

CS 88: Security and Privacy

11: Symmetric Key Cryptography

02-29-2024

slides adapted from Dave Levine, Vitaly Shmatikov, Christo Wilson, and Franz Roesner



Cryptography

- Cryptography: An ancient art
 - 500BC – 20th century: Design -> break -> repair -> break -> repair ->.....
- Modern Cryptography: Cryptography as a science
 - relies on rigorous threat models
 - firm theoretical foundations and proofs!

Modern Cryptography

*Design, analysis and implementation of **mathematical techniques** for **securing** information, systems and computation against **adversarial attacks**.*

Modern Cryptography: How many of the following actions involve cryptography ?

1. Git cloning your lab repo
2. Connecting to Swarthmore's WiFi
3. Updating software on your device
4. Making online purchases

- A. One of these
- B. Two of these
- C. Three of these
- D. Four of these

Modern Cryptography

If you don't understand what you want to achieve, how can you possibly know when (or if) you have achieved it?

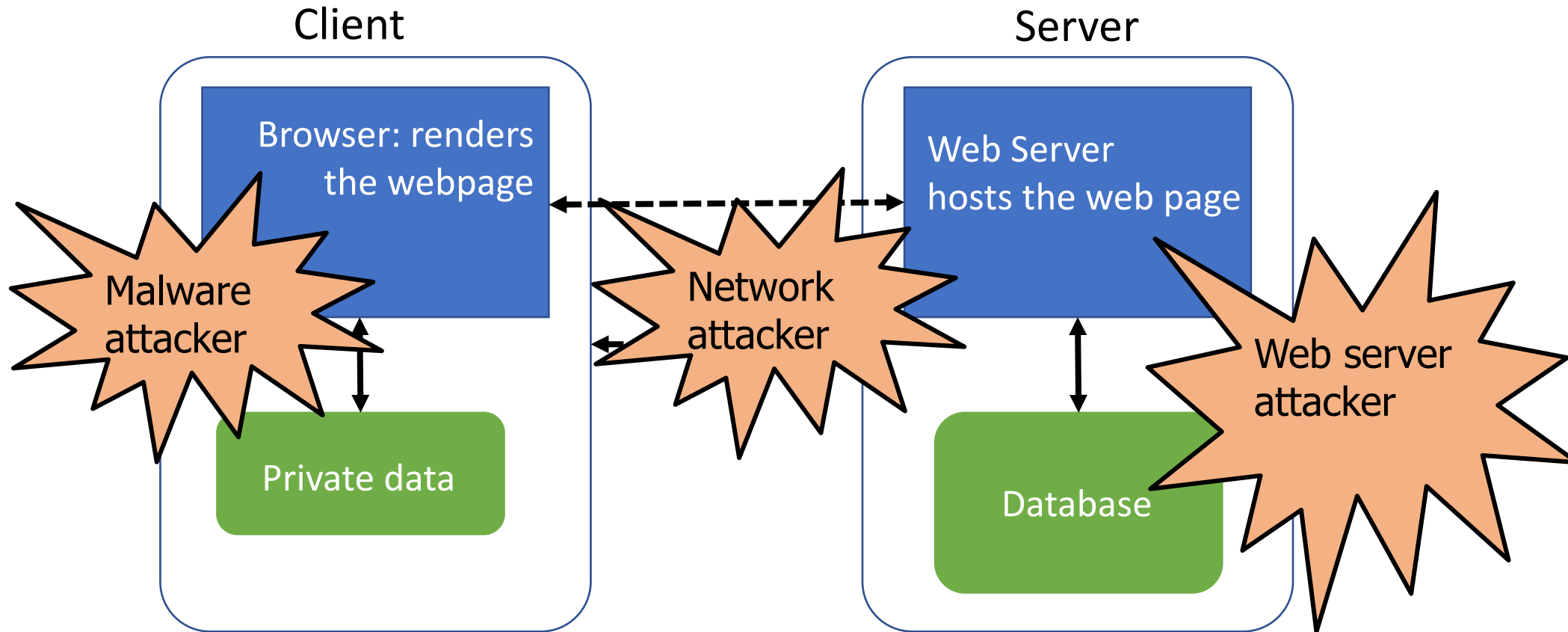
Modern Cryptography

Importance of clear assumptions:

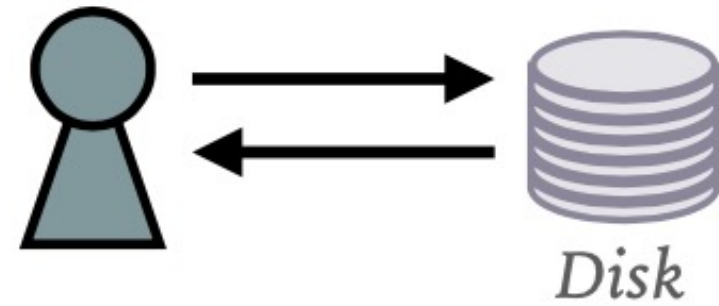
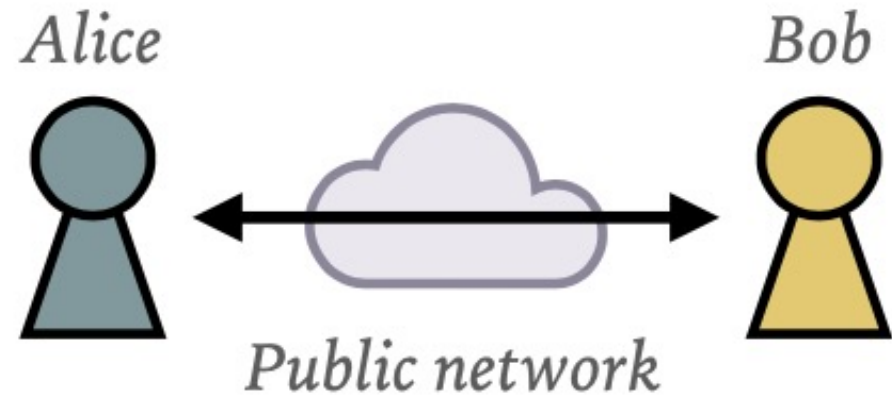
- allows researchers to validate assumptions
- comparison between schemes based on different assumptions
- re-adjust for weaknesses in assumptions

any new cryptographic construction should be proven secure with respect to a specific definition, and a set of clearly stated assumptions

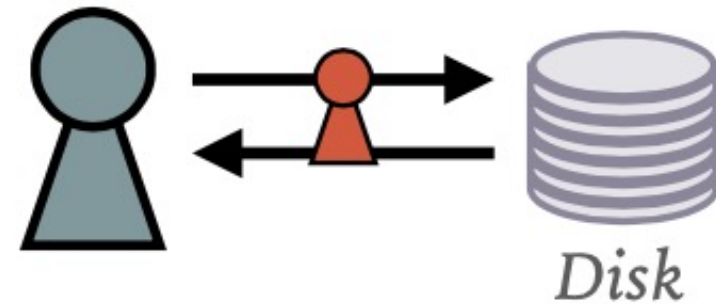
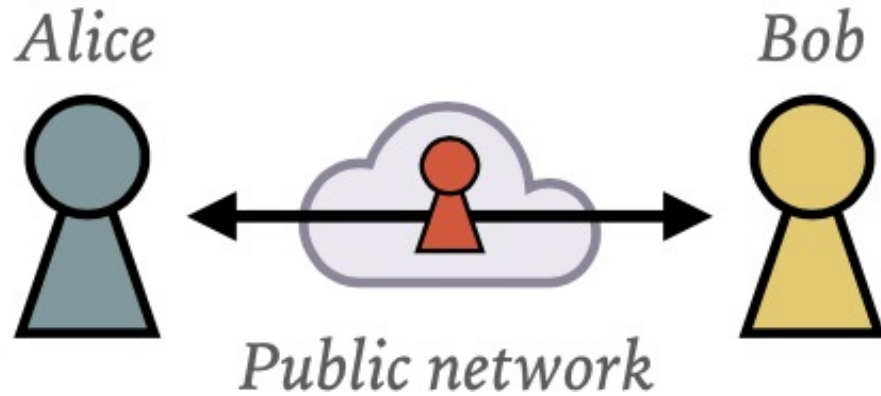
Where Does the Attacker Live?



Scenarios and Goals



Scenarios and Goals



- Confidentiality** Keep others from reading Alice's messages/data
- Integrity** Keep others from undetectably tampering with Alice's messages/data
- Authenticity** Keep others from undetectably impersonating Alice (keep her to her word too!)

