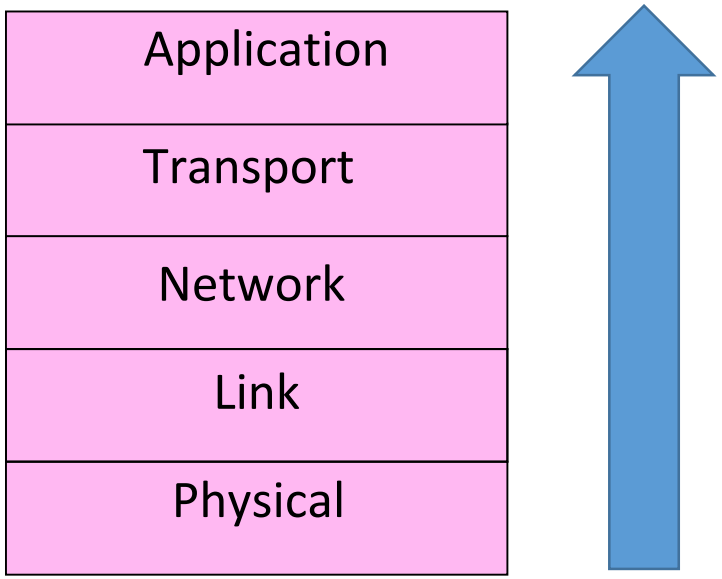
Network Security

Where we are in the Course

- Security crosses all layers



Encryption/Decryption (2)

- Encryption is a reversible mapping
  - Ciphertext is confused plaintext
- Assume attacker knows algorithm
  - Security does not rely on its secrecy
- Algorithm is parameterized by keys
  - Security does rely on key secrecy
  - Must be distributed (Achilles' heel)

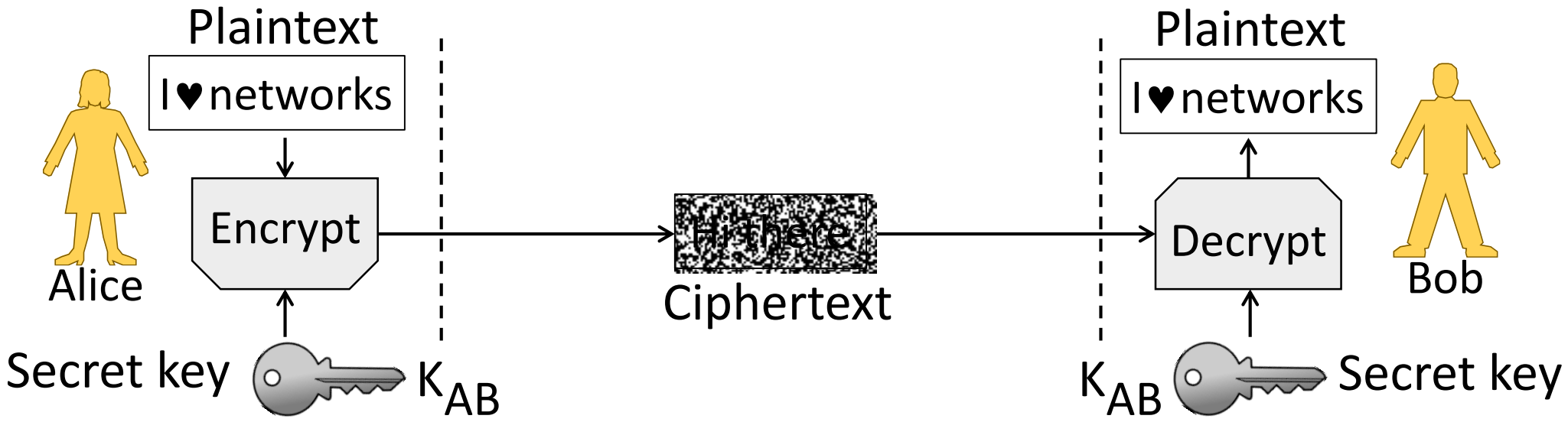
Encryption/Decryption (3)

Two main kinds of encryption:

1. Symmetric key encryption », e.g., AES
   - Alice and Bob share secret key
   - Encryption is a bit mangling box
2. Public key encryption », e.g., RSA
   - Alice and Bob each have a key in two parts: a public part (widely known), and a private part (only owner knows)
   - Encryption is based on mathematics (e.g., RSA is based on difficulty of factoring)

