Announcements

- Complete Homework 0 if you haven’t

- Teaching feedback welcome

Last Time: Introduction and Security Principles

- Course Logistics

- Security Principles

Today’s Cunning Plan: Cryptography

- Introduction to Cryptography
  - What is cryptography?
  - Definitions of key properties
  - IND-CPA security
  - A brief history of cryptography
  - Symmetric-key crypto
    - One-time pads

What is cryptography?

What is Cryptography?

- **Older definition**: The study of secure communication over insecure channels
- **Newer definition**: Provide rigorous guarantees on the security of data and computation in the presence of an attacker
  - Not just *confidentiality* but also *integrity* and *authenticity* (we'll see these definitions today)
- Modern cryptography is heavily based on math
  - The math is hard
  - We won't cover the details. Wikipedia is a great resource
  - Some system / algorithms are provably secure

Definitions of Key Properties

Meet Alice, Bob, Eve, and Mallory

- Alice and Bob: The main characters trying to send messages to each other over an insecure communication channel

- Eve: An **eavesdropper** who can read any data sent over the channel

- Mallory: A **manipulator** who can read and modify any data sent over the channel

Meet Alice, Bob, Eve, and Mallory

- We often describe cryptographic problems using a common cast of characters
- One scenario:
  - Alice wants to send a message to Bob.
  - However, Eve is going to *eavesdrop* on the communication channel.
  - How does Alice send the message to Bob without Eve learning about the message?
- Another scenario:
  - Bob wants to send a message to Alice.
  - However, Mallory is going to *tamper* with the communication channel.
  - How does Bob send the message to Alice without Mallory changing the message?

Three Goals of Cryptography

- In cryptography, there are three main properties that we want on our information
- **Confidentiality**: An adversary cannot *read* our messages.
- **Integrity**: An adversary cannot *change* our messages without being detected.
- **Authenticity**: I can prove that this message came from the person who claims to have written it.
  - Integrity and authenticity are closely related properties…
    - Before I can prove that a message came from a certain person, I have to prove that the message wasn't changed!

Keys

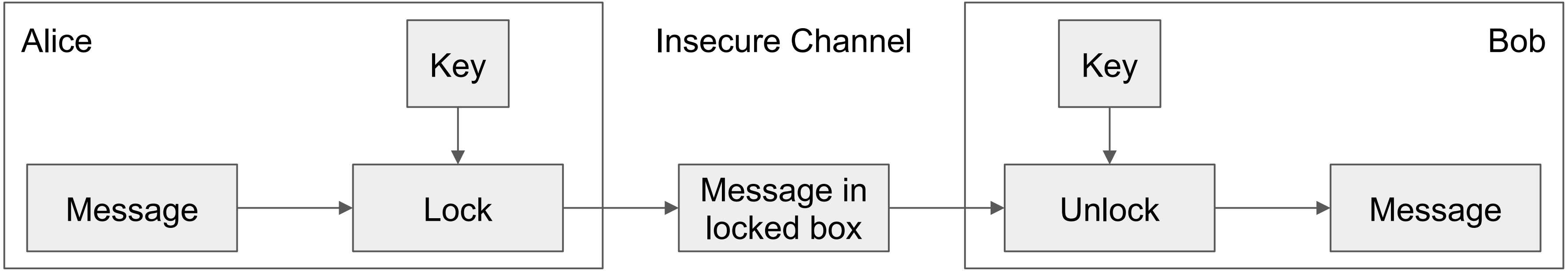
- The most basic building block of any cryptographic scheme: The **key**
- We use the key in our algorithms to secure messages
- Two models of keys:
  - **Symmetric key model**: Alice and Bob both know the value of the same secret key.
    - Example: One-time pad, AES encryption
  - **Asymmetric key model**: Everybody has two keys, a secret key and a public key.
    - Example: RSA encryption

Security Principle: Kerckhoff’s Principle

- This principle is closely related to Shannon’s Maxim
  - Don’t use security through obscurity. Assume the attacker knows the system.
- Kerckhoff’s principle says:
  - Cryptosystems should remain secure even when the attacker knows all internal details of the system
  - The key should be the only thing that must be kept secret
  - The system should be designed to make it easy to change keys that are leaked (or suspected to be leaked)
    - If your secrets are leaked, it is usually a lot easier to change the key than to replace every instance of the running software
- Our assumption: The attacker knows all the algorithms we use. The only information the attacker is missing is the secret key(s).

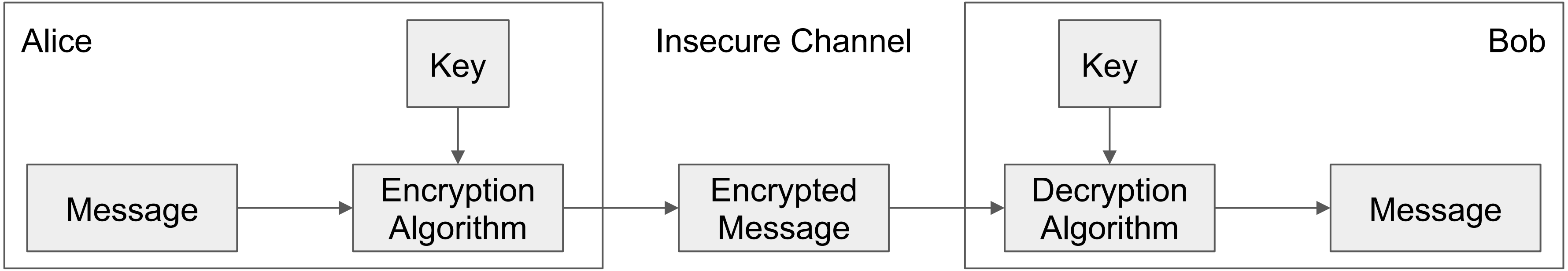
Confidentiality

- **Confidentiality**: An adversary cannot *read* our messages.
- Analogy: Locking and unlocking the message
  - Alice uses the key to lock the message in a box
  - Alice sends the message (locked in the box) over the insecure channel
  - Eve sees the locked box, but cannot access the message without the key
  - Bob receives the message (locked in the box) and uses the key to unlock the message