Recall the Bigger Picture

- Cryptography: small piece of a larger system
- Protect the entire system (recall: the weakest link)
 - physical security
 - OS security
 - Network security
 - Users
 - Cryptography
- Cryptography is a crucial part of this toolbox



Cryptography: Terms



- Encryption (E): The process of transforming a message so that its meaning is not obvious
- Decryption (D): The process of transforming an encrypted message back into its original form.
- Plaintext (P): Original, unencrypted form of a message
- Ciphertext (C): The encrypted form of a message

Formal Notation: We seek a cryptosystem for which $P = D (E (P))$

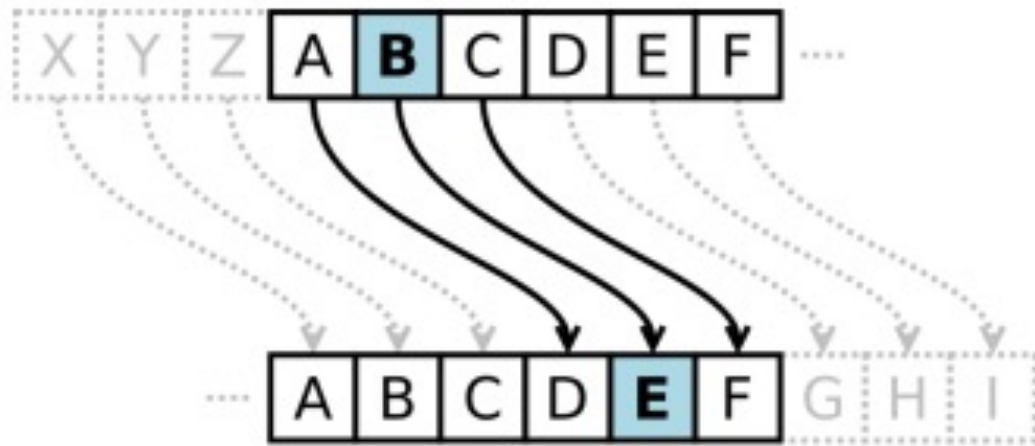
Historical Ciphers

- Substitution Cipher
 - Monoalphabetic - Ceasar's Cipher – fixed subst. over the entire message
 - Polyalphabetic – a number of substitutions at different positions in the message
- Transposition Ciphers
- Codebooks
- Machines

Recommended Reading: The Codebreakers by David Kahn, The Code Book by Simon Singh

Ceasar Cipher: Substitution Cipher

Plaintext letters replaced with letters fixed shift way in the alphabet.



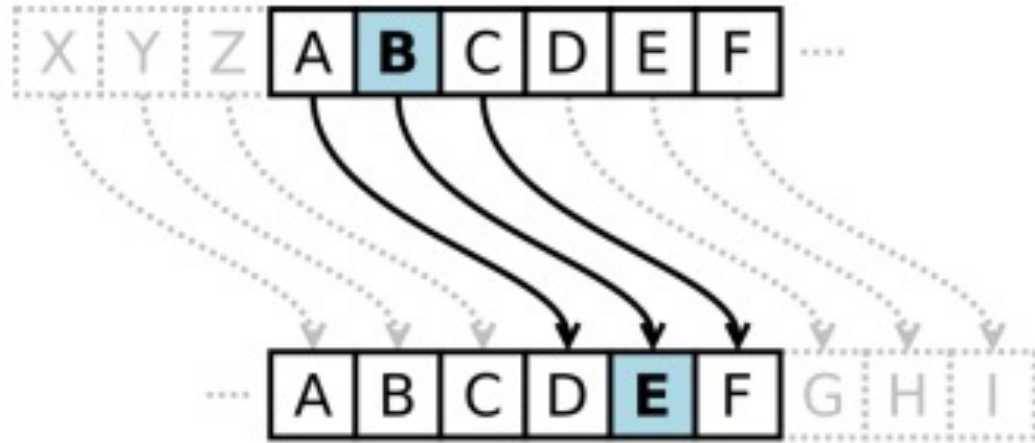
Example:

- Plaintext: HEY BRUTUS BRING A KNIFE TO THE PARTY.
- Ciphertext: **KHB EUXWXV EULQJ D NQLIH WR WKH SDUWB**
- Key Shift 3:
 - ABCDEFGHIJKLMNOPQRSTUVWXYZ
 - DEFPGHIJKLMNOPQRSTUVWXYZABC



Ceasar Cipher: Substitution Cipher

Plaintext letters replaced with letters fixed shift way in the alphabet.



Example:

- Plaintext: HEY BRUTUS BRING A KNIFE TO THE PARTY.
- Ciphertext: **KHB EUXWXV EULQJ D NQLIH WR WKH SDUWB**
- Key Shift 3:
 - ABCDEFGHIJKLMNOPQRSTUVWXYZ
 - DEF_GHIJKLMNOPQRSTUVWXYZABC



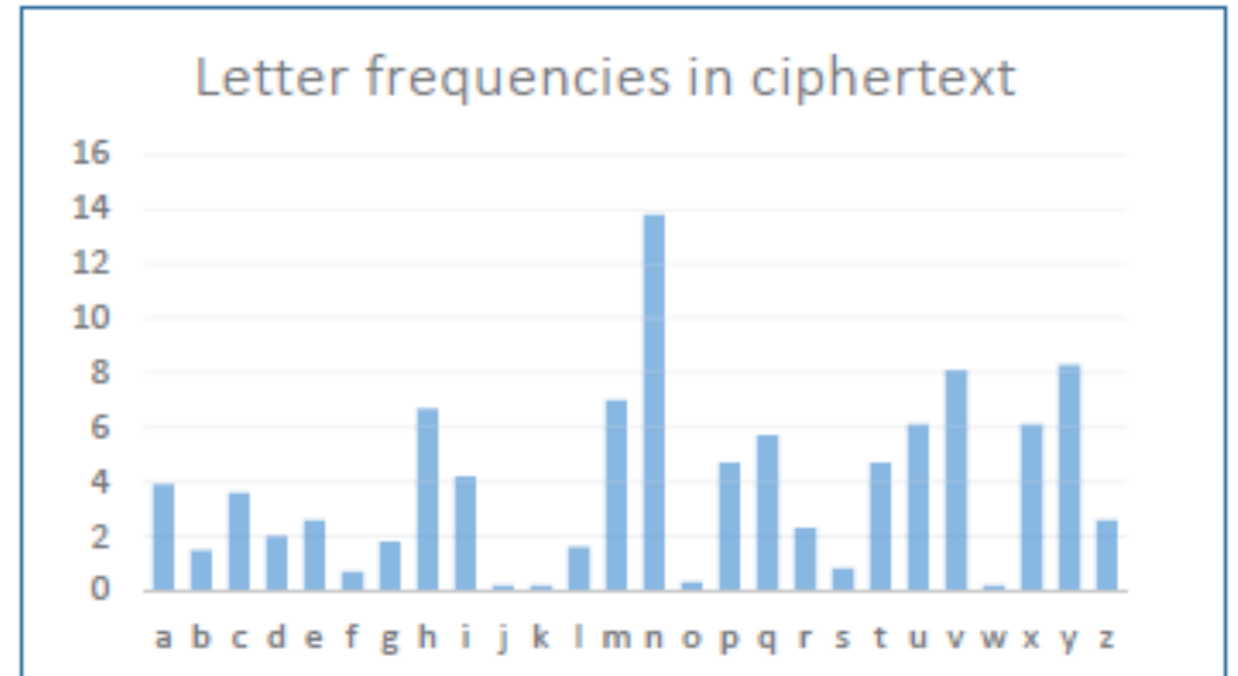
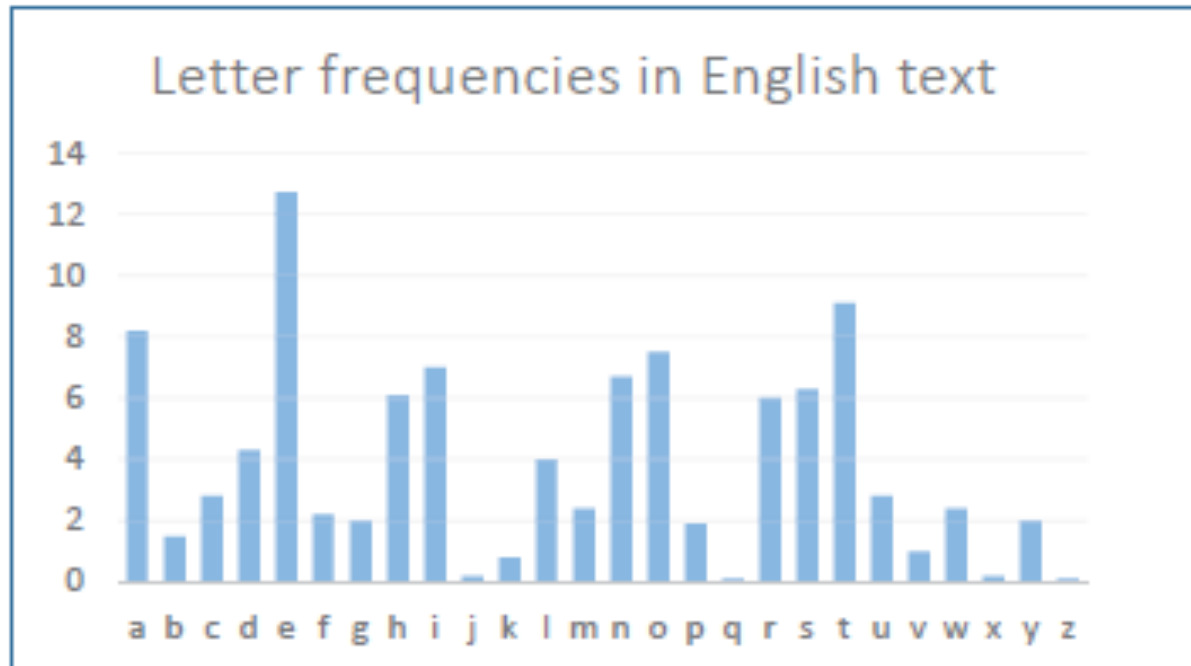
- Encryption and Decryption are symmetric.
- Key space?
 - 26
- Attack shift ciphers?
 - brute force