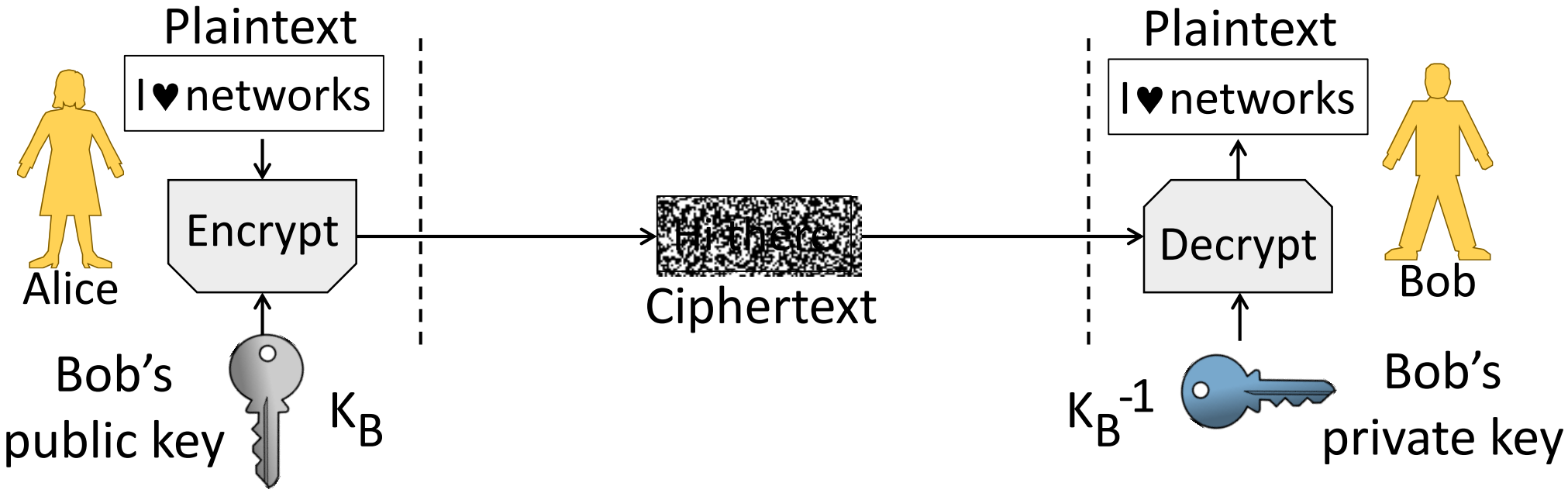
Symmetric (Secret Key) Encryption

- Alice and Bob have the same secret key, K_{AB}
  - Anyone with the secret key can encrypt/decrypt



