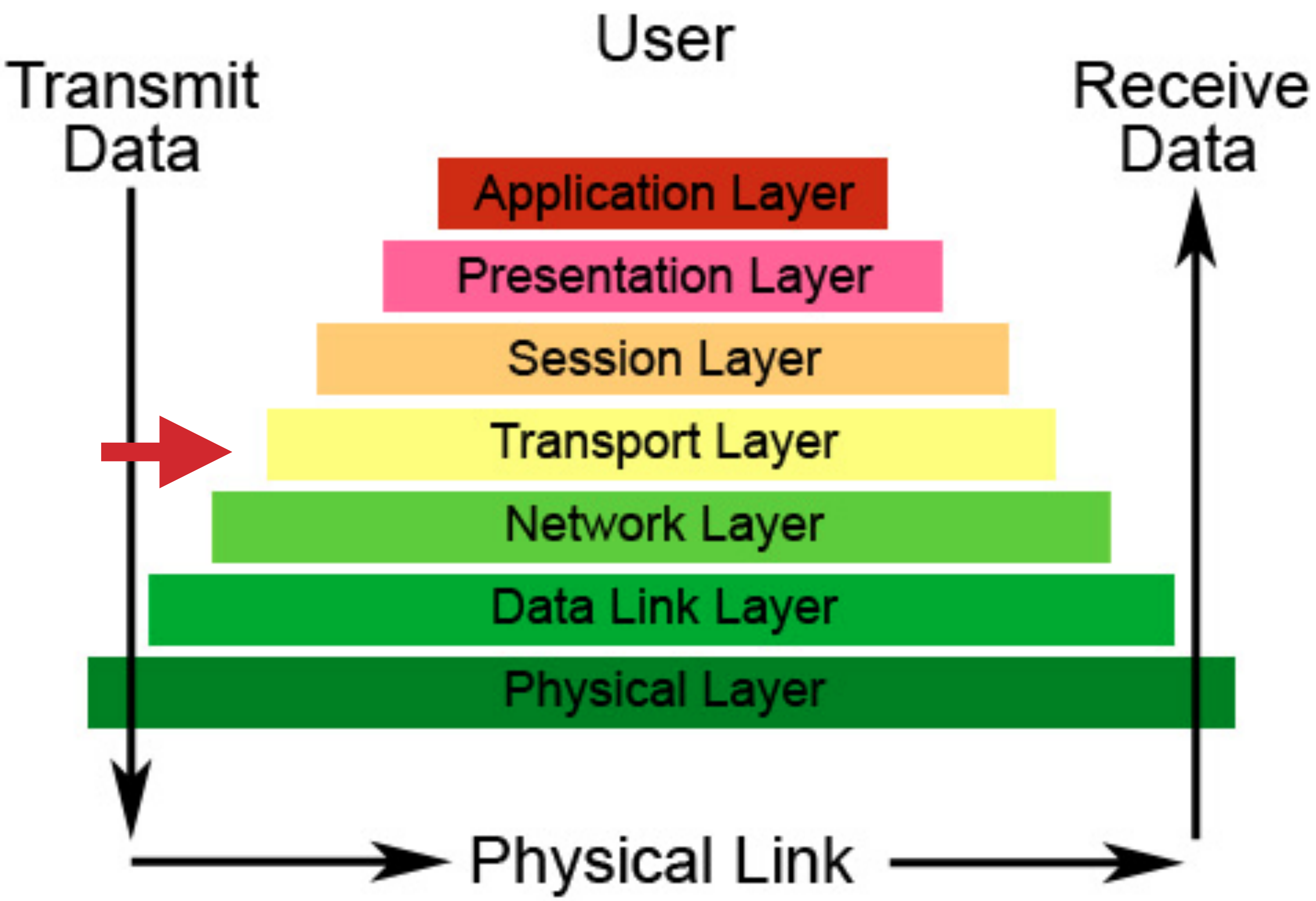
# The Seven Layers of OSI



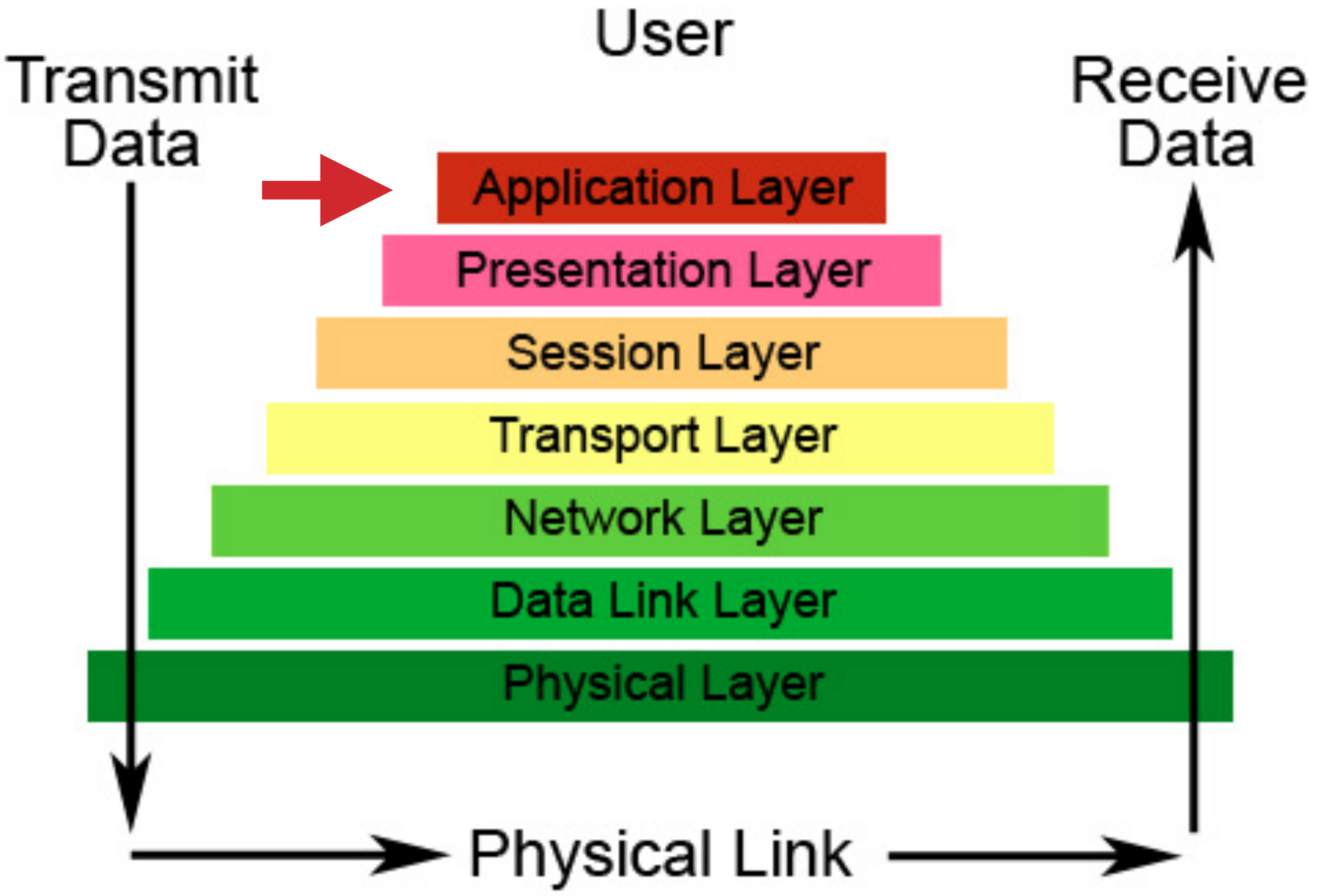
# The Seven Layers of OSI



# The Seven Layers of OSI



# The Seven Layers of OSI



# What about security?

- Where is confidentiality and authenticity?
- No relevant "security" fields in IP, TCP, or UDP headers.
- Why not?

