

IERG 4130 – 2026 Fall



Intro to Cryptography

Yajin ZHOU

<https://yajin.org>



What is cryptography?

- The study of secure communication over **insecure channels**, and secure cooperation among distrustful parties
- Modern cryptography involves a lot of math



Don't try this at home!

- We will teach you the basic building blocks of cryptography, but you should never try to write your own cryptographic algorithms. Why?
 - It's very easy to make a mistake that makes your code insecure
 - Lots of tricky edge cases that we won't cover
 - One small bug could compromise the security of your code
- Instead, use existing well-vetted cryptographic libraries



Why do we still need to take this lecture?

- Educating you **as a consumer (how to use)**, NOT educating you as an engineer (how to invent/implement your own cryptographic algorithms)
- You might also be asked to evaluate the difference between alternatives. For that, you will need to understand the underlying cryptographic engineering involved
- Similarly, there are sometimes applications that take advantage of cryptographic primitives in non-cryptographic ways, so it is useful to know the primitives
 - Example: how can we determine whether two strings (very long) are equal?
 - strcmp?
 - SHA-1/SHA-256(string)

Definitions



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 data
- **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!
 - ... but they're not identical properties



Three Goals of Cryptography

- **Integrity and authenticity**
- Examples
 - Download software from a website, and the website shows the MD5/SHA-256 hash
 - We check the SHA-256 value of the downloaded software to verify its integrity
 - However, you cannot ensure the **authenticity**
 - **Why?**



Keys

- The most basic building block of any cryptographic scheme: **key**
- We can 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.
 - **Asymmetric key model**: Everybody has two keys, a secret key and a public key.
 - Example: RSA encryption



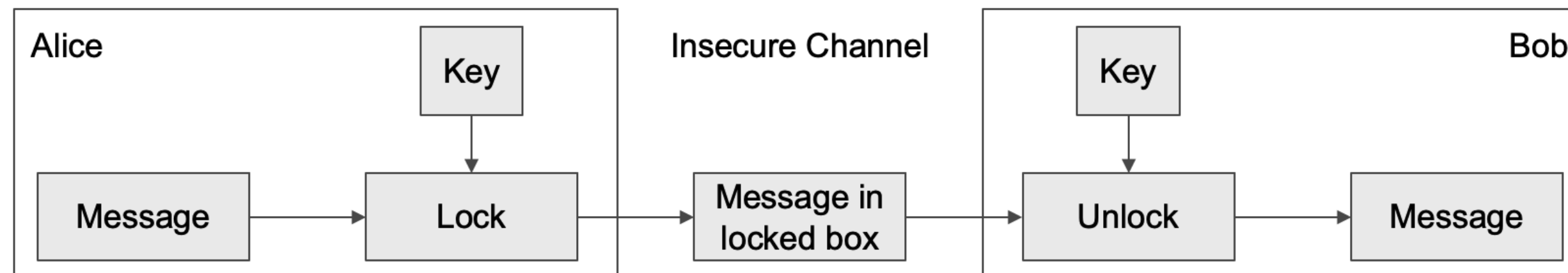
Security Principle: Kerckhoffs's Principle

- This principle is closely related to Shannon's Maxim
 - **Don't use security through obscurity.** Assume the attacker knows the system.
- Kerckhoffs'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

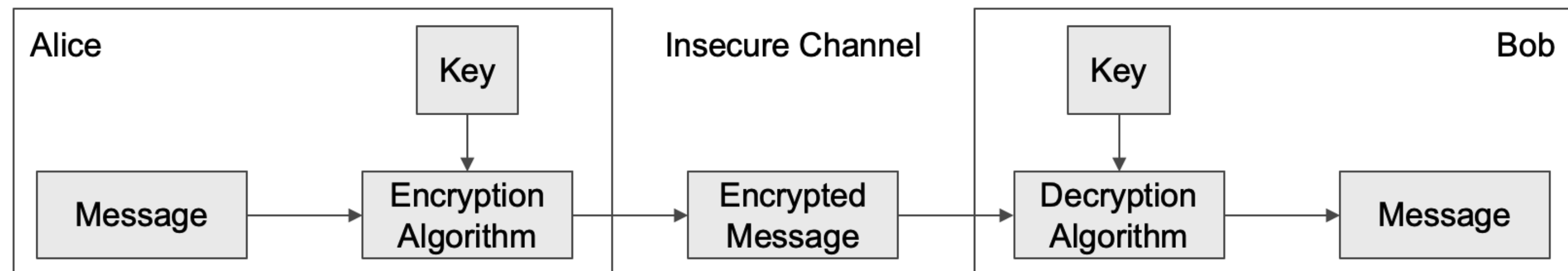
- **Confidentiality:** An adversary cannot read our messages.
- Analogy: Locking and unlocking the message
 - Alice uses the key to lock the message in a box
 - Alice sends the message (locked in the box) over the insecure channel
 - Eve sees the locked box, but cannot access the message without the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