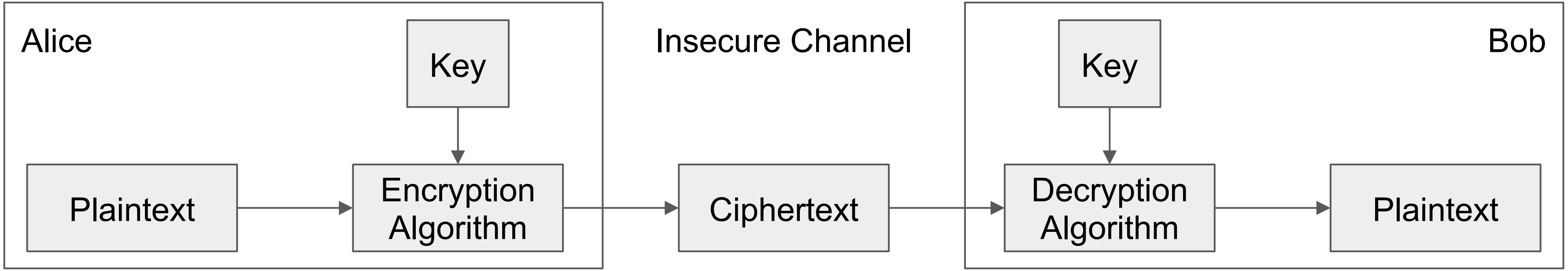
Confidentiality

- Schemes provide confidentiality by **encrypting** messages
  - Alice uses the key to **encrypt** the message: Change the message into a scrambled form
  - Alice sends the encrypted message over the insecure channel
  - Eve sees the encrypted message, but cannot figure out the original message without the key
  - Bob receives the encrypted message and uses the key to **decrypt** the message back into its original form



Confidentiality

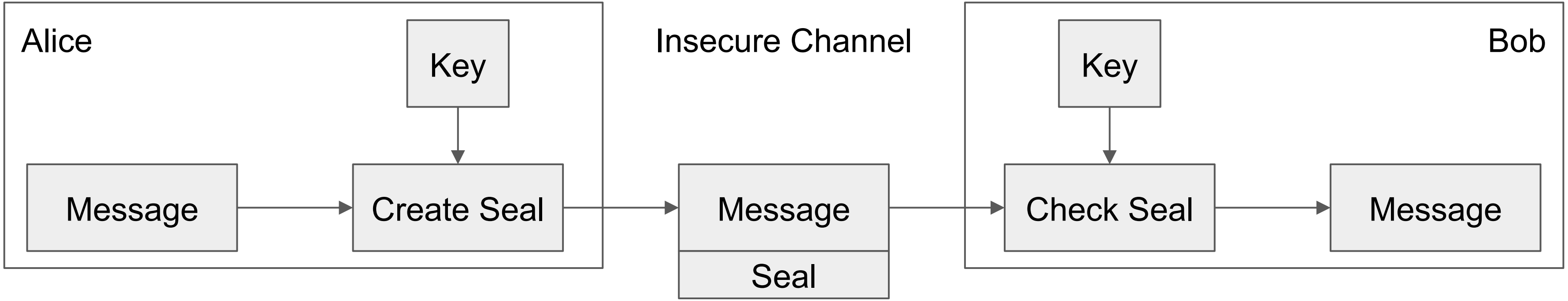
- **Plaintext**: The original message
- **Ciphertext**: The encrypted message



Integrity (and Authenticity)

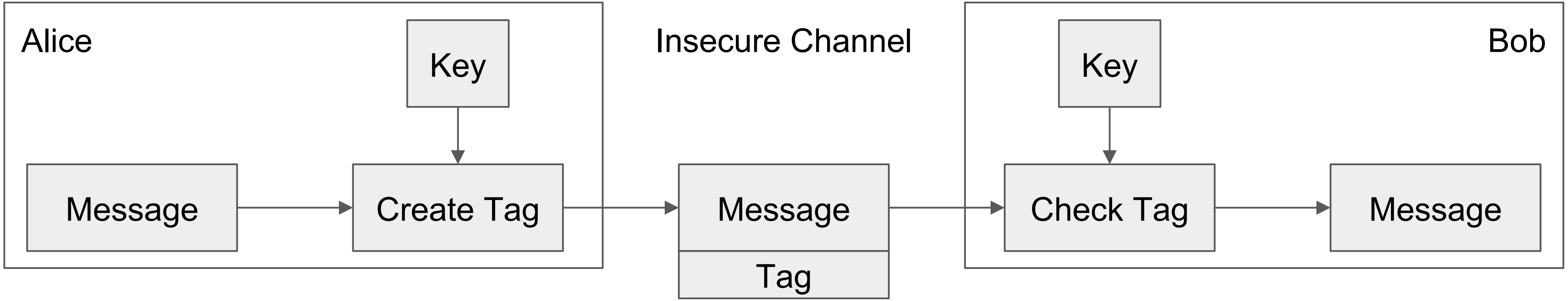
ITIS 6200 / 8200

- **Integrity**: An adversary cannot *change* our messages without being detected.
- Analogy: Adding a seal on the message
  - Alice uses the key to add a special seal on the message (e.g. puts tape on the envelope)
  - Alice sends the message and the seal over the insecure channel
  - If Mallory tampers with the message, she'll break the seal (e.g. break the tape on the envelope)
  - Without the key, Mallory cannot create her own seal
  - Bob receives the message and the seal and checks that the seal has not been broken



Integrity (and Authenticity)

- Schemes provide integrity by adding a **tag** or **signature** on messages
    - Alice uses the key to generate a special tag for the message
    - Alice sends the message and the tag over the insecure channel
    - If Mallory tampers with the message, the tag will no longer be valid
    - Bob receives the message and the tag and checks that the tag is still valid



Threat Models (Types of Cryptanalysis)

- Ciphertext only
  - Attackers only have ciphertext, need to decrypt it
- Known-plaintext
  - Attackers know **some** ciphertext and its corresponding plaintext, needs to decrypt some other cyphertext
- Chosen-plaintext
  - Attackers can get **any** ciphertext they want encrypted, and need to decrypt some specific cyphertext
  - e.g., Eve can trick Alice into encrypting arbitrary messages of Eve's choice
- In this class, we often use the chosen plaintext attack (CPA) model

Cryptography Roadmap

	Symmetric-key	Asymmetric-key
Confidentiality	• One-time pads • Block ciphers with chaining modes (e.g. AES-CBC) • Stream ciphers	• RSA encryption • ElGamal encryption
Integrity, Authentication	• MACs (e.g. HMAC)	• Digital signatures (e.g. RSA signatures)

- Hash functions
- Pseudorandom number generators
- Public key exchange (e.g. Diffie-Hellman)
- Key management (certificates)
- Password management

Symmetric-Key Encryption

Cryptography Roadmap

	Symmetric-key	Asymmetric-key
Confidentiality	• One-time pads • Block ciphers with chaining modes (e.g. AES-CBC) • Stream ciphers	• RSA encryption • ElGamal encryption
Integrity, Authentication	• MACs (e.g. HMAC)	• Digital signatures (e.g. RSA signatures)