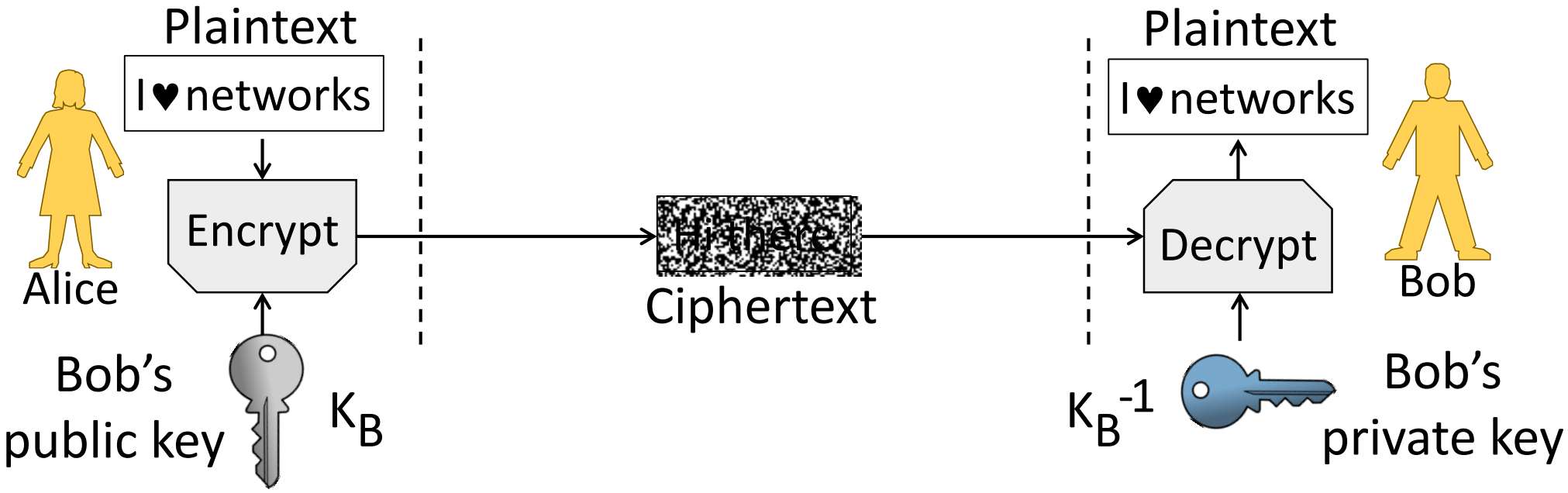
Public Key (Asymmetric) Encryption

- Alice and Bob have public/private key pairs (K_B / K_B^{-1})
  - Public keys are well-known, private keys are secret

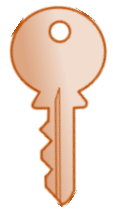


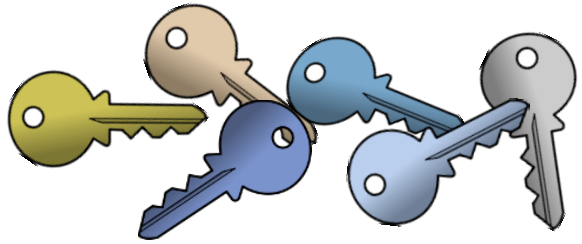
Public Key Encryption (2)

- Alice encrypts w/ Bob's pubkey K_B; anyone can send
- Bob decrypts w/ his private key K_B^{-1}; only he can



Key Distribution





- This is a big problem on a network!
  - Often want to talk to new parties
- Symmetric encryption problematic
  - Have to first set up shared secret
- Public key idea has own difficulties
  - Need trusted directory service
  - We'll look at certificates later

Symmetric vs. Public Key

- Have complementary properties
  - Want the best of both!

Property	Symmetric	Public Key
Key Distribution	Hard – share secret per pair of users	Easier – publish public key per user
Runtime Performance	Fast – good for high data rate	Slow – few, small, messages

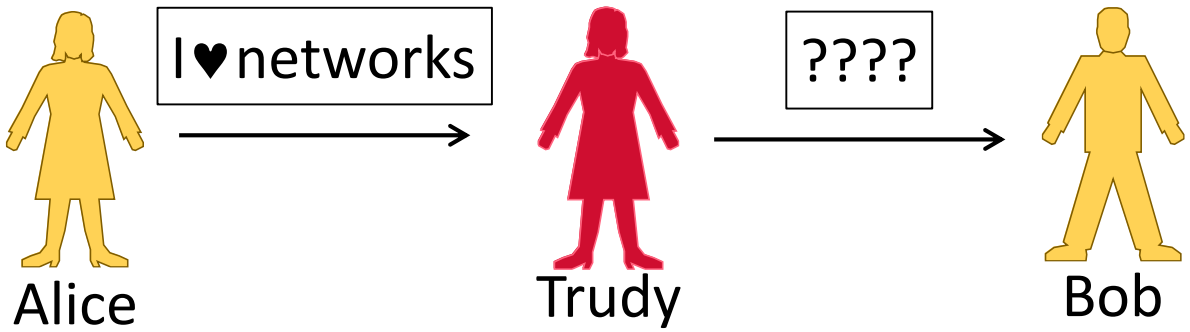
Winning Combination

- Alice uses public key encryption to send Bob a small private message
  - It's a key! (Say 256 bits.)
- Alice/Bob send messages with symmetric encryption
  - Using the key they now share
- The key is called a session key
  - Generated for short-term use

Message Authentication

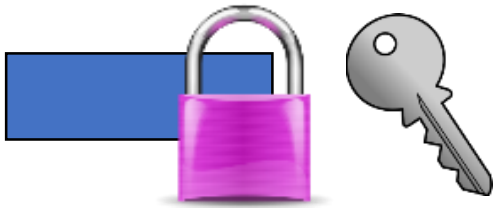
Goal and Threat Model

- Goal is for Bob to verify the message is from Alice and unchanged
  - This is called integrity/authenticity
- Threat is Trudy will tamper with messages
  - Trudy is an active adversary (interferes)



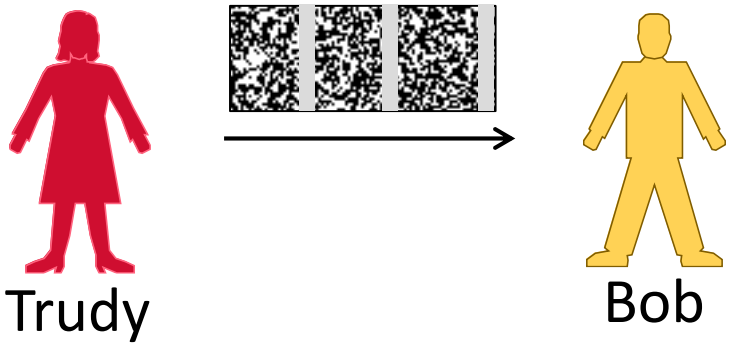
Wait a Minute!