# Cryptography

# cryptography < security

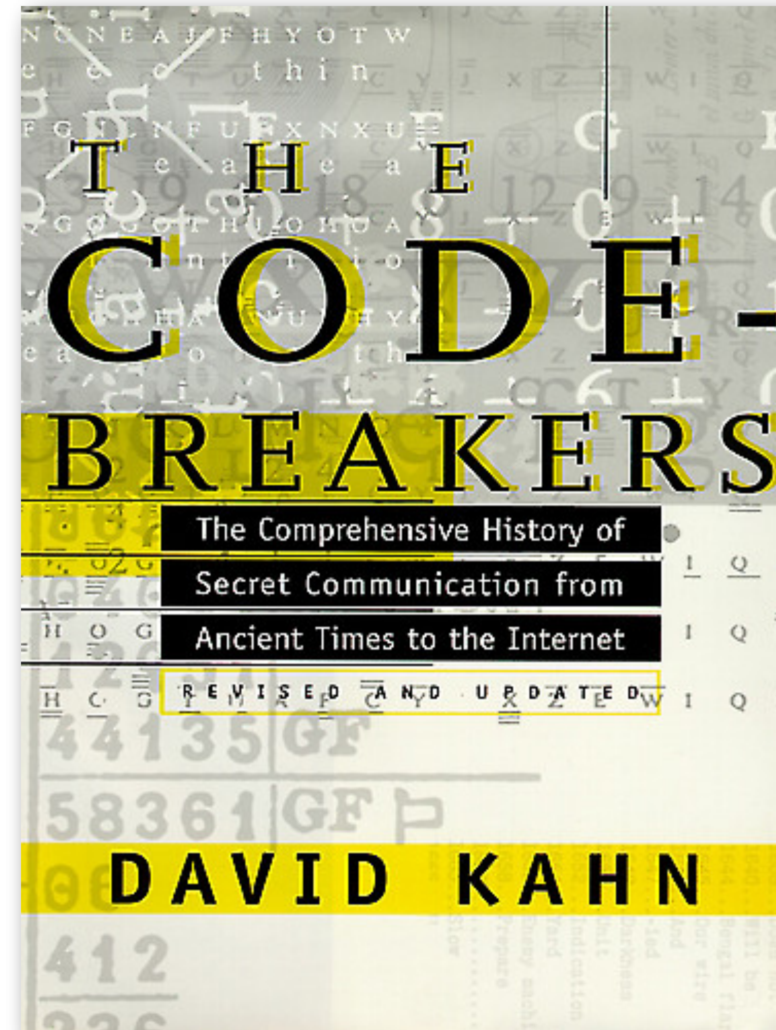
- Cryptography isn't the solution to security
  - Buffer overflows, worms, viruses, trojan horses, SQL injection attacks, cross-site scripting, bad programming practices, etc.
- It's a tool, not a solution
- Even when used, difficult to get right
  - Choice of encryption algorithms
  - Choice of parameters
  - Implementation
  - Hard to detect errors
    - Even when crypto fails, the program may still work
    - May not learn about crypto problems until after they've been exploited

88% of Android Apps using crypto make a mistake [Egele 2013]

Misunderstanding encryption is more common than not using it [Votipka 2020]