- Hash functions
- Pseudorandom number generators
- Public key exchange (e.g. Diffie-Hellman)
- Key management (certificates)
- Password management

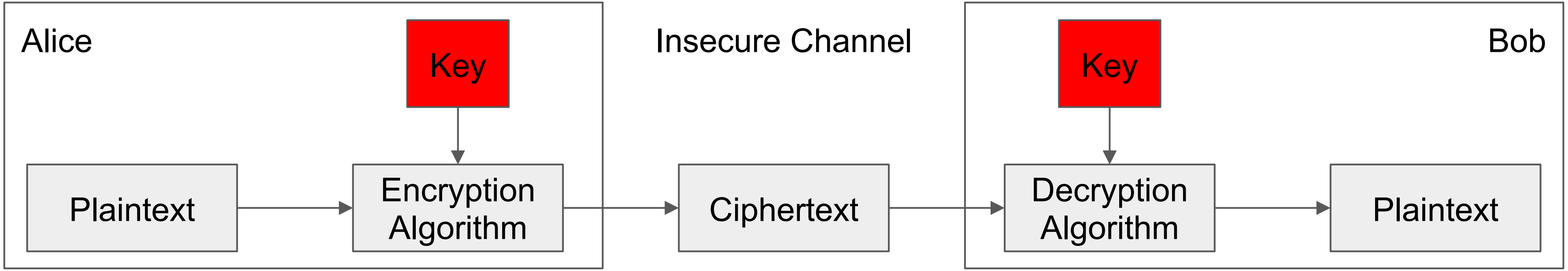
Symmetric-Key Encryption

- The next few schemes are symmetric-key encryption schemes
  - **Encryption schemes** aim to provide *confidentiality* (but not integrity or authentication)
  - **Symmetric-key** means Alice and Bob share the same secret key that the attacker doesn't know
    - Don't worry about how Alice and Bob share the key for now

- For modern schemes, we're going to assume that messages are *bitstrings*
  - **Bitstring**: A sequence of bits (0 or 1), e.g. `110101010010010`
  - Text, images, etc. can usually be converted into bitstrings before encryption, so bitstrings are a useful abstraction. After all, everything in a computer is just a sequence of bits!

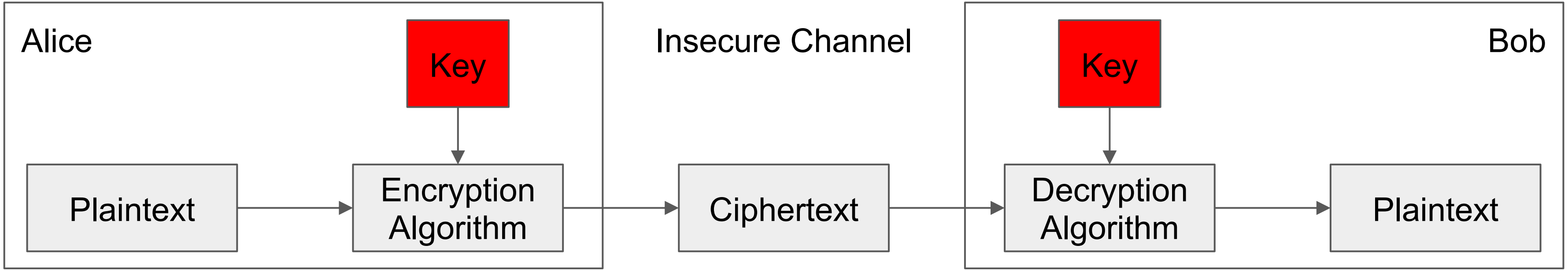
Symmetric-Key Encryption: Definition

- A symmetric-key encryption scheme has three algorithms:
  - KeyGen() → K: Generate a key K
  - Enc(K, M) → C: Encrypt a **plaintext** M using the key K to produce **ciphertext** C
  - Dec(K, C) → M: Decrypt a ciphertext C using the key K



Symmetric-Key Encryption: Definition

- What properties do we want from a symmetric encryption scheme?
  - **Correctness**: Decrypting a ciphertext should result in the message that was originally encrypted
    - $\mathrm{Dec}(K, \mathrm{Enc}(K, M)) = M$ for all $K \leftarrow \mathrm{KeyGen}()$ and M
  - **Efficiency**: Encryption/decryption algorithms should be fast: >1 Gbps on a standard computer
  - **Security**: Confidentiality



Defining Confidentiality

- Recall our definition of confidentiality from earlier: “An adversary cannot read our messages”
  - This definition isn’t very specific
    - What if Eve can read the first half of Alice’s message, but not the second half?
    - What if Eve figures out that Alice’s message starts with “Dear Bob”?
  - This definition doesn’t account for prior knowledge
    - What if Eve already knew that Alice’s message ends in “Sincerely, Alice”?
    - What if Eve knows that Alice’s message is “BUY!” or “SELL” but doesn't know which?

Defining Confidentiality

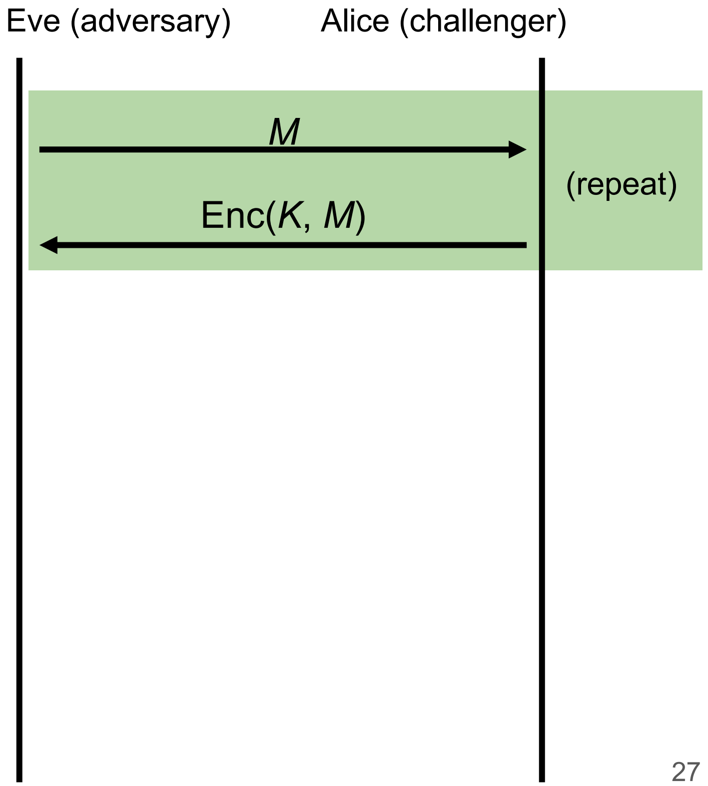
- A better definition of confidentiality: The ciphertext should not give the attacker *any **additional** information* about the plaintext.
- Let's design an experiment/security **game** to test our definition:
  - Eve chooses two messages M_0 and M_1 of the same length
  - Alice chooses one message at random M_b, encrypts it, and sends the ciphertext
  - Eve knows either M_0 or M_1 was sent, but doesn't know which
  - Eve reads the ciphertext and tries to guess which message was sent
  - If the probability that Eve correctly guesses which message was sent is 1/2, then the encryption scheme is confidential
- Intuition
  - If the scheme is confidential, Eve can only guess with probability 1/2, which is no different than if Eve hadn't sent the ciphertext at all
  - In other words: the ciphertext gave Eve no ***additional*** information about which plaintext was sent!