- We're already encrypting messages to provide confidentiality
- Why isn't this enough?



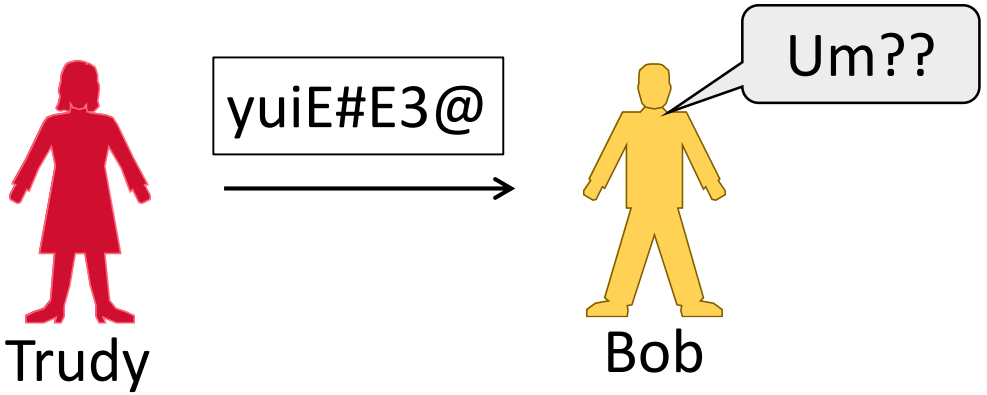
Encryption Issues

- What will happen if Trudy flips some of Alice's message bits?
  - Bob will decrypt it, and …



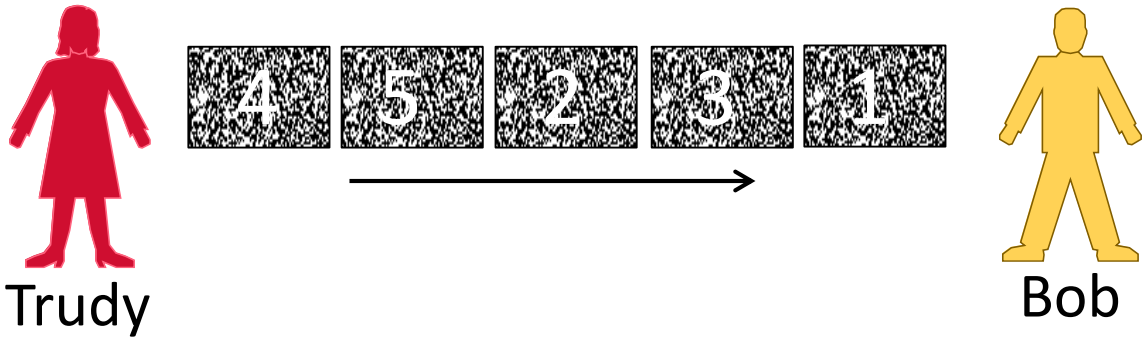
Encryption Issues (2)

- What will happen if Trudy flips some of Alice's message bits?
  - Bob will receive an altered message



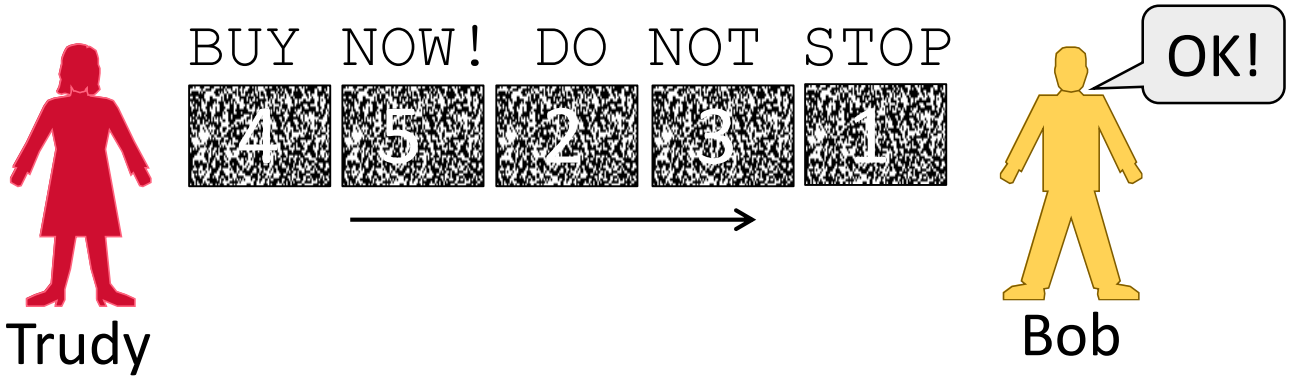
Encryption Issues (3)