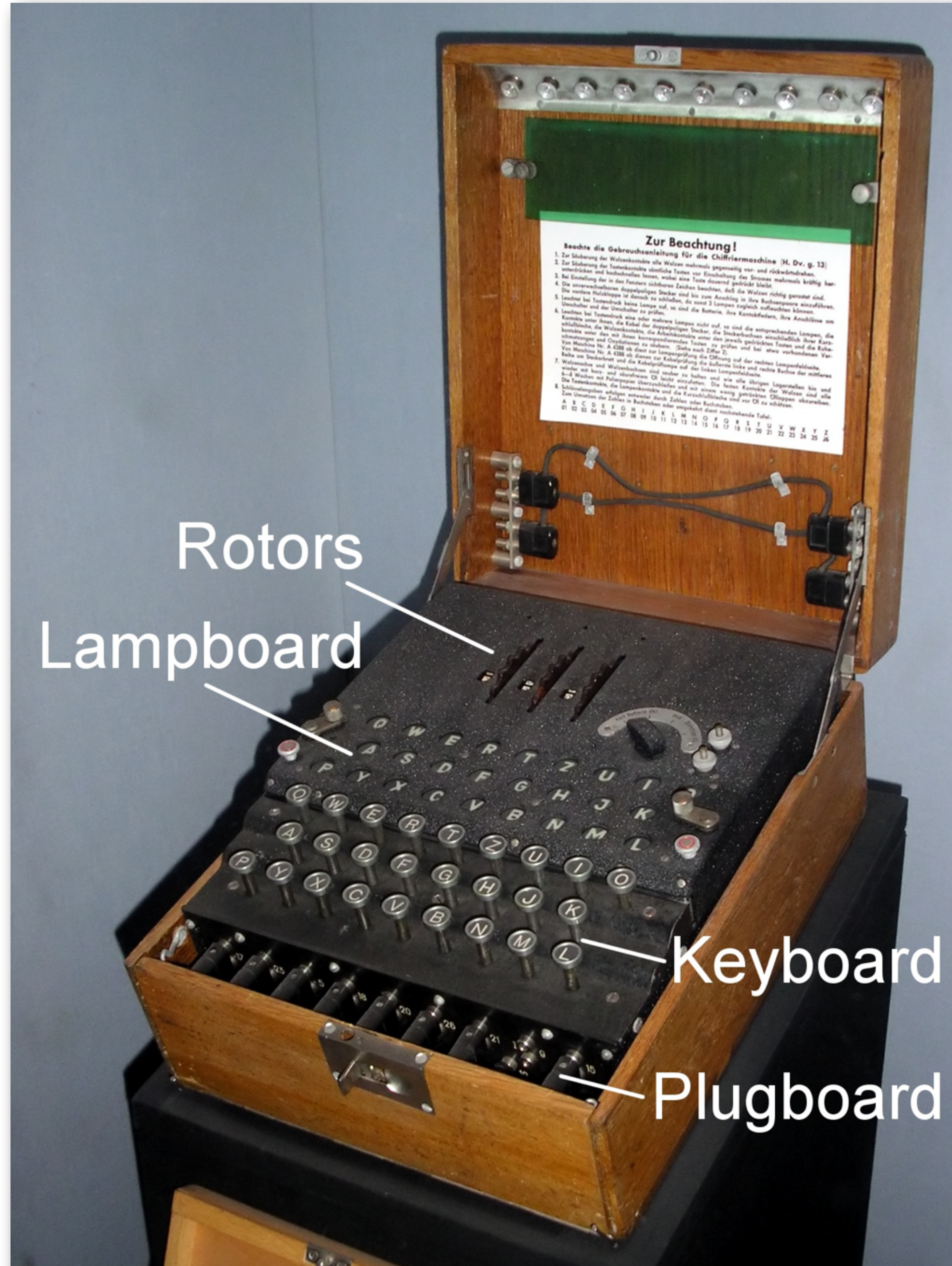
# Cryptographic History

- hide secrets from your enemy
- ~4000 year old discipline
    - Egyptians' use of non-standard hieroglyphics
    - Spartans used *scytale* to perform transposition cipher
    - Italian Leon Battista Alberti ("father of western cryptography") invents polyalphabetic ciphers in 1466




# Enigma

- German WWII encryption device
- Used polyalphabetic substitution cipher
- Broken by Allied forces
- Intelligence called Ultra
- Codebreaking at Bletchley Park
- See original at the International Spy Museum (bring your wallet) or NSA's National Cryptologic Museum (free!)



# What can crypto do?

- **Confidentiality**
  - Keep data and communication secret
  - Encryption / decryption
- **Integrity**
  - Protect reliability of data against tampering
  - “Was this the original message that was sent?”
- **Authenticity**
  - Provide evidence that data/messages are from their purported originators
  - “Did Alice really send this message?”

# Why is crypto useful?

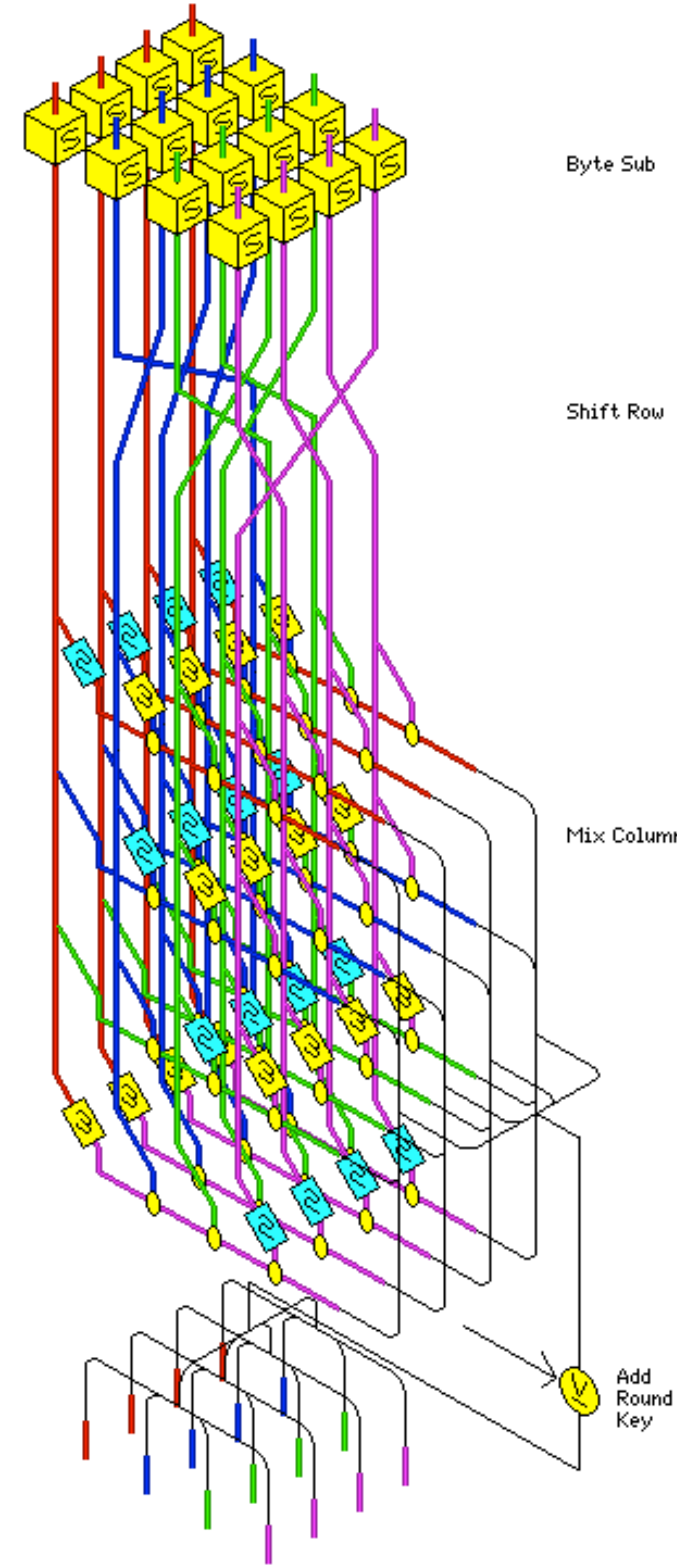
- Networks designed for data transport, not for data confidentiality (privacy) or authenticity
- Internet eavesdropping is (relatively) easy
- Crypto enables:
  - e-commerce and e-banking
  - confidential messaging
  - digital identities
  - protection of personal data
  - electronic voting
  - anonymity

# Some terminology

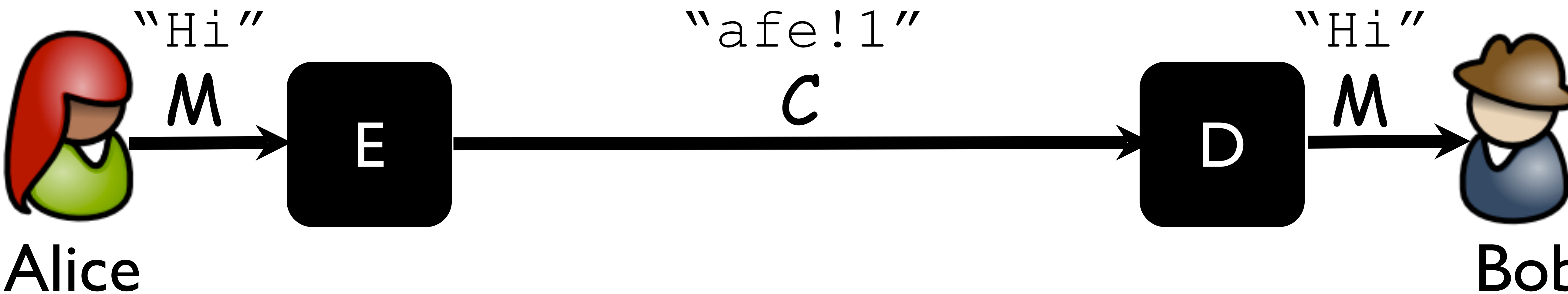
- **cryptosystem**: method of disguising (encrypting) plaintext messages so that only select parties can decipher (decrypt) the ciphertext
- **cryptography**: the art/science of developing and using cryptosystems
- **cryptanalysis**: the art/science of breaking cryptosystems
- **cryptology**: the combined study of cryptography and cryptanalysis