Defining Confidentiality: IND-CPA

- Recall our threat model: Eve can also perform a **chosen plaintext attack (CPA)**
  - It means, Eve can trick Alice into encrypting arbitrary messages of Eve's choice
  - We can adapt our experiment to account for this threat model
- A better definition of confidentiality: Even if Eve is able to trick Alice into encrypting messages, Eve can still only guess what message Alice sent with probability 1/2.
  - This definition is called **IND-CPA** (indistinguishability under chosen plaintext attack)
- Cryptographic properties are often defined in terms of “games” that an adversary can either “win” or “lose”

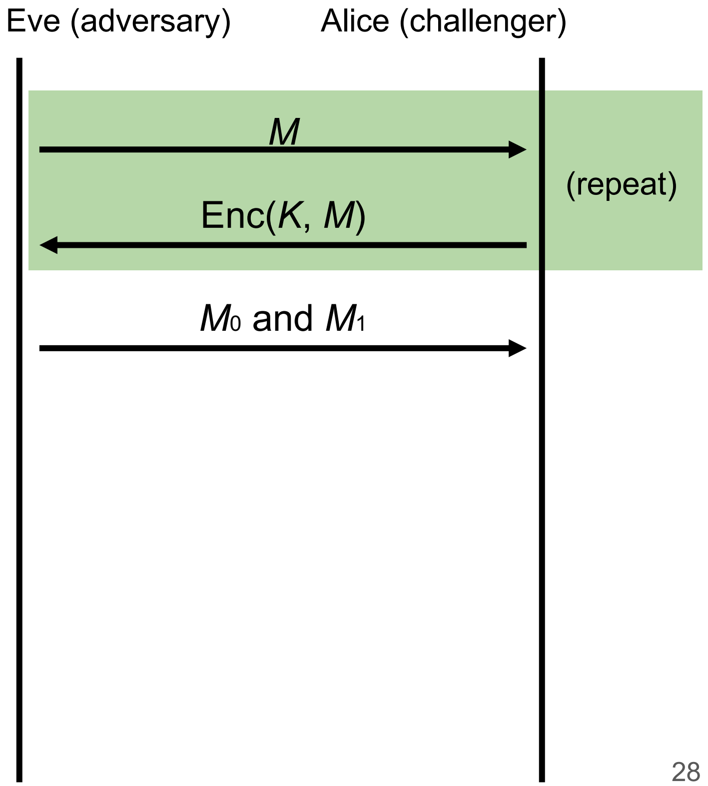
Defining Confidentiality: IND-CPA

1. Eve may choose plaintexts to send to Alice and receives their ciphertexts



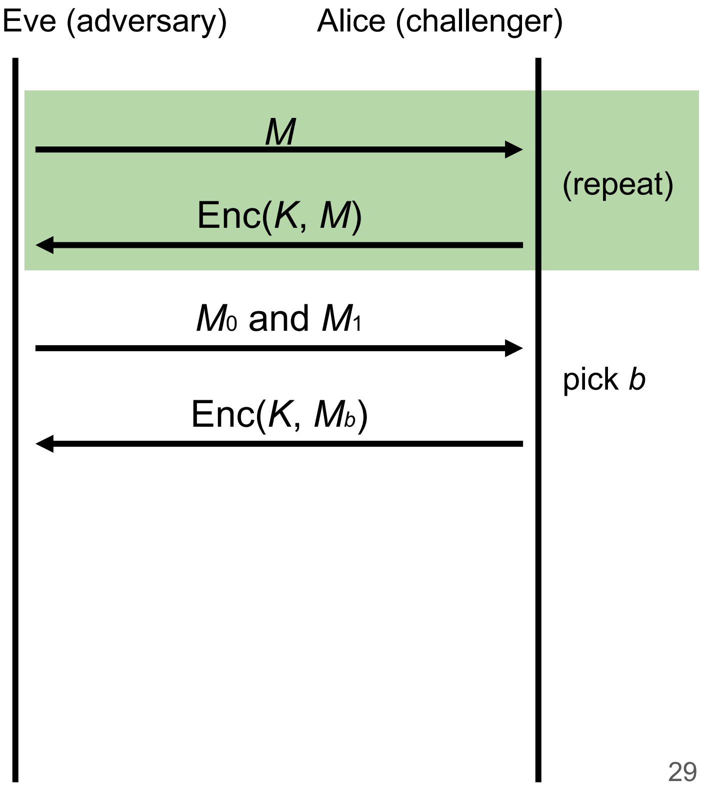
Defining Confidentiality: IND-CPA

1. Eve may choose plaintexts to send to Alice and receives their ciphertexts
2. Eve issues a pair of plaintexts M_0 and M_1 to Alice