- Typically encrypt blocks of data
- What if Trudy reorders message?
  - Bob will decrypt, and …



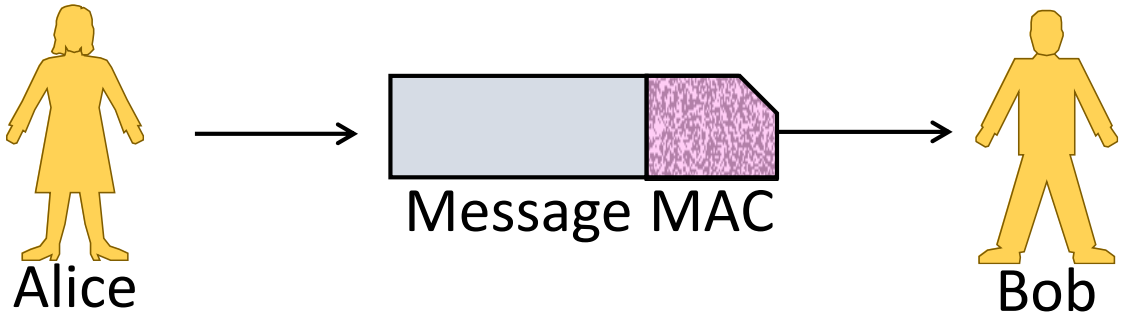
Encryption Issues (4)

- What if Trudy reorders message?
  - Bob will receive altered message



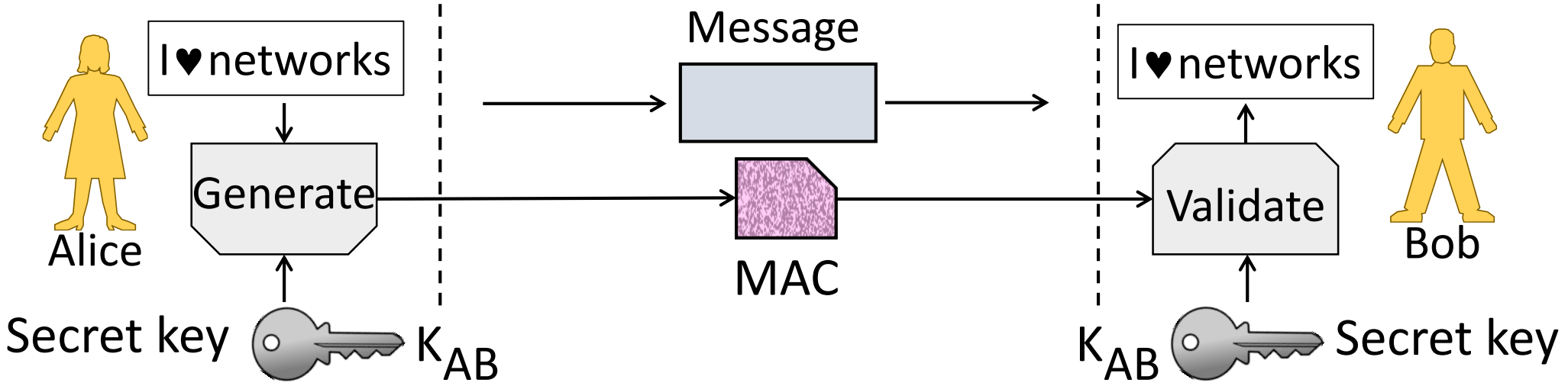
MAC (Message Authentication Code)

- MAC is a small token to validate the integrity/authenticity of a message
  - Conceptually ECCs again
  - Send the MAC along with message
  - Validate MAC, process the message
  - Example: HMAC scheme