Confidentiality

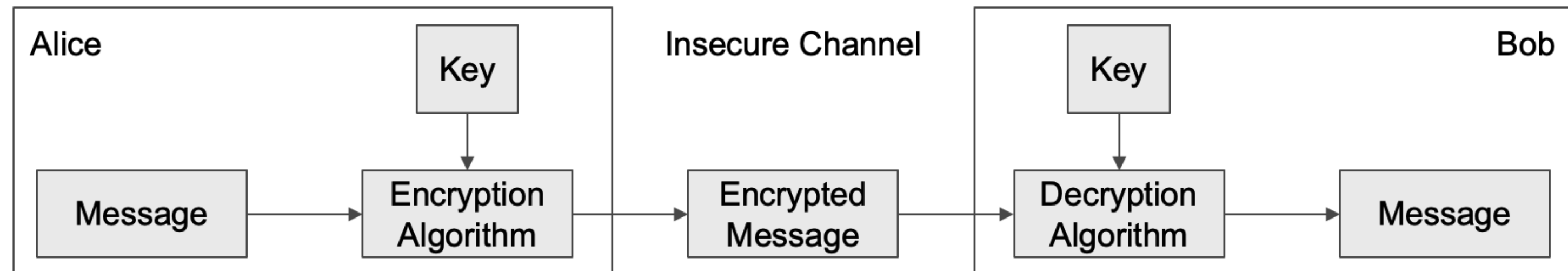
- Schemes provide confidentiality by encrypting messages
 - Alice uses the key to encrypt the message: Change the message into a scrambled form
 - Alice sends the encrypted message over the insecure channel
 - Eve sees the encrypted message, but cannot figure out the original message without the key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





Confidentiality

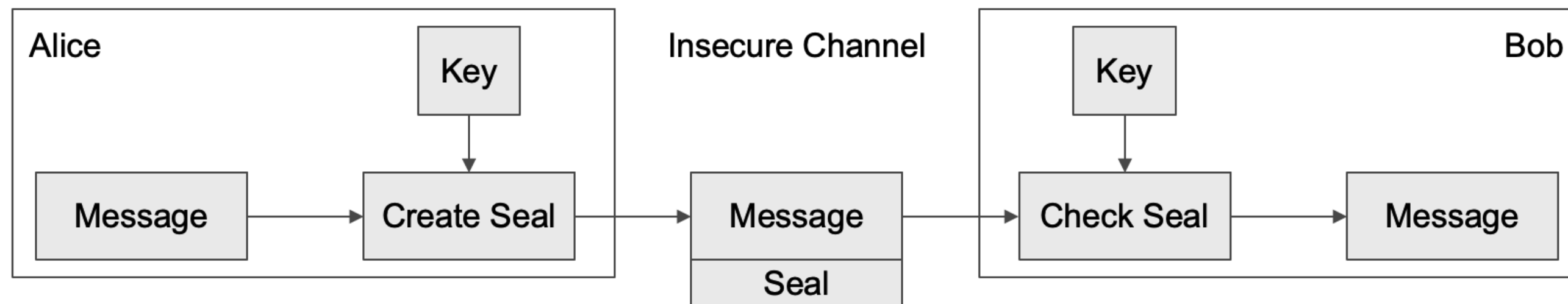
- **Plaintext:** The original message
- **Ciphertext:** The encrypted message





Integrity (and Authenticity)

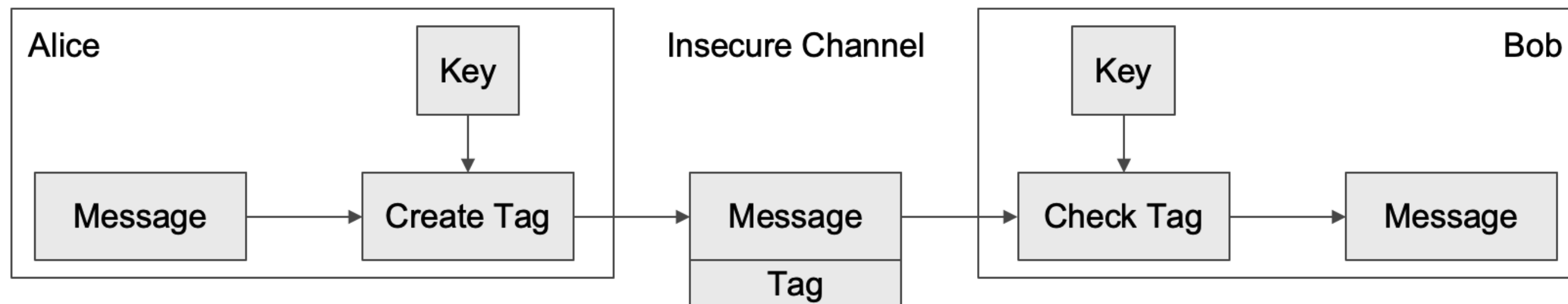
- **Integrity:** An adversary cannot change our messages without being detected.
- **Analogy:** Adding a seal on the message
 - Alice uses the key to add a special seal on the message (e.g. puts tape on the envelope)
 - Alice sends the message and the seal over the insecure channel
 - If Mallory tampers with the message, she'll break the seal (e.g. break the tape)
 - 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
 - And we are sure the message was sent by Alice





Threat Models

- What if Eve can do more than eavesdrop?
- Real-world schemes are often vulnerable to more sophisticated attackers, so cryptographers have created more sophisticated threat models too
- Chosen-plaintext: Eve asks Alice to encrypt the word “EvanBot”
- Chosen-ciphertext: Eve asks Bob: tell me the decrypted value of “garbage values” (because of some vulnerabilities) and Bob does it (Then Eve learns extra information)
 - Suppose $\text{Enc}(K, \text{“ABCD”}) = \text{“JKLM”}$ and $\text{Enc}(K, \text{“A_B_C_D”}) = \text{“JAKALAMA”}$
 - Eve to Bob: Can you decrypt “JAKALAMA” for me?
 - Bob to Eve: “Yes sir, it’s A_B_C_D”
 - Then we will know that the decrypted value of JKLM is ABCD