Substitution Cipher

- **Superset of shift ciphers:** each letter is substituted for another one.
- One implementation: Add a secret key
- Example
 - Plaintext: ABCDEFGHIJKLMNOPQRSTUVWXYZ
 - Cipher: ZEBRASCDFGHIJKLMNOPQTUVWXY
- “state-of-the-art” for thousands of years

Monoalphabetic Substitution Cipher

- What is the key space?
 - $26!$ approx. = 2^{88}
- Launching an attack?
 - frequency analysis: the study of frequency of letters or groups of letters (grams).
 - Common letters: T, A, E, I, O
 - Common 2-letter combinations (bi-grams): TH, HE, IN, ER
 - Common 3-letter combinations (tri-grams): THE, AND, ING.



Cryptanalysis of Monoalphabetic Substitution

- Dominates cryptography through the first millennium
- Frequency analysis (Al-Kindi from 800 AD)
- Lessons?
 - Use large blocks: instead of replacing ~6 bits at a time, replace 64 or 128 bits
 - Leads to block ciphers like DES and AES
 - Use different substitutions to prevent frequency analysis
 - Leads to polyalphabetic substitution ciphers and stream ciphers

Vigenère Cipher (1596)




- Main weakness of monoalphabetic substitution ciphers:
 - Each letter in the ciphertext corresponds to only one letter in the plaintext
- Polyalphabetic substitution cipher
 - Given a key $K = (k_1, k_2, \dots, k_m)$
 - Shift each letter p in the plaintext by k_i , where i is modulo m

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

Plaintext CRYPTOGRAPHY
Key LUCKLUCKLUCK (Shift 11 20 2 10 11 20 2 11 ...)
Ciphertext NLAZEIIBLJJI

Kasisky Test

Plaintext	T H E S U N A N D	T H E	M A N I N	T H E	M O O N
Key	K I N G K I N G K	I N G	K I N G K	I N G	K I N G
Ciphertext	D P R Y E V N T N	B U K	W I A O X	B U K	W W B T


Distance = 8

- Repeating patterns (of length >2) in ciphertext are a tell
 - Likely due to repeated plaintext encrypted under repeated key characters
 - The distance is likely to be a multiple of the key length

Cryptanalysis of Vigenère Cipher

- Cracking Vigenère (1854 or 1863)
 1. Guess the key length x using Kasisky test or index of coincidence
 2. Divide the ciphertext into x shift cipher encryptions
 3. Use frequency analysis on each shift cipher
- Lessons?
 - As key length increases, letter frequency becomes more random
 - If key never repeated, Vigenère wouldn't be breakable!

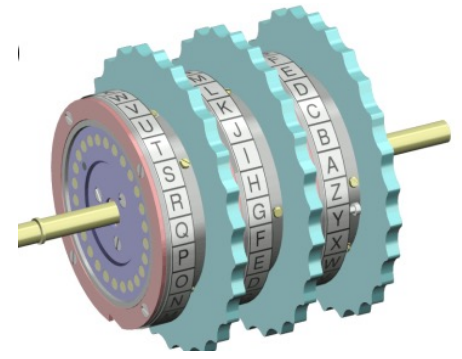




- WW2 German Enigma machine
 - Polyalphabetic substitution cipher
 - Substitution table changes from character to character
 - Rotors control substitutions
- Allies broke Enigma (even before the war), significant intelligence impact
- Computers were built to break WW2 ciphers, by Alan Turing and others



Enigma Machine



- Use rotors that change position after each key
- Key: initial setting of the rotors
- Key space?
 - 26^n for n rotors
- KeyGen:
 - Choose rotors, rotor orders, rotor positions, and plugboard settings
 - 158,962,555,217,826,360,000 possible keys!

Cryptanalysis: Enigma

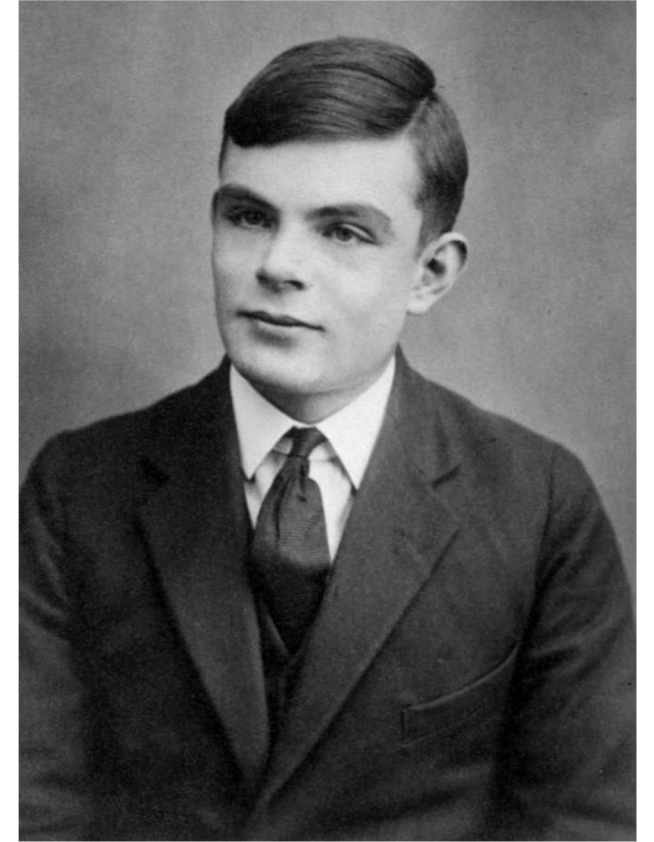
- Polish and British cryptographers built BOMBE, a machine to brute-force Enigma keys
- Why was Enigma breakable?
 - **Kerckhoff's principle**: The Allies stole Enigma machines, so they knew the algorithm
 - **Known plaintext attacks**: the Germans often sent predictable messages (e.g. the weather report every morning)
 - **Chosen plaintext attacks**: the Allies could trick the Germans into sending a message (e.g. "soldiers at Normandy")
 - **Brute-force**: BOMBE would try many keys until the correct one was found