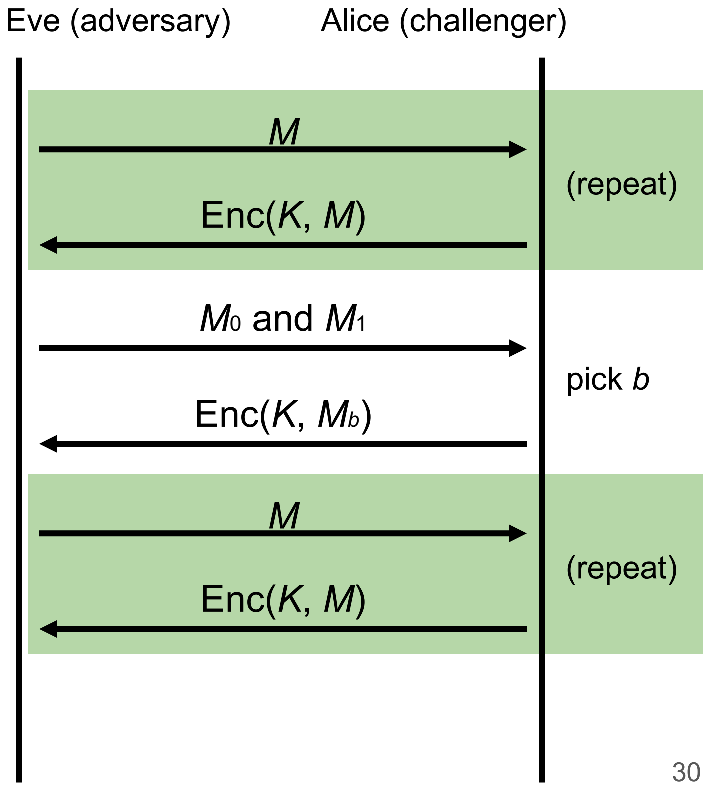
Defining Confidentiality: IND-CPA

1. Eve may choose plaintexts to send to Alice and receives their ciphertexts
2. Eve issues a pair of plaintexts M_0 and M_1 to Alice
3. Alice randomly chooses either M_0 or M_1 to encrypt and sends the encryption back
    a. Alice does not tell Eve which one was encrypted!



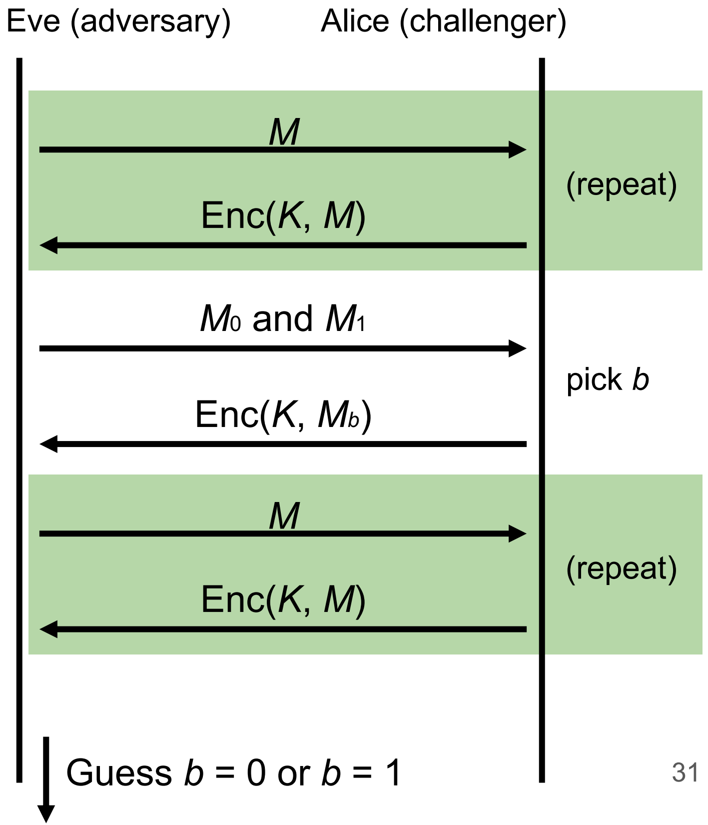
Defining Confidentiality: IND-CPA

1. Eve may choose plaintexts to send to Alice and receives their ciphertexts
2. Eve issues a pair of plaintexts M_0 and M_1 to Alice
3. Alice randomly chooses either M_0 or M_1 to encrypt and sends the encryption back
    a. Alice does not tell Eve which one was encrypted!
4. Eve may again choose plaintexts to send to Alice and receives their ciphertexts



Defining Confidentiality: IND-CPA

1. Eve may choose plaintexts to send to Alice and receives their ciphertexts
2. Eve issues a pair of plaintexts M_0 and M_1 to Alice
3. Alice randomly chooses either M_0 or M_1 to encrypt and sends the encryption back
   a. Alice does not tell Eve which one was encrypted!
4. Eve may again choose plaintexts to send to Alice and receives their ciphertexts
5. Eventually, Eve outputs a guess as to whether Alice encrypted M_0 or M_1



Defining Confidentiality: IND-CPA

- If Eve correctly guesses which message Alice encrypted, then Eve wins. Otherwise, she loses.
- How does Eve guess whether M_0 or M_1 was encrypted? What strategy does she use?
  - We don't *assume* she uses a particular strategy; Eve represents all possible strategies
- Proving insecurity: There exists at least *one* strategy that can win the IND-CPA game with probability > 1/2
  - 1/2 is the probability of winning by random guessing
  - If you can be better than random, then the ciphertext has leaked information, and Eve is able to learn it and use it to gain an advantage!
- Proving security: For *all* attackers/Eve-s, the probability of winning the IND-CPA game is at most 1/2

Edge Cases: Length

- Cryptographic schemes are (usually) allowed to leak the length of the message
  - To hide length: All messages must always be the same length
    - 16-byte messages: We can’t encrypt large messages (images, videos, etc.)!
    - 1-GB messages: Sending small messages (text, Tweets, etc.) needs 1 GB of bandwidth!
    - This is unpractical
- In the IND-CPA game: M_0 and M_1 must be the same length
  - To break IND-CPA, Eve must learn something other than message length

Edge Cases: Attacker Runtime

- Some schemes are theoretically vulnerable, but secure in any real-world setting
  - If an attack takes longer than the life of the solar system to complete, it probably won't happen!
  - Or if it would require a computer made out of a literal galaxy worth of science-fiction nanotech
- In the IND-CPA game: Eve is limited to a practical runtime
  - One common practical limit: Eve is limited to polynomial runtime algorithms (no exponential-time algorithms)