Threat Models

- Some threat models for analyzing confidentiality:

	Can Eve trick Alice into encrypting messages of Eve's choosing?	Can Eve trick Bob into decrypting messages of Eve's choosing?	Can Eve eavesdrop on Alice's encrypted communications?
Ciphertext-only	No	No	Yes
Chosen-plaintext	Yes	No	Yes
Chosen-ciphertext	No	Yes	Yes
Chosen plaintext-ciphertext	Yes	Yes	Yes



Threat Models

- In this class, we'll focus primarily on chosen plaintext attacks, for simplicity
- In practice, cryptographers use the chosen plaintext-ciphertext model

	Can Eve trick Alice into encrypting messages of Eve's choosing?	Can Eve trick Bob into decrypting messages of Eve's choosing?	Can Eve eavesdrop on Alice's encrypted communications?
Ciphertext-only	No	No	Yes
Chosen-plaintext	Yes	No	Yes
Chosen-ciphertext	No	Yes	Yes
Chosen plaintext-ciphertext	Yes	Yes	Yes

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

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



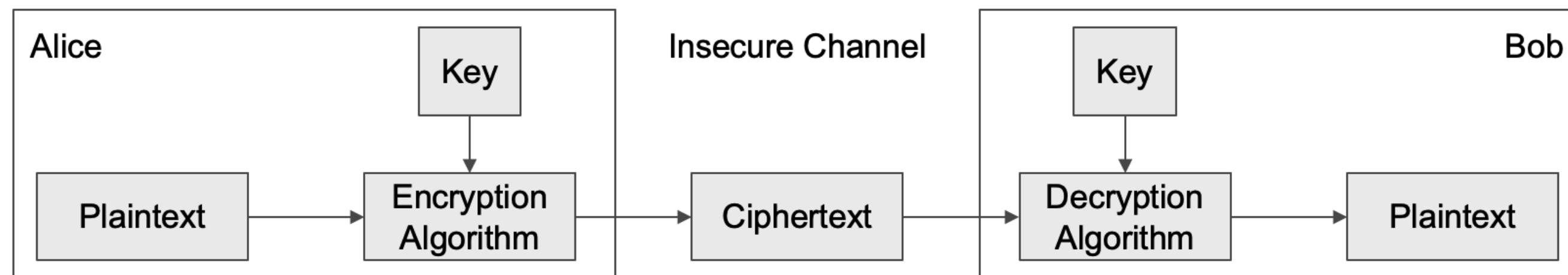
Symmetric-Key Encryption

- 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. 11010101001001010
 - 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

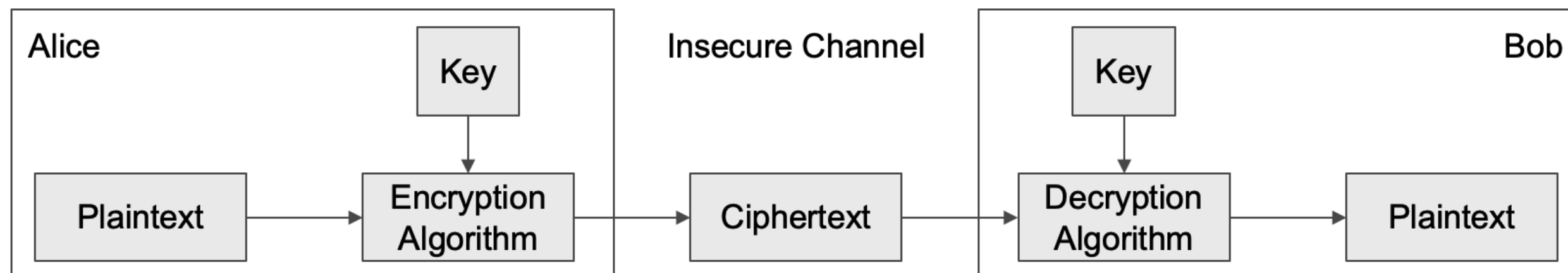
- A symmetric-key encryption scheme has three algorithms:
 - $\text{KeyGen}() \rightarrow K$: Generate a key K
 - $\text{Enc}(K, M) \rightarrow C$: Encrypt a **plaintext** M using the key K to produce **ciphertext** C
 - $\text{Dec}(K, C) \rightarrow M$: Decrypt a ciphertext C using the key K





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
 - $\text{Dec}(K, \text{Enc}(K, M)) = M$ for all $K \leftarrow \text{KeyGen}()$ and all M
 - **Efficiency**: Encryption/decryption algorithms should be fast: >1 Gbps on a standard computer
 - **Security**: Confidentiality of M from attackers who don't know K





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?

26





Defining Confidentiality

- A better definition of confidentiality: The ciphertext should not give the attacker *any additional information* about the plaintext, beyond what the attacker already knew.
- We make this precise with an experiment/security game:
 - 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



Defining Confidentiality

- Intuition
 - If the scheme is confidential, Eve can only guess with probability $1/2$, which is no different than if Eve hadn't seen the ciphertext at all
 - In other words: the ciphertext gave Eve no *additional* information about which plaintext was sent!