BOMBE machine

Legacy of Enigma

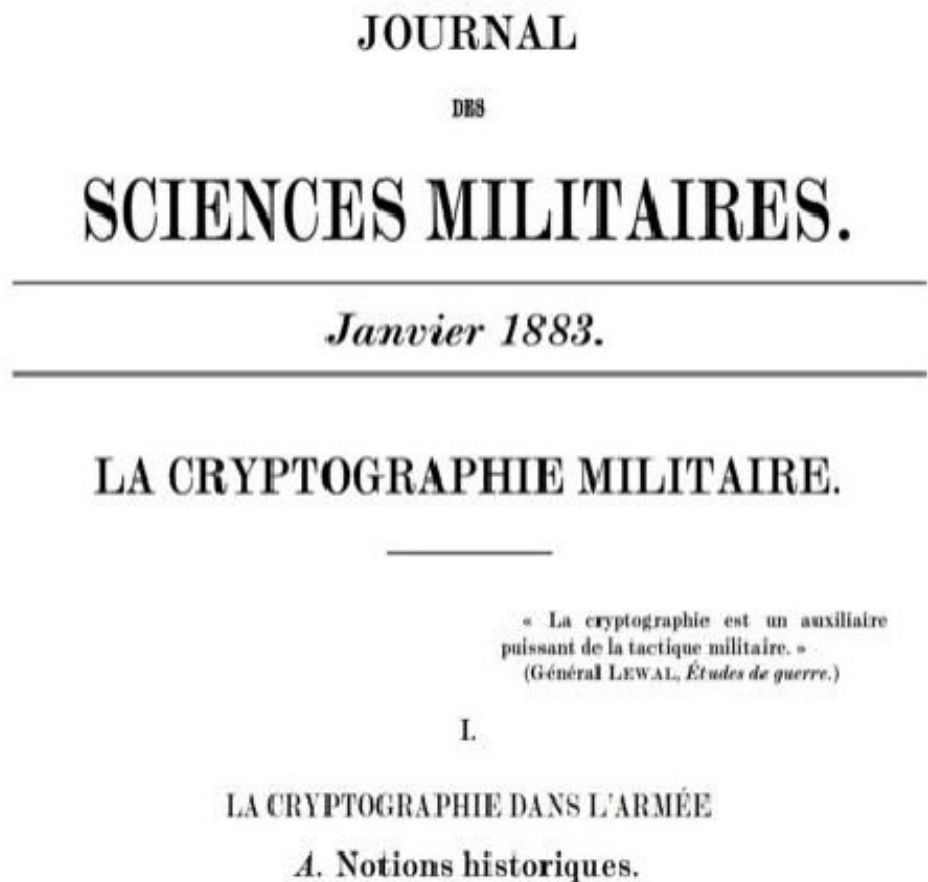
- Alan Turing, one of the cryptographers who broke Enigma, would go on to become one of the founding fathers of computer science
- Most experts agree that the Allies breaking Enigma shortened the war in Europe by about a year



Alan Turing

Kerckhoffs' principle

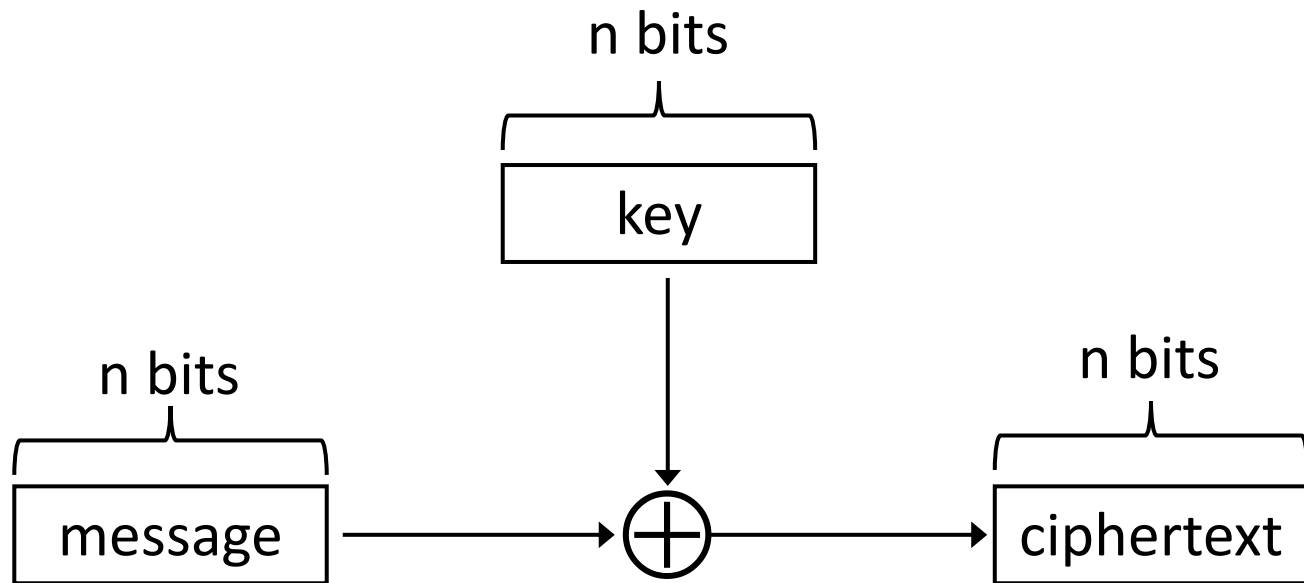
Encryption and Decryption and Key Generation Algorithm are publicly known. *The only unknown is the shared secret key*



Wikipedia

One Time Pad (1920s)

- Fix vulnerability in Vigenère cipher: use very long keys
- Key is a random string: **at least as long as the plaintext**
- Plaintext: Message that is n bits long
- Key: $\{0, 1\}^n$ sequence of n bits chosen uniformly at random.



Review: XOR

The XOR operator takes two bits and outputs one bit

$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$

Useful properties of XOR

$$x \oplus 0 = x$$

$$x \oplus x = 0$$

$$x \oplus y = y \oplus x$$

$$(x \oplus y) \oplus z = x \oplus (y \oplus z)$$

$$(x \oplus y) \oplus x = y$$

Review: XOR Algebra

Algebra works on XOR too

$y \oplus 1 = 0$	Goal: Solve for y
$y \oplus 1 \oplus 1 = 0 \oplus 1$	XOR both sides by 1
$y = 1$	Simplify with identities

One-Time Pads: Key Generation

Alice

K	0	1	1	0	0	1	0	1	0	1	1	1
-----	---	---	---	---	---	---	---	---	---	---	---	---

The key K is a randomly-chosen bitstring.

Recall: We are in the symmetric-key setting, so we'll assume Alice and Bob both know this key.

One-Time Pads: Encryption

Alice

K	0	1	1	0	0	1	0	1	0	1	1	1
M	1	0	0	1	1	0	0	1	0	1	0	0

The plaintext M is the bitstring that Alice wants to encrypt.

Idea: Use XOR to scramble up M with the bits of K .

One-Time Pads: Encryption

Alice												
K	0	1	1	0	0	1	0	1	0	1	1	1
	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus
M	1	0	0	1	1	0	0	1	0	1	0	0
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
C	1	1	1	1	1	1	0	0	0	0	1	1

Encryption algorithm: XOR each bit of K with the matching bit in M .

The ciphertext C is the encrypted bitstring that Alice sends to Bob over the insecure channel.

One-Time Pads: Decryption

Bob

K	0	1	1	0	0	1	0	1	0	1	1	1
C	1	1	1	1	1	1	0	0	0	0	1	1

Bob receives the ciphertext C . Bob knows the key K . How does Bob recover M ?

One-Time Pads: Decryption

Bob

K	0	1	1	0	0	1	0	1	0	1	1	1
	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus	\oplus
C	1	1	1	1	1	1	0	0	0	0	1	1
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
M	1	0	0	1	1	0	0	1	0	1	0	0

Decryption algorithm: XOR each bit of K with the matching bit in C .

Cryptography by Computers

- The modern era of cryptography started after WWII, with the work of Claude Shannon
- “New Directions in Cryptography” (1976) showed how number theory can be used in cryptography
 - Its authors, Whitfield Diffie and Martin Hellman, won the Turing Award in 2015 for this paper
- *This is the era of cryptography we’ll be focusing on.*



How Cryptosystems work today

- Layered approach:
 - Cryptographic protocols (e.g., CBC mode encryption)
 - Built on: Cryptographic primitives (block ciphers)
- Flavors of cryptography:
 - Symmetric: private key
 - Asymmetric: public key
- Public algorithms: Kerckhoff's principle
- *Security proofs based on assumptions (not this course)*
- **Warning! Here be dragons!**
 - careful about inventing your own
 - Use vetted libraries to apply crypto algorithms



Cryptosystem Stack

- Primitives:
 - AES/DES
 - ESA / ElGamal / Elliptic Curve
- Modes
 - Block mode (CBC, ECV, CTR, GCM..)
 - Padding structures
- Protocols:
 - TLS, SSL, SSH
- Usage of protocols:
 - Browser security
 - Secure remote logins

Flavors of Cryptography

- Symmetric cryptography
 - Both communicating parties have access to a shared random string K , called the key.
- Asymmetric cryptography
 - Each party creates a public key p_k and a secret key s_k
 - Inventors won Turing Award!