# Crypto is really, really, really, really, wicked hard

- Task: develop a cryptosystem that is secure against all conceivable (and inconceivable) attacks, and will be for the foreseeable future
- If you are inventing your own crypto, you're doing it wrong
- Common security idiom: "no one ever got fired for using AES"



# Encryption and Decryption



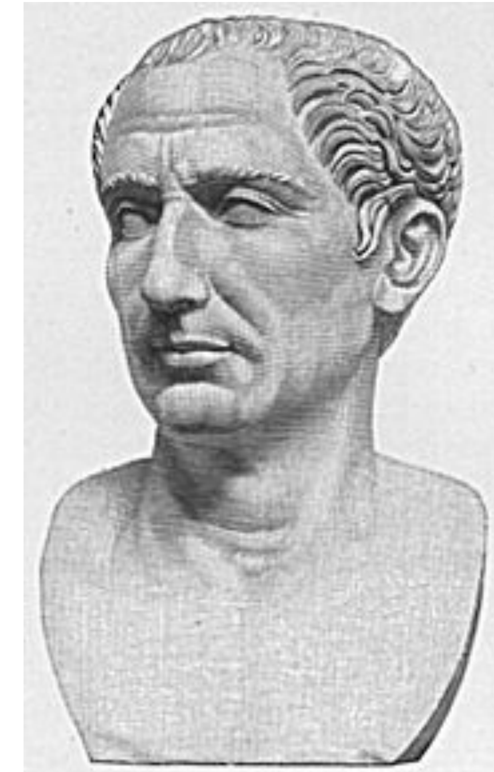

C=E(M)
M=D(C)
i.e.,
M=D(E(M))

where

M = plaintext
C = ciphertext
E(x) = encryption function
D(y) = decryption function

# Let’s look at some old crypto algorithms
# (don’t use these)

# Caesar Cipher



- A.K.A. Shift Cipher
- Used by Julius to communicate with his generals
- Encryption: Right-shift every character by x
- Decryption: Left-shift every character by x

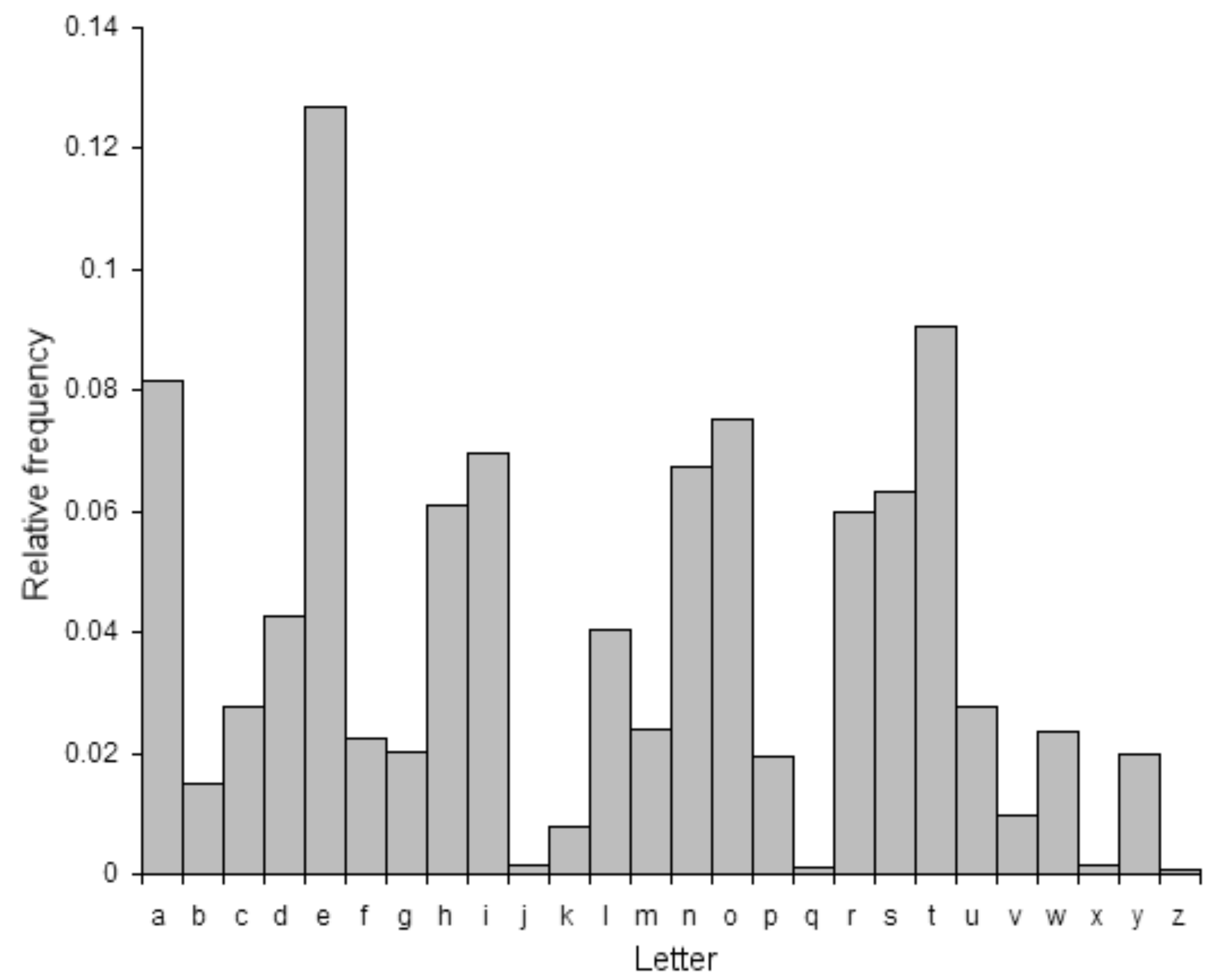
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y |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

**S E C U R I T Y   A N D   P R I V A C Y**

**V H F X U L W B   D Q G   S U L Y D F B**

# Cryptanalyzing the Caesar Cipher

- Cryptanalysis:
  - **Brute-force attack:** try all 26 possible shifts (i.e., values of x)
  - Frequency analysis: look for frequencies of characters



# Substitution Cipher

- Map each letter of the alphabet to another letter of the alphabet according to some fixed (but random) permutation
- E.g., cryptogram puzzles
- “Key size” is 26! (that's factorial, not, holy cow, 26!)
- E.g., (H→U,E→F,L→Z,O→A): HELLO →UFZZA
- Cryptanalysis:
  - frequency analysis
  - pattern analysis: (double Zs could be double Ds, Es, Ls, etc.)

| A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| C | M | T | E | F | H | P | U | D | X | N | Z | L |

| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| O | A | J | R | Y | I | G | W | V | B | S | Q | K |

# Substitution Cipher

- Vg gbbx n ybg bs oybbq, fjrng naq grnef gb trg gb jurer jr ner gbqnl, ohg jr unir whfg ortha. Gbqnl jr ortva va rnearfg gur jbex bs znxvat fher gung gur jbeyq jr yrnir bhe puvyqera vf whfg n yvggyr ovg orggre guna gur bar jr vaunovg gbqnl.