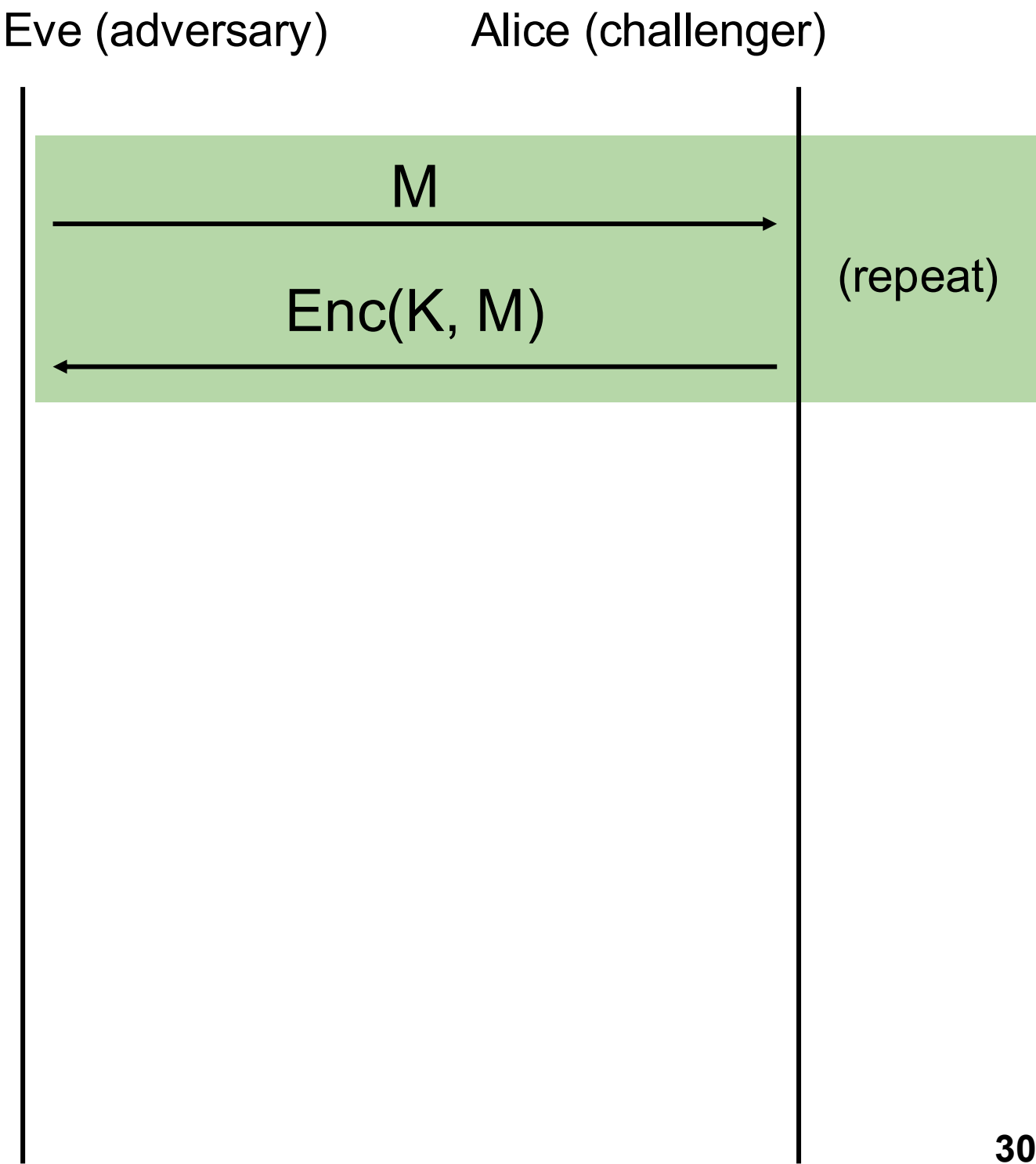
Defining Confidentiality: IND-CPA

- Recall our threat model: Eve can also perform a **chosen plaintext attack**
 - 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 other 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”
 - We will use one to define confidentiality precisely