Kerckhoff's Principle

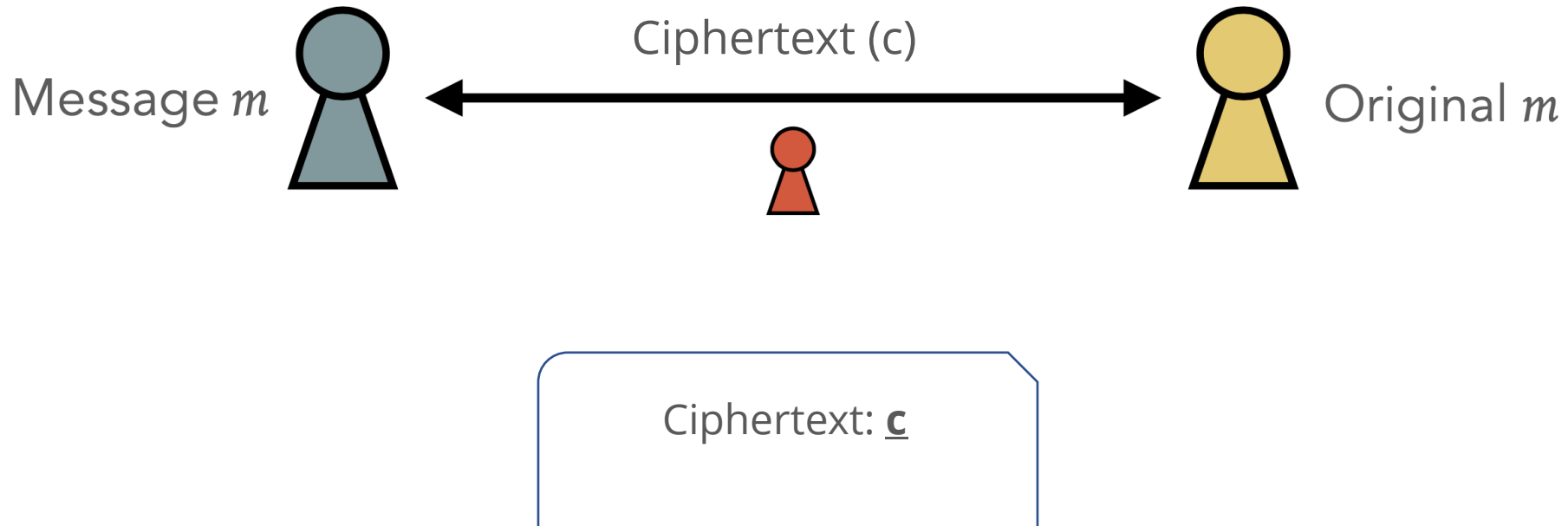
- Security of a cryptographic object should depend only on the secrecy of the secret (private) key.
- Security should not depend on the secrecy of the algorithm itself.
- Foreshadow: Need for randomness – the key to keep private

Modern Cryptography

any new cryptographic construction should be proven secure with respect to a specific definition, and a set of clearly stated assumptions

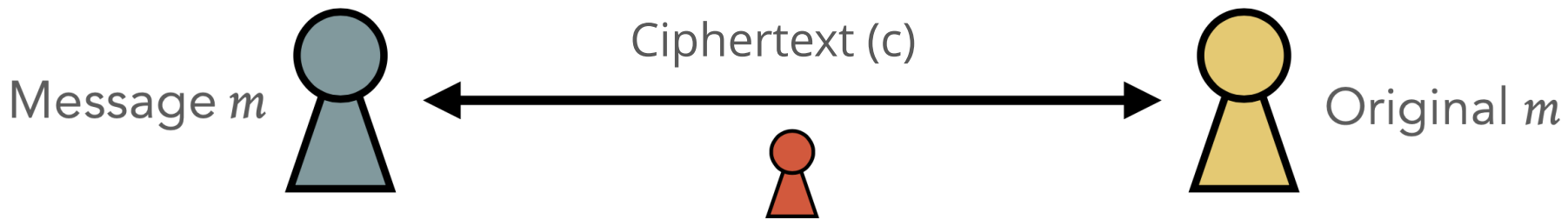
Threat Models

- **Ciphertext-only attack:** An attacker (Eve) observes ciphertext and nothing else
 - Can Eve observe more than one ciphertext?
 - this distinction can make a big difference!



Threat Models

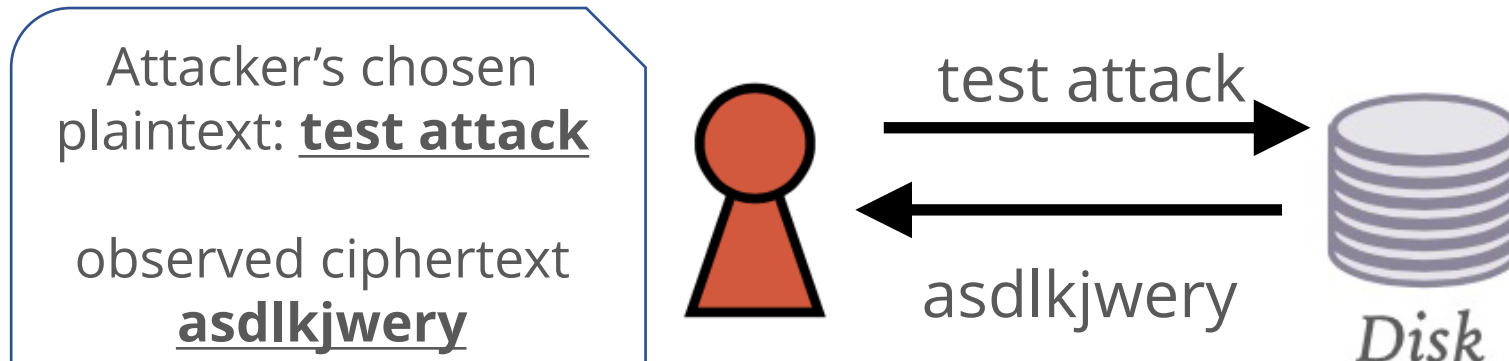
- **Known-Plaintext Attack:** An attacker (Eve) observes ciphertext and knows underlying plaintext
 - e.g., Alice: plaintext: Hello! ciphertext: 23asdf1941
 - Bob: plaintext: Hello! ciphertext: 23asdf1941



Alice's plaintext: Hello!
ciphertext: 23asdf1941

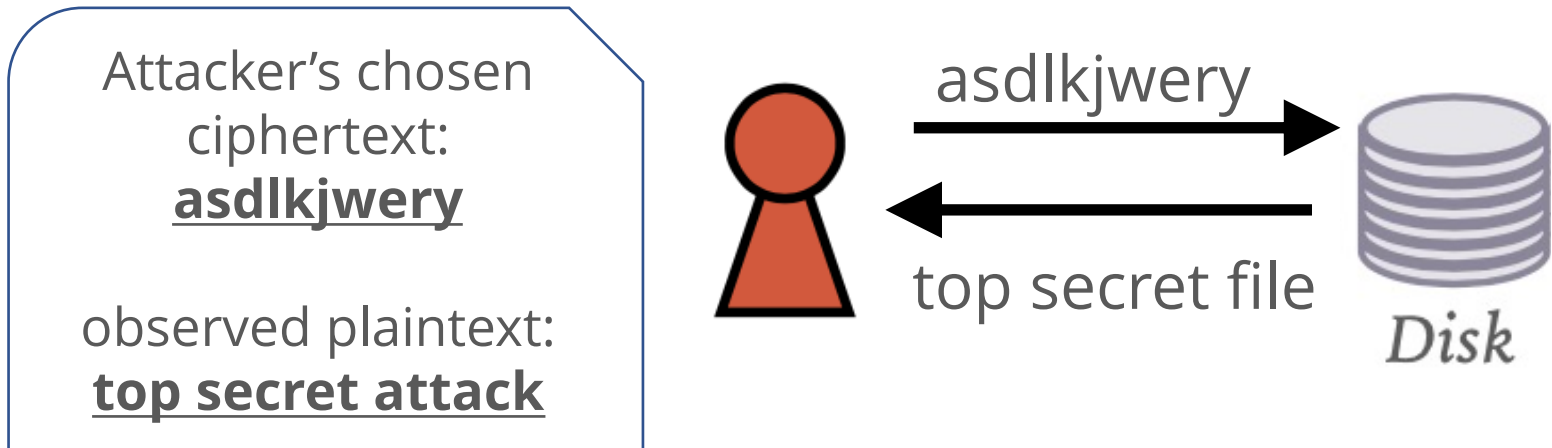
Threat Models

- **Chosen-Plaintext Attack:**
 - Observe one or more ciphertext, where plaintext is unknown
 - Also observe ciphertext for plaintext of attacker's choosing.