Edge Cases: Negligible Advantage

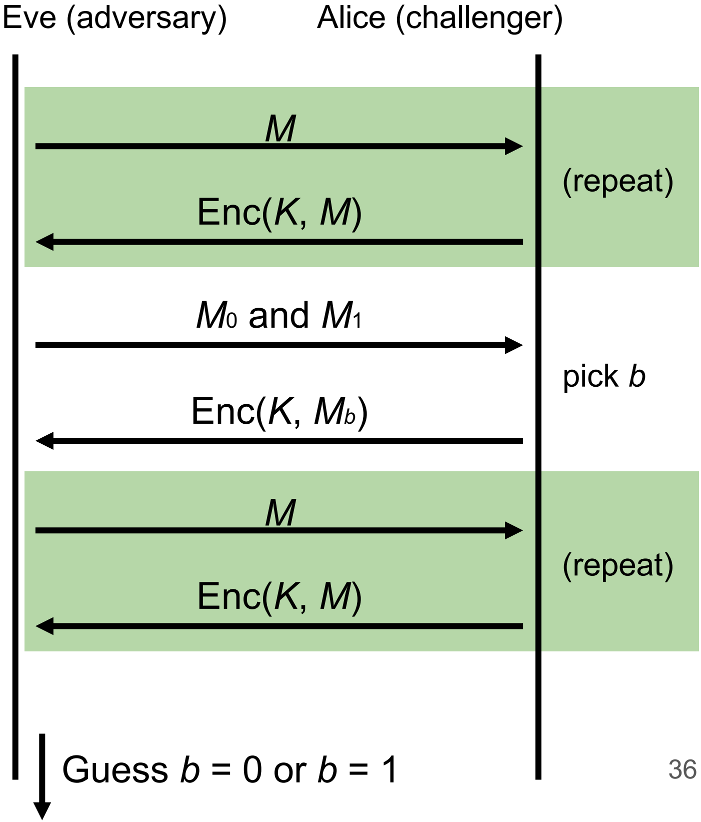
- Sometimes it's possible for Eve to win with probability $1/2 + 1/2^{128}$
  - This probability is greater than 1/2, but it's so close to 1/2 that it's as good as 1/2.
  - Eve's advantage is so small that she can't use it for any practical attacks
- In the IND-CPA game: The scheme is secure even if Eve can win with probability $\leq 1/2 + \varepsilon$, where ε is *negligible*
  - The actual mathematical definition of negligible is out of scope
  - Example: $1/2 + 1/2^{128}$: Negligible advantage
  - Example: 2/3: Non-negligible advantage

IND-CPA: Putting it together

ITIS 6200 / 8200

1. Eve may choose plaintexts to send to Alice and receives their ciphertexts
2. Eve issues a pair of plaintexts M_0 and M_1 to Alice
3. Alice randomly chooses either M_0 or M_1 to encrypt and sends the encryption back
    - Alice does not tell Eve which one was encrypted!
4. Eve may again choose plaintexts to send to Alice and receives their ciphertexts
5. Eventually, Eve outputs a guess as to whethe
encrypted M_0 or M_1

- An encryption scheme is IND-CPA secure if for all polynomial time attackers Eve:
    - Eve can win with probability ≤ 1/2 + Ɛ, where Ɛ is *negligible.*



A Brief History of Cryptography

Cryptography by Hand: Caesar Cipher

- One of the earliest cryptographic schemes was the **Caesar cipher**
  - Used by Julius Caesar over 2,000 years ago
- KeyGen():
  - Choose a key *K* randomly between 0 and 25
- Enc(*K*, *M*):
  - Replace each letter in *M* with the letter *K* positions later in the alphabet
  - If *K* = 3, plaintext DOG becomes GRJ
- Dec(*K*, *C*):
  - Replace each letter in *C* with the letter *K* positions earlier in the alphabet
  - If *K* = 3, ciphertext GRJ becomes DOG

K = 3			
M	*C*	*M*	*C*
A	D	N	Q
B	E	O	R
C	F	P	S
D	G	Q	T
E	H	R	U
F	I	S	V
G	J	T	W
H	K	U	X
I	L	V	Y
J	M	W	Z
K	N	X	A
L	O	Y	B
M	P	Z	C

Cryptography by Hand: Attacks on the Caesar Cipher

- Eve sees the ciphertext JCKN ECGUCT, but doesn’t know the key *K*
- If you were Eve, how would you try to break this algorithm?
- Brute-force attack: Try all 26 possible keys!
- Use existing knowledge: Assume that the message is in English

+1	IBJM DBFTBS	+9	ATBE VTXLTK	+17	SLTW NLPDLC
+2	$\boxed{\text{HAIL CAESAR}}$	+10	ZSAD USWKSJ	+18	RKSV MKOCKB
+3	GZHK BZDRZQ	+11	YRZC TRVJRI	+19	QJRU LJNBJA
+4	FYGJ AYCQYP	+12	XQYB SQUIQH	+20	PIQT KIMAIZ
+5	EXFI ZXBPXO	+13	WPXA RPTHPG	+21	OHPS JHLZHY
+6	DWEH YWAOWN	+14	VOWZ QOSGOF	+22	NGOR IGKYGX
+7	CVDG XVZNVM	+15	UNVY PNRFNE	+23	MFNQ HFJXFW
+8	BUCF WUYMUL	+16	TMUX OMQEMD	+24	LEMP GEIWEV
				+25	KDLO FDHVDU

Cryptography by Hand: Attacks on the Caesar Cipher

- Eve sees the ciphertext JCKN ECGUCT, but doesn't know the key K
- Chosen-plaintext attack: Eve tricks Alice into encrypting plaintext of her choice
  - Eve sends a message M = AAA and receives C = CCC
  - Eve can deduce the key: C is 2 letters after A, so K = 2
  - Eve has the key, so she can decrypt the ciphertext

Cryptography by Hand: Substitution Cipher

- A better cipher: create a mapping of each character to another character.
  - Example: A = N, B = Q, C = L, D = Z, etc.
  - Unlike the Caesar cipher, the shift is no longer constant!
- KeyGen():
  - Generate a random, one-to-one mapping of characters
- Enc(*K*, *M*):
  - Map each letter in *M* to the output according to the mapping *K*
- Dec(*K*, *C*):
  - Map each letter in *C* to the output according to the *reverse* of the mapping *K*

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

Cryptography by Hand: Attacks on Substitution Ciphers

ITIS 6200 / 8200

- Does the brute-force attack still work?
  - There are $26! \approx 2^{88}$ possible mappings to try
    - Too much for most modern computers… for now
- How about the chosen-plaintext attack?
  - Trick Alice into encrypting ABCDEFGHIJKLMNOPQRSTUVWXYZ, and you'll get the whole mapping!
- Another strategy: *cryptanalysis*
  - The most common English letters in text are E, T, A, O, I, N