Defining Confidentiality: IND-CPA

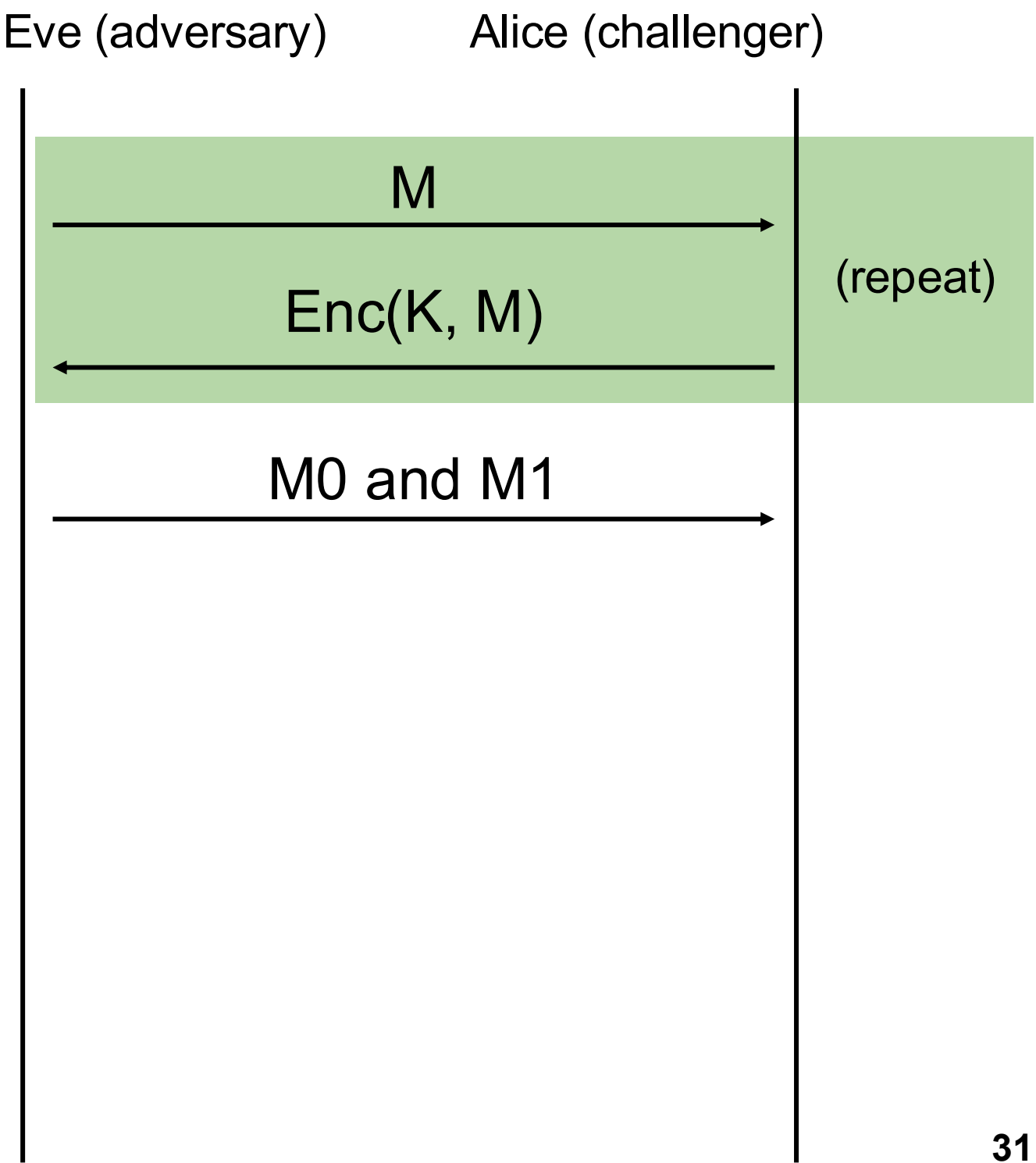
- 1. Eve may choose plaintexts to send to Alice and receives their ciphertexts





Defining Confidentiality: IND-CPA

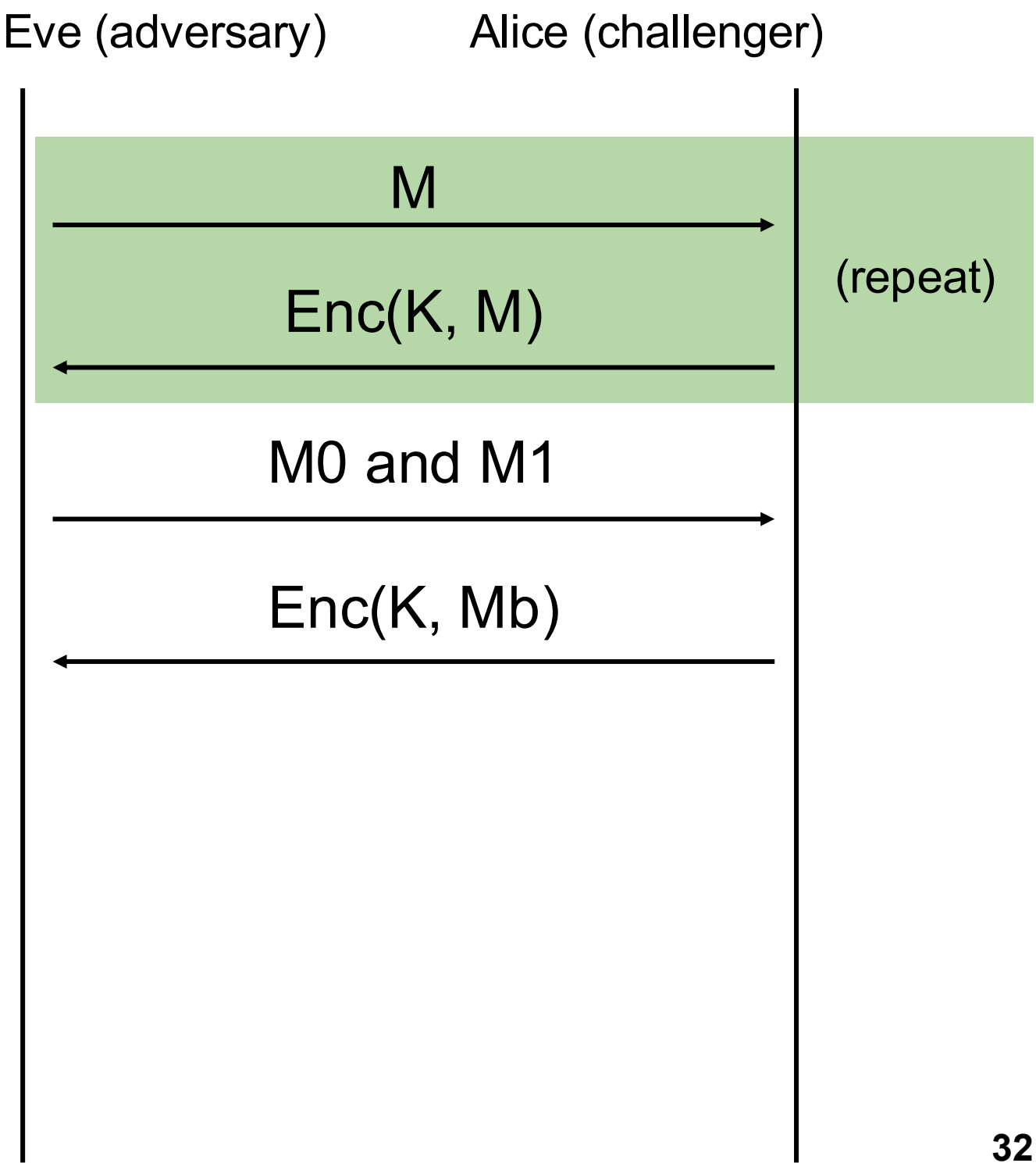
- 1. Eve may choose plaintexts to send to Alice and receives their ciphertexts
- 2. Eve issues a pair of plaintexts M_0 and M_1 to Alice