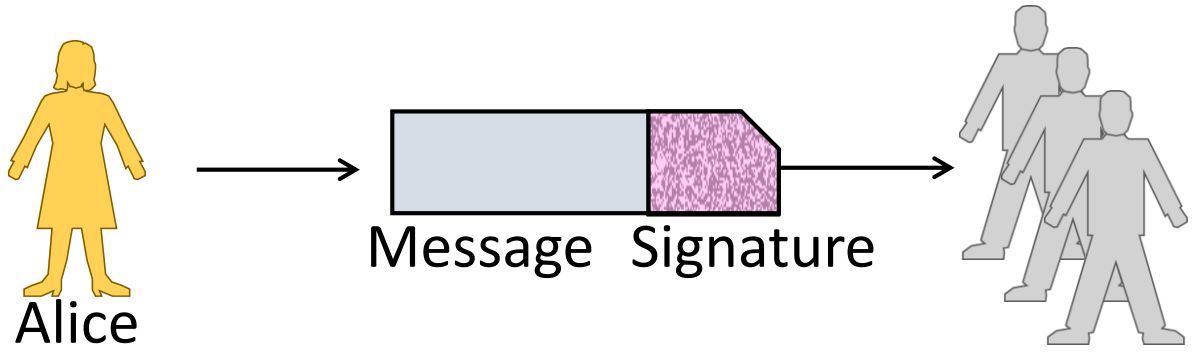
MAC (2)

- Sorta symmetric encryption operation – key shared
  - Lets Bob validate unaltered message came from Alice
  - Doesn't let Bob convince Charlie that Alice sent the message



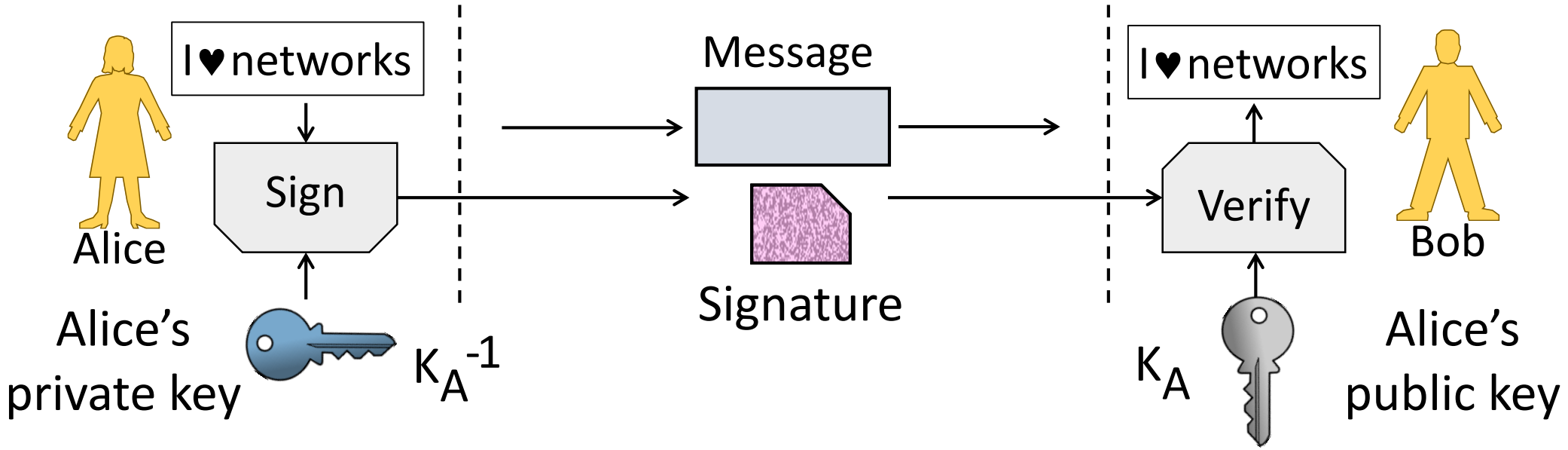
Digital Signature

- Signature validates the integrity/authenticity of message
  - Send it along with the message
  - Lets all parties validate
  - Example: RSA signatures



Digital Signature (2)

- Kind of public key operation – pub/priv key parts
  - Alice signs w/ private key, K_A^{-1}, Bob verifies w/ public key, K_A
  - Does let Bob convince Charlie that Alice sent the message