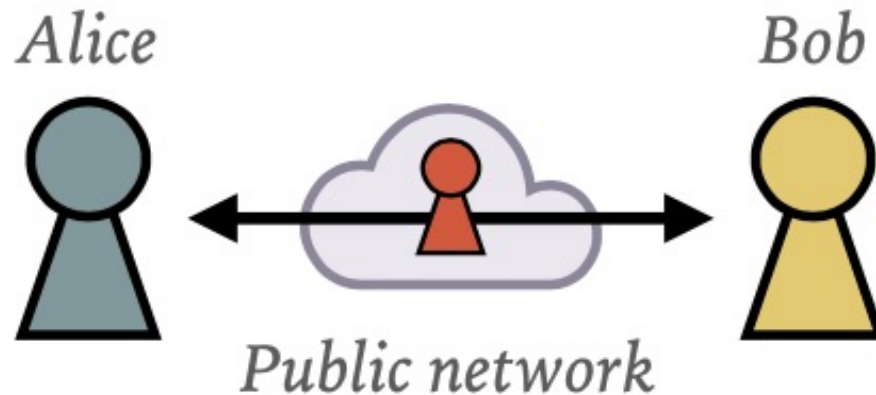
Threat Models

- **Chosen-Ciphertext Attack:**
 - Attacker is able to get the parties to decrypt certain cipher texts of that attacker's choice.



A Perfectly Secure Encryption Scheme

*Regardless of any prior information the attacker has about the plaintext
the ciphertext observed by the attacker
should **leak no additional information** about the plaintext.*



Alice can only observe one ciphertext going over the network

A Secure Encryption Scheme

An encryption scheme given by: (key gen alg., encryption alg, decryption alg.) over message space \mathbf{M} is perfectly secure iff

\forall probability distribution over \mathbf{M}

\forall message $m \in \mathbf{M}$

\forall ciphertext $c \in \mathbf{C}$ for which $\Pr[\mathbf{C} = c] > 0$

we have

$$\Pr[\mathbf{M} = m \mid \mathbf{C} = c] = \Pr[\mathbf{M} = m]$$

One Time Pad: Perfectly Secure?

- OTP achieves Perfect Secrecy
 - Shannon or Information Theoretic Security
 - Basic idea: **ciphertext reveals no “additional information” about plaintext**

Proving Perfect Security: One Time Pads

Problem Statement:

- Suppose Alice has sent one of two messages M_0 or M_1 , and Eve has no idea which was sent.
- Eve tries to guess which was sent by looking at the ciphertext.

To Show:

- Eve's probability of guessing correctly is $\frac{1}{2}$
- This is no different than it would be if she had not intercepted the ciphertext at all.

Proving Perfect Security: One Time Pads

Alice randomly chooses a bit string $\in \{0, 1\}^n$, and Alice sends the encryption of M_b .

If Eve observes that the ciphertext has some specific value C , what is the conditional probability that b = the input bit string given her observation?

Fix arbitrary distribution over $\mathbf{M} = \{0,1\}^n$, and arbitrary $m, c \in \{0,1\}^n$

$$\Pr[M = m \mid C = c] = ?$$

$$= \Pr[C = c \mid M = m] \cdot \Pr[M = m] / \Pr[C = c]$$

$$\Pr[M = m \mid C = c] = ?$$

$$= \Pr[C = c \mid M = m] \cdot \Pr[M = m] / \Pr[C = c]$$

$$= \Pr[K = m \oplus c] \cdot \Pr[M = m] / 2^{-n}$$

$$= 2^{-n} \cdot \Pr[M = m] / 2^{-n}$$

$$= \Pr[M = m]$$

$$\Pr[C = c]$$

$$= \sum_{m'} \Pr[C = c \mid M = m'] \cdot \Pr[M = m']$$

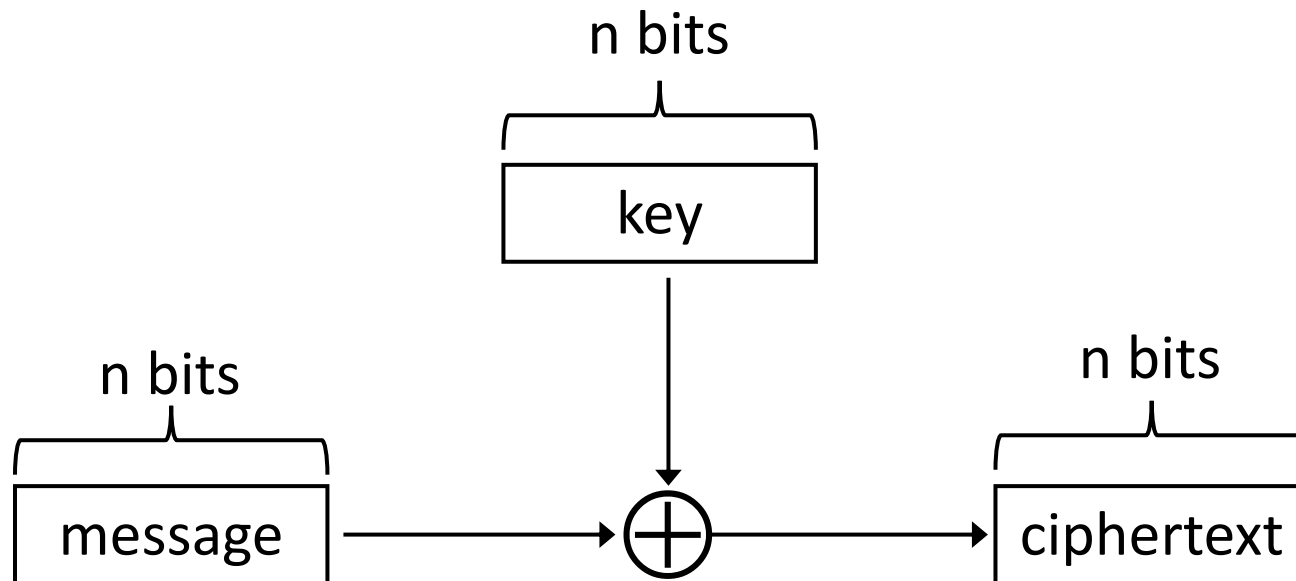
$$= \sum_{m'} \Pr[K = m' \oplus c] \cdot \Pr[M = m']$$

$$= \sum_{m'} 2^{-n} \cdot \Pr[M = m']$$

$$= 2^{-n}$$

One Time Pad: Limitations

- *The key is as long as the message*
- Only secure if **each key is used to encrypt a single message**
- Parties must share keys of (total) length = the (total) length of all the messages they might ever send!



Using the same key twice?

$$\text{Say } c_1 = k \oplus m_1$$

$$c_2 = k \oplus m_2$$

Attacker can compute

$$c_1 \oplus c_2 = (k \oplus m_1) \oplus (k \oplus m_2) = m_1 \oplus m_2$$

This leaks information about m_1, m_2 !

Limitations of Perfect Security

Regardless of any prior information the attacker has about the plaintext the ciphertext observed by the attacker should **leak no additional information** about the plaintext.

- The key is as long as the message
- Only secure if each key is used to encrypt a single message

Limitations are not only of One Time Pads, but inherent to any perfectly secure encryption scheme.

Assumes the attacker as unlimited computational power

Computational Security/Secrecy

Would be okay if a scheme leaked information with a **tiny probability** to eavesdroppers with **bounded computational resources**.

- Allowing security to fail with a tiny probability (negligible in key length n)
 - how tiny is tiny? 2^{-60} : probability of an event occurring every 100 billion years!
- Only consider efficient attackers (bounded in polynomial time by key length)
 - attackers that can brute-force the key space in bounded time.
 - try testing 2^{112} keys? Would take a supercomputer since Big Bang!
 - modern key space? 2^{128} or more!