Defining Confidentiality: IND-CPA

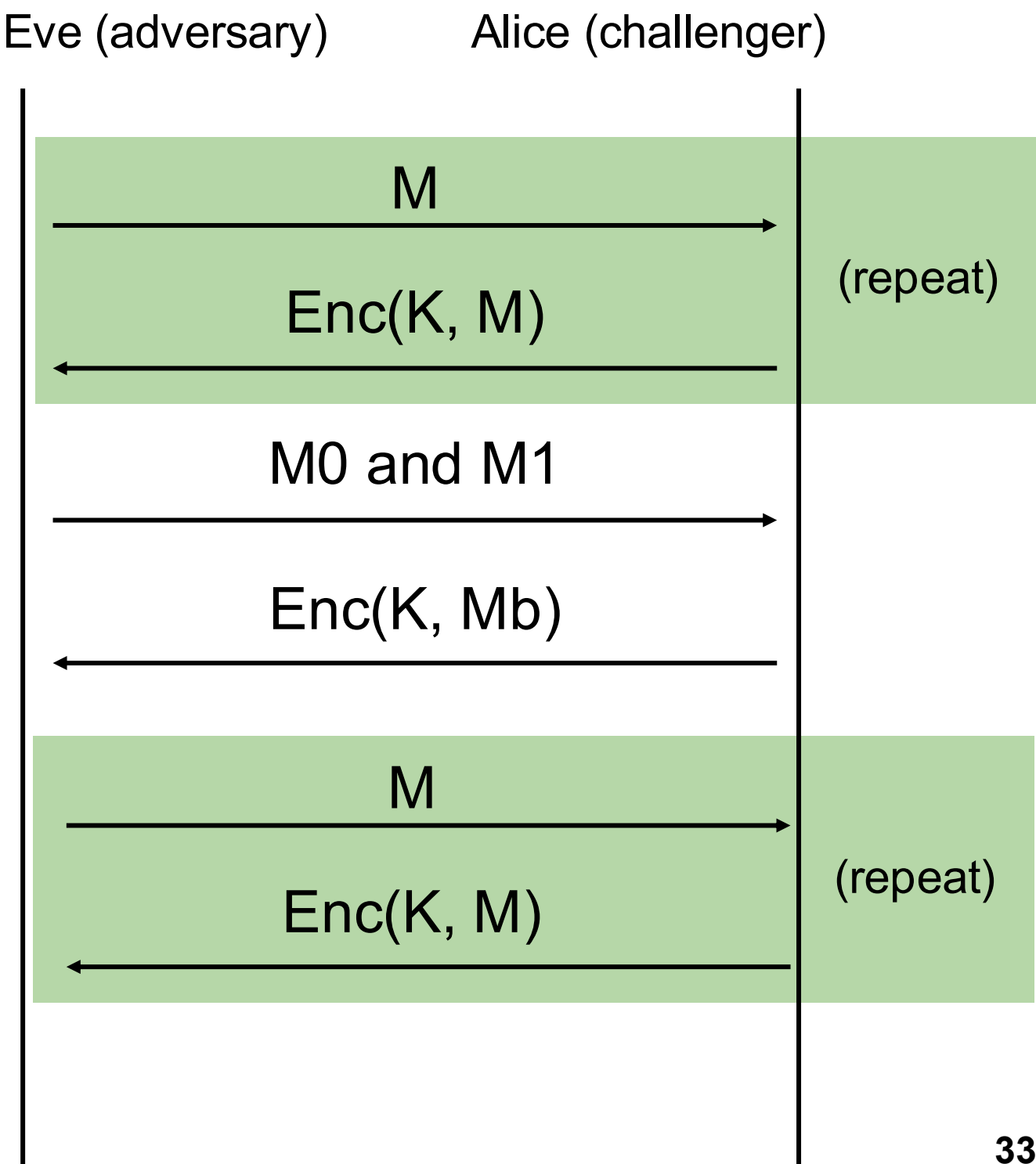
- 1. Eve may choose plaintexts to send to Alice and receives their ciphertexts
- 2. Eve issues a pair of plaintexts M_0 and M_1 to Alice
- 3. Alice randomly chooses either M_0 or M_1 to encrypt and sends the encryption back
 - a) Alice does not tell Eve which one was encrypted!