# Substitution Cipher

- Vg gbbx n ybg bs oybbq, fjrng naq grnef gb trg gb jurer jr ner gbqnl, ohg jr unir whfg ortha. Gbqnl jr ortva va rnearfg gur jbex bs znxvat fher gung gur jbeyq jr yrnir bhe puvyqera vf whfg n yvggyr ovg orggre guna gur bar jr vaunovg gbqnl.

# Substitution Cipher

- Vg gbbx n ybg bs oybbq, fjrng naq grnef gb trg gb jurer jr ner gbqnl, ohg jr unir whfg ortha. Gbqnl jr ortva va rnearfg gur jbex bs znxvat fher gung **gur** jbeyq jr yrnir bhe puvyqera vf whfg n yvggyr ovg orggre guna **gur** bar jr vaunovg gbqnl.
- It took a lot of blood, sweat and tears to get to where we are today, but we have just begun. Today we begin in earnest the work of making sure that **the** world we leave our children is just a little bit better than **the** one we inhabit today.

# One-time Pads

- To produce ciphertext, XOR the plaintext with the **one-time pad** (secret key)
  - $E(M) = M \oplus Pad$
  - $D(E(M)) = E(M) \oplus Pad$
- Requires sizeof(pad) == sizeof(plaintext)

- Offers **perfect secrecy**:
  - *a posteriori* probability of guessing plaintext given ciphertext equals the *a priori* probability
  - given a ciphertext without the pad, any plaintext of same length is possible input (there exists a corresponding pad)
  - $Pr[M=m|C=c] = Pr[M=m]$ (you learn nothing from the ciphertext)

# Proof that OTP achieves perfect secrecy

- Goal: $\Pr[M=m|C=c] = \Pr[M=m]$
  - Knowing the ciphertext should not improve our ability to determine the original message

# Proof that OTP achieves perfect secrecy

- Goal: $Pr[M=m|C=c] = Pr[M=m]$

1. $Pr[M=m|C=c] = \dfrac{Pr[C=c|M=m] * Pr[M=m]}{Pr[C=c]}$ Bayes' Theorem

2. $Pr[C=c|M=m] = Pr[c = m \oplus k] = \dfrac{1}{2^n}$
   - Given a message, the probability of picking a particular cipher is equal to the probability of picking a particular key

# Proof that OTP achieves perfect secrecy

- Goal: $Pr[M=m|C=c] = Pr[M=m]$

1. $Pr[M=m|C=c] = \dfrac{Pr[C=c|M=m] * Pr[M=m]}{Pr[C=c]}$

2. $Pr[C=c|M=m] = Pr[c = m \oplus k] = \dfrac{1}{2^n}$

3. $Pr[C=c] = \sum_{M} \underline{Pr[C=c|M=m]}*Pr[M=m]$

   - Probability of picking any ciphertext is the sum of the probability of picking a specific ciphertext for each possible message.

# Proof that OTP achieves perfect secrecy

- Goal: Pr[M=m|C=c] = Pr[M=m]

1. $\Pr[M=m|C=c] = \dfrac{\Pr[C=c|M=m] * \Pr[M=m]}{\Pr[C=c]}$

2. $\Pr[C=c|M=m] = \Pr[c = m \oplus k] = \dfrac{1}{2^n}$

3. $\Pr[C=c] = \sum_{M} 1/2^n * \Pr[M=m]$

# Proof that OTP achieves perfect secrecy

- Goal: Pr[M=m|C=c] = Pr[M=m]

1. $\Pr[M=m|C=c] = \dfrac{\Pr[C=c|M=m] * \Pr[M=m]}{\Pr[C=c]}$

2. $\Pr[C=c|M=m] = \Pr[c = m \oplus k] = \dfrac{1}{2^n}$

3. $\Pr[C=c] = 1/2^n \underbrace{\sum_M \Pr[M=m]}_{1}$

# Proof that OTP achieves perfect secrecy

- Goal: $\Pr[M=m|C=c] = \Pr[M=m]$

1. $\Pr[M=m|C=c] = \dfrac{\Pr[C=c|M=m] * \Pr[M=m]}{\Pr[C=c]}$

   $\dfrac{\cancel{1/2^n} * \Pr[M=m]}{\cancel{1/2^n}}$

2. $\Pr[C=c|M=m] = \Pr[c = m \oplus k] = \dfrac{1}{2^n}$

3. $\Pr[C=c] = \dfrac{1}{2^n}$

# Proof that OTP achieves perfect secrecy

- Goal: $Pr[M=m|C=c] = Pr[M=m]$

1. $Pr[M=m|C=c] =$ $\qquad$ $Pr[M=m]$

2. $Pr[C=c|M=m] = Pr[c = m \oplus k] = \frac{1}{2^n}$

3. $Pr[C=c] = \frac{1}{2^n}$

- more generally, if all ciphertexts are equally likely (i.e., $Pr[C|M]=1/2^n$), then cryptosystem achieves perfect secrecy!

# One-time Pads

- To produce ciphertext, XOR the plaintext with the **one-time pad** (secret key)
  - $E(M) = M \oplus Pad$
  - $D(E(M)) = E(M) \oplus Pad$
- Requires sizeof(pad) == sizeof(plaintext)

- Offers **perfect secrecy**:
  - *a posteriori* probability of guessing plaintext given ciphertext equals the *a priori* probability
  - given a ciphertext without the pad, any plaintext of same length is possible input (there exists a corresponding pad)
  - $Pr[M=m|C=c] = Pr[M=m]$ (you learn nothing from the ciphertext)

- Never reuse the pad (hence "one-time")! Why not?

# Modern Cryptography

# Two flavors of confidentiality

- **Unconditional** or **probabilistic security**: cryptosystem offers provable guarantees, irrespective of computational abilities of an attacker
  - given ciphertext, the probabilities that bit i of the plaintext is 0 is p and the probability that it is 1 is (1-p)
  - e.g., one-time pad
  - often requires key sizes that are equal to size of plaintext
- **Conditional** or **computational security**: cryptosystem is secure assuming a computationally bounded adversary, or under certain hardness assumptions (e.g., P<>NP)
  - e.g., DES, 3DES, AES, RSA, DSA, ECC, DH, MD5, SHA
  - Key sizes are much smaller (~128 bits)
- Almost all deployed modern cryptosystems are conditionally secure

# An aside about key sizes

- Original DES used 56-bit keys
- 3DES uses 168-bit keys
- AES uses 128-, 192- or 256-bit keys
- Are these numbers big enough?
  - DES has $2^{56}$ = 72,057,594,037,927,936 possible keys
    - In Feb 1998, distributed.net cracked DES in 41 days
    - In July 1998, the Electronic Frontier Foundation (EFF) and distributed.net cracked DES in 56 hours using a $250K machine
    - In Jan 1999, the team did in less than 24 hours
  - **Each additional bit adds 2X brute-force work factor (exponential security for linear keysize increase)**
  - There are approximately $2^{250}$ atoms in the universe, so don't expect 256-bit keys to be brute forced anytime in the next trillion years.
- Takeaway: 128-keys are reasonably secure