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

Takeaways

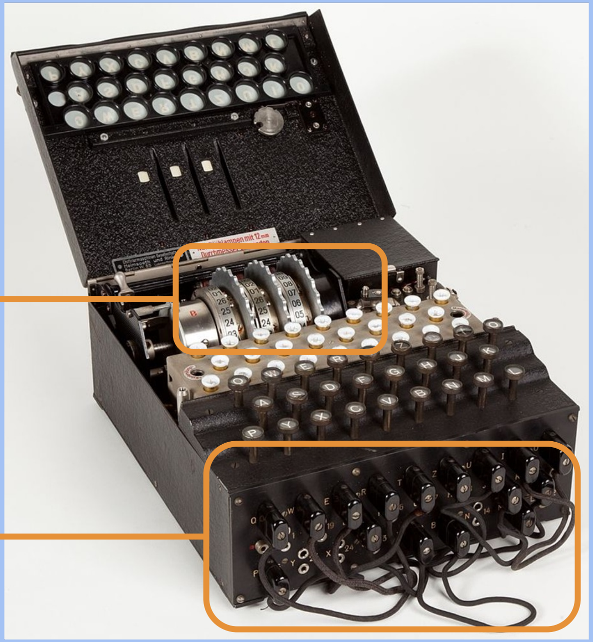
- Cryptography started with paper-and-pencil algorithms (Caesar cipher)
- Then cryptography moved to machines (Enigma)
- Finally, cryptography moved to computers (which we're about to study)
- Hopefully you gained some intuition for some of the cryptographic definitions

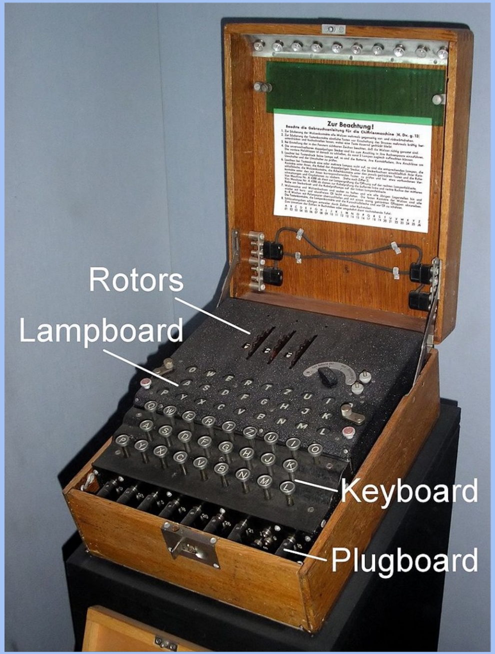
Cryptography by Machines: Enigma

- A mechanical encryption machine used by the Germans in WWII

Rotors

Plugboard



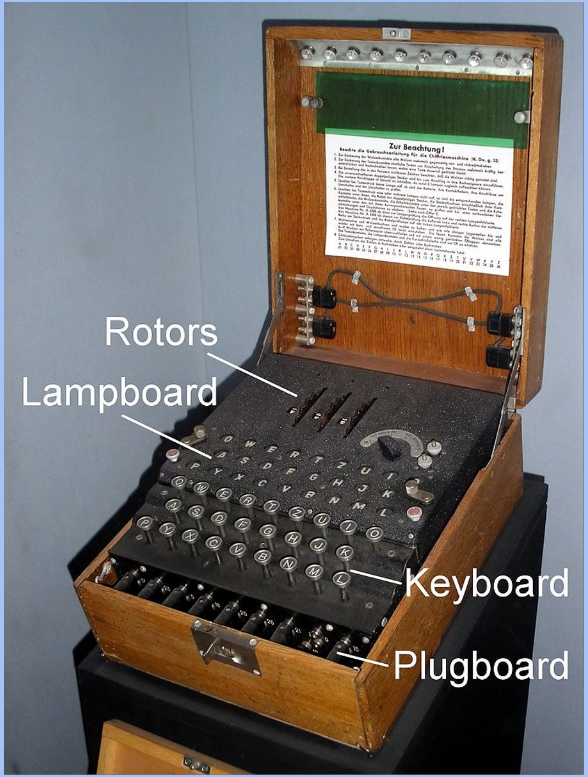


Enigma Operating Principle: Rotor Machine

- The encryption core was composed of 3 or 4 rotors
  - Each rotor was a fixed permutation (e.g. A maps to F, B maps to Q...)
  - And the end was a "reflector", a rotor that sent things backwards
  - Plus a fixed-permutation plugboard
- A series of rotors were arranged in a sequence
  - Each keypress would generate a current from the input to one light for the output
  - Each keypress also advanced the first rotor
    - When the first rotor makes a full rotation, the second rotor advances one step
    - When the second rotor makes a full rotation, the third rotor advances once step

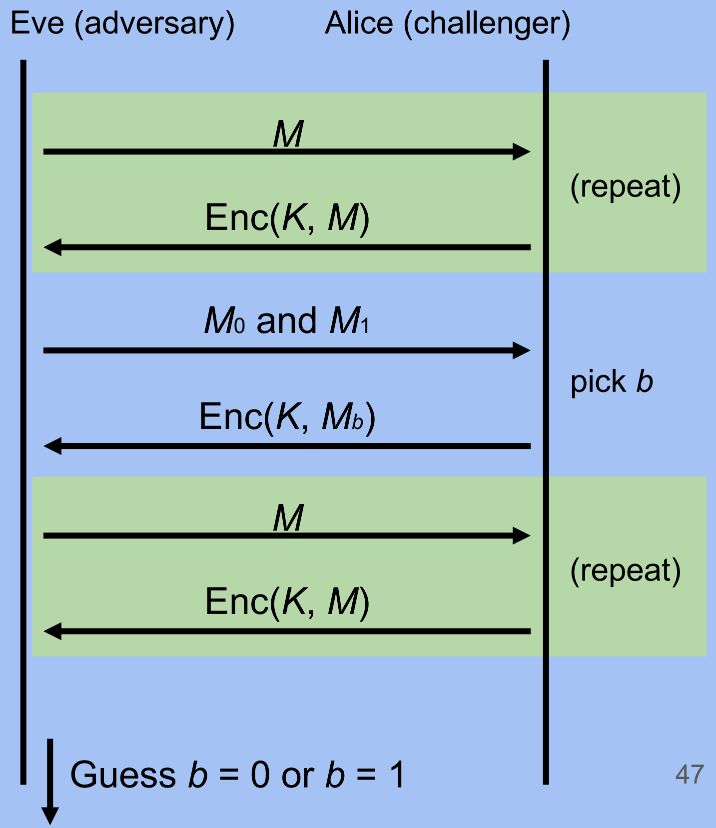
Cryptography by Machines: Enigma

- KeyGen():
  - Choose rotors, rotor orders, rotor positions, and plugboard settings
  - 158,962,555,217,826,360,000 possible keys
- Enc(*K*, *M*) and Dec(*K*, *C*):
  - Input the rotor settings *K* into the Enigma machine
  - Press each letter in the input, and the lampboard will light up the corresponding output letter
  - Encryption and decryption are the same algorithm!
- Germans believed that Enigma was an "unbreakable code"