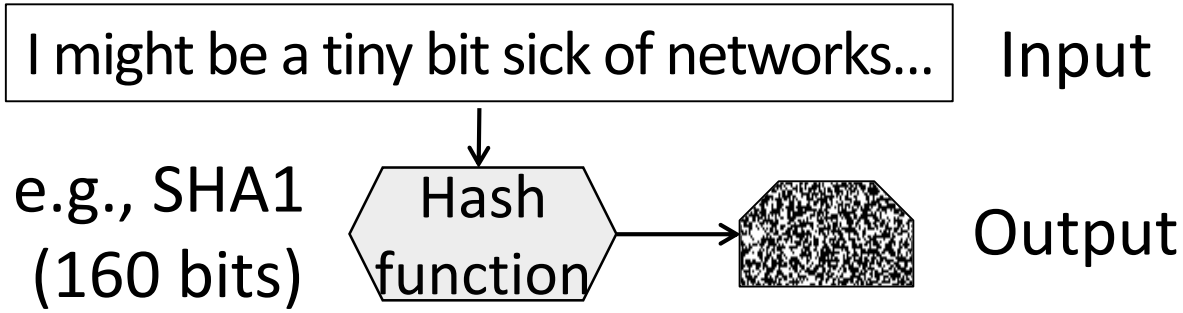


Speeding up Signatures

- Same tension as for confidentiality:
  - Public key has keying advantages
  - But it has slow performance!
- Use a technique to speed it up
  - Message digest stands for message
  - Sign the digest instead of full message

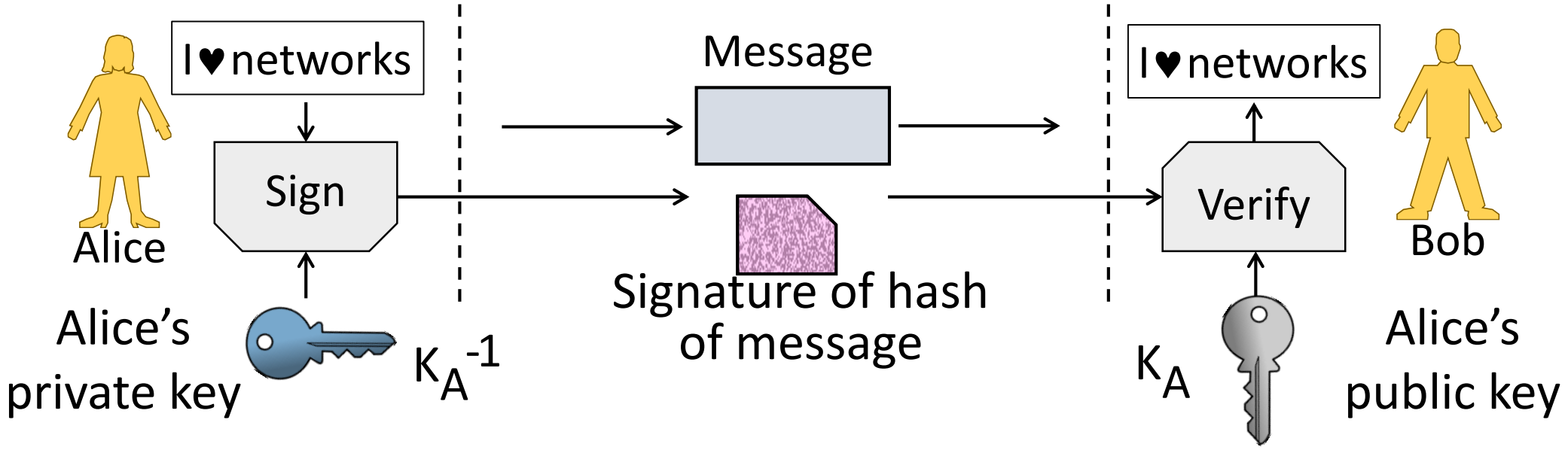
Message Digest or Cryptographic Hash

- Digest/Hash is a secure checksum
  - Deterministically mangles bits to pseudo-random output (like CRC)
  - Can't find messages with same hash
  - Acts as a fixed-length descriptor of message – very useful!



Speeding up Signatures (2)

- Conceptually similar except sign the hash of message
  - Hash is fast to compute, so it speeds up overall operation
  - Hash stands for msg as can't find another w/ same hash

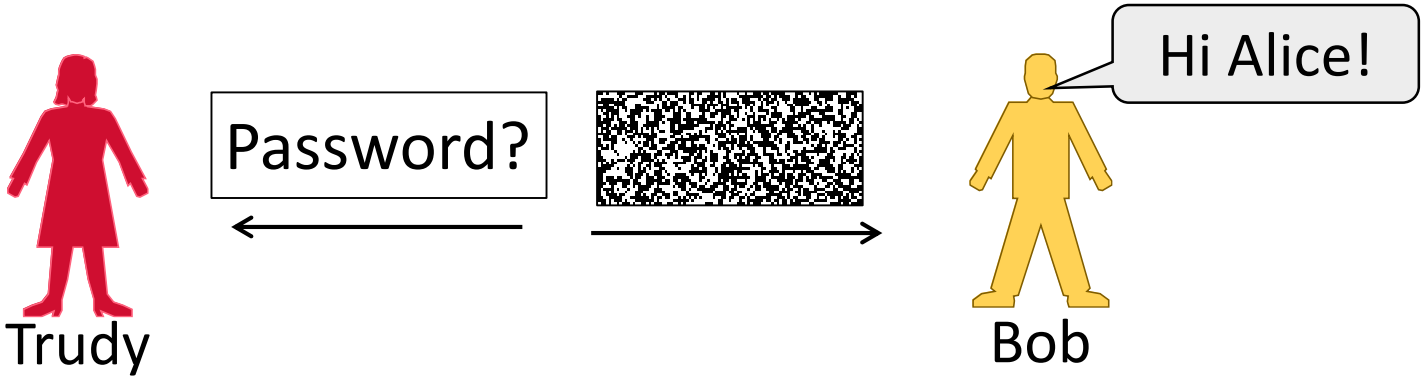


Preventing Replays

- We normally want more than confidentiality, integrity, and authenticity for secure messages!
  - Want to be sure message is fresh
- Need to distinguish message from replays
  - Repeat of older message
  - Acting on it again may cause trouble

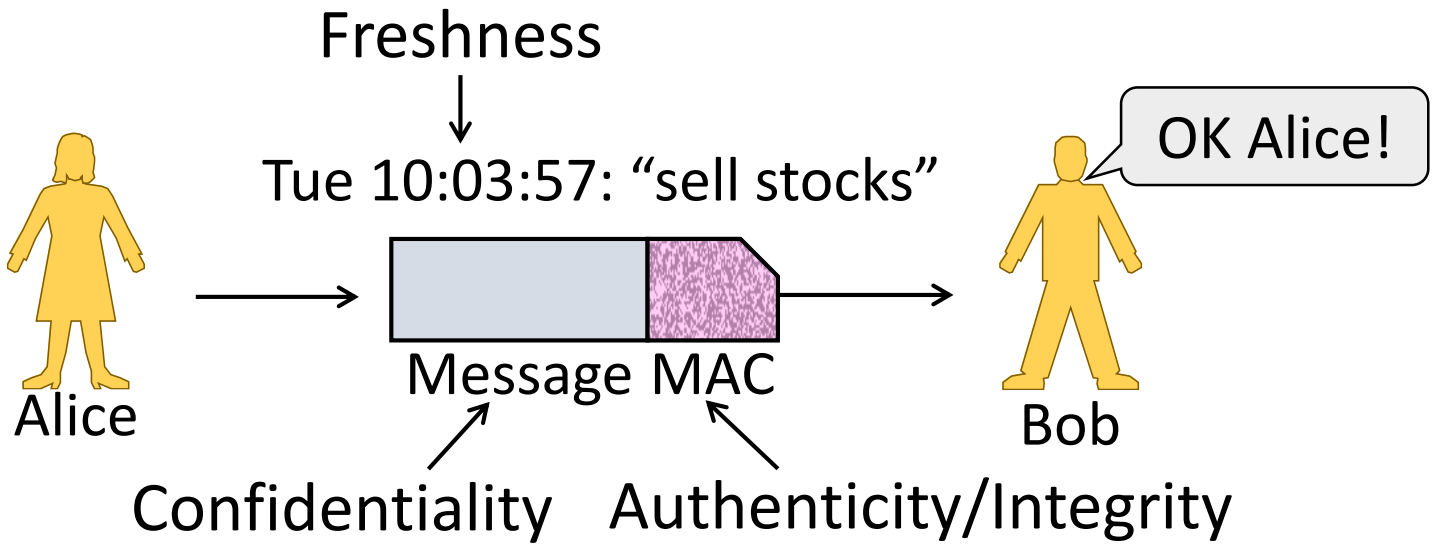
Preventing Replays (2)

- Replay attack:
  - Trudy records Alice's messages to Bob
  - Trudy later replays them (unread) to Bob
    - She pretends to be Alice



Preventing Replays (3)

- To prevent replays, include a proof of freshness in the messages
  - Use a timestamp, or nonce



Takeaway

- Cryptographic designs can give us integrity, authenticity and freshness as well as confidentiality.
- Real protocol designs combine the properties in different ways
  - We'll see some examples
  - Note many pitfalls in how to combine, as well as in the primitives themselves