# Cryptanalysis

- Goal: learn the key
- Classifications:
  - **ciphertext-only** attack: Eve has access only to ciphertext
  - **known-plaintext** attack: Eve has access to plaintext and corresponding ciphertext
  - **chosen-plaintext** attack: Eve can choose plaintext and learn ciphertext
  - **chosen-ciphertext** attack: Eve can choose ciphertext and learn plaintext

# Other cryptanalysis ...

- Brute force cryptanalysis
  - Just keep trying different keys and check result
- Not covered in this class:
  - Linear cryptanalysis
    - Construct linear equations relating plaintext, ciphertext and key bits that have a high bias
    - Use these linear equations in conjunction with known plaintext-ciphertext pairs to derive key bits
  - Differential cryptanalysis
    - Study how differences in an input can affect the resultant difference at the output
    - Use chosen plaintext to uncover key bits

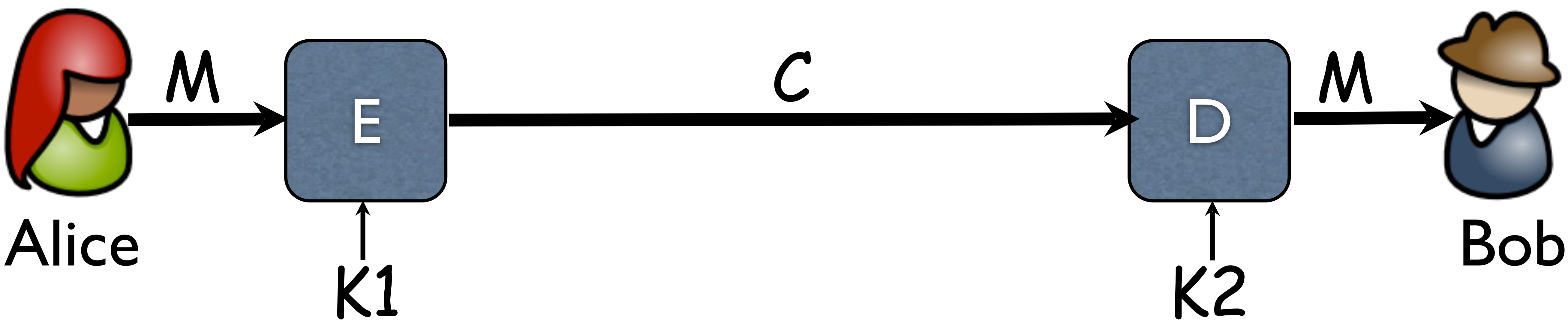
# Kerckhoffs' Principles

- Modern cryptosystems use a key to control encryption and decryption
- Ciphertext should be undecipherable without the correct key
- Encryption key may be different from decryption key.
- **Kerckhoffs' principles** [1883]:
  - Assume Eve knows cipher algorithm
  - Security should rely on choice of key
  - If Eve discovers the key, a new key can be chosen

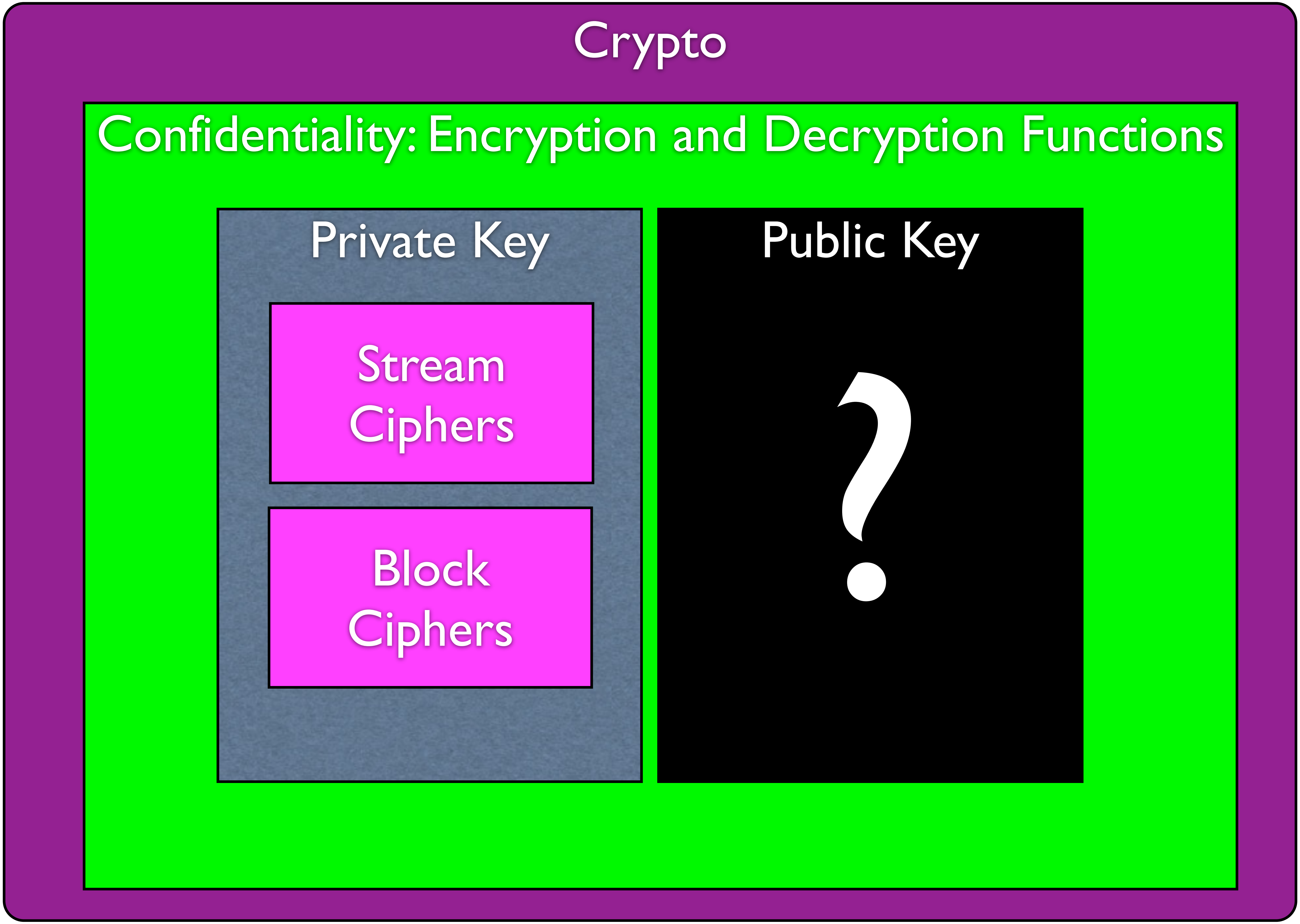
# Kerckhoffs' Principles

- Kerckhoffs' Principles are contrary to the principle of "**security by obscurity**", which relies only upon the secrecy of the algorithm/cryptosystem
  - If security of a keyless algorithm compromised, cryptosystem becomes permanently useless (and unfixable)
  - Algorithms relatively easy to reverse engineer