Cryptography by Machines: Enigma

- Enigma has a significant weakness: a letter never maps to itself!
    - No rotor maps a letter to itself
    - The reflector never maps a letter to itself
    - This property is necessary for Enigma's mechanical system to work
- What pair of messages should Eve send to Alice in the challenge phase?
    - Send $M_0 = A^k$, $M_1 = B^k$
    - M_0 is a string of k 'A' characters, M_1 is a string of k 'B' characters
- How can Eve probably know which message Alice encrypted?
    - If there are no 'A' characters, it was M_0
    - If there are no 'B' characters, it was M_1



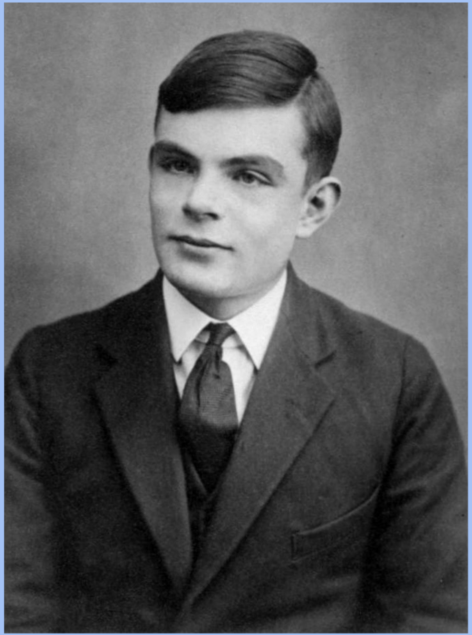
Cryptography by Machines: Attack on Enigma

- 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. "newly deployed minefield")
  - Brute-force: BOMBE would try many keys until the correct one was found
    - Plus a weakness: You'd be able to try multiple keys with the same hardware configuration

BOMBE machine

Cryptography by Machines: Legacy of Enigma

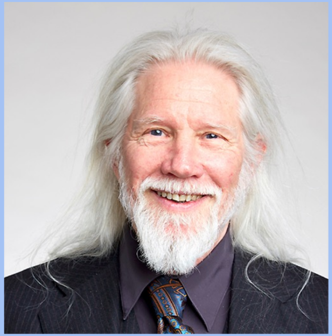
- Alan Turing, one of the cryptographers who broke Enigma, would go on to become one of the founding fathers of computer science
- Most experts agree that the Allies breaking Enigma shortened the war in Europe by about a year



Alan Turing

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



One of these is Diffie, and the other one is Hellman.

One-Time Pads

Cryptography Roadmap

	Symmetric-key	Asymmetric-key
Confidentiality	• One-time pads • Block ciphers • Stream ciphers	• RSA encryption
Integrity, Authentication	• MACs	• Digital signatures

- Hash functions
- Pseudorandom number generators
- Public key exchange (e.g. Diffie-Hellman)
- Key management (certificates)
- Password management

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:

$x \oplus 0 = x$
$x \oplus x = 0$
$x \oplus y = y \oplus x$
$(x \oplus y) \oplus z = x \oplus (y \oplus z)$
$(x \oplus y) \oplus x = y$

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.

One-Time Pad

- KeyGen()
  - Randomly generate an *n*-bit key, **where *n* is the length of your message**
    - Recall: For today, we assume that Alice and Bob can securely share this key
    - For one-time pads, we generate a *new* key for every message
- Enc(K, M) = $K \oplus M$
  - Bitwise XOR M and K to produce C
    - In other words: XOR the *i*th bit of the plaintext with the *i*th bit of the key.
    - $C_i = K_i \oplus M_i$
  - Alice and Bob use a different key (or pad) for each encryption (this is the "one-time" in one-time pad).
- Dec(K, C) = $K \oplus C$
  - Bitwise XOR C and K to produce M
    - $M_i = K_i \oplus C_i$

One-Time Pad: Correctness

- Correctness: If we encrypt and then decrypt, we should get the original message back

$\text{Enc}(K, M)$	$= K \oplus M$	Definition of encryption
$\text{Dec}(K, \text{Enc}(K, M))$	$= \text{Dec}(K, K \oplus M)$	Decrypting the ciphertext
	$= K \oplus (K \oplus M)$	Definition of decryption
	$= M$	XOR property

One-Time Pad: Security

- Recall our definition of confidentiality: The ciphertext should not give the attacker any additional information about the plaintext
- Recall our experiment to test confidentiality from earlier:
  - Alice has encrypted and sent either $M0$ or $M1$
  - Eve knows either $M0$ or $M1$ was sent, but doesn't know which
  - Eve reads the ciphertext and tries to guess which message was sent
  - If the probability that Eve correctly guesses which message was sent is 1/2, then the encryption scheme is confidential

One-Time Pad: Security

Possibility 0: Alice sends Enc(K, M_0)

The ciphertext is $C = K \oplus M_0$

Therefore, $K = C \oplus M_0$

Possibility 1: Alice sends Enc(K, M_1)

The ciphertext is $C = K \oplus M_1$

Therefore, $K = C \oplus M_1$

K was chosen randomly, so both possibilities are equally possible

Eve has learned no new information, so the scheme is *perfectly secure*

One-Time Pad is IND-CPA secure if K was randomly chosen for every encryption!

One-Time Pad: Security

Possibility 0: Alice sends Enc(K, M_0)

The ciphertext is $C = K \oplus M_0$

Therefore, $K = C \oplus M_0$

Possibility 1: Alice sends Enc(K, M_1)

The ciphertext is $C = K \oplus M_1$

Therefore, $K = C \oplus M_1$

What if:

We reuse the same key for different messages? Do we still have IND-CPA?

Answer: No. Eve can derive the key easily, thus all the plaintext.

Impracticality of One-Time Pads

- Problem #1: Key generation
  - For security to hold, keys must be randomly generated for every message, and never reused
  - Randomness is expensive, as we'll see later
- Problem #2: Key distribution
  - To communicate an *n*-bit message, we need to securely communicate an *n*-bit key first
  - But if we have a way to securely communicate an *n*-bit key, we could have communicated the message directly!
- Only practical application: Communicate keys in advance
  - You have a secure channel now, but you won't have it later
  - Use the secure channel now to communicate keys in advance
  - Use one-time pad later to communicate over the insecure channel
  - And people can compute this by hand without computers!

Summary

- What is cryptography?
- Definitions of key properties
- A better definition of confidentiality: IND-CPA security
- A brief history of cryptography
- Symmetric-key crypto