Defining Confidentiality: IND-CPA

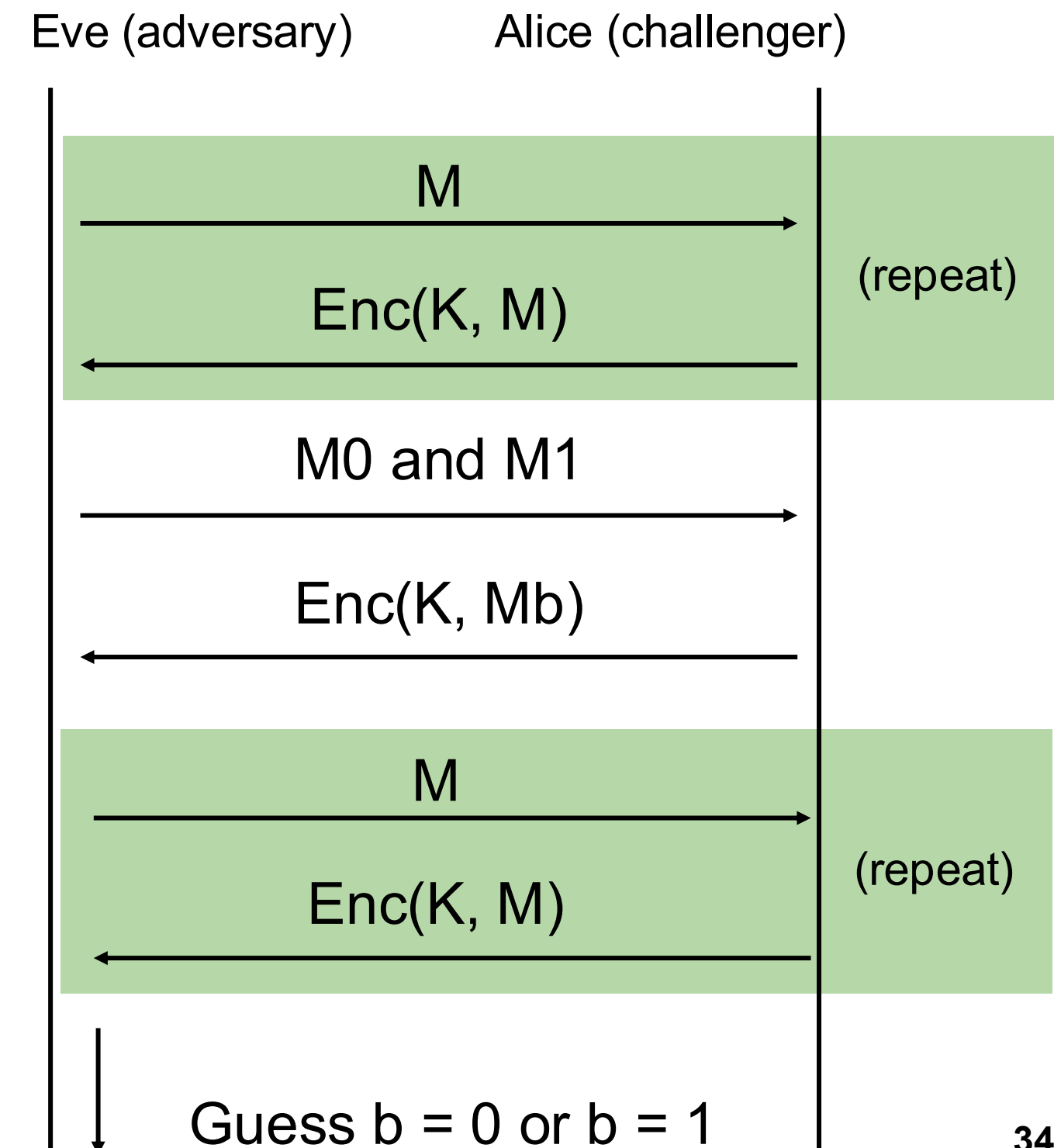
- 1. Eve may choose plaintexts to send to Alice and receives their ciphertexts
- 2. Eve issues a pair of plaintexts M_0 and M_1 to Alice
- 3. Alice randomly chooses either M_0 or M_1 to encrypt and sends the encryption back
 - a) Alice does not tell Eve which one was encrypted!
- 4. Eve may again choose 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 strategy³⁵; Eve could use any possible strategy



Defining Confidentiality: IND-CPA

- **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 do better than random, then the ciphertext³⁶ has leaked information, and Eve is able to learn it and use it to gain an advantage!
- **Proving security:** For all attackers (Eves), 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
 - Why? 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 impractical
- Applications that wish to hide length must choose to *pad* their own messages to the maximum possible length before encrypting



Edge Cases: Length

- 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 way to formalize this: Eve is limited to polynomial runtime algorithms (no exponential-time algorithms)