# Symmetric and Asymmetric Crypto




- **Symmetric crypto:** (also called **private key crypto**)
  - Alice and Bob share the same key (K=K1=K2)
  - K used for both encrypting and decrypting
  - Doesn't imply that encrypting and decrypting are the same algorithm
  - Also called **private key** or **secret key** cryptography, since knowledge of the key reveals the plaintext
- **Asymmetric crypto:** (also called **public key crypto**)
  - Alice and Bob have different keys
  - Alice encrypts with K1 and Bob decrypts with K2
  - Also called **public key** cryptography, since Alice and Bob can publicly post their *public* keys


Crypto
Confidentiality: Encryption and Decryption Functions
Private Key
Stream Ciphers
Block Ciphers
Public Key
?

# Stream ciphers vs. Block ciphers

- **Stream Ciphers**
  - Combine (e.g., XOR) plaintext with pseudorandom stream of bits
  - Pseudorandom stream generated based on key
  - XOR with same bit stream to recover plaintext
  - E.g., RC4, FISH
- **Block Ciphers**
  - Fixed block size
  - Encrypt block-sized portions of plaintext
  - Combine encrypted blocks (more on this later)
  - E.g., DES, 3DES, AES

# Stream Ciphers

- Useful when plaintext arrives as a stream (e.g., 802.11's WEP)
- Vulnerable if used incorrectly

# Stream Ciphers

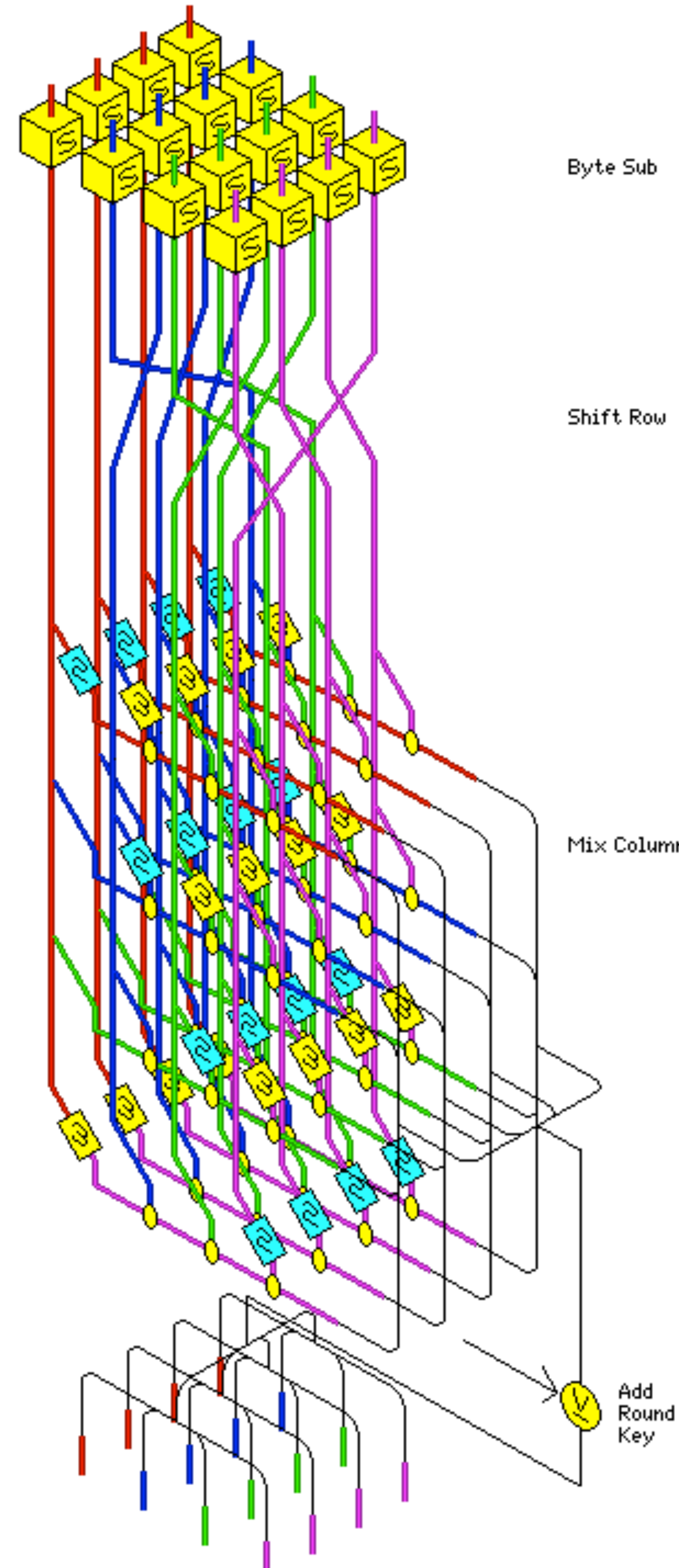
- **Key reuse:** [C(K) = pseudorandom stream produced using key K]
  - $E(M1) = M1 \oplus C(K)$
  - $E(M2) = M2 \oplus C(K)$
  - Suppose Eve knows ciphertexts E(M1) and E(M2)
  - $E(M1) \oplus E(M2) = M1 \oplus C(K) \oplus M2 \oplus C(K) = M1 \oplus M2$
  - M1 and M2 can be derived from $M1 \oplus M2$ using frequency analysis
- Countermeasure is to use IV (**initialization vector**)
  - IV <u>sent in clear</u> and is combined with K to produce pseudorandom sequence
  - E.g., replace C(K) with $C(K \oplus IV)$ or C(f(K,IV))
  - IVs should never be reused and should be sufficiently large
  - WEP broken partly because IVs were insufficiently large
  - modern stream ciphers take IVs, but it's up to the programmer to generate them

# Stream Ciphers

- **Substitution Attack:**
  - M = “Pay Eve $100.00”
  - $E(M) = M \oplus C(K,IV)$
  - Suppose Eve knows M and E(M) but doesn't know K
  - She can substitute M for M' by replacing E(M) with:
    - $E'(M) = E(M) \oplus M \oplus M' = (M \oplus C(K)) \oplus M \oplus M' = C(K) \oplus M'$
    - Eve can then replace E(M) with E'(M), which Bob will decrypt message as M' (“Pay Eve $900.00”)
  - Countermeasure is to include message authentication code (more on this later) that helps detect manipulation (i.e., provides integrity and authenticity)

# Block Ciphers

- Plaintext broken into fixed-sized blocks
- Each block individually encrypted
- Substitution-Permutation Networks
  - **S-Box**
    - Input: sequence of x bits
    - Output: new sequence of x bits
    - Mapping from one bit string to another
  - **Permutation**
    - Input: sequence of x bits
    - Output: permutation of the input
- Symmetric key encryption typically uses many rounds of S-Boxes and permutations, incorporating the key



# Advanced Encryption Standard (AES)

- International NIST bakeoff in 2001 between cryptographers
- Replaced DES as the "accepted" symmetric key cipher
  - Substitution-permutation network
  - Variable key lengths
  - Fast implementation in both hardware and software
  - Small code and memory footprint

# Modes of Operation

- Modes of operation allow encryption of arbitrary length plaintext

# Modes of Operation: Electronic Codebook (ECB)

- Blocks are individually encrypted and concatenated together
- Problems:
  - Identical plaintext blocks produce identical ciphertext blocks
  - Encrypted blocks can be shuffled without detection

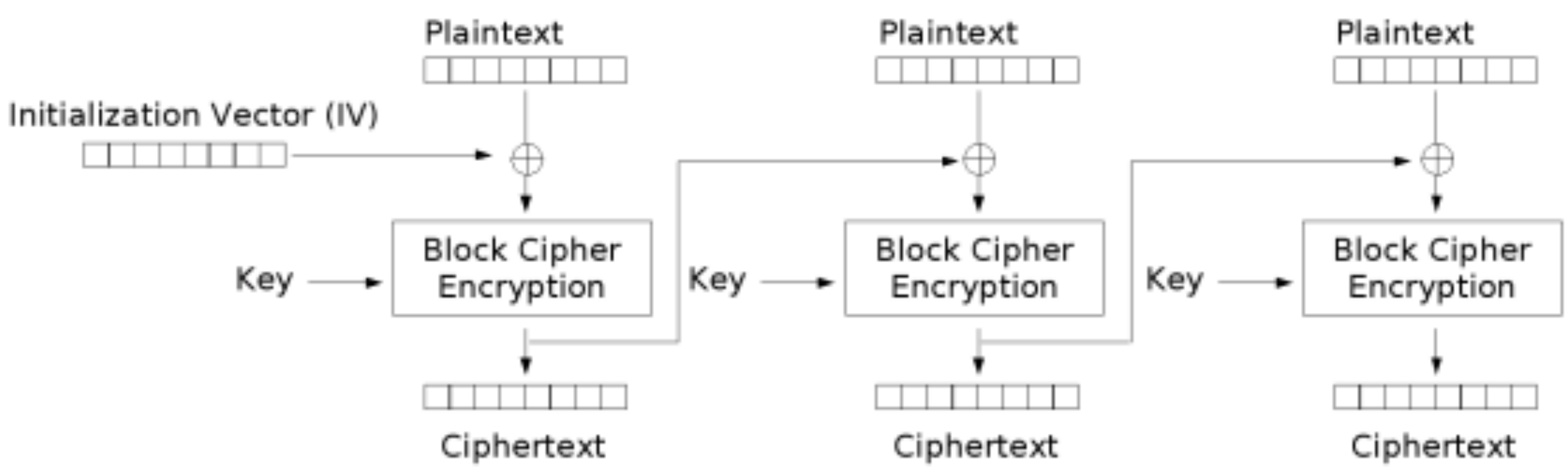
Plaintext

ECB

Other modes

# Modes of Operation: Cipher-block Chaining (CBC)

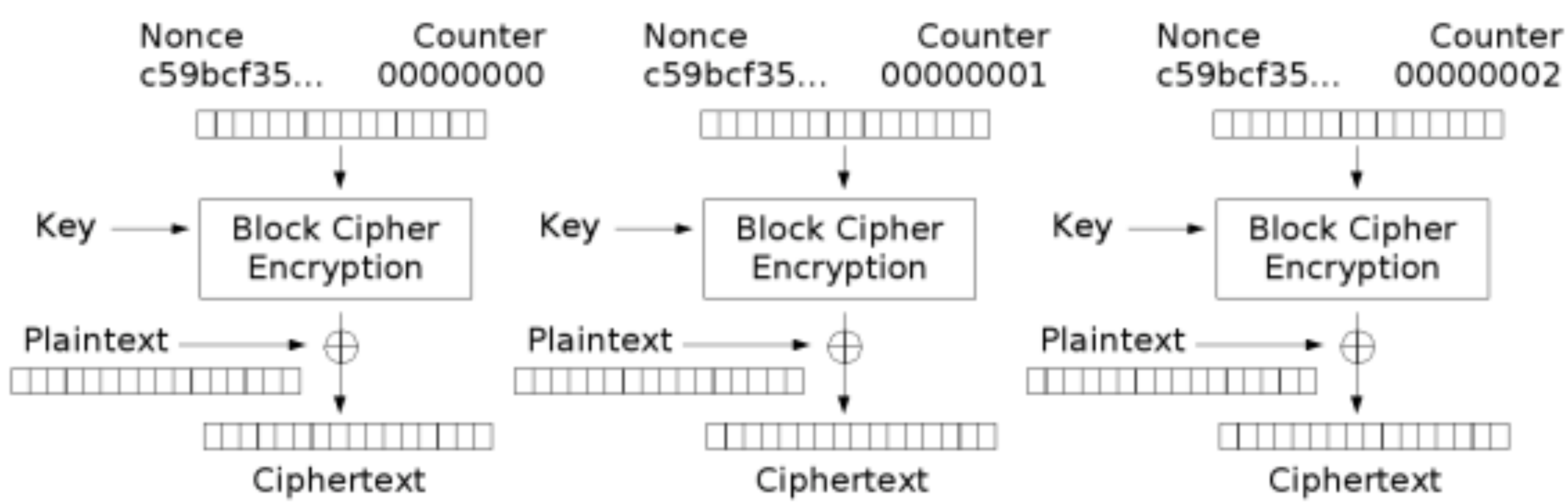
- Each block xor'd with ciphertext of previous block before encrypting
- Uses **initialization vector** (IV) to kickoff randomness
- IVs sent in the clear; should be randomly chosen for each session




Cipher Block Chaining (CBC) mode encryption

# Modes of Operation: Counter Mode (CTR)

- Allows random-access encryption/decryption
- Encrypts the IV plus a counter (incremented with each block), and xor the result with the plaintext
- Causes block cipher to function as a stream cipher




Counter (CTR) mode encryption

# Basic truths of cryptography


- Cryptography is not frequently the source of security problems
  - Algorithms are well known and widely studied
  - Vetted through crypto community
  - Avoid any "proprietary" encryption
  - Claims of "new technology" or "perfect security" are almost assuredly **snake oil**

# Building systems with cryptography

- Use quality libraries
  - SSLeay, cryptolib, openssl
  - Find out what cryptographers think of a package before using it
- Code review like crazy
- Educate yourself on how to use library
  - Understand caveats by original designer and programmer

# Common pitfalls

- Generating randomness
- Storage of secret keys
- Virtual memory (pages secrets onto disk)
- Protocol interactions
- Poor user interface
- Poor choice of parameters or modes

# What encryption does and does not

- Does:
  - confidentiality
- Doesn't do:
  - data integrity
  - source authentication
- **Need:** ensure that data is not altered and is from an authenticated source

Hashes and Message Authentication