Computational Secrecy: One Time Pads

Key Insight: Randomness –

- something an adversary won't know, can't predict and can't figure out.

Randomness



Explicit Uses of Randomness:

- Generate secret cryptographic keys
- Generate random initialization vectors or nonces for encryption

Use cases

- Generate passwords for new users
- Shuffle the order of votes in an electronic voting machine
- Shuffle cards etc. (for online games)

What does “random” mean?

- What does “uniform” mean?
- Which of the following is a uniform string?
 - 0101010101010101
 - 0010111011100110
 - 0000000000000000
- If we generate a uniform string, each of the above occurs with probability 2^{-16}

What does “random” mean?

- “Randomness” is not a property of a string, but a property of a distribution.
- The uniform distribution on n -bit strings is the distribution U_n where
 - $U_n(x) = 2^{-n}$ for all $x \in \{0,1\}^n$

What does “pseudorandom” mean?

- Informal: cannot be distinguished from uniform (i.e., random)
- Which of the following is pseudorandom?
 - 0101010101010101
 - 0010111011100110
 - 0000000000000000
- Pseudorandomness is a property of a distribution

How Random is "Random"?

OXFFFFFFF EVERY TIME IS OXDEADBEEF —

How a months-old AMD microcode bug destroyed my weekend [UPDATED]

AMD shipped Ryzen 3000 with a serious microcode bug in its random number generator.

JIM SALTER - 10/29/2019, 7:00 AM



Uptime

IT and business computing insights

Crypto shocker: four of every 1,000 public keys provide no security (updated)

By Dan Goodin | Published 7 days ago

WORLD U.S. N.Y. / REGION BUSINESS TECHNOLOGY SCIENCE HEALTH SPORTS OPINION A

Flaw Found in an Online Encryption Method

JOHN MARKOFF

Published: February 14, 2012

SAN FRANCISCO — A team of European and American mathematicians and cryptographers have discovered an unexpected weakness in the encryption system widely used worldwide for online shopping, banking, e-mail and other Internet services intended to remain private and secure.

RECOMMEND

TWITTER

LINKEDIN

COMMENTS
(127)

E-MAIL

C's built-in rand() function

```
unsigned long int next = 1;
/* rand: return pseudo-random integer on 0..32767 */
int rand(void){
    next = next * 11-3515245 + 12345;
    return (unsigned int) (next/65536) % 32768;
}
/* srand: set seed for rand() */
void srand(unsigned int seed){
    next = seed;
}
```

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin." -- John von Neumann

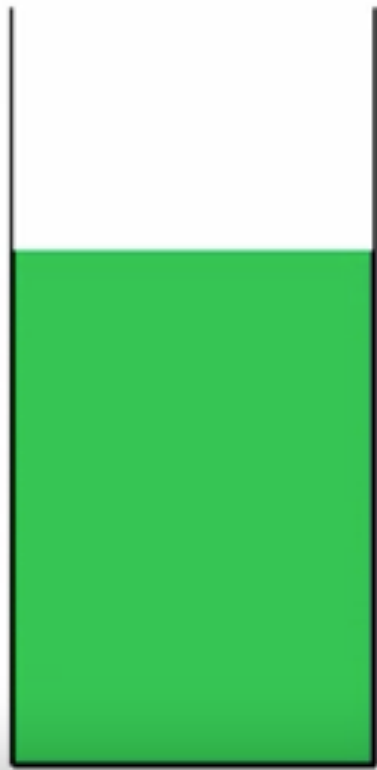
Random-number generation

- Two steps:
 1. Continually collect a “pool” of high-entropy (i.e., “unpredictable”) data from external inputs
 1. Delays between network events
 2. Hard-disk access times
 3. Keystroke/mouse movements
 2. When random bits are requested, process this data to generate a sequence of uniform, independent bits/bytes
 - May “block” if insufficient entropy available
- Other... – Hardware random-number generator (e.g., Intel)

How might we get “good” random numbers?

- For security applications, want “cryptographically secure pseudorandom numbers”
- Libraries include cryptographically secure pseudorandom number generators (CSPRNG)
- Linux:
 - `/dev/random`: blocking: waits for enough entropy
 - `/dev/urandom`: nonblocking, possibly less entropy
 - `getrandom()` – syscall! – by default blocking
- Internally:
 - Entropy pool: gathered from multiple sources
 - e.g.: mouse/keyboard/network timings
- Better idea:
 - AMD/Intel’s on-chip random number generator: RDRAND
 - Hopefully no hardware bugs!

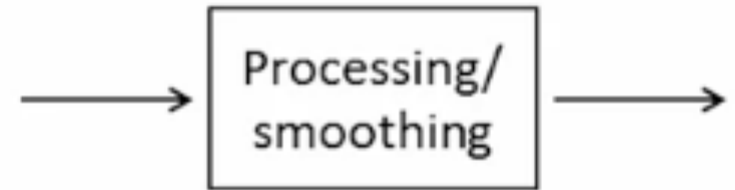
Random-number generation



Request random bits
←

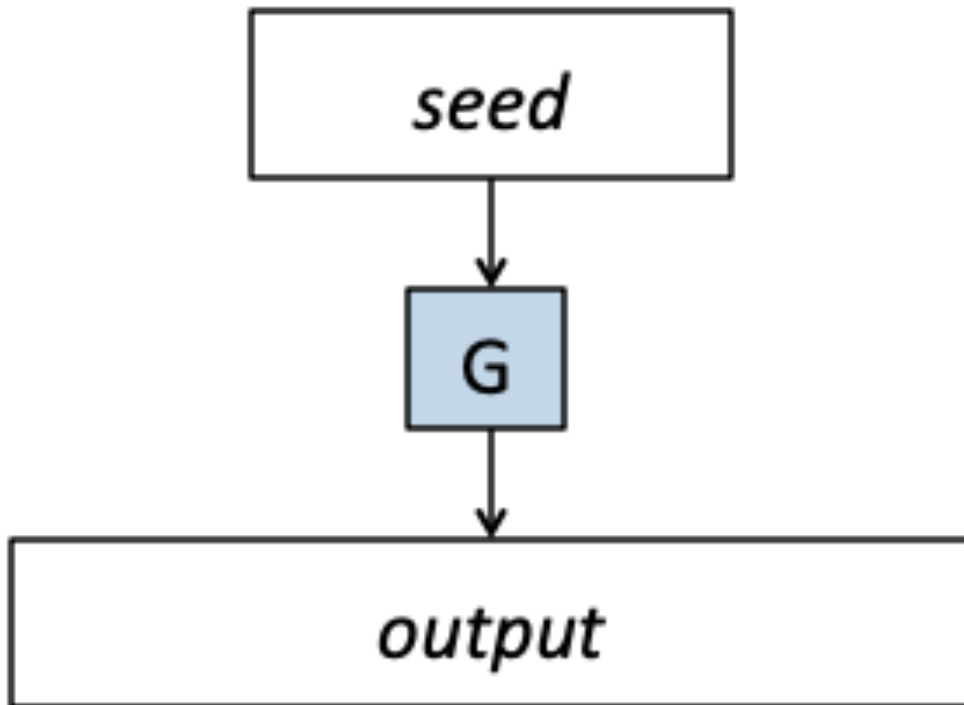


Request random bits
←



Pseudorandom (number) generators: PRG/PRNGs

- A PRG is an efficient, **deterministic algorithm that expands a short, uniform seed into a longer, pseudorandom output**
 - Useful whenever you have a “small” number of true random bits, and want lots of “random looking” bits