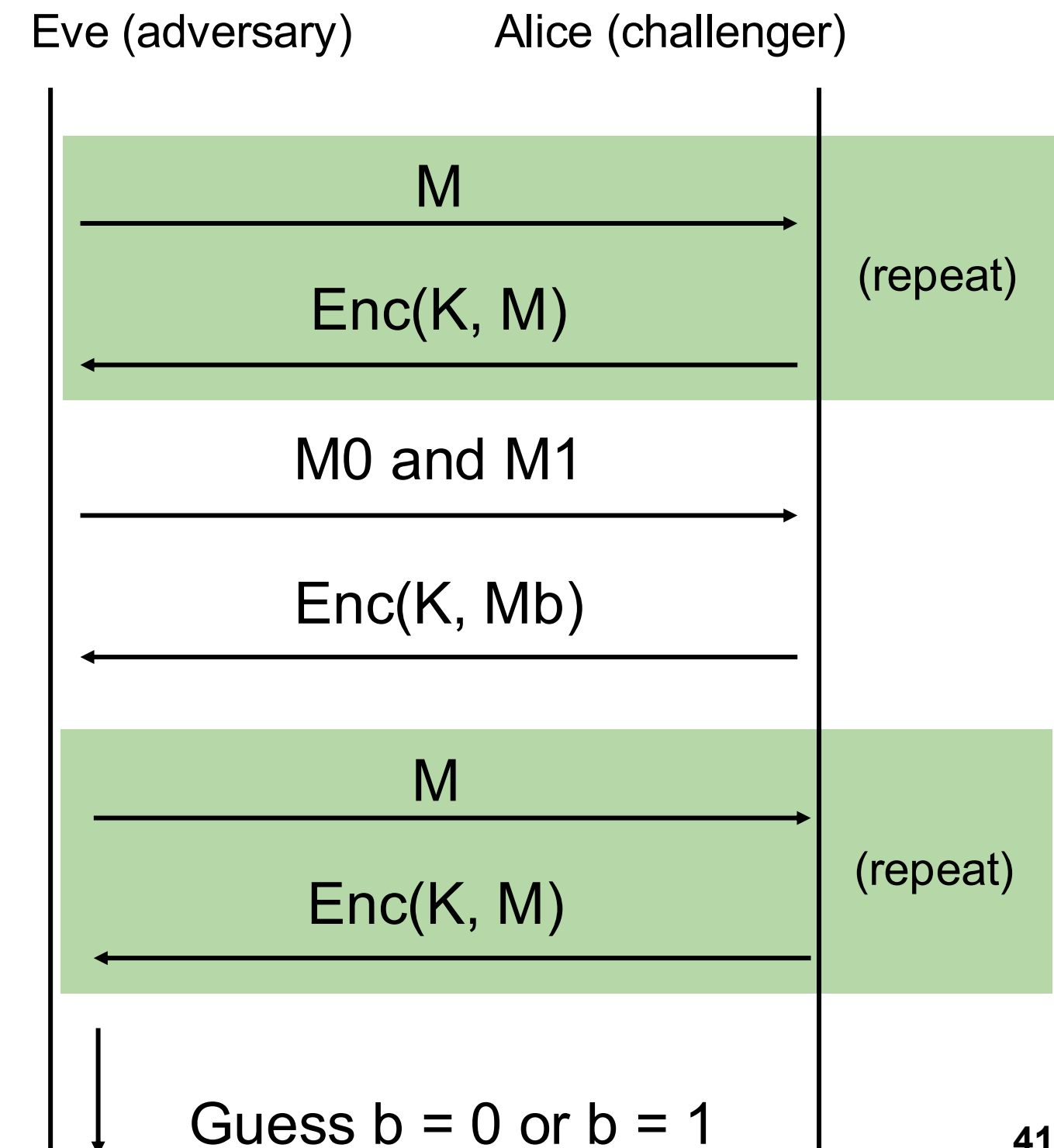
Edge Cases: Negligible Advantage

- 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 attack
- In the IND-CPA game: The scheme is secure even if Eve can win with probability $\leq 1/2 + \epsilon$, 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

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
- An encryption scheme is IND-CPA secure if for all polynomial-time attackers Eve
 - Eve can win with probability $\leq 1/2 + \epsilon$, where ϵ is negligible.



Summary



- What's cryptography?
 - Communicating securely over insecure channels
 - You should never write your own crypto! Use existing libraries instead.



Summary

- Definitions
 - Alice and Bob want to send messages over an insecure channel. Eve can read anything sent over the insecure channel. Mallory can read or modify anything sent over the insecure channel.
 - We want to ensure confidentiality (adversary can't read message), integrity (adversary can't modify message), and authenticity (prove message came from sender)
 - Crypto uses secret keys. Kerckhoffs's principle says to assume the attacker knows the entire system, except the secret keys.
 - There are several different threat models. We'll focus on the chosen plaintext attack, where Eve tricks Alice into encrypting some messages.

Summary



- IND-CPA security
 - Even if Eve can trick Alice into encrypting some messages of Eve's choosing, given the encryption of either M_0 or M_1 , Eve cannot distinguish which message was sent with probability greater than $1/2$.
 - We can use the IND-CPA game to test for IND-CPA security
 - Edge cases: IND-CPA secure schemes can leak length. Eve is limited to polynomial-time algorithms, and must have a non-negligible advantage to win.