cryptography

cs642

adam everspaugh
ace@cs.wisc.edu

computer security

today

* Cryptography intro

* Crypto primitives
  / Symmetric and asymmetric crypto
  / MACs
  / Digital signatures
  / Key exchange

* Provable security

crypto

* Cryptography: "hidden writing"
* Study and practice of building security protocols that resist adversarial behavior
* Blend of mathematics, engineering, and computer science

US diplomatic cables

Internet

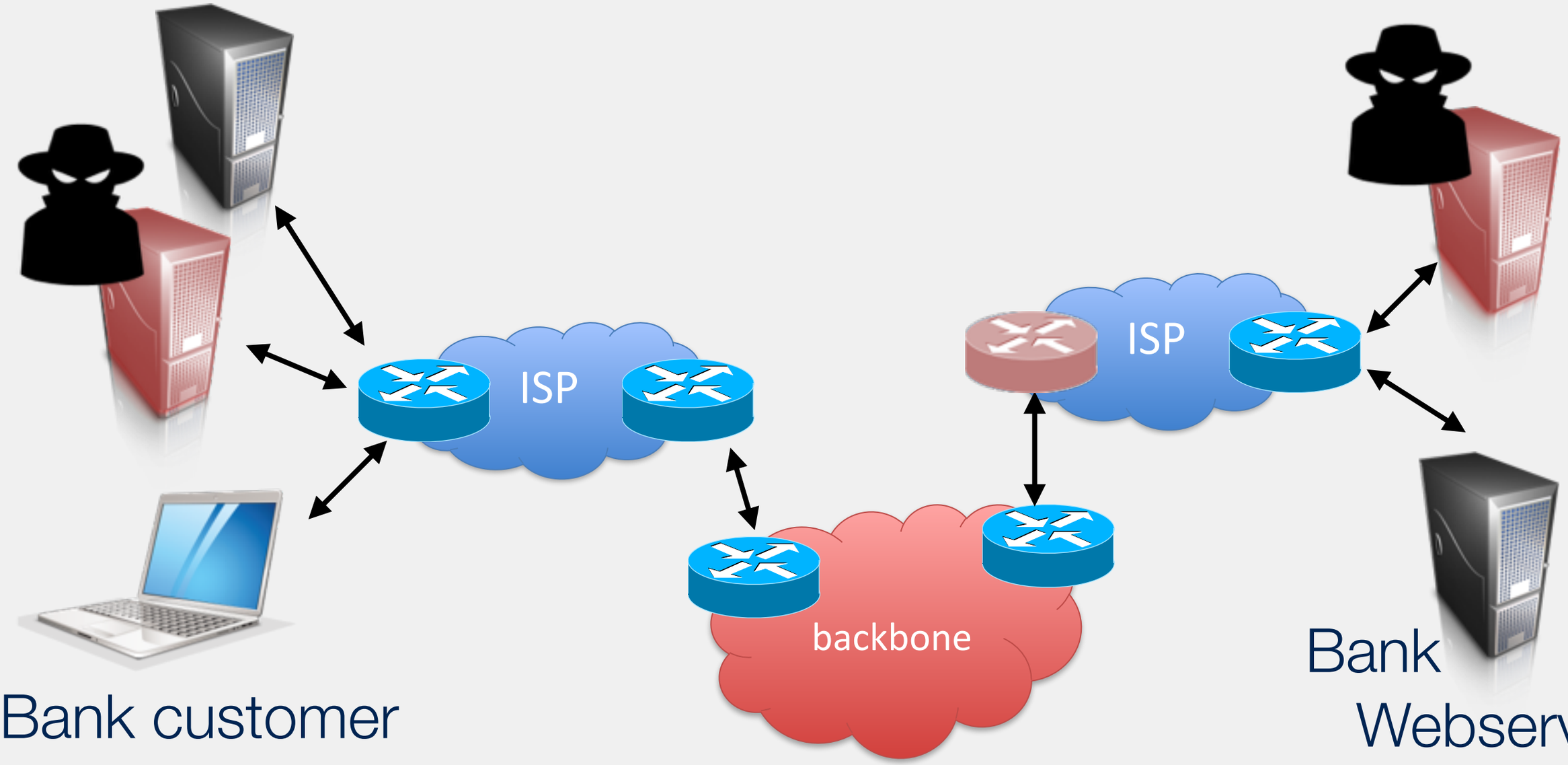
Doesn't want to reveal contents early

Wants info stored in a way that can be quickly revealed at the right time

Modern cryptography enables this:

- Encrypt the file
- Store key in a safe place

example 1



Customer and bank want to communicate securely:

- Confidentiality (messages are private)
- Integrity (messages aren't modified)
- Authenticity (is this the bank? is this the customer?)
- Sometimes: anonymity (hide identities)

example 2

Encrypted hard drives

Corporate intellectual property
Customer financial records
Personal notes

Encrypt hard drives or individual files

- Confidentiality
- Even if attacker has physical access to device

Bitlocker, TrueCrypt, OSX, iOS, Seagate

example 3

* Cryptography is a powerful tool

* Helps provide:
  / Confidentiality
  / Integrity
  / Authenticity
  / even more

* Limitations
  / Not the (complete) solution to every security problem
  / Must be designed securely
  / Must be implemented properly
  / Must be used properly

cryptography

> A cryptosystem should be secure even if **everything** about the system, except the secret key, is **public knowledge**.
>
> —Auguste Kerckhoffs, 19th century

kerckhoff's principle

flavors

* Symmetric cryptography
  / All parties have access to a **shared random string** K, called the key

* Asymmetric cryptography
  / Each party creates a pair of keys: a public key pk and a secret key sk

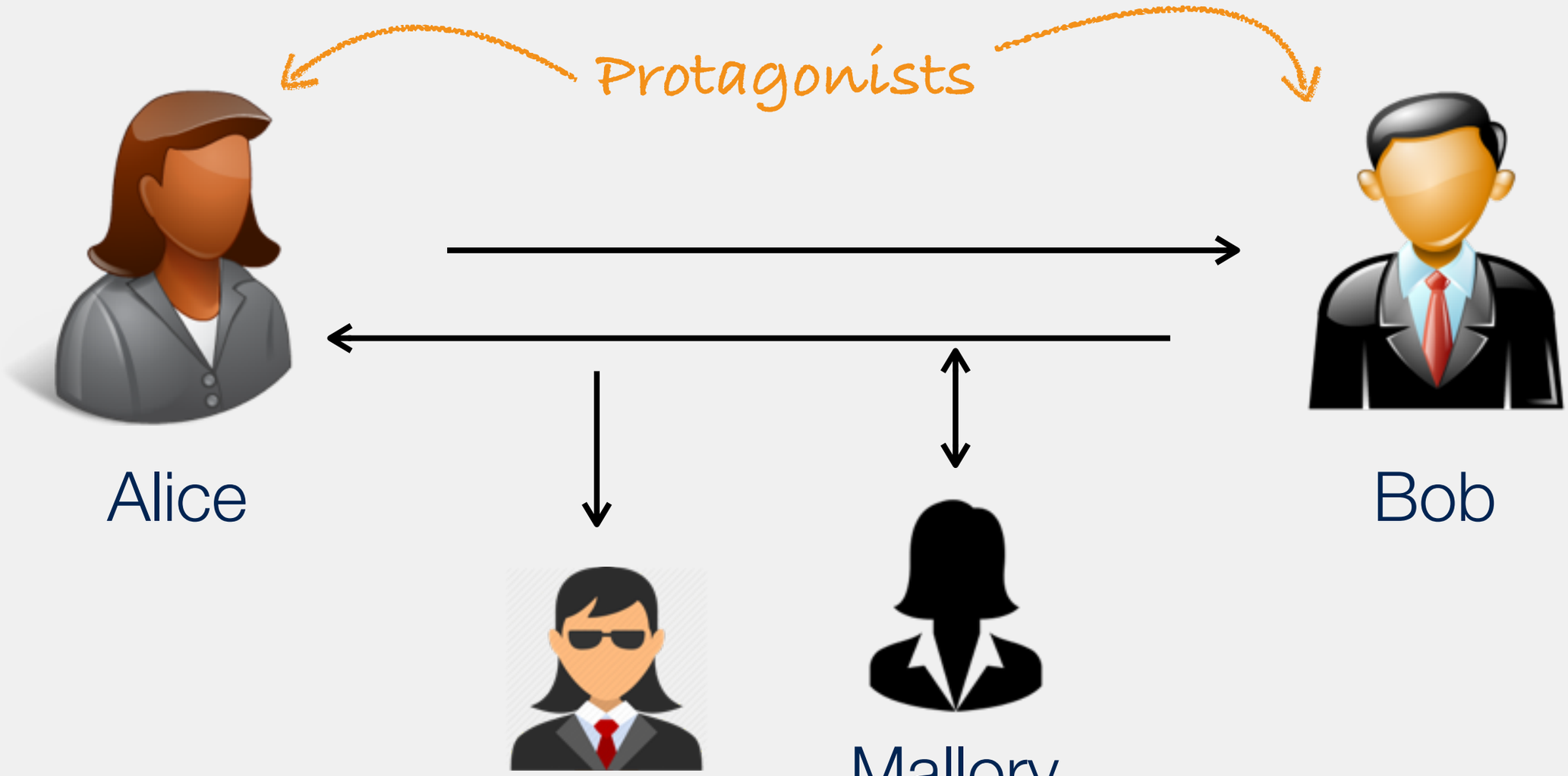
primitives

* Encryption
  / confidentiality
  / symmetric + asymmetric versions

* Message authentication codes
  / integrity, authentication
  / symmetric

* Digital signatures
  / integrity, authentication
  / asymmetric

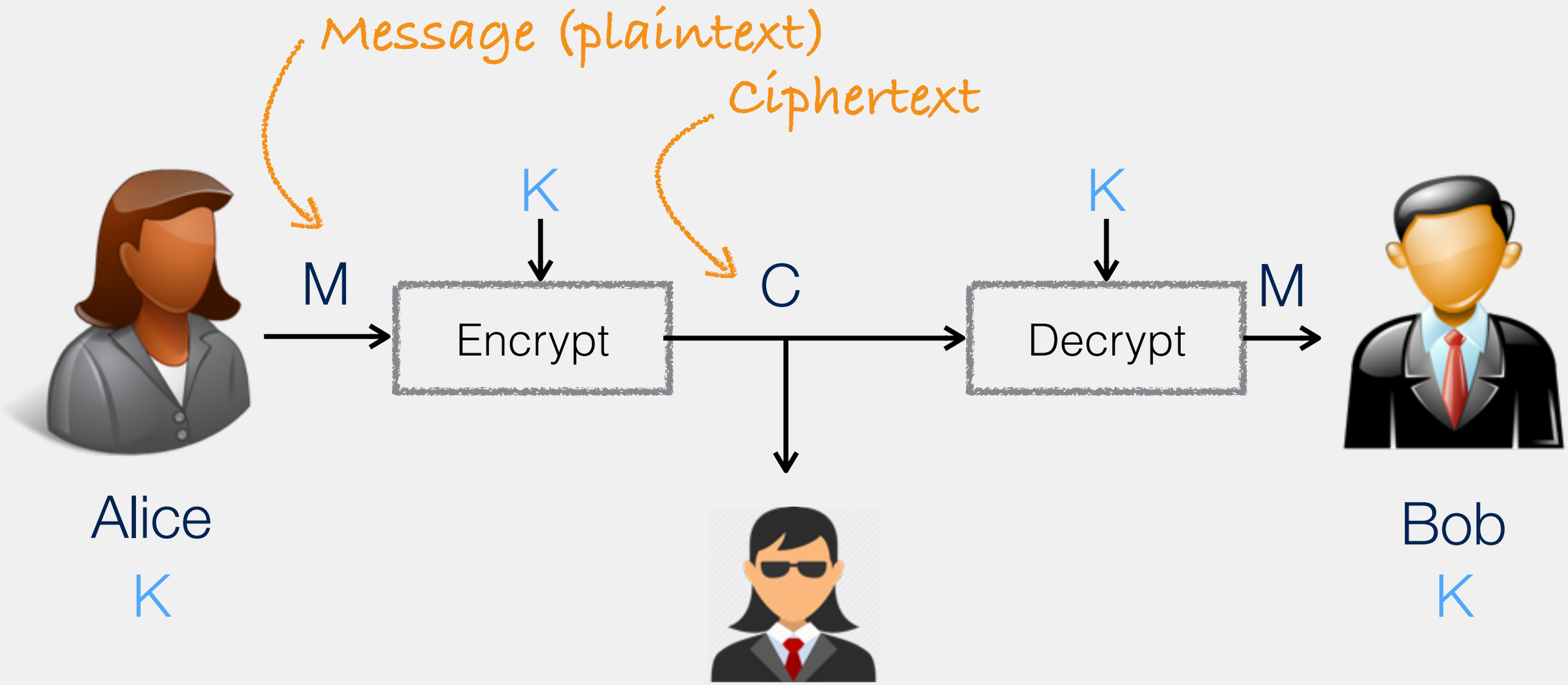
* Key exchange



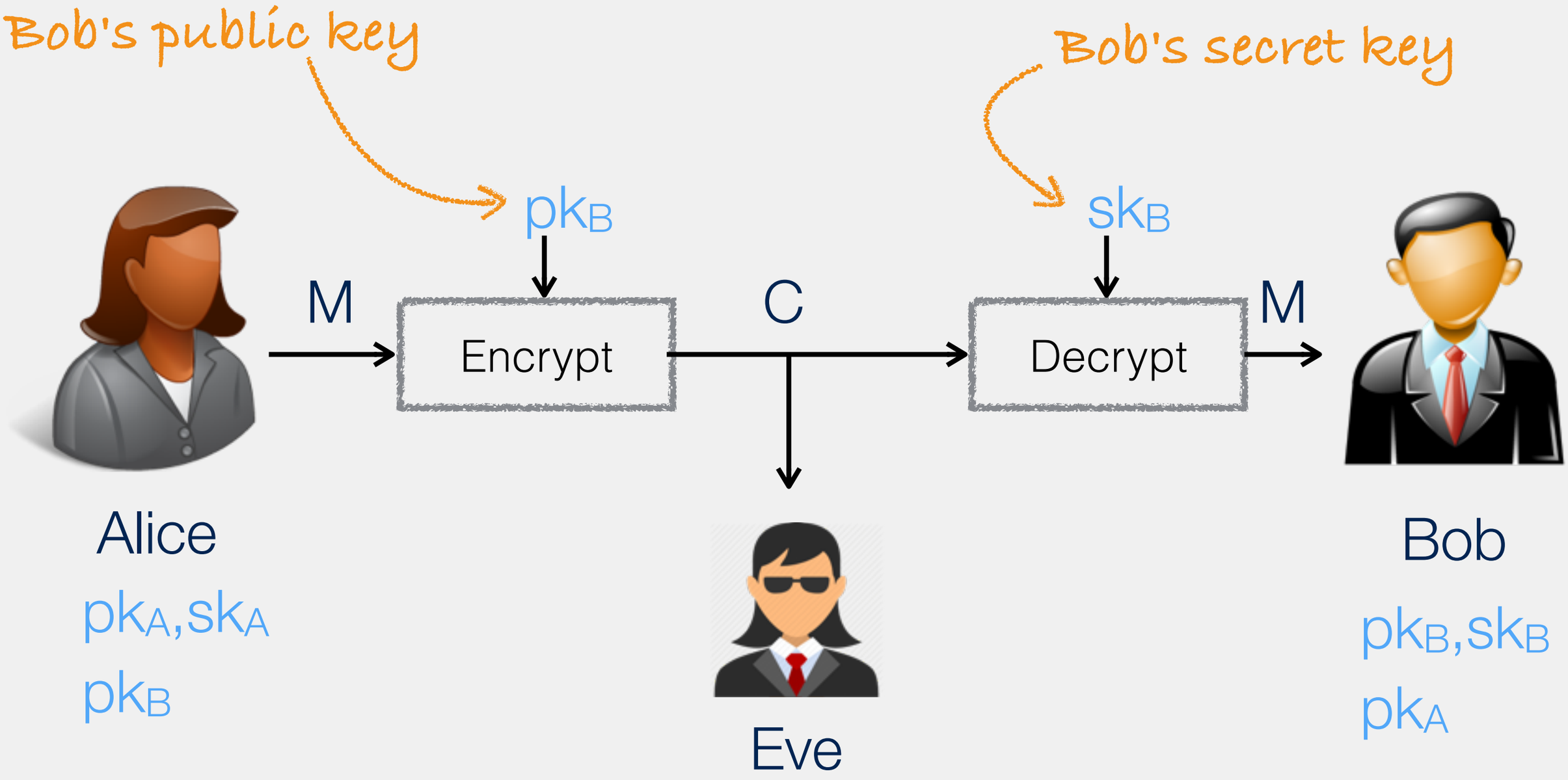
Eavesdropper
(passive adversary)

Malicious Actor
(active adversary)

