Lecture 9:

Network Security

Katerina Argyraki, EPFL

Security properties

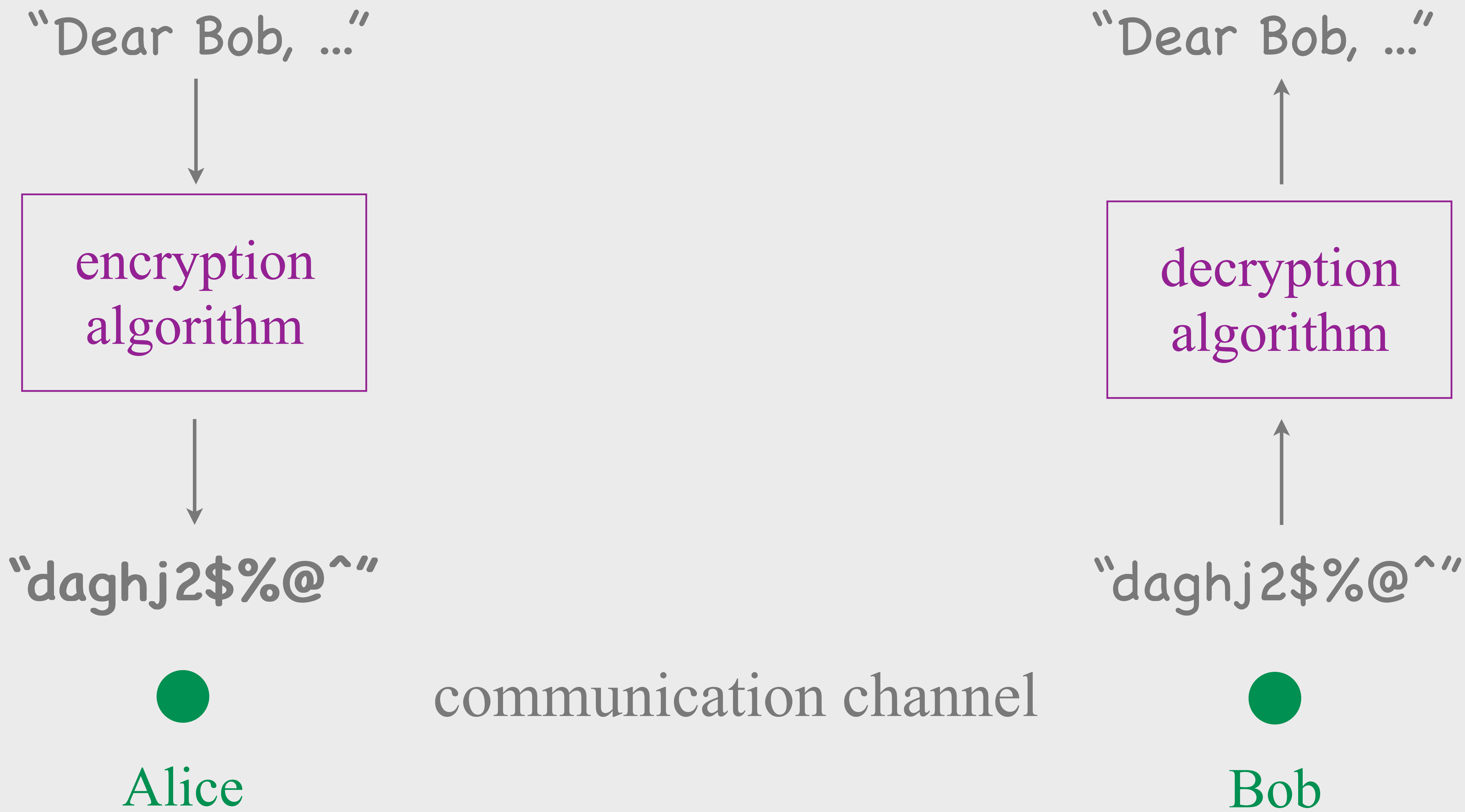
- Confidentiality
  - only the sender and the receiver understand the contents of the message
- Authenticity
  - the message is from whom it claims to be
- Integrity
  - the message was not changed along the way

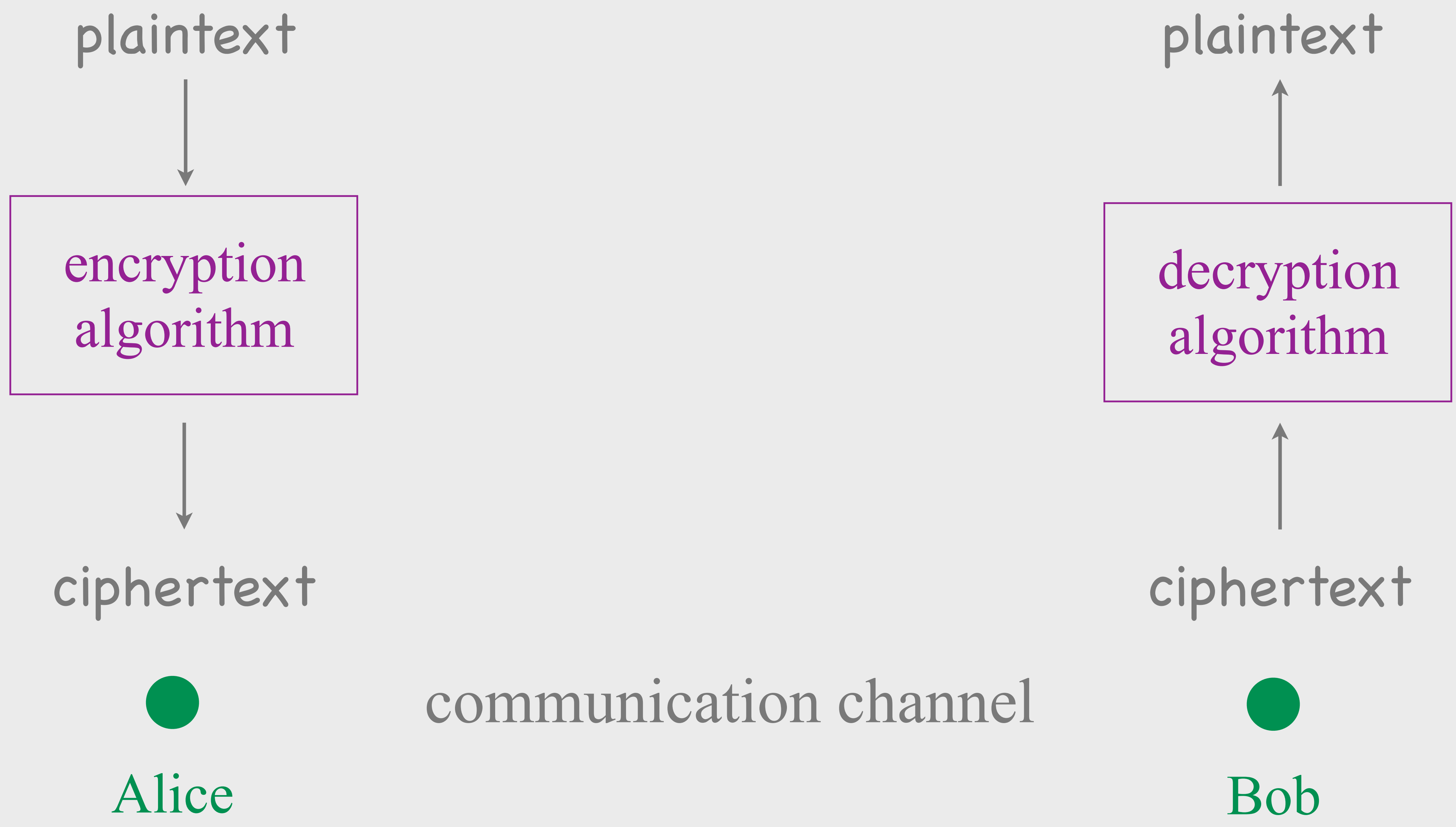
Outline

- Building blocks
- Providing security properties
- Securing Internet protocols
- Operational security

Outline

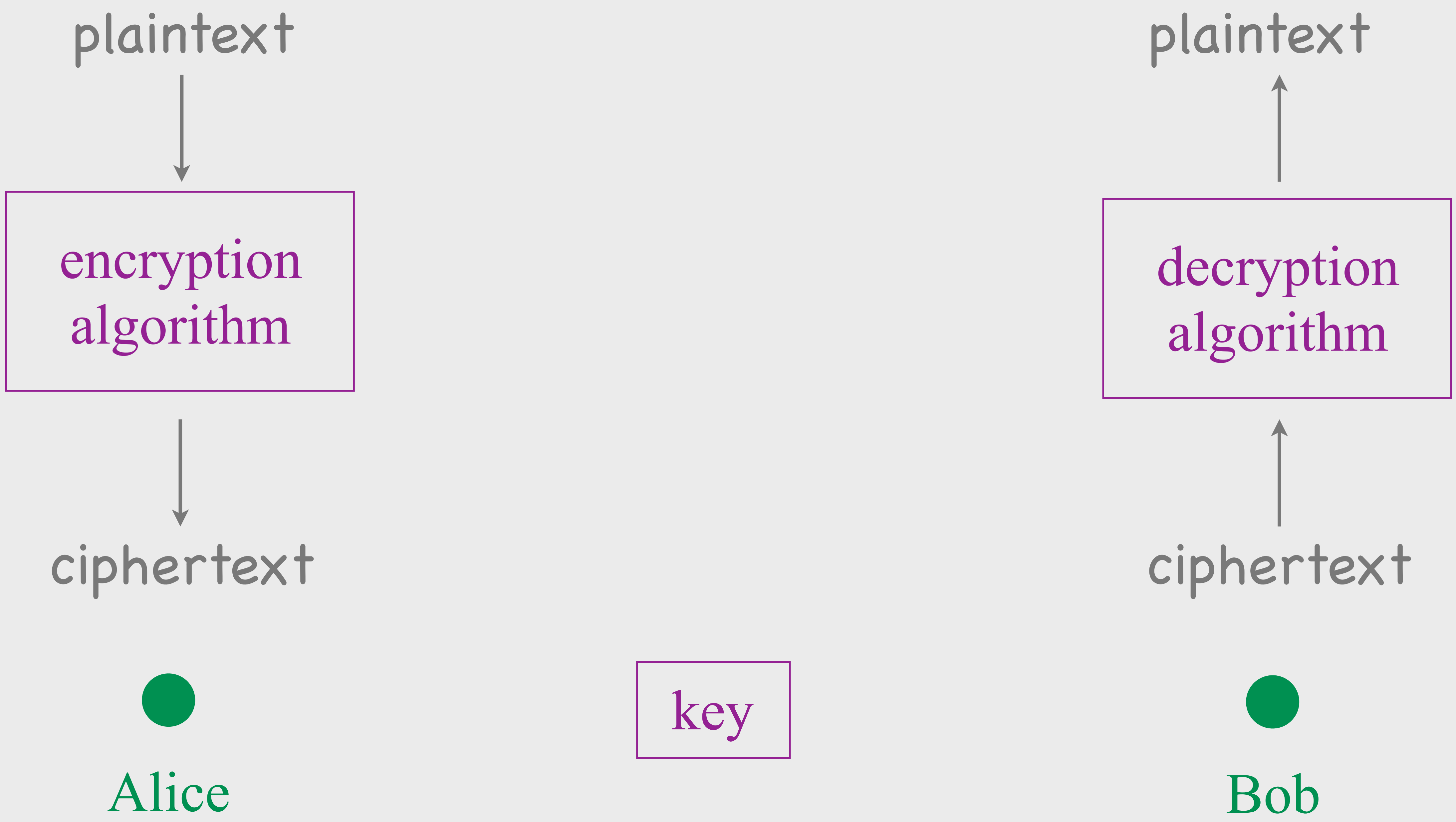
- Building blocks
- Providing security properties
- Securing Internet protocols
- Operational security

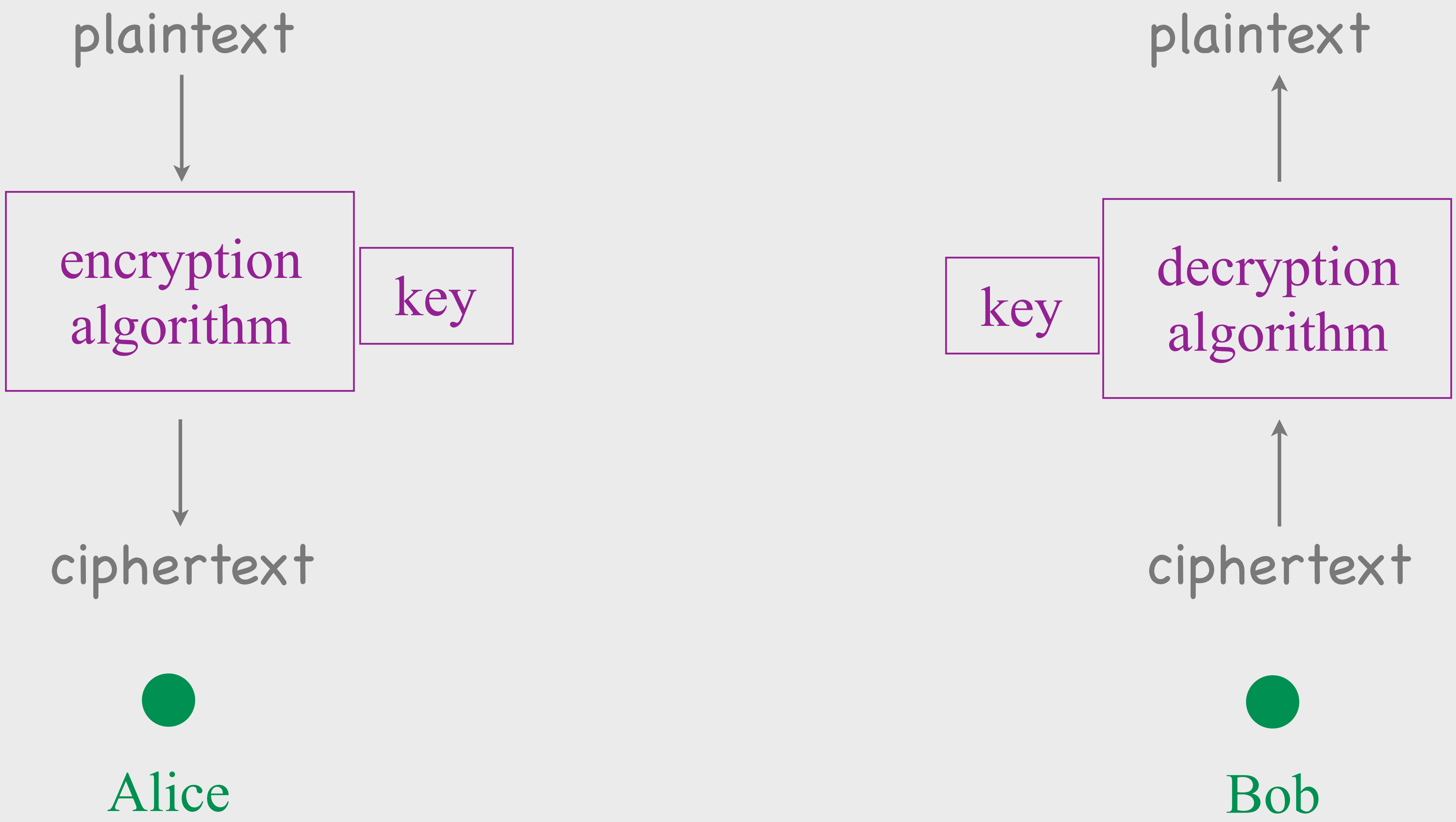

"Dear Bob, …"
encryption algorithm
"daghj2$%@^"
Alice
communication channel
"Dear Bob, …"
decryption algorithm
"daghj2$%@^"
Bob


plaintext
encryption algorithm
ciphertext
plaintext
decryption algorithm
ciphertext
communication channel
Alice
Bob

Encryption & decryption

- Encryption: plaintext in, ciphertext out
- Decryption: ciphertext in, plaintext out
- Ciphertext: ideally, should reveal no information about the plaintext


plaintext
encryption algorithm
ciphertext
Alice
key
plaintext
decryption algorithm
ciphertext
Bob



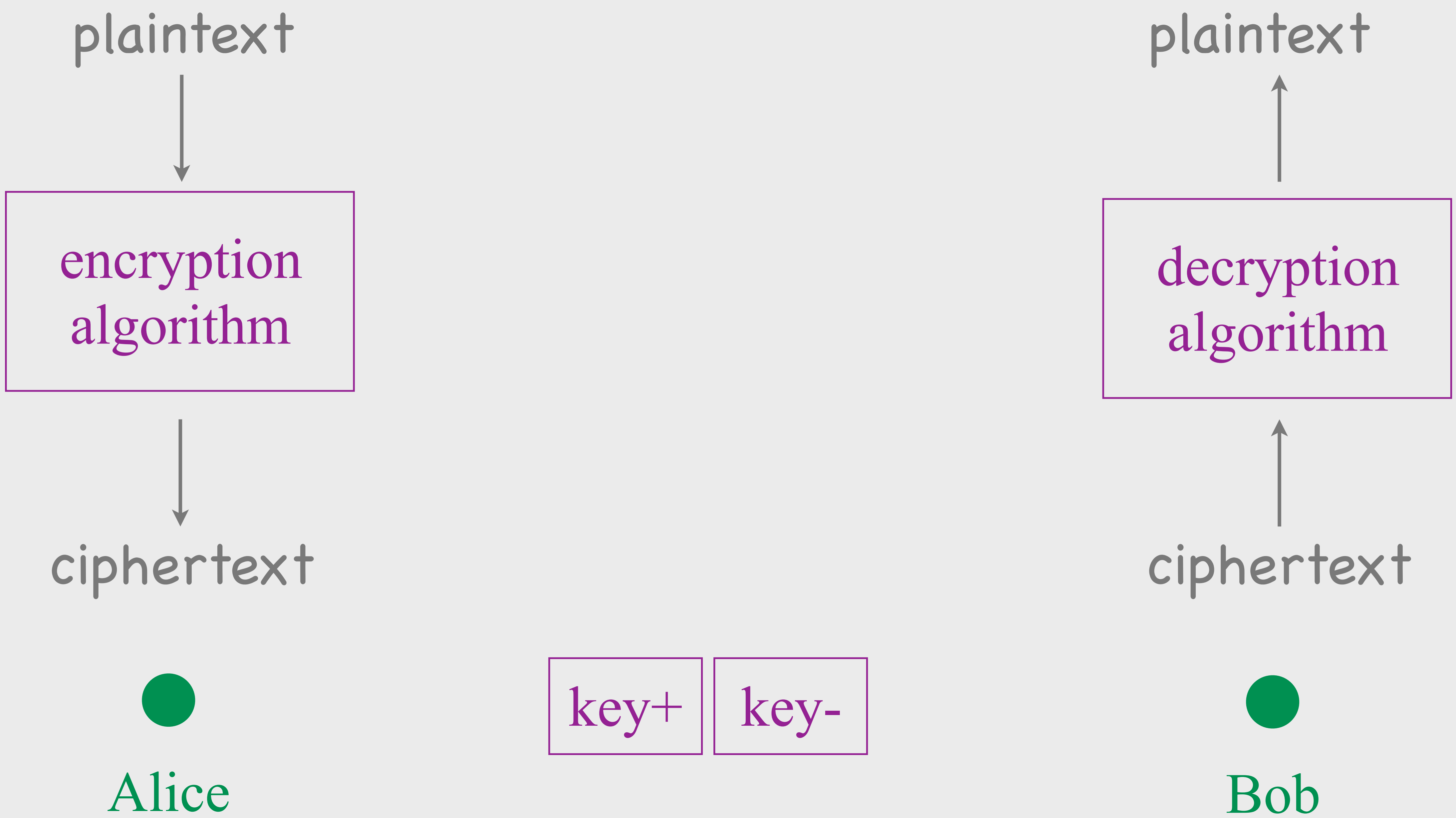
key{ key{ plaintext } } = plaintext

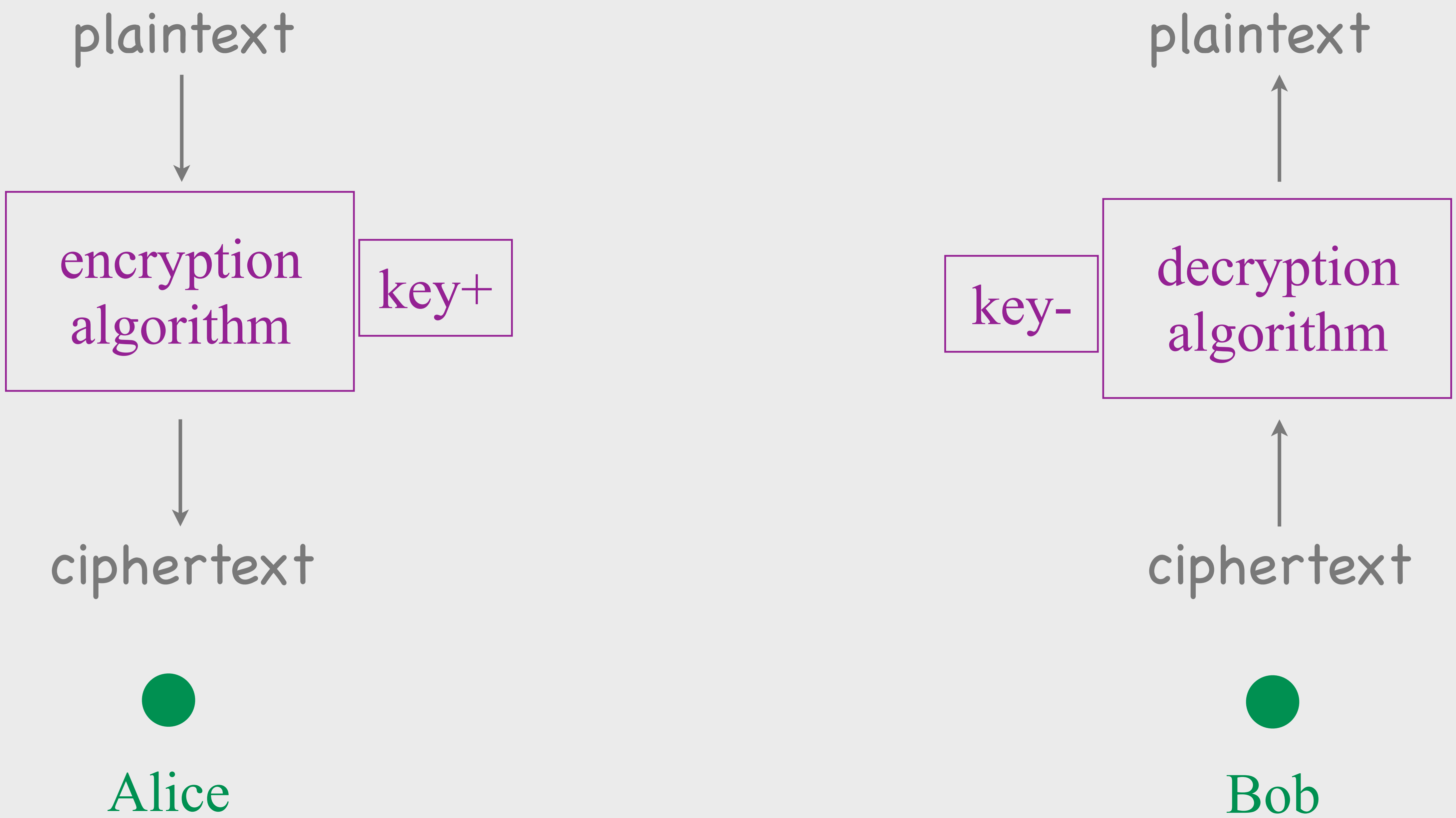
Symmetric key cryptography

- Alice and Bob share the same key
  - used both for the encryption and decryption algorithm
- Use key to "scramble" the plaintext
  - stream ciphers & block ciphers
  - RC4, AES, Blowfish

Symmetric key cryptography

- Challenge: how to share a key?
  - out of band
  - not always an option


plaintext
encryption algorithm
ciphertext
Alice
key+
key-
plaintext
decryption algorithm
ciphertext
Bob



key-{ key+{ plaintext } } = plaintext

key+{ key-{ plaintext } } = plaintext

Asymmetric key cryptography

- Alice and Bob use different keys
  - public (key+) and private (key-) key
- There is a special relationship between them
  - key-{ key+{ plaintext } } = plaintext
  - key+{ key-{ plaintext } } = plaintext
  - RSA, DSA

Asymmetric key cryptography

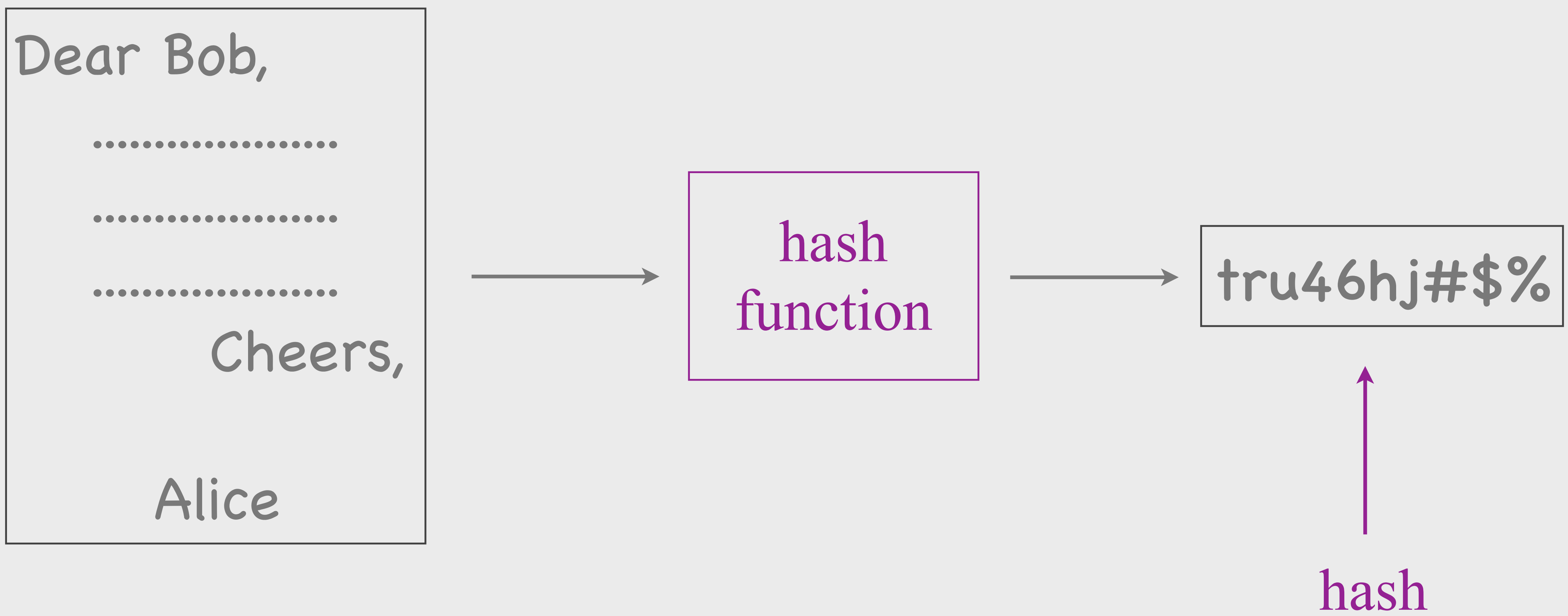
- Public key is not secret
  - only private key is secret
  - enough to guarantee secrecy
- But you can’t guess one from the other
  - Alice/Bob can share key+ with everyone
  - without revealing information about key-

Asymmetric key cryptography

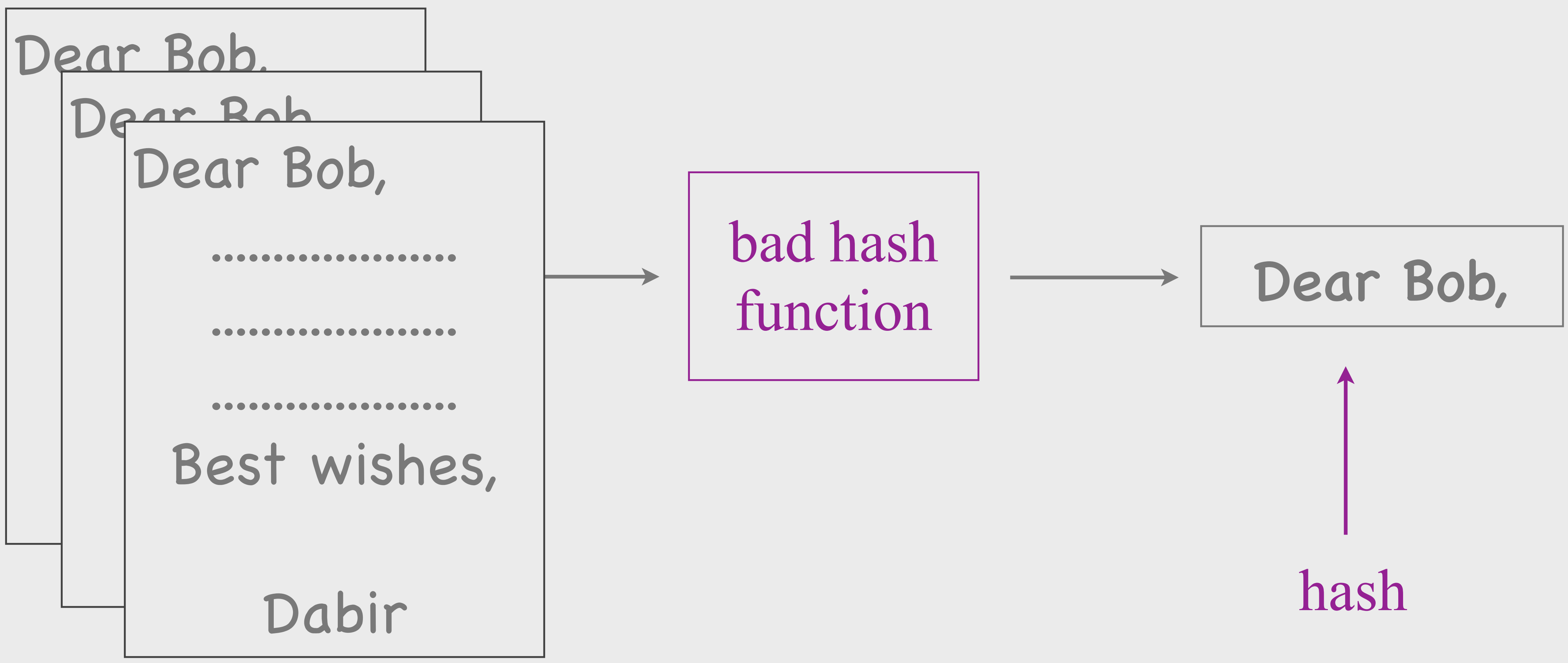
- Challenge: computationally expensive
  * sophisticated encryption/decryption algorithms based on number theory

Two approaches to crypto

- Symmetric: faster but out-of-band secret sharing
- Asymmetric: no out-of-band secret sharing but slower



Dear Bob,
....
....
....
Cheers,
Alice
hash function
tru46hj#$%
hash



Dear Bob,
Dear Bob,
Dear Bob,
....
....
....
Best wishes,
Dabir
bad hash function
Dear Bob,
hash

Cryptographic hash function

- Maps larger input space to smaller hash space
- Hash ideally reveals no information on input
- Should be hard to identify two inputs that lead to the same hash

How is hashing different from encryption?

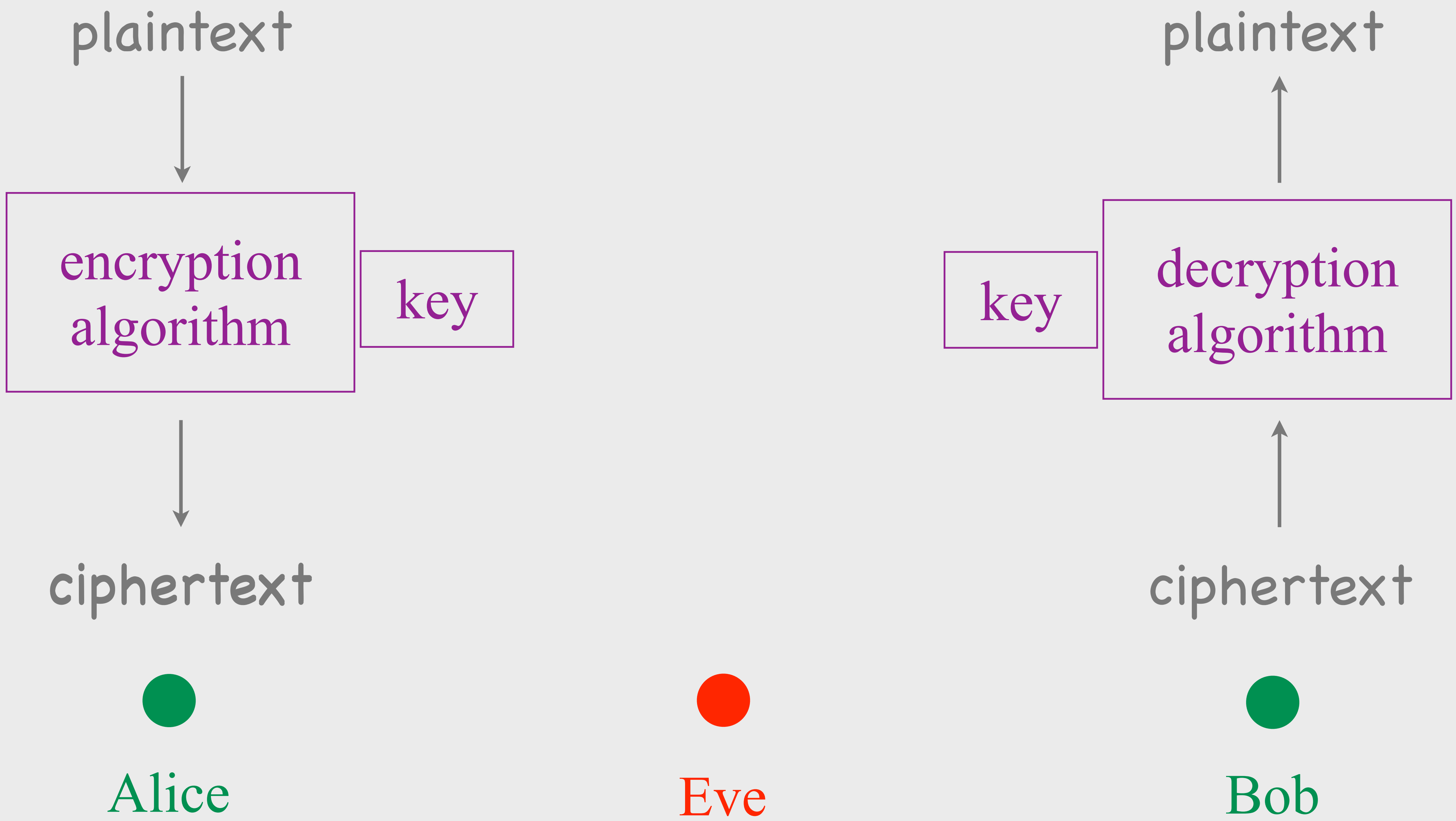
Building blocks

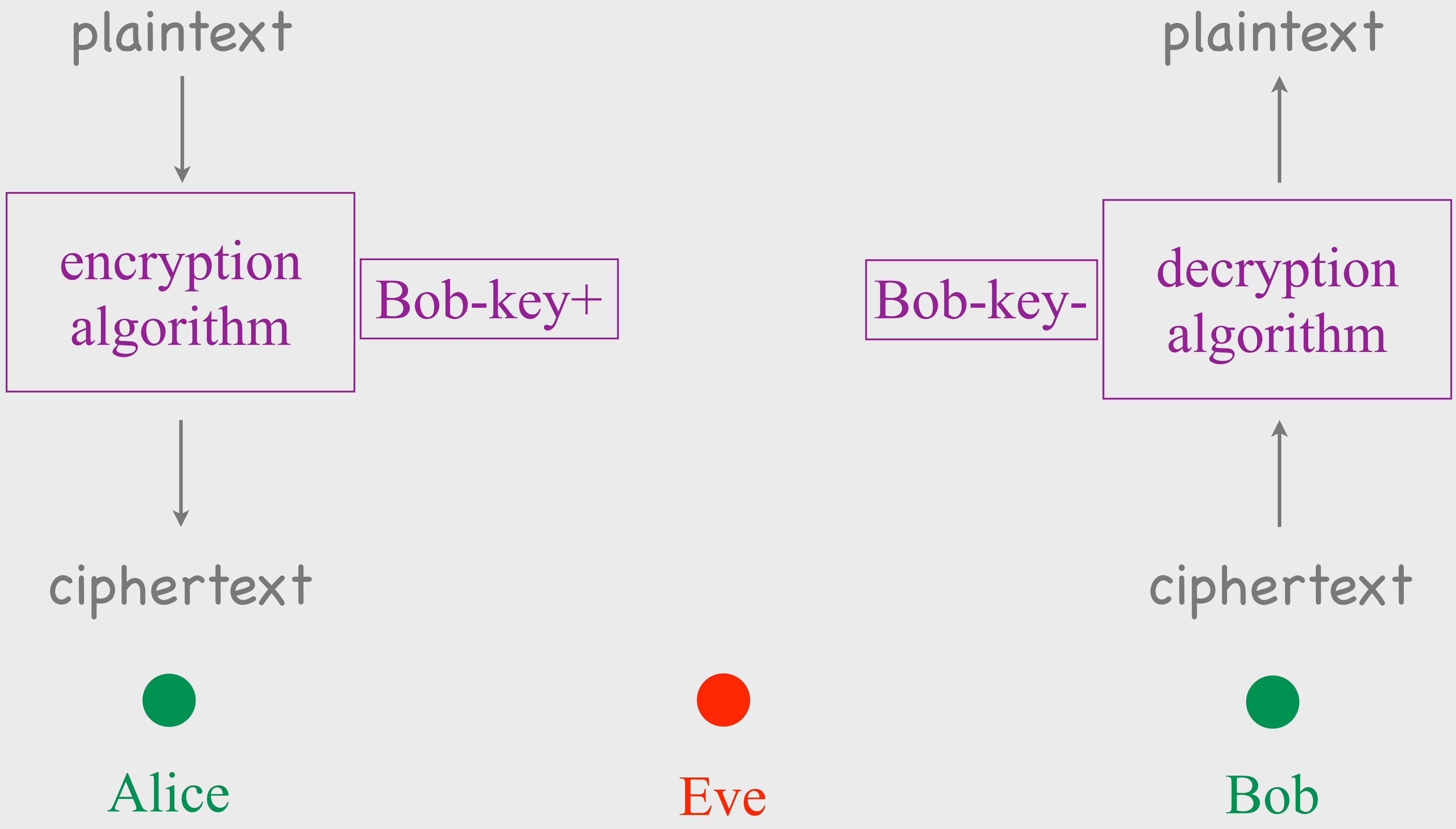
- Symmetric key encryption/decryption
  - Alice and Bob share the same secret key
  - challenge: exchanging the secret key
- Asymmetric key encryption/decryption
  - Alice and Bob use different keys
  - challenge: computationally more expensive
- Cryptographic hash function
  - produces a hash of the original message

Outline

- Building blocks
- Providing security properties
- Securing Internet protocols
- Operational security

Providing confidentiality


plaintext
encryption algorithm
key
ciphertext
Alice
Eve
plaintext
key
decryption algorithm
ciphertext
Bob


plaintext
encryption algorithm
Bob-key+
ciphertext
Alice
Eve
plaintext
Bob-key-
decryption algorithm
ciphertext
Bob

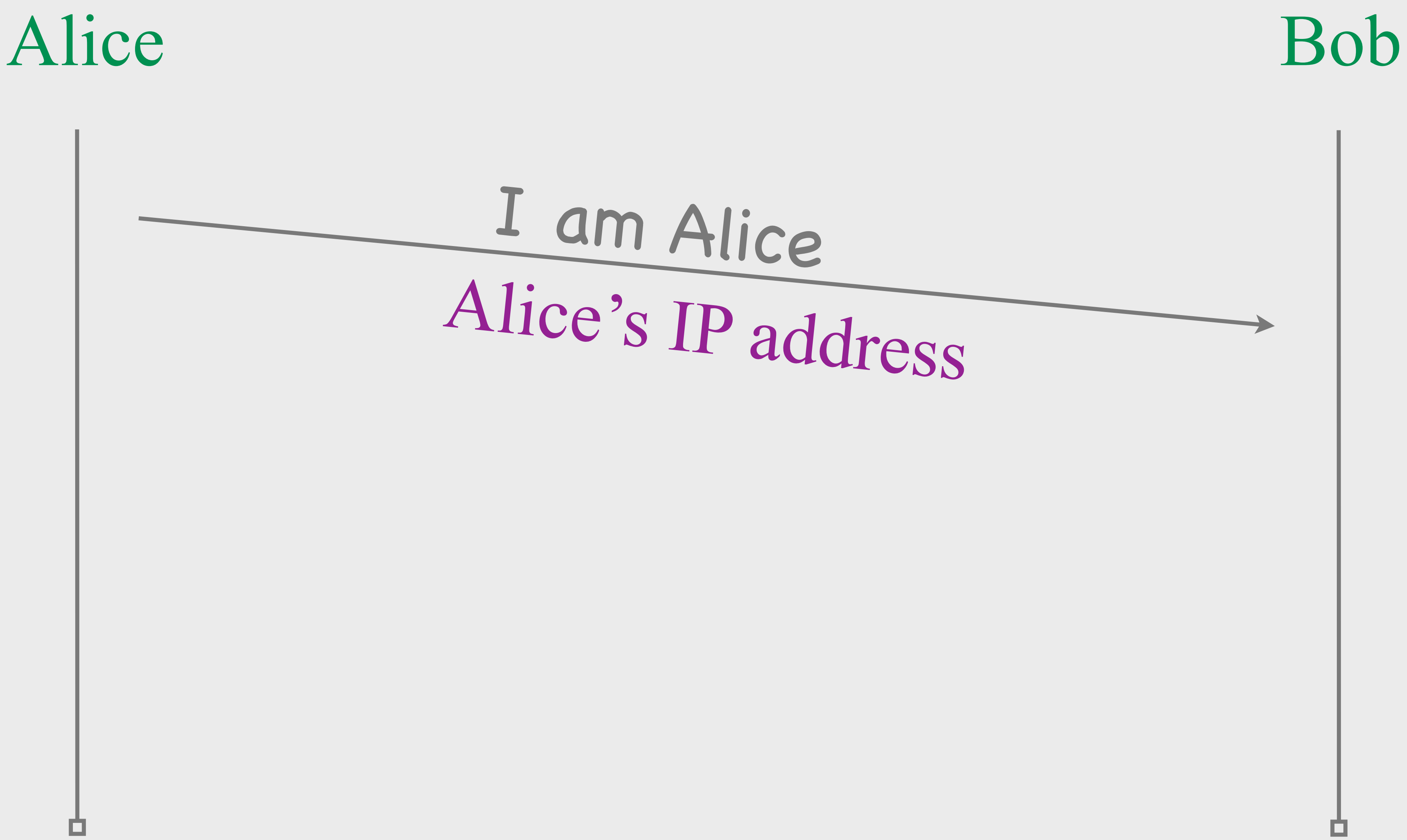
Providing confidentiality

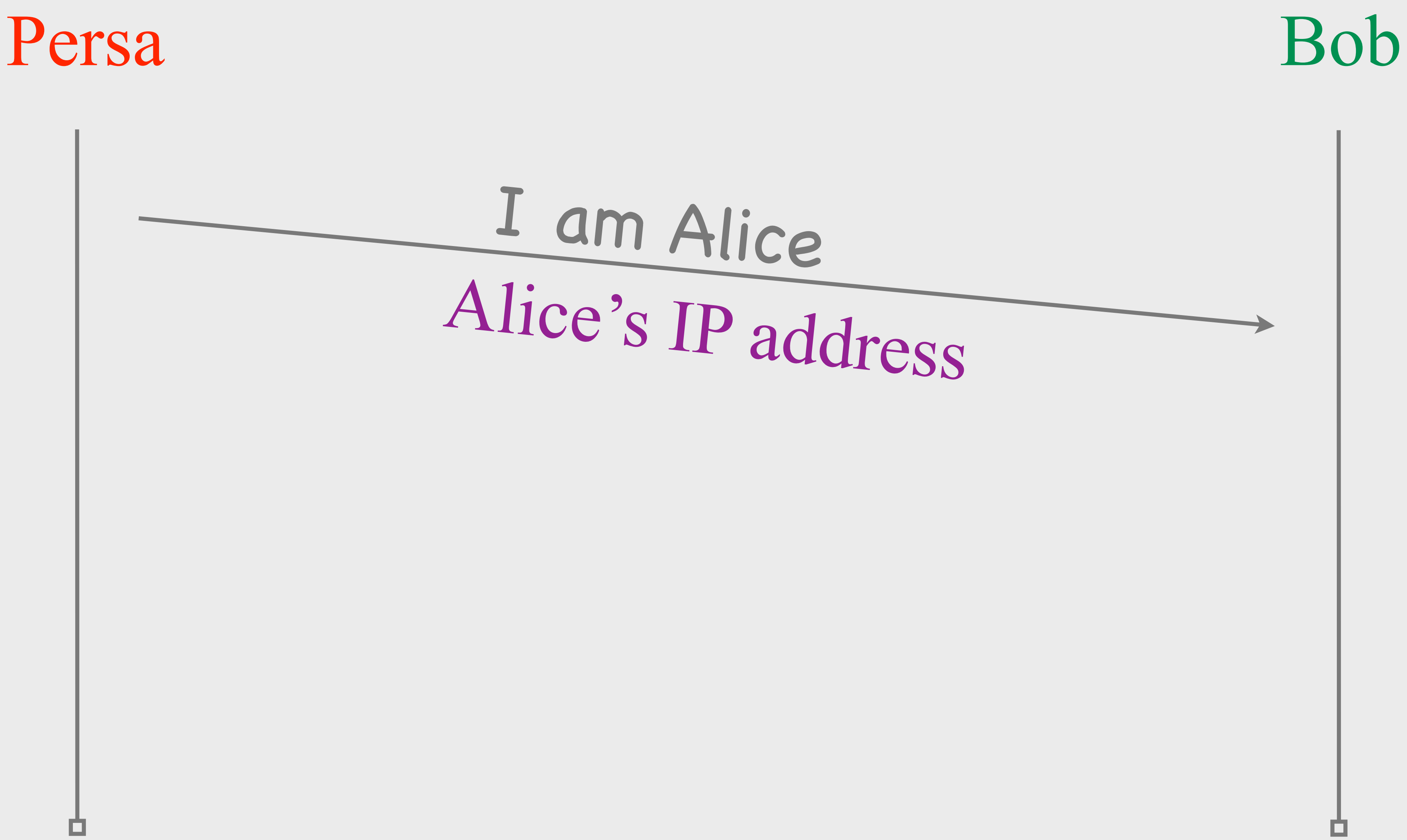
- With symmetric key crypto
  - Alice encrypts message with shared key
  - only Bob can decrypt it (with shared key)
- With asymmetric key crypto
  - Alice encrypts message with Bob's public key
  - only Bob can decrypt it (with his private key)

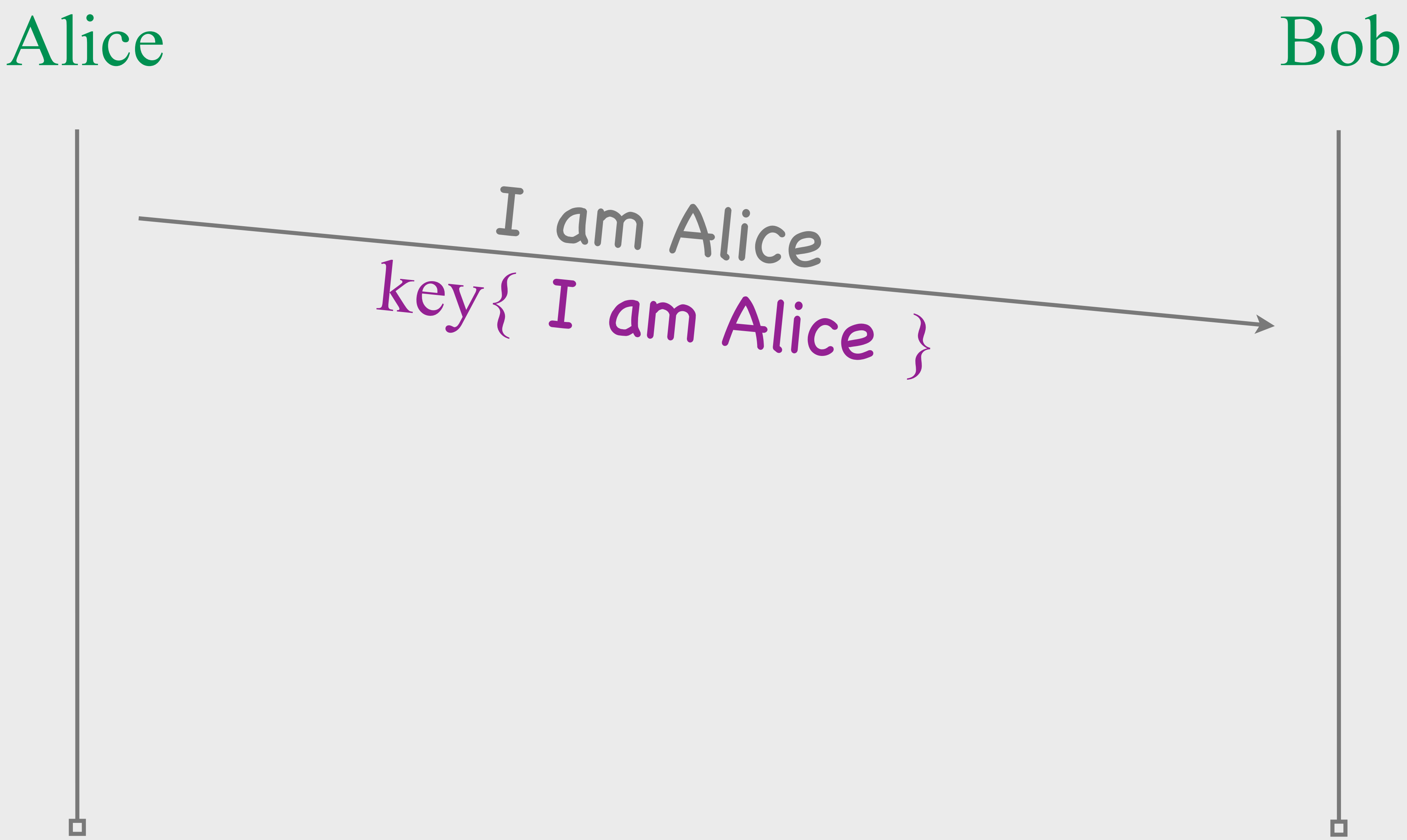
Providing authenticity


Alice
Bob
I am Alice

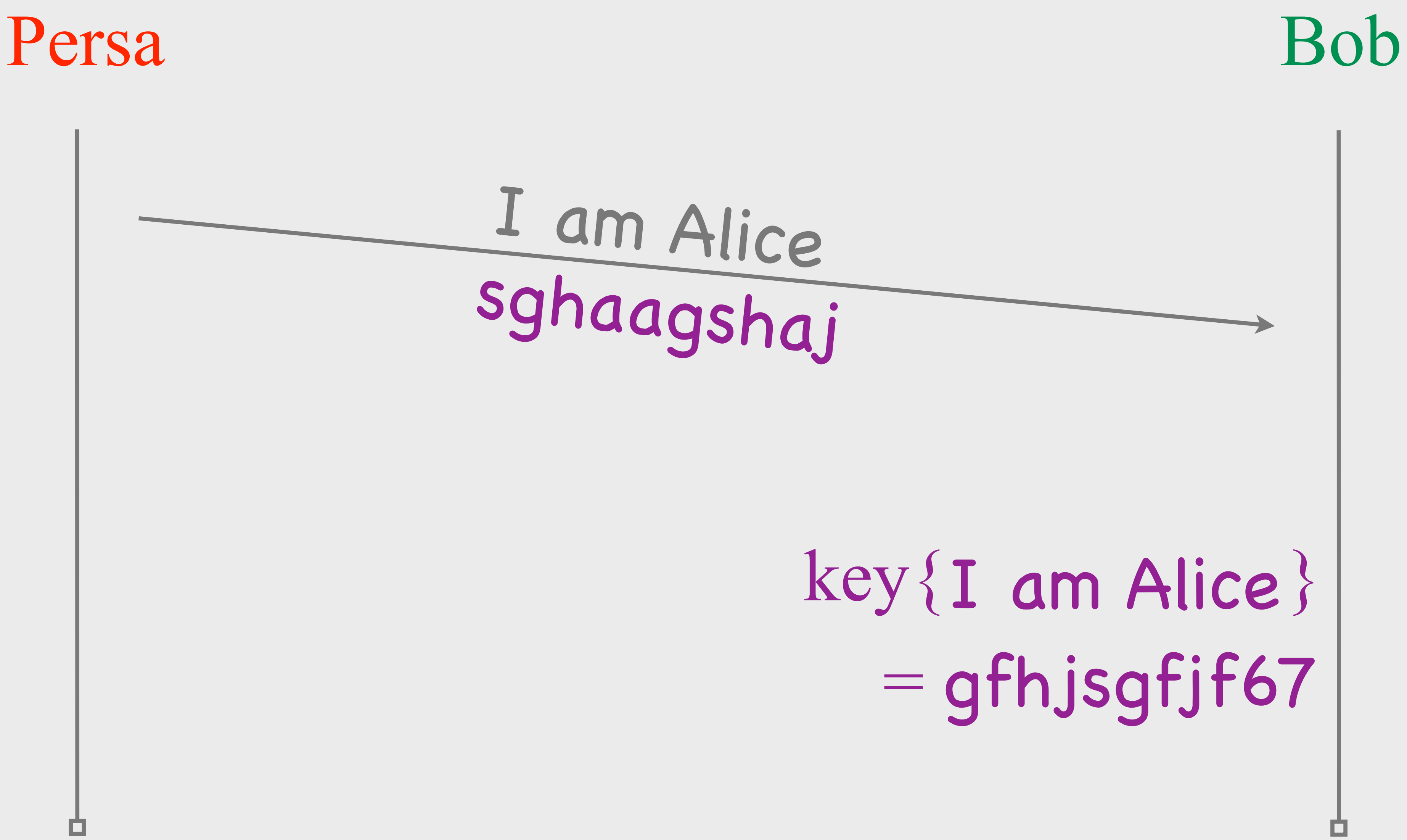

Persa
Bob
I am Alice

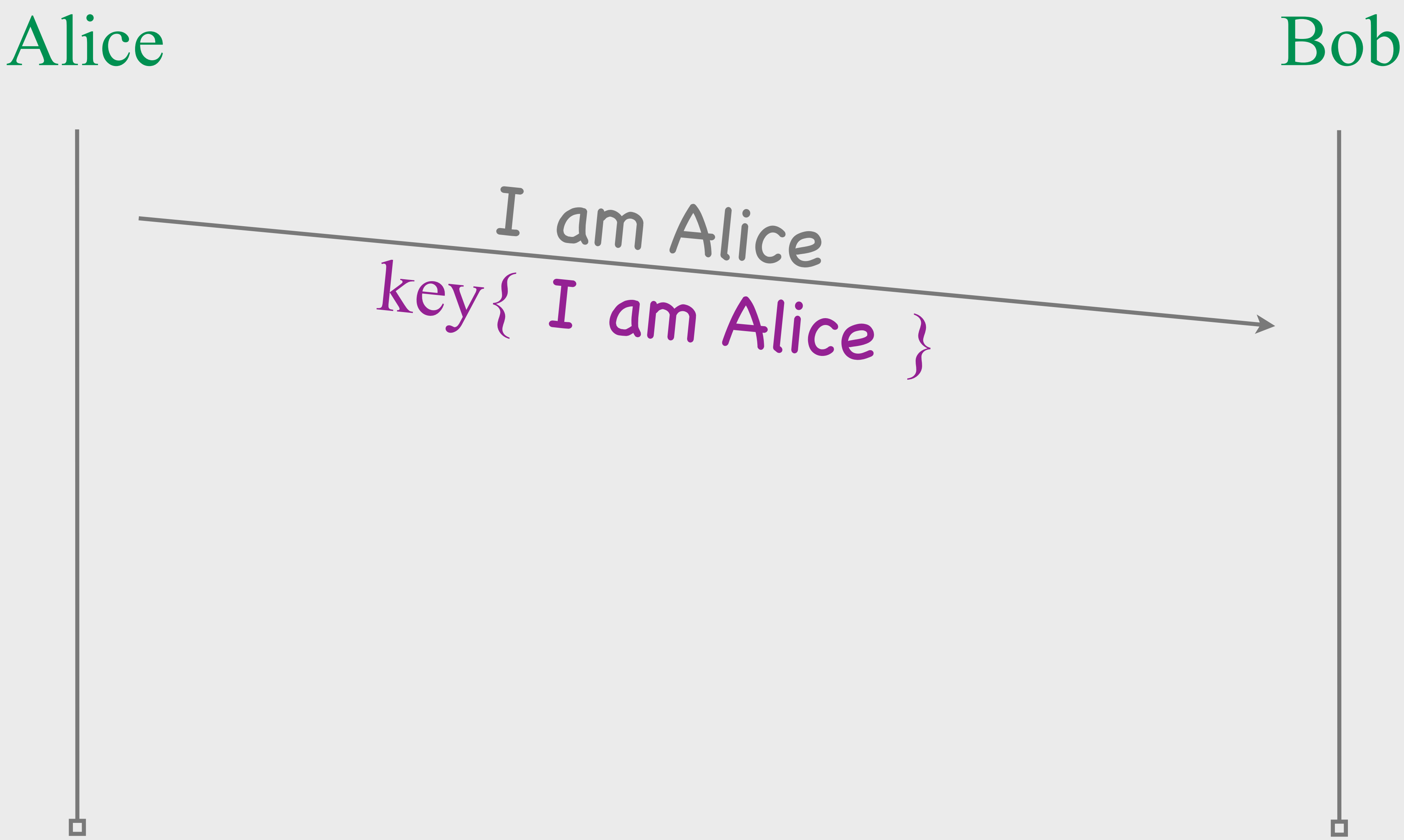

Alice
Bob
I am Alice
Alice's IP address

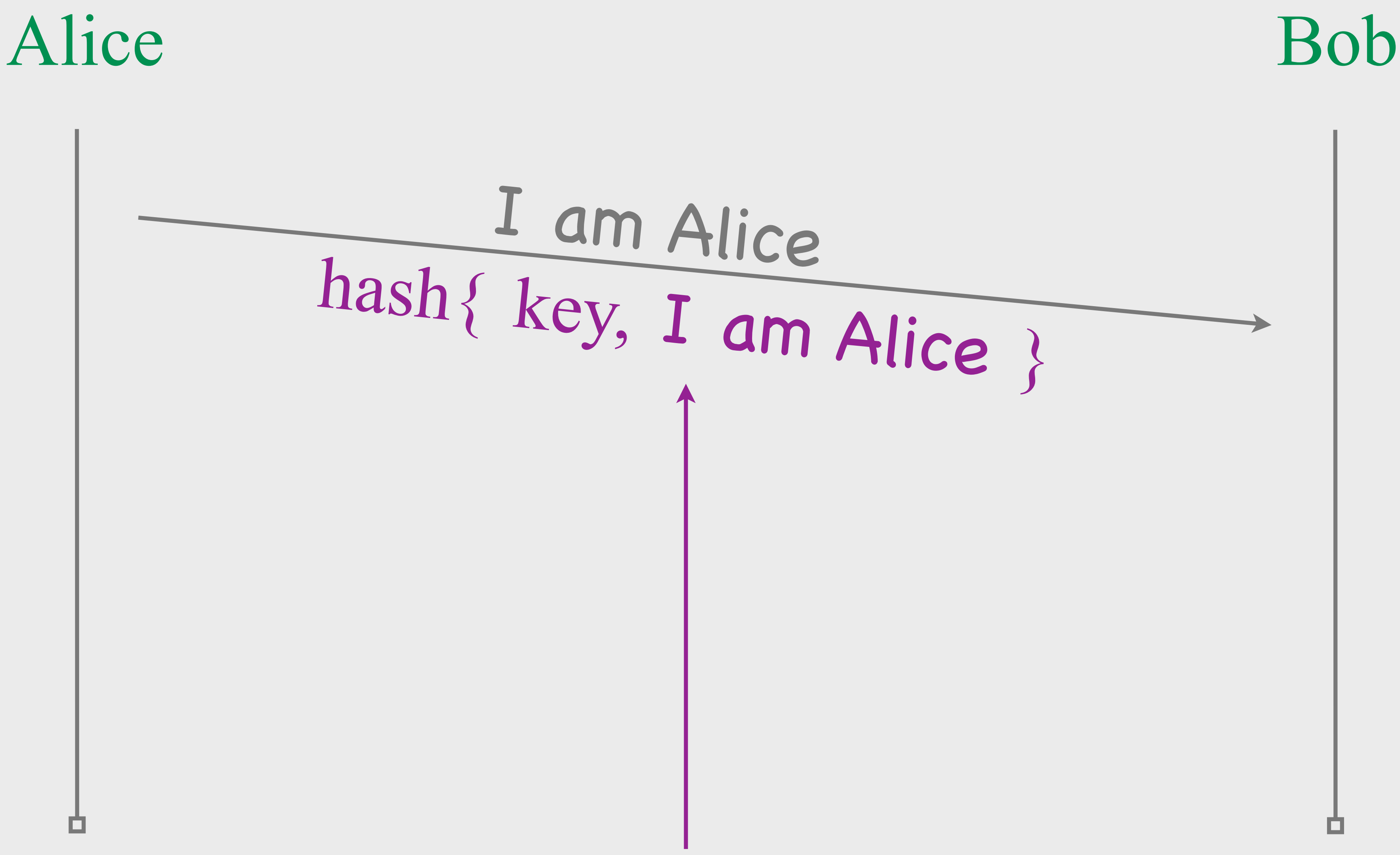

Persa
Bob
I am Alice
Alice's IP address


Alice
Bob
I am Alice
key{ I am Alice }

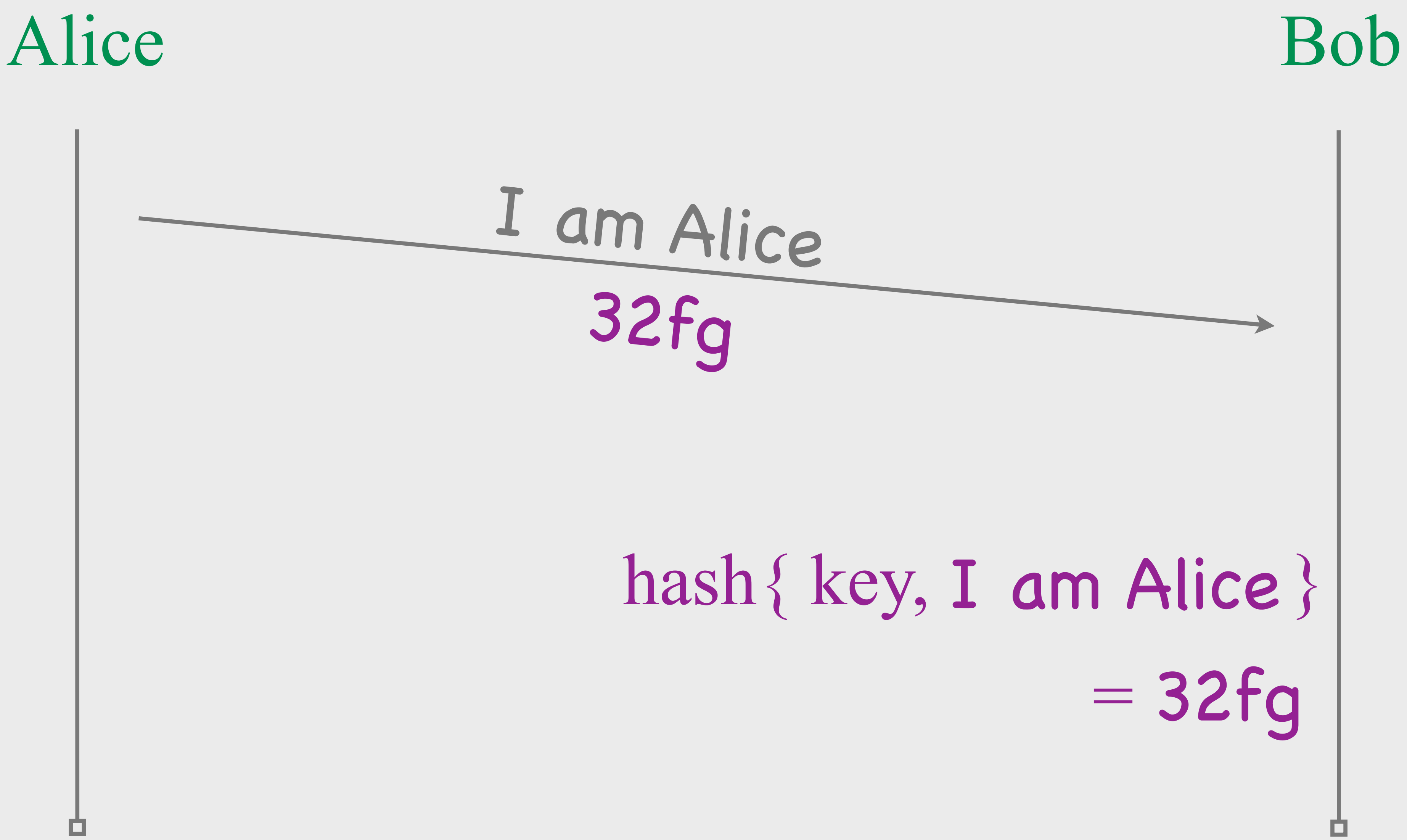

Alice
Bob
I am Alice
gfhjsgfjf67
key{I am Alice}
= gfhjsgfjf67


Persa
Bob
I am Alice
sghaagshaj
key{I am Alice}
= gfhjsgfjf67


Alice
Bob
I am Alice
key{ I am Alice }

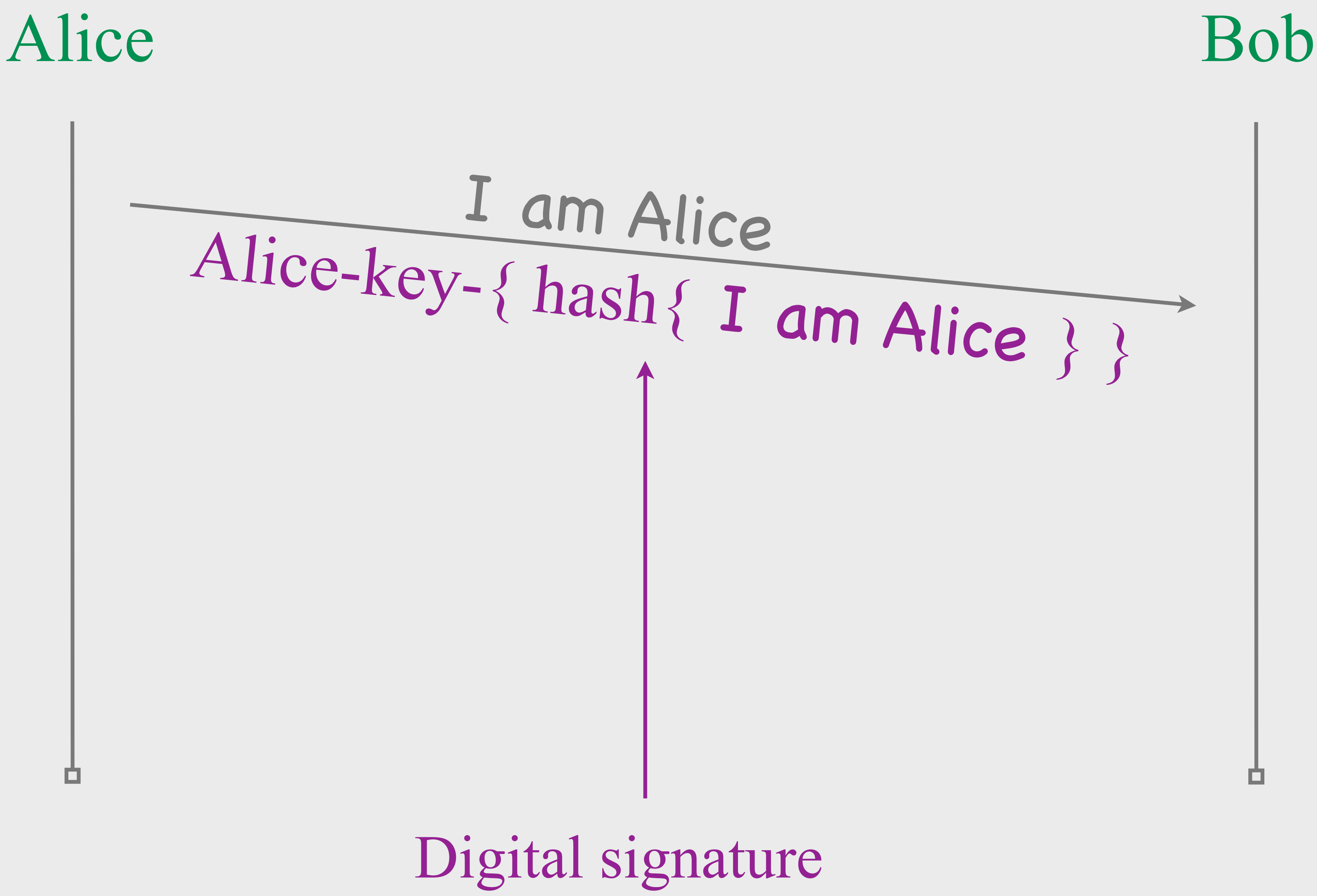


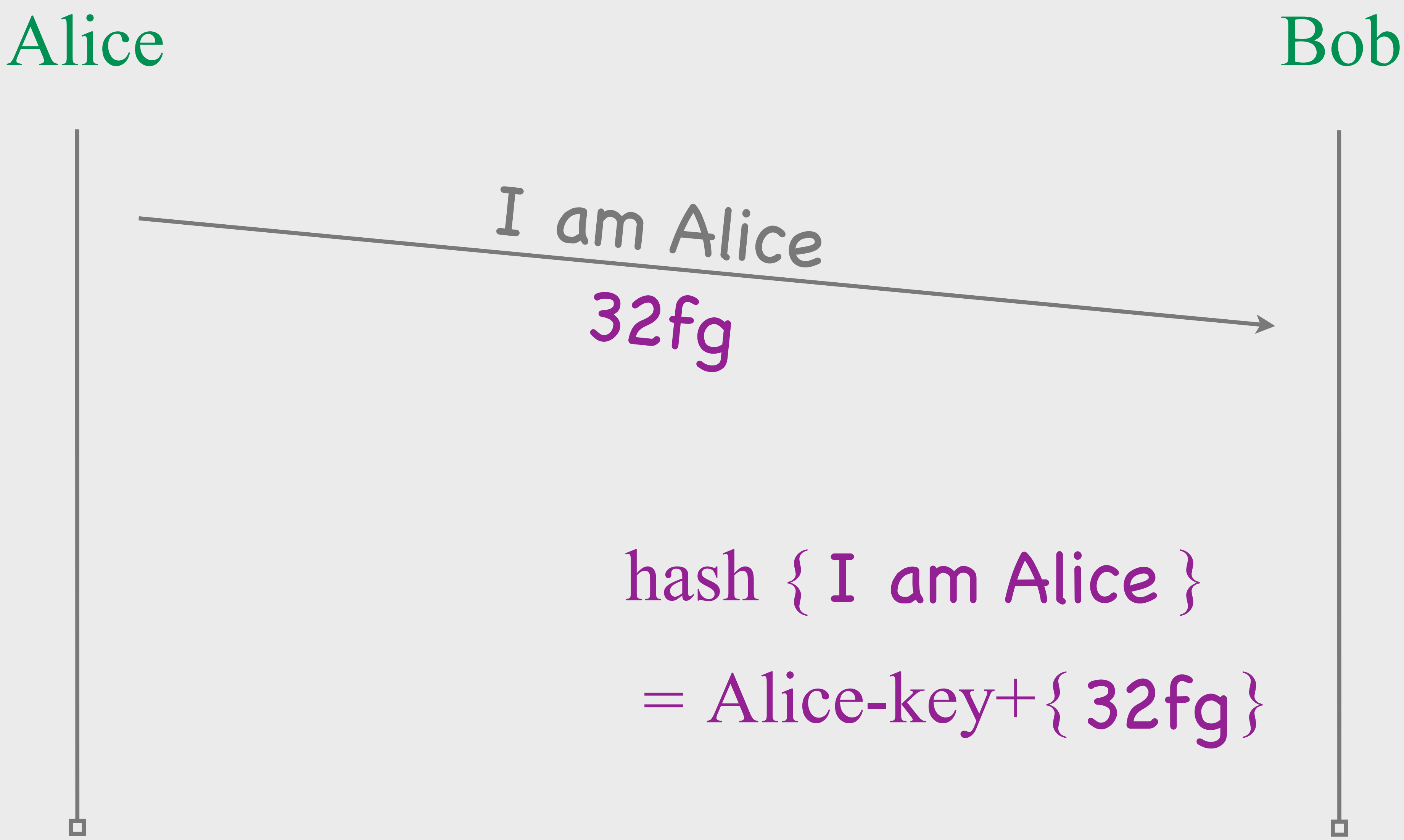
Message Authentication Code
(MAC)


Alice
Bob
I am Alice
32fg
hash{ key, I am Alice }
= 32fg

Message Authentication Code

- hash{ key, plaintext }
- Proof that this particular plaintext was sent by an entity that knows the key

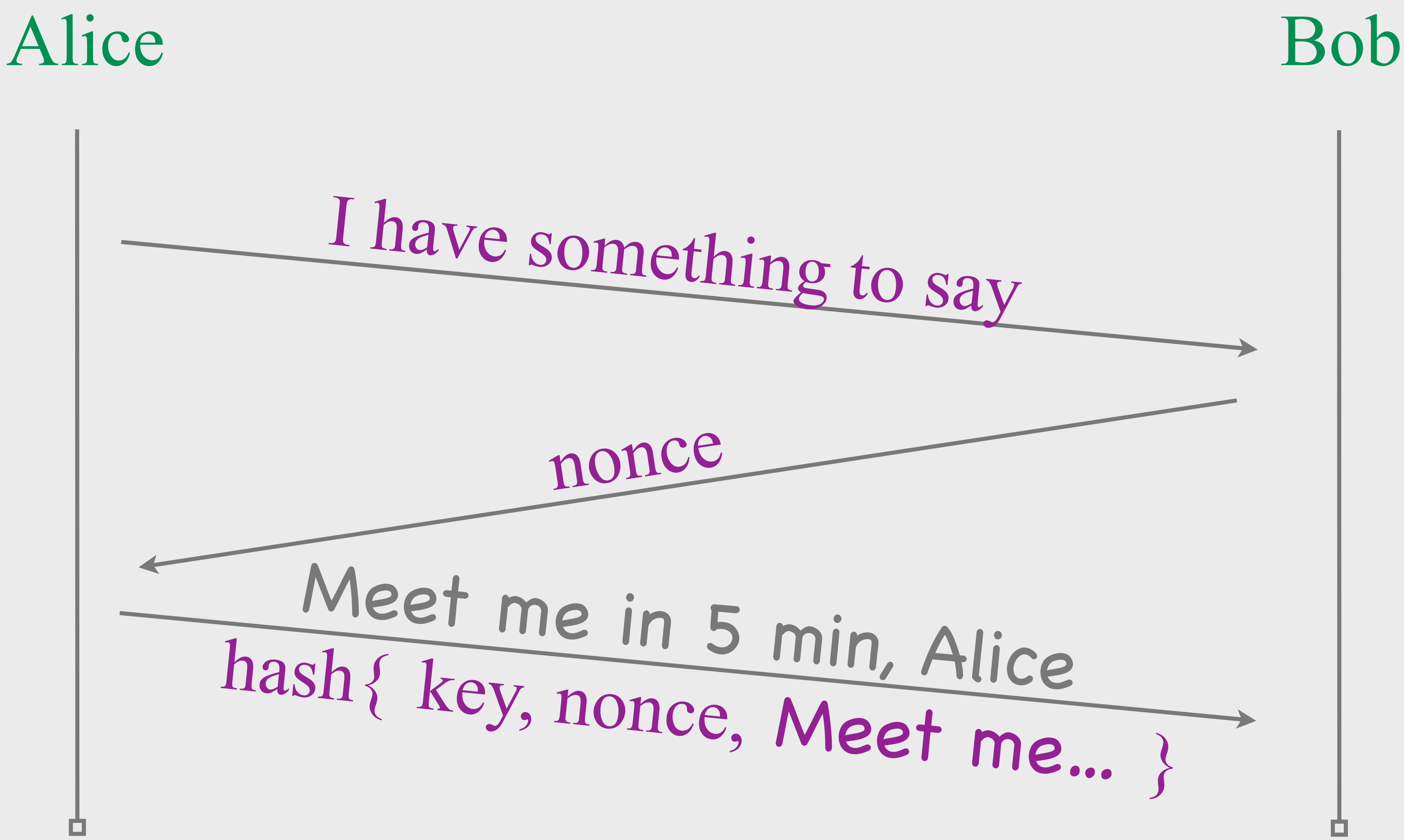

Alice
Bob
I am Alice
Alice-key-{ hash{ I am Alice } }
Digital signature


Alice
Bob
I am Alice
32fg
hash { I am Alice }
= Alice-key+{ 32fg }

Digital signature

- Generate: key-{ hash{ message } }
- Verify: key+{…} == hash { message }
- Proof that this particular message was sent by an entity who knows the private key that matches public key key+


Alice
Bob
Meet me in 5 min, Alice
hash{ key, Meet me… }



Alice
Bob
I have something to say
nonce
Meet me in 5 min, Alice
hash{ key, nonce, Meet me… }

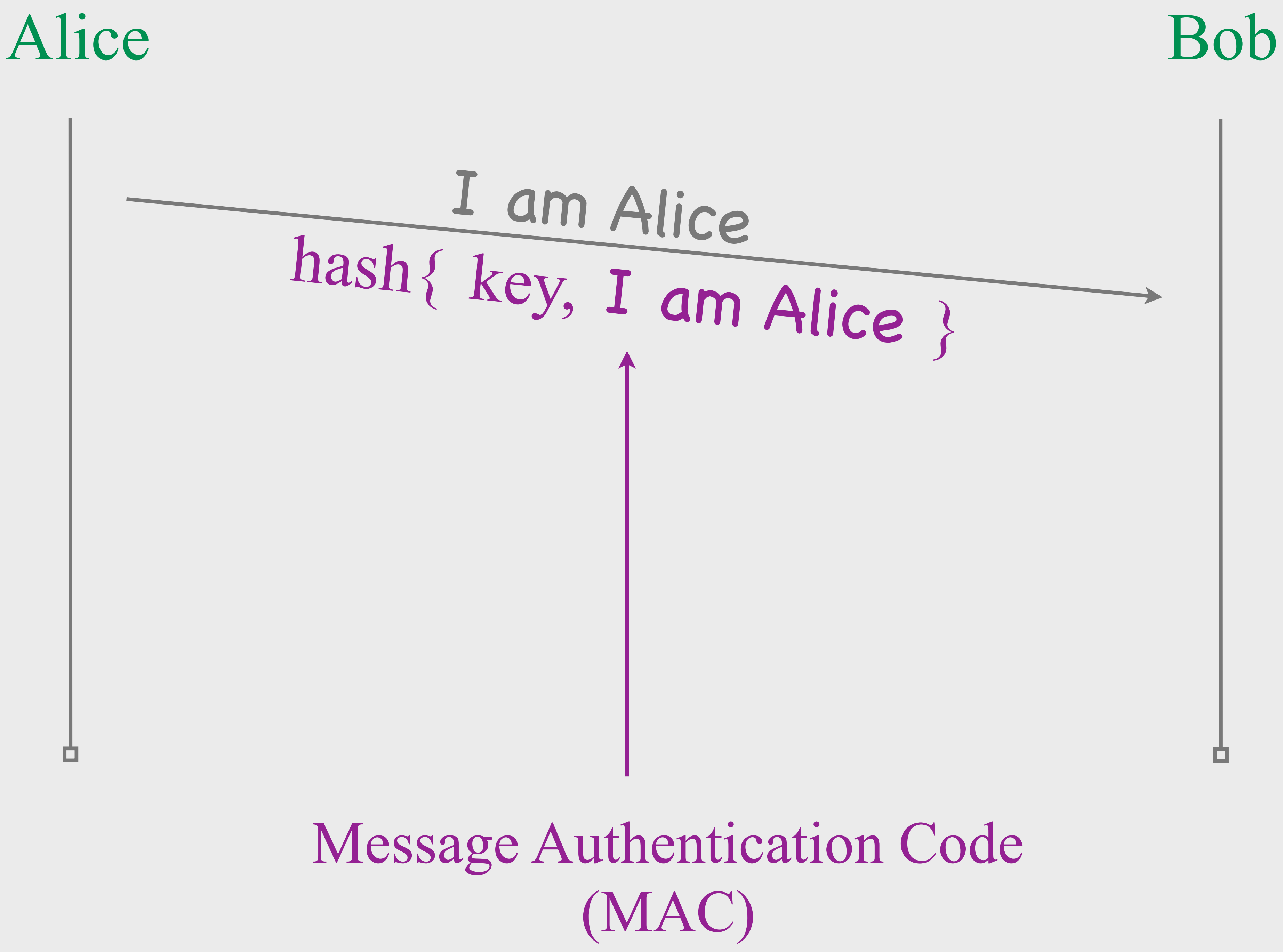
Providing authenticity

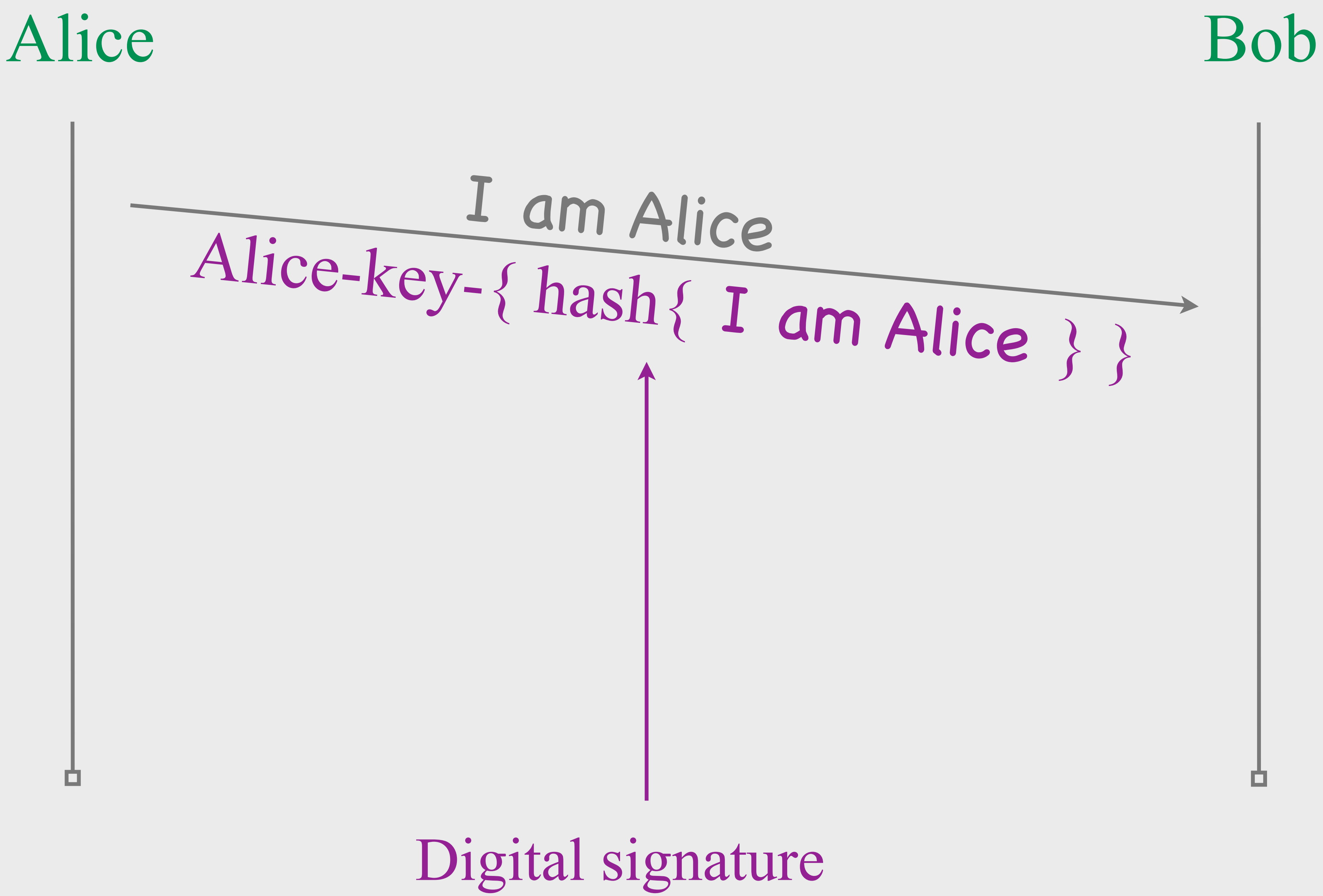
- With symmetric key crypto
  - Alice appends MAC
  - Bob checks that it is correct (using shared key)
- With asymmetric key crypto
  - Alice appends digital signature (using her private key)
  - Bob checks that it is correct (using Alice's public key)

Providing authenticity

- Use nonce to prevent replay attacks
  - Alice appends MAC of nonce + message
  - Bob verifies that it is correct (using shared key)

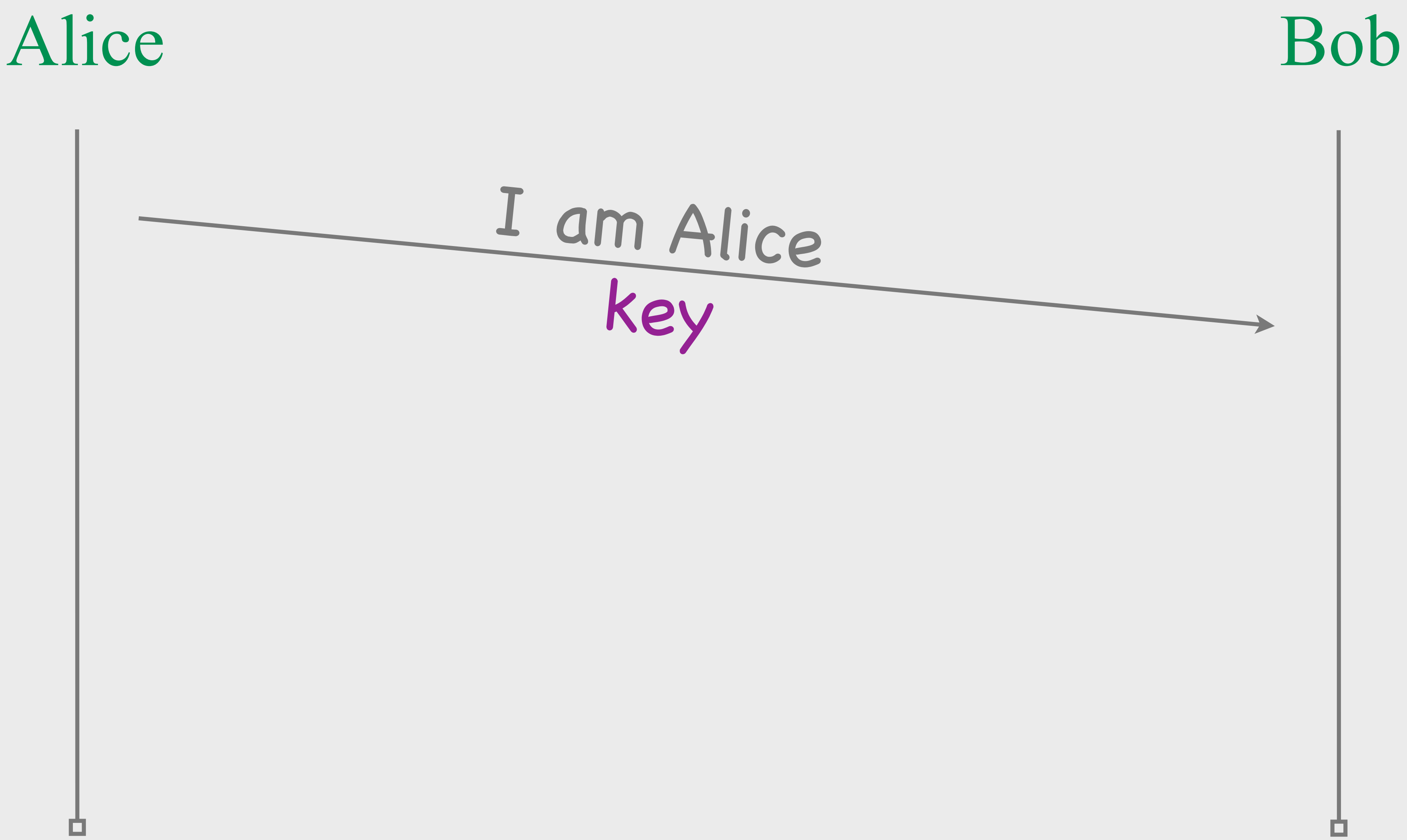
Providing data integrity


Alice
Bob
I am Alice
hash{ key, I am Alice }
Message Authentication Code
(MAC)

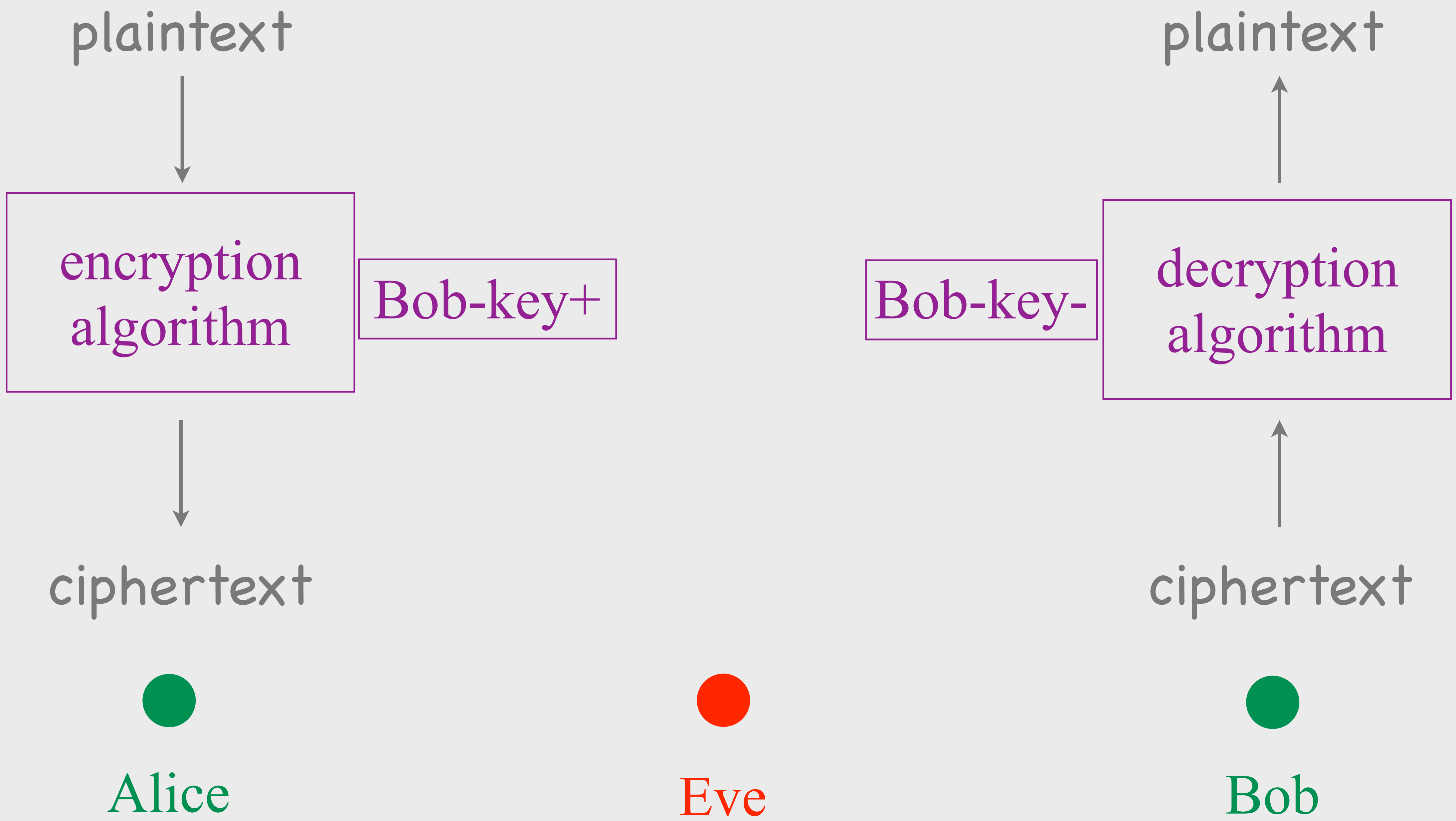

Alice
Bob
I am Alice
Alice-key-{ hash{ I am Alice } }
Digital signature

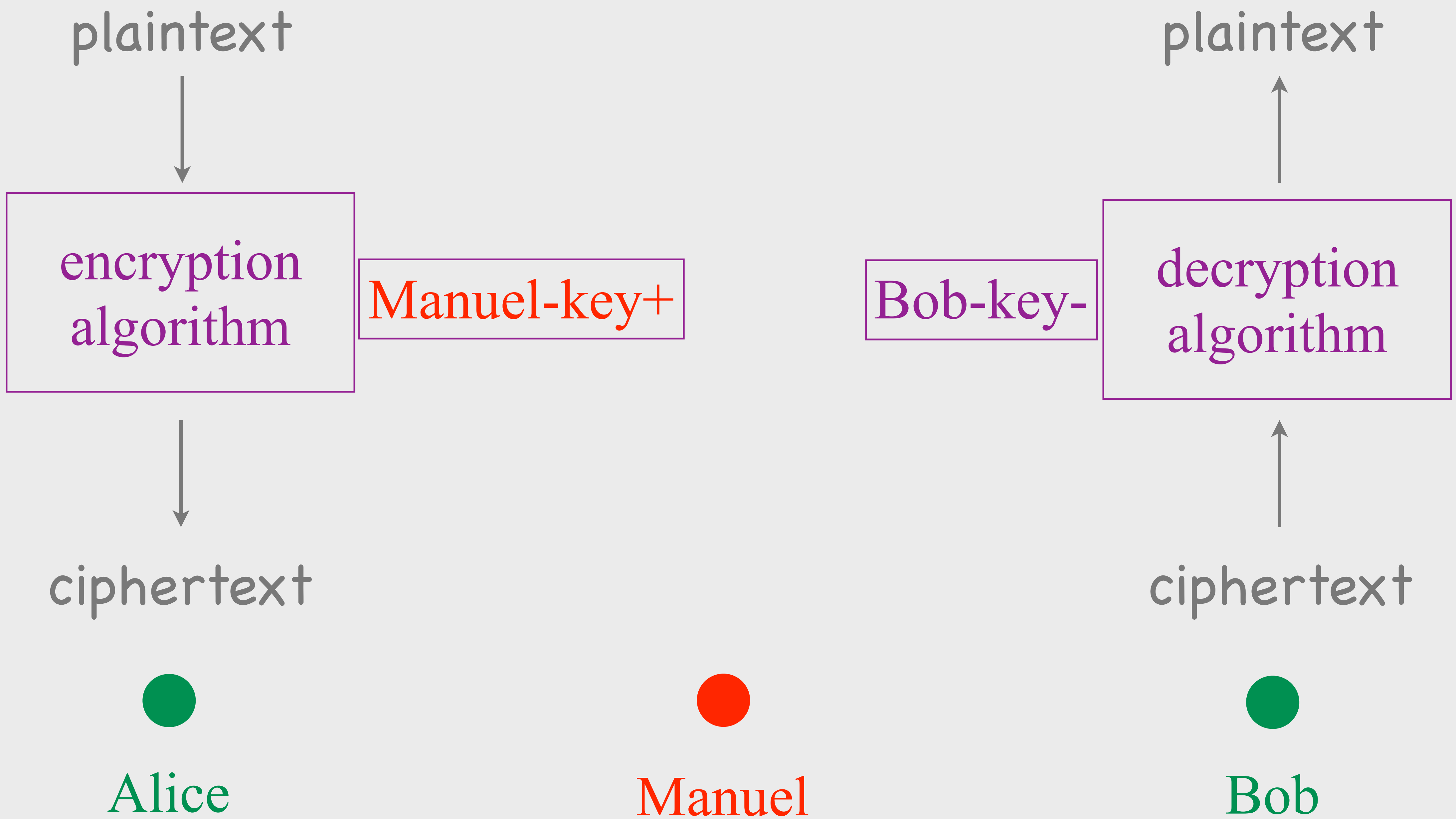
Providing integrity

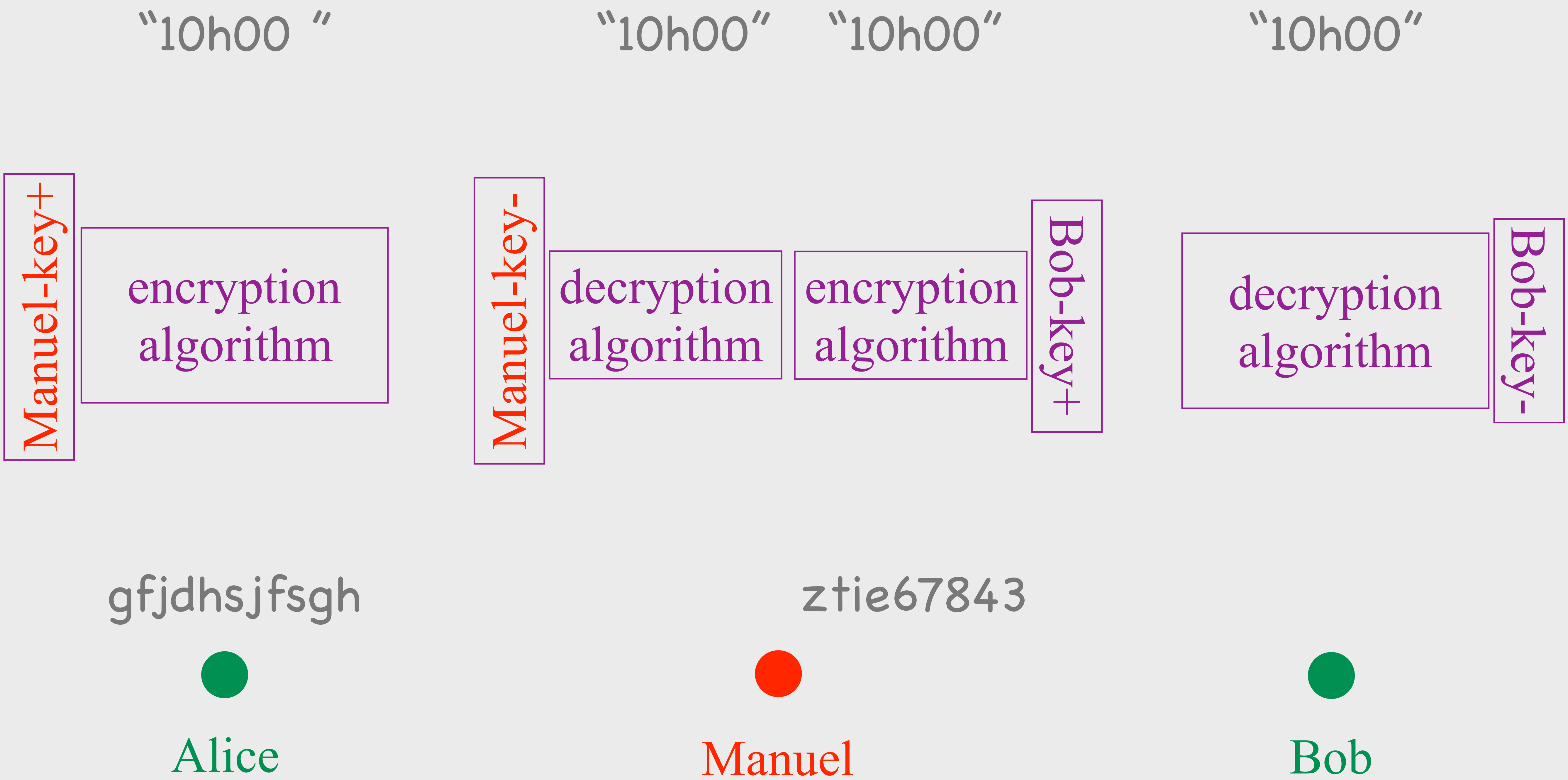
- With exactly the same mechanisms that provide authenticity


Alice
Bob
I am Alice
key

Preventing man-in-the-middle attacks

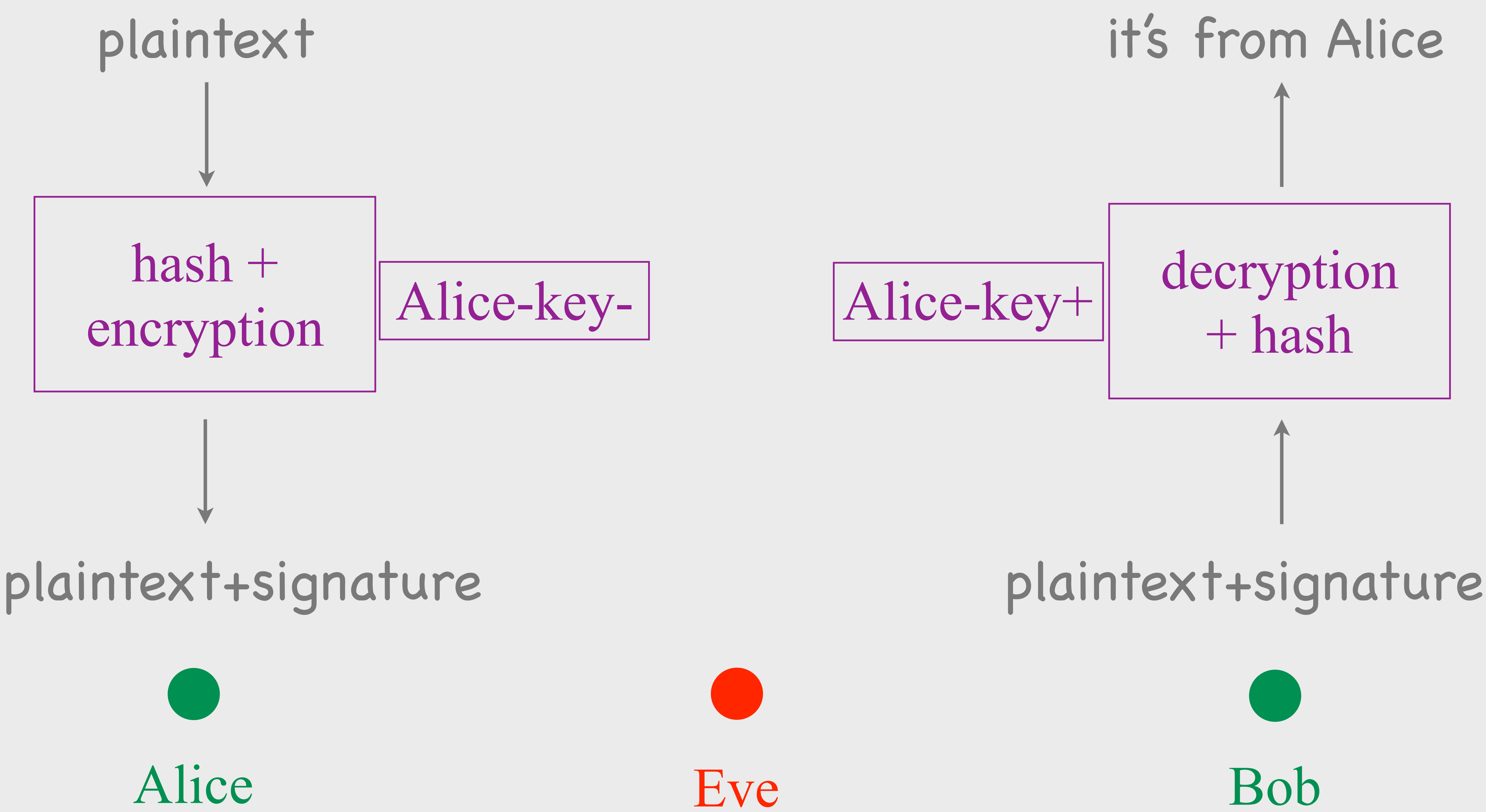

plaintext
encryption algorithm
Bob-key+
ciphertext
Alice
Eve
plaintext
decryption algorithm
Bob-key-
ciphertext
Bob

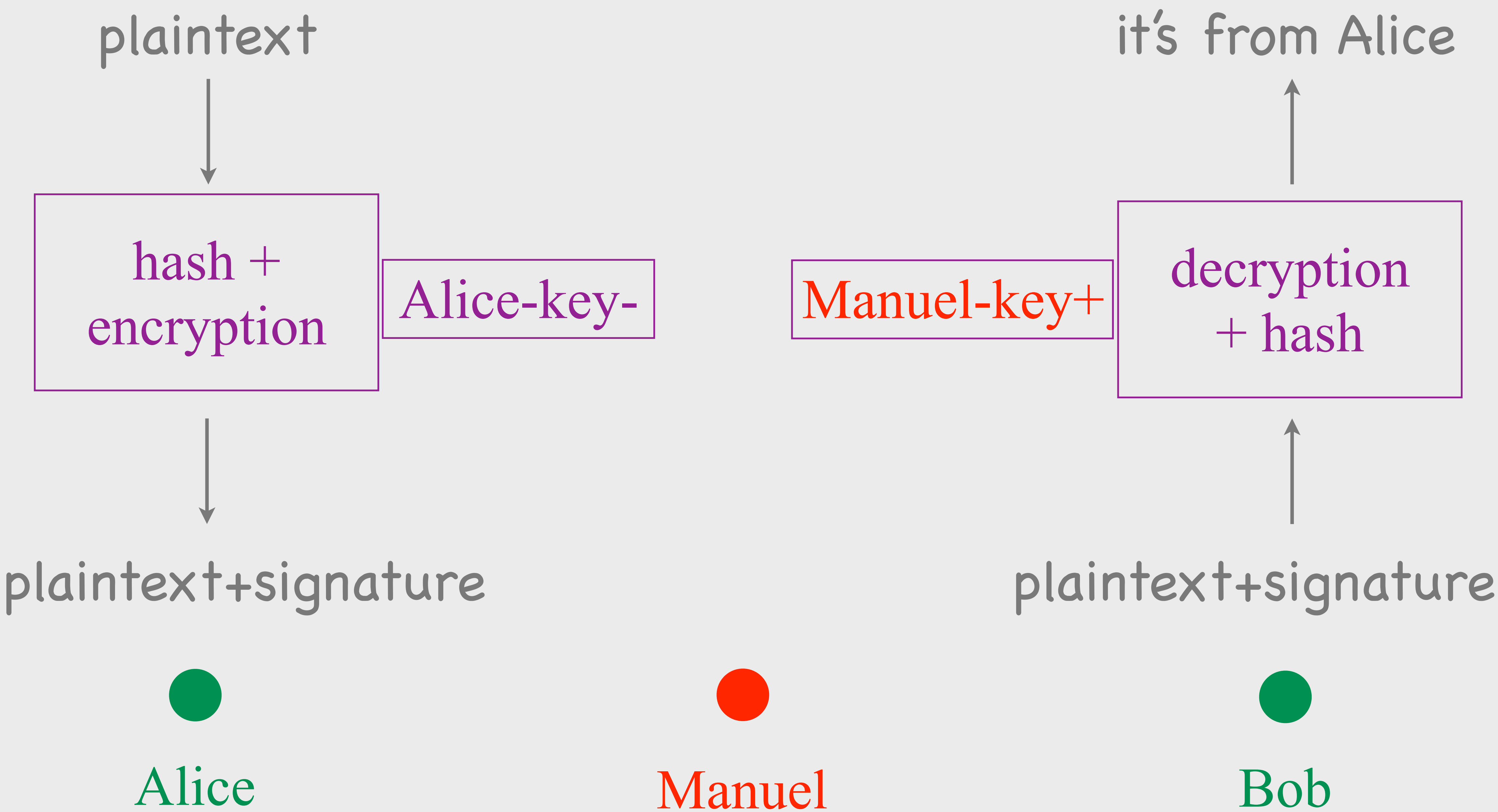

plaintext
encryption algorithm
Manuel-key+
ciphertext
Alice
Manuel
plaintext
Bob-key-
decryption algorithm
ciphertext
Bob

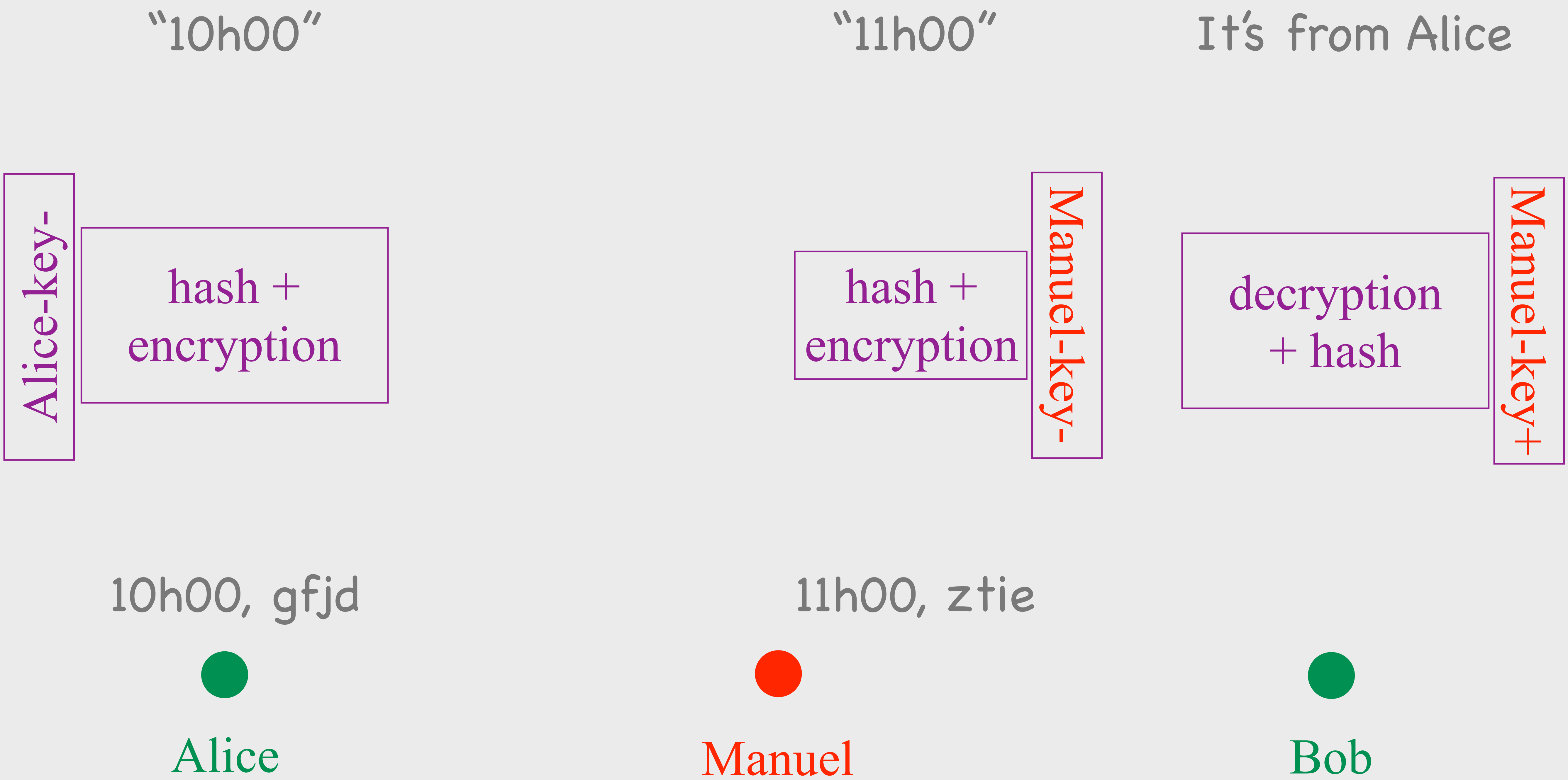

"10h00 "
"10h00"
"10h00"
"10h00"
Manuel-key+
encryption algorithm
Manuel-key-
decryption algorithm
encryption algorithm
Bob-key+
decryption algorithm
Bob-key-
gfjdhsjfsgh
ztie67843
Alice
Manuel
Bob

Man in the middle

- Can break confidentiality
  - Manuel convinces Alice to use his public key instead of Bob's
  - decrypts and re-encrypts Alice-Bob messages
- Cause: no way to verify public-keys
  - when Alice learns Bob's public key, she must verify that it is indeed his


plaintext
hash + encryption
Alice-key-
plaintext+signature
Alice
Eve
it's from Alice
Alice-key+
decryption + hash
plaintext+signature
Bob


plaintext
hash + encryption
Alice-key-
plaintext+signature
it's from Alice
Manuel-key+
decryption + hash
plaintext+signature
Alice
Manuel
Bob


"10h00"
"11h00"
It's from Alice
Alice-key-
hash + encryption
hash + encryption
Manuel-key-
decryption + hash
Manuel-key+
10h00, gfjd
11h00, ztie
Alice
Manuel
Bob

Man in the middle

- Can break authenticity/data integrity
  - Manuel convinces Bob to use his public key instead of Alice's
  - re-computes the digital signatures on Alice-Bob messages
- Cause: no way to verify public-keys
  - when Bob learns Alice's public key, he must verify that it is indeed hers

Solution: public-key certificates

- Rely on trusted certificate authority (CA)
  - an entity that both Alice & Bob trust
- CA produces certificate of Bob's public key
  - { Bob owns Bob-key+ }
- CA digitally signs the certificate
  - CA-key-{ hash{ Bob owns Bob-key+ } }

Solution: public-key certificates

- Alice needs Bob's true public key
  - to communicate with Bob with confidentiality and/or authenticity/data integrity
- Bob sends public key & certificate
  - CA-key-{ hash{ Bob owns Bob-key+, … } }
  - guarantees this is Bob's public key
- Alice needs CA's true public key
  - to check CA-key-{ hash{ Bob owns Bob-key+, … } }

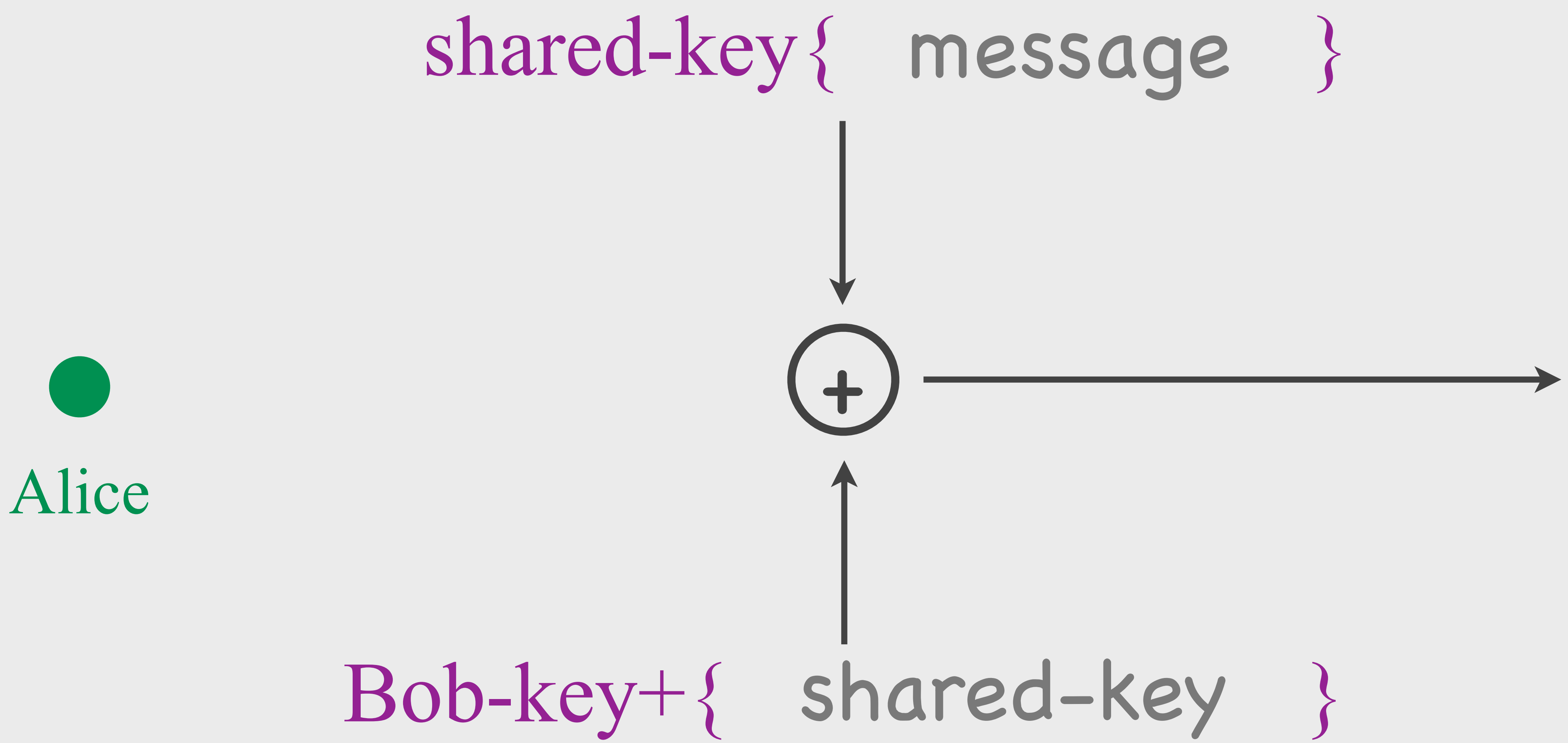
Bootstrapping is unavoidable

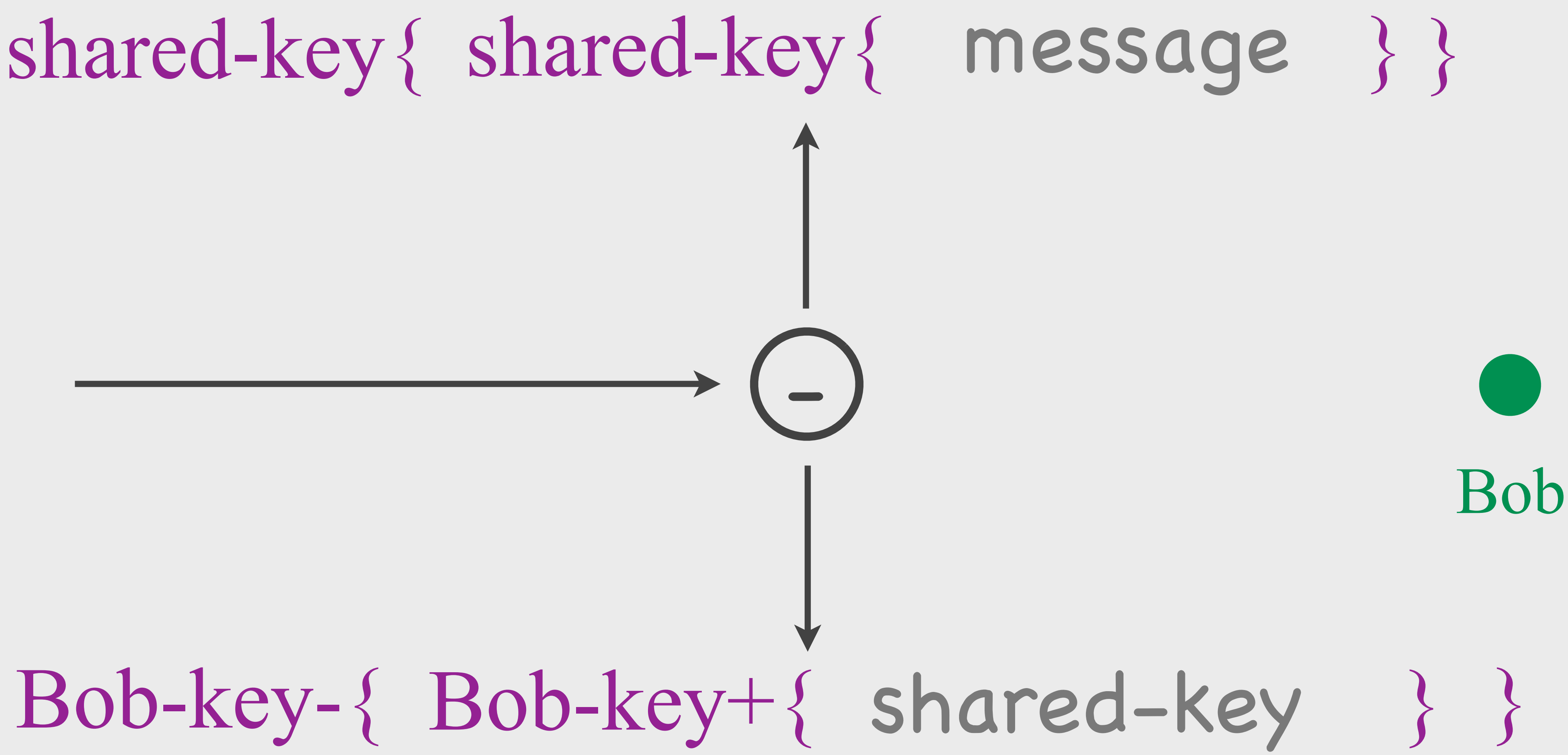
- Secure communication requires **some form of shared state**
- Symmetric crypto: secret key
- Asymmetric crypto: CA's public key
  - typically stored in browser

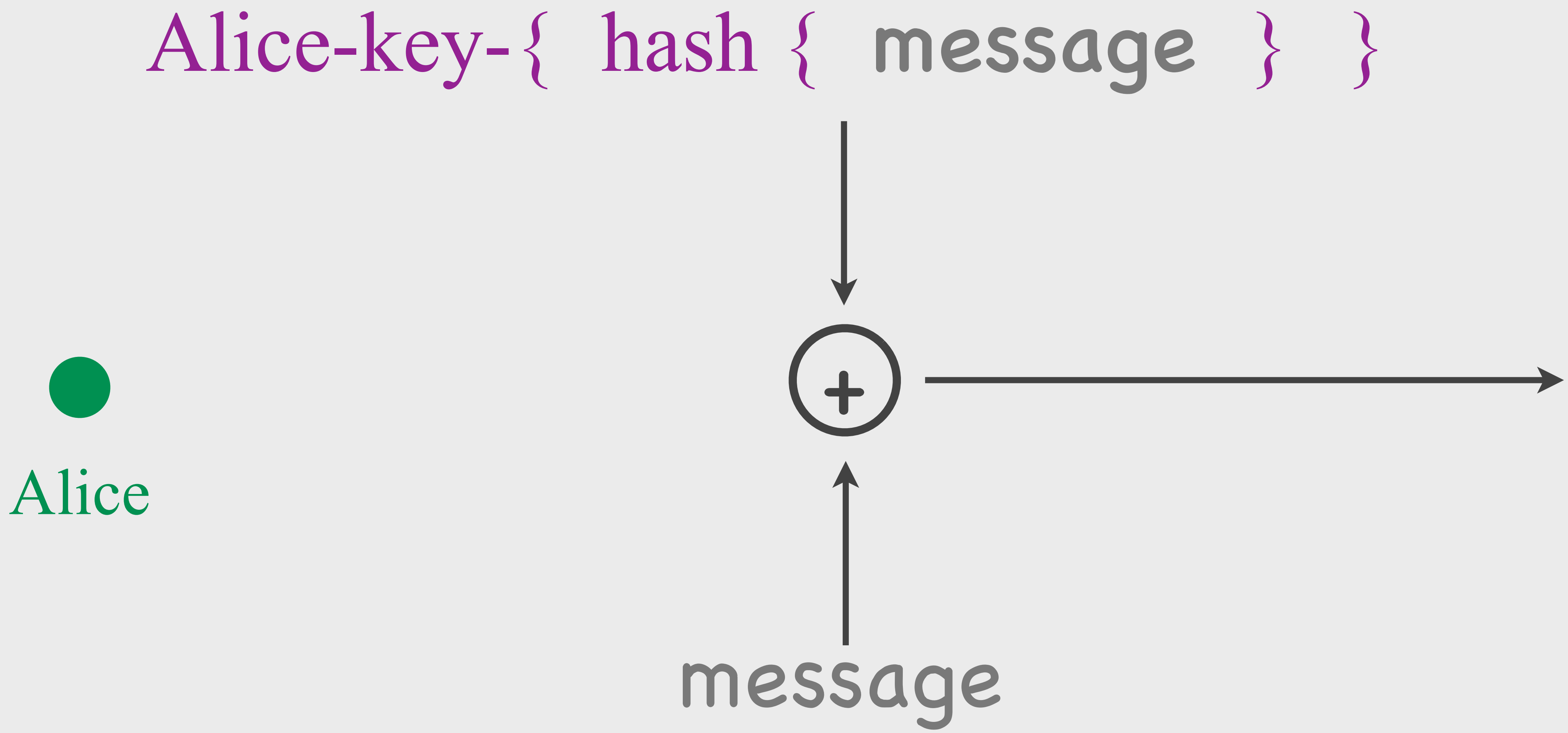
Asymmetric crypto reduces bootstrapping information

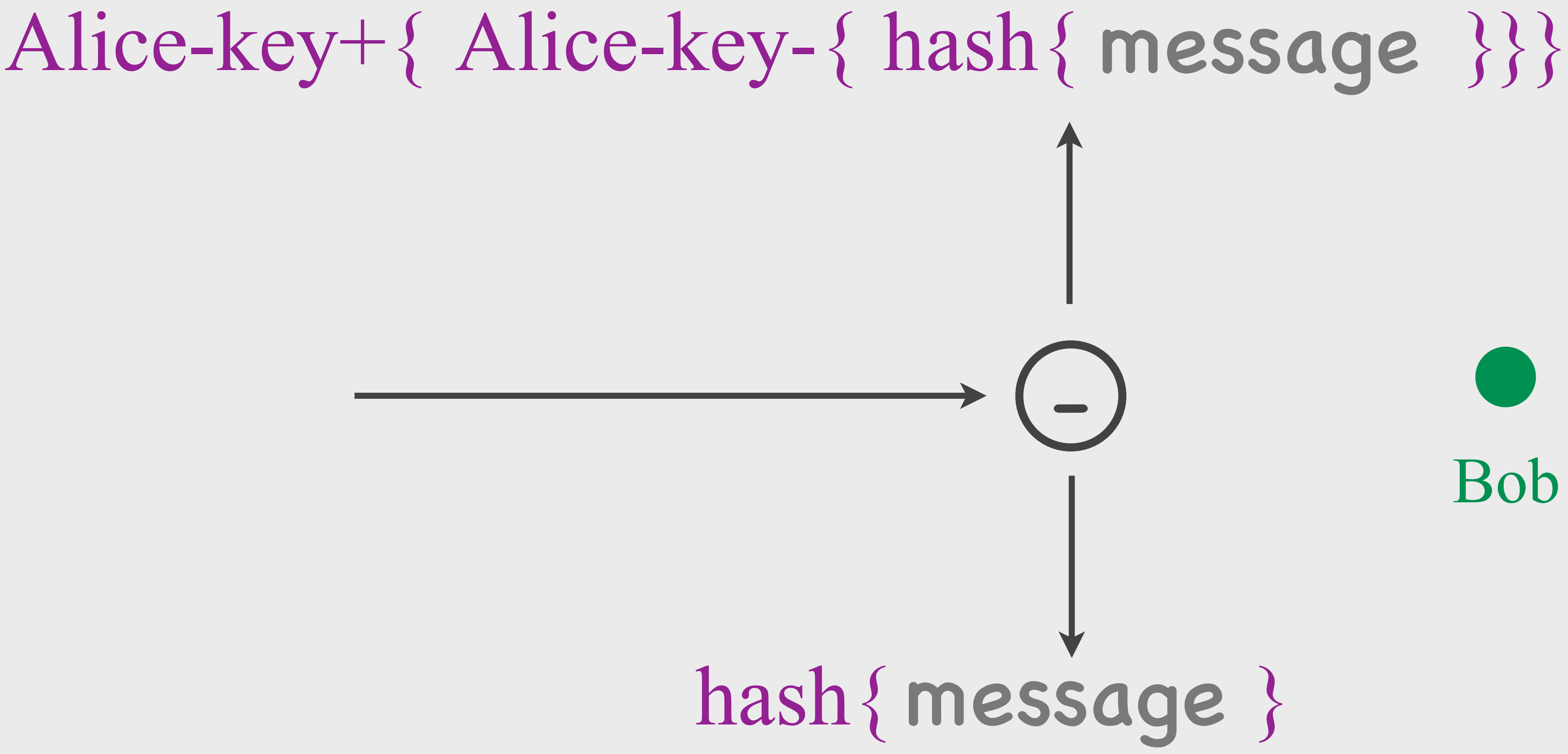
Outline

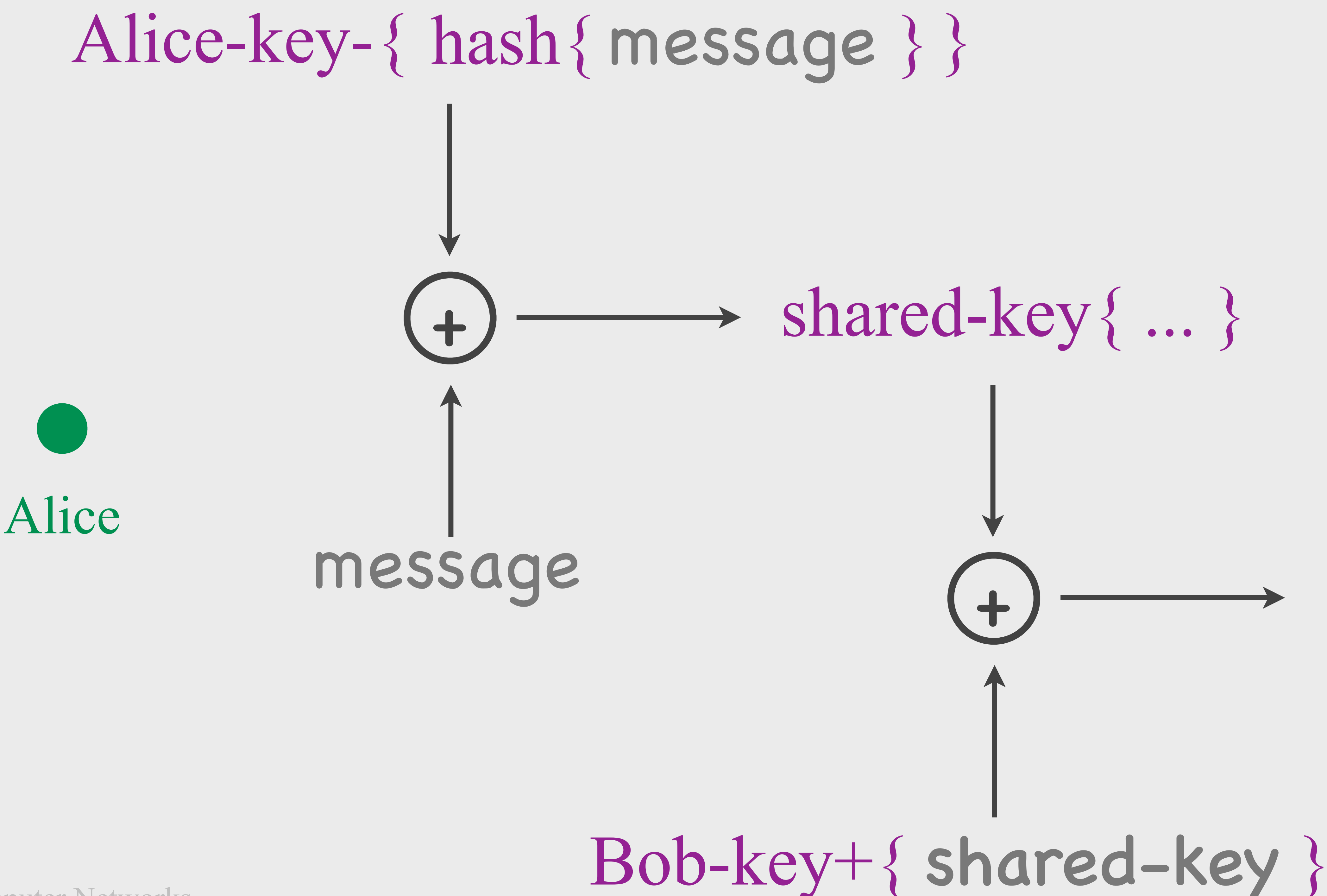
- Building blocks
- Providing security properties
- Securing Internet protocols
- Operational security

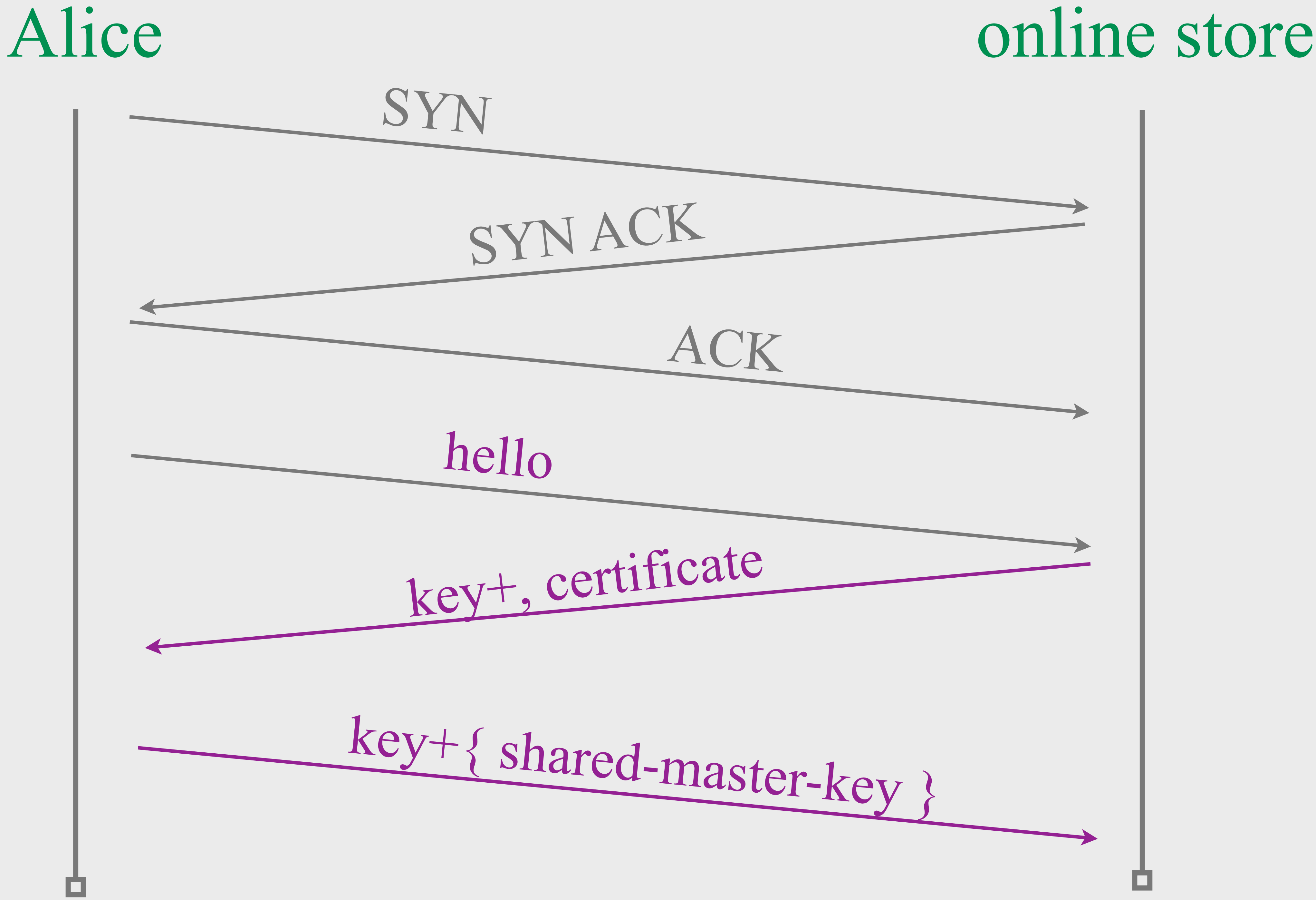

shared-key{ message }
Alice
Bob-key+{ shared-key }


shared-key{ shared-key{ message } }
Bob
Bob-key-{ Bob-key+{ shared-key } }


Alice-key-{ hash { message } }
Alice
message

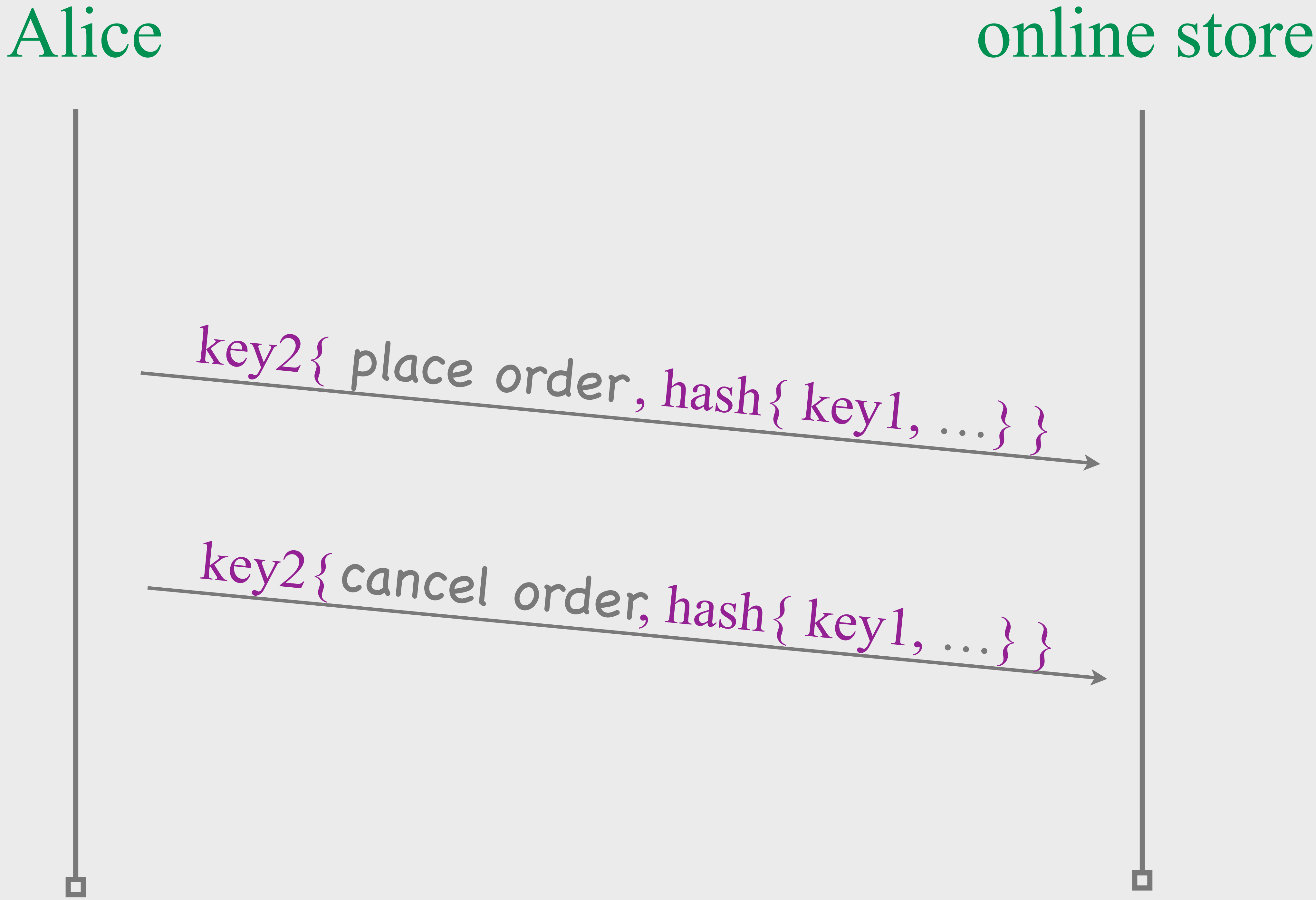

Alice-key+{ Alice-key-{ hash{ message }}}
Bob
hash{ message }


Alice-key-{ hash{ message } }
shared-key{ ... }
Alice
message
Bob-key+{ shared-key }


Alice
online store
SYN
SYN ACK
ACK
hello
key+, certificate
key+{ shared-master-key }

Securing TCP applications

- Server sends its public key & certificate
- Client creates and sends a shared master key
  - encrypts it with server's public key
- Both use master key to create 4 session keys
  - 1 key for encrypting client --> server data
  - 1 key for creating MAC for client --> server data
  - same for server --> client data


Alice
online store
key2{ place order, hash{ key1, ...} }
key2{cancel order, hash{ key1, ...} }


Alice
online store
key2{place order , hash{ key1, #1 ...} }
key2{cancel order, hash{ key1, #2 ...} }

Securing TCP applications

- Client organizes data in records
  - each record has a sequence number
- Creates MAC for each record + sequence #
  - using one of the 4 session keys
- Encrypts the data + MAC for each record
  - using (another) one of the 4 session keys

Key ideas

- Combination of symmetric/asymmetric keys
  - asymmetric key crypto to exchange shared keys
  - symmetric key crypto for confidentiality, authenticity, & integrity
  - symmetric key crypto is faster
- Seq. numbers to avoid reordering attacks
  - organize data in records with seq. numbers
  - compute MAC on record data + seq. number

Outline

- Building blocks
- Providing security properties
- Securing Internet protocols
- Operational security

action	src IP	dst IP	proto	src port	dst port
allow	167.67/16	any	TCP	any	80
allow	any	167.67/16	TCP	80	any
deny	all	all	all	all	all