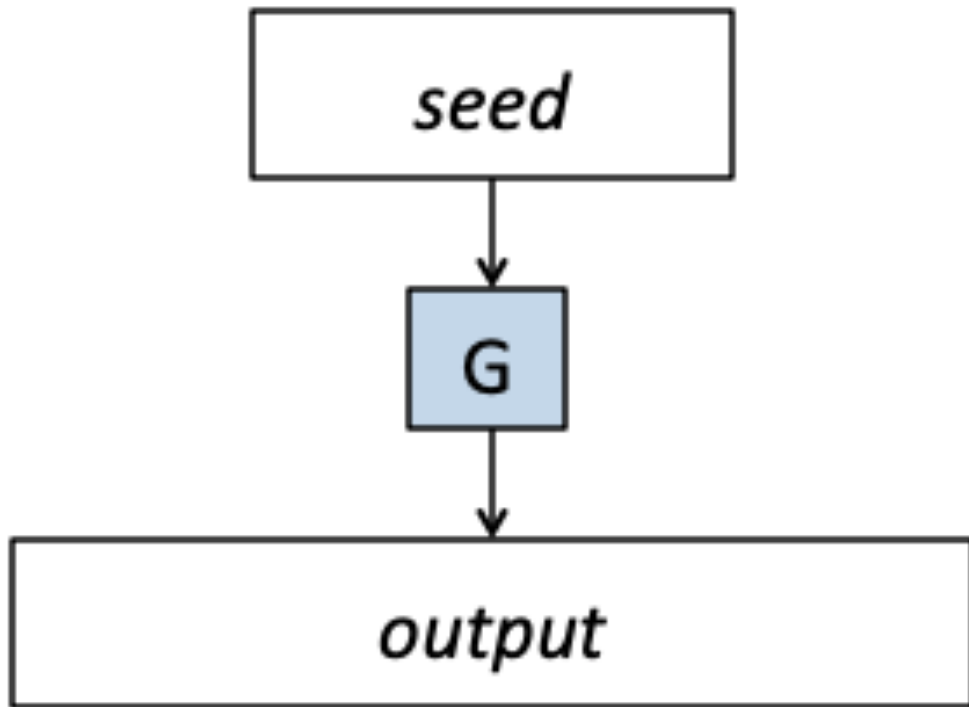
seed: a small number of true random bits

G: deterministic polynomial time algorithm

output: pseudorandom bits of length n , i.e., cannot be distinguished from truly random bits, by any efficient statistical test.

Do PRNGs exist?

- We actually don't know!
- Assume that there exist *some* functions G that are PRNG.

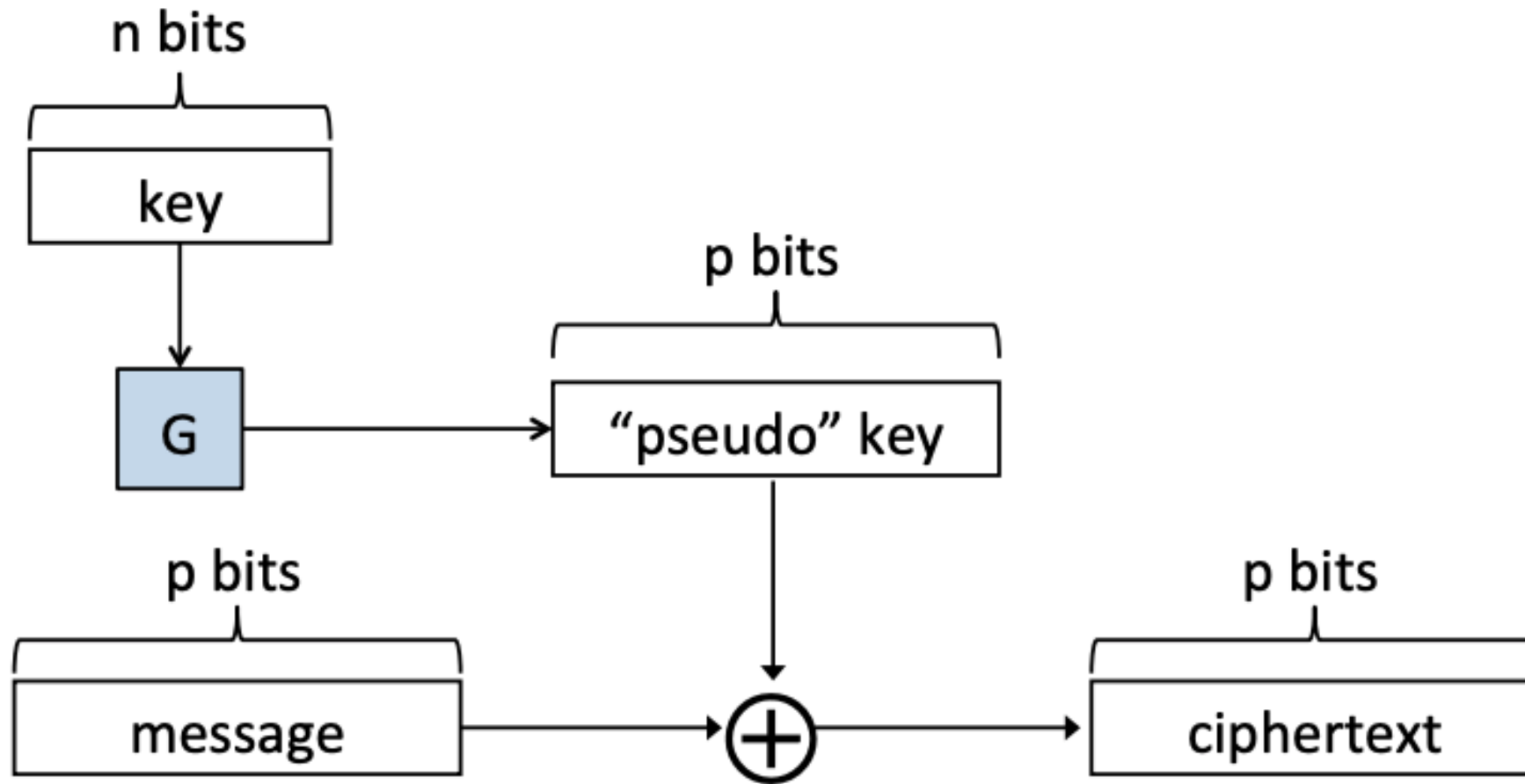


seed: a small number of true random bits

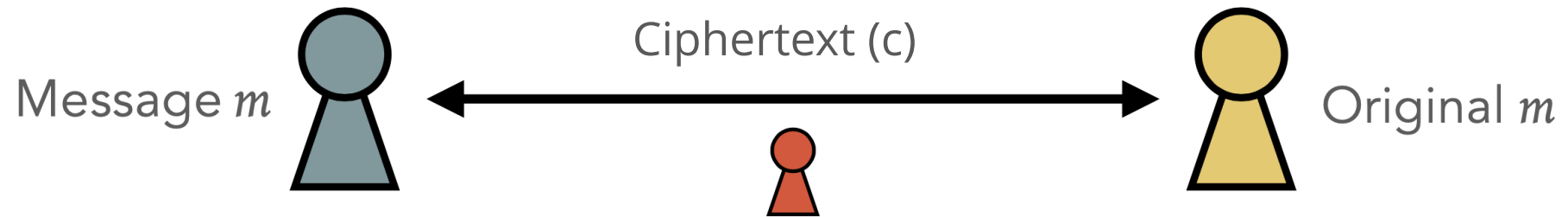
G : deterministic polynomial time algorithm

output: pseudorandom bits of length n , i.e., cannot be distinguished from truly random bits, by any efficient statistical test.

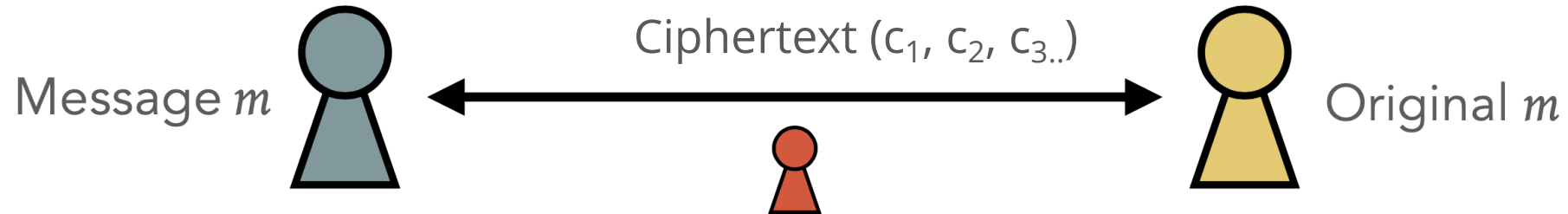
Applying Pseudo-randomness to the one-time pad



Single-message secrecy



Multiple message secrecy



- We are not going to formally define a notion of multiple-message secrecy
- Instead, define something stronger: security against chosen-plaintext attacks (CPA-security)
 - *minimal notion of security an encryption scheme should satisfy*

Security against Chosen Plaintext Attack: Impossible?



It really is a problem if an attacker can tell when the same message is encrypted twice!

This attack only works if encryption is deterministic!