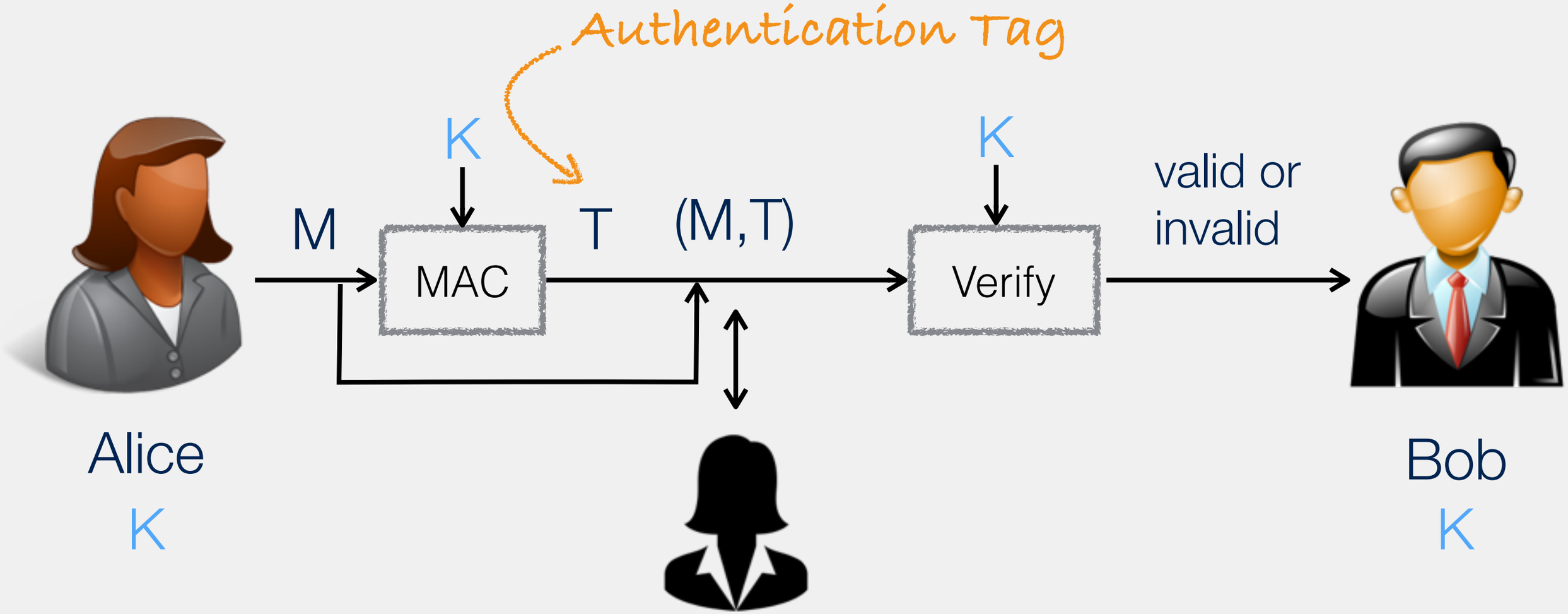
conventions



symmetric encryption

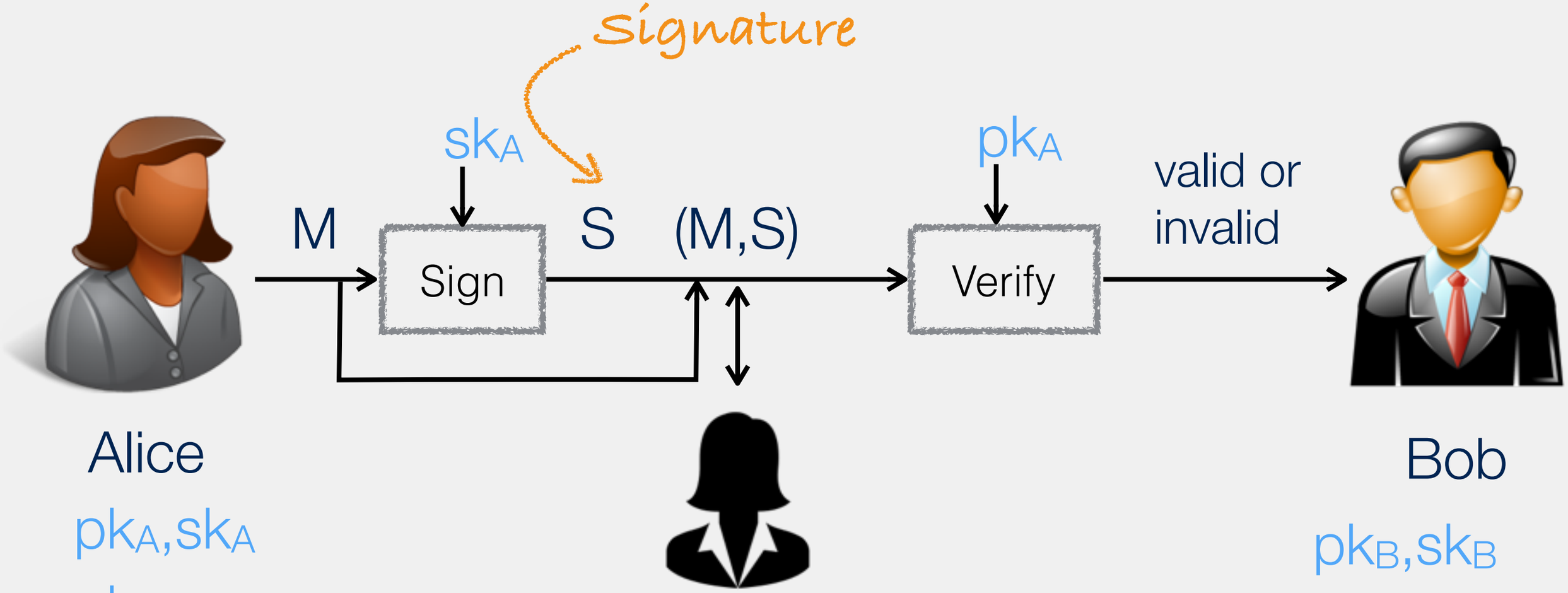


asymmetric encryption



Message Authentication Code (MAC)
message integrity & authenticity / symmetric

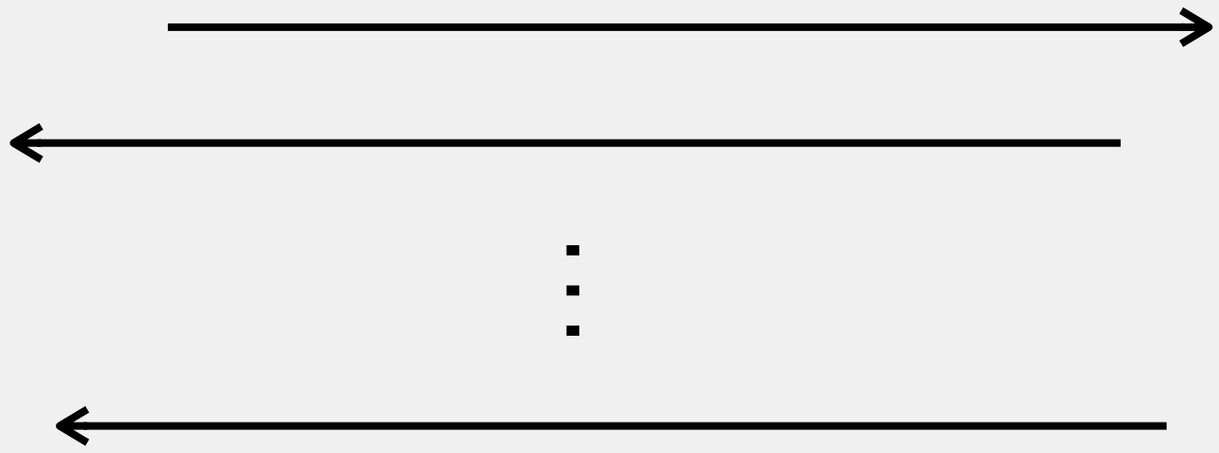
mac



message integrity & authenticity / asymmetric

digital signatures

Eve



Alice

Bob

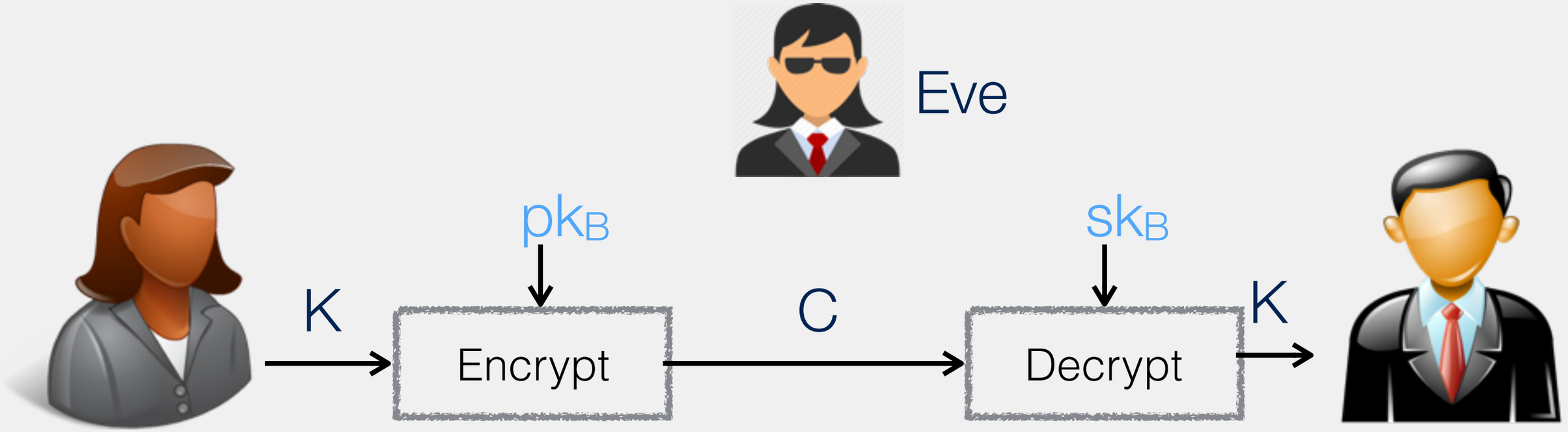
K

think-*pair*-share

K

Alice and Bob exchange messages in the presence of an eavesdropper, and (magically) both generate an identical secret (symmetric) key that Eve cannot know

key exchange



pk_B pk_A

K := rand()

Two main techniques for key exchange

1. Public key transport (shown here)
2. Diffie-Hellman key agreement

key transport

transport layer security (tls) protects http connections (https)

bank.com

ClientHello, MaxVer, Nc, Ciphers/CompMethods

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

CERT = (pk of bank, signature over it)

C

ChangeCipherSpec,
{ Finished, PRF(MS, “Client finished” || H(transcript)) }

ChangeCipherSpec,
{ Finished, PRF(MS, “Server finished” || H(transcript’)) }

Ciphersuite negatiation

Server authentication

MS <- PRF(PMS, “master secret” || Nc || Ns)

K1,K2 <- PRF(MS, “key expansion” || Ns || Nc)

Key exchange completed

tls

SSL ver 2.0 designed by **Hickman** at Netscape

SSL ver 2

1994

Wagner, Goldberg break SSL ver 2

1995

Freier, Karlton, Kocher design SSL ver 3.0

SSL ver 3

Bleichenbacher breaks RSA PKCS #1 encryption, used in SSL ver 3

1998

1999

TLS ver 1 released as IETF standard, based on SSL 3, **many cryptographers** involved

TLS ver 1.0

2001

Vaudenay, Klima et al. padding attacks

2002

Rogaway IV re-use insecurity

2003

Brumley, Boneh remote timing attacks

2006

TLS ver 1.1 released as standard

TLS ver 1.1

⋮

ancient history

iteration

* TLS was built via "design-break-redesign-break" iteration

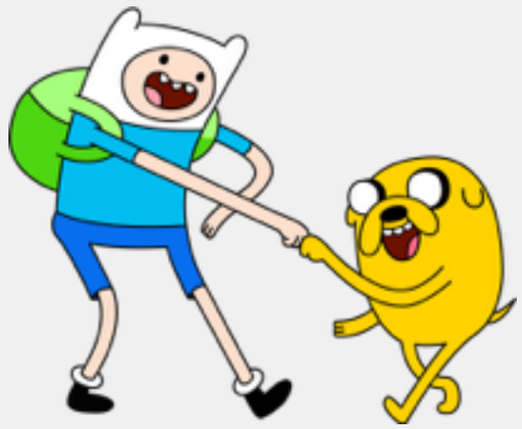
* Some amount iteration is fundamental

* Designing secure protocols is **really hard**
  / the problems are rarely in the primitives

* Many other tools have similar stories:
  / SSH, IPsec, kerberos, WEP + WPA (WiFi), GSM (cell phone)

provable security

* Provable security supplements "design-break-redesign-break" iteration with a mathematical approach



1. Design a cryptographic scheme
2. Provide a proof of it's security

[Shannon, 1946]

Formal definitions
- Scheme semantics and assumption
- Security

Security Proofs
- Scheme cannot be broken if assumption hold

enigma

* Put yourself in Shannon's place in 1946

* Enigma is state of the art cryptography developed by the Germans

* Broken by the Allies

otp

* Shannon's one-time pad
* Fix message length L
* Kg: output random bit string K of length L

$E(K,M) = M \oplus K = C$ $D(K,C) = C \oplus K = M$

security notion

Dfn. A symmetric encryption is *perfectly secure* if for all messages M,M' and ciphertexts C

$$\Pr[\, E(K,M)=C \,] = \Pr[\, E(K,M') = C \,]$$

where probabilities are over choice of K.

* Shannon's "perfect security" notion
* Each message is equally likely to map to a given ciphertext
* Also: seeing a ciphertext leaks nothing about what message was encrypted

otp proof

Dfn. A symmetric encryption is *perfectly secure* if for all messages M,M' and ciphertexts C

$$\Pr[\, E(K,M)=C \,] = \Pr[\, E(K,M') = C \,]$$

where probabilities are over choice of K.

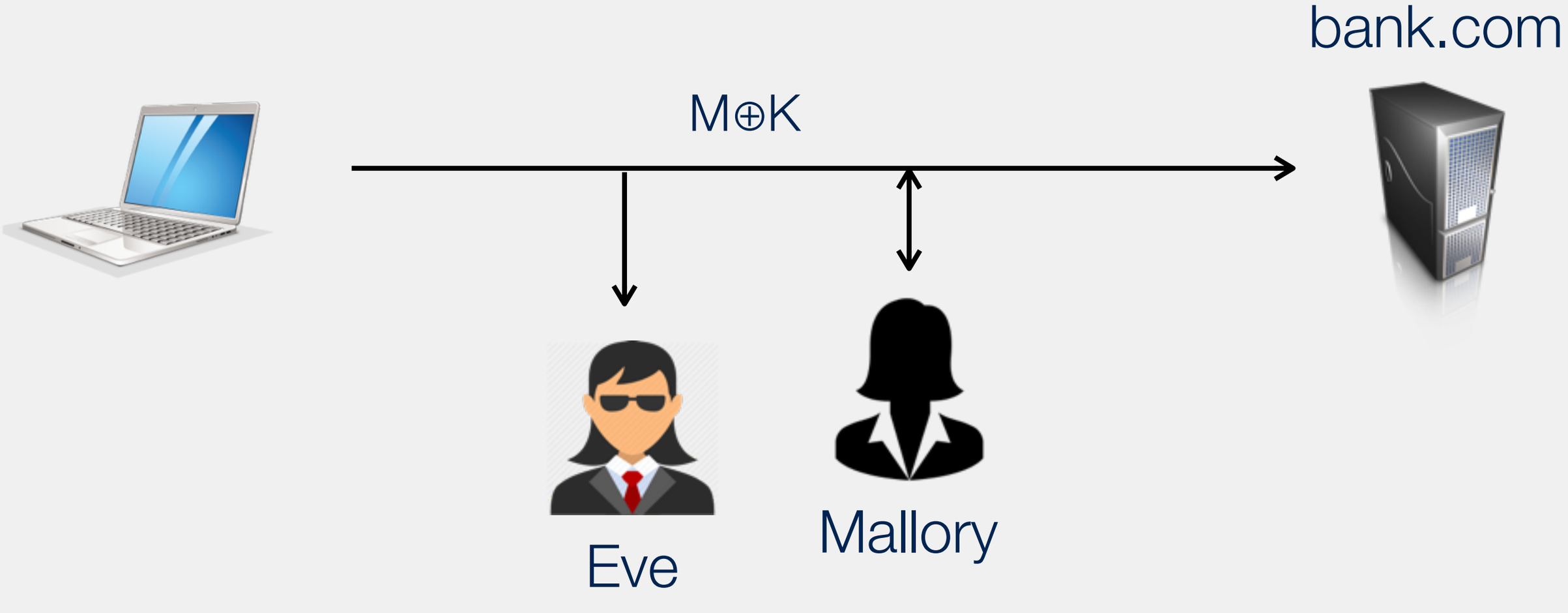
* Thm. OTP is perfectly secure.

* For any C,M of length L:

$$\Pr[\, E(K,M)=C \,] = 1/2^L$$

$$\Pr[\, E(K,M')=C \,] = 1/2^L$$

$$\Pr[\, E(K,M)=C \,] = \Pr[\, E(K,M')]$$



K must be as large as M

Reusing K for M,M' leaks M⊕M'

Message length is obvious

Mallory can make undetected modifications

limitations

provable security

* Cryptography as a ***computational science***
* Use computational intractability as basis for confidence

1. Design a cryptographic scheme
2. Provide a proof that no attacker with bounded computational resources can break it

[Goldwasser, Micali, Blum, 1980s]

Formal definitions

- Scheme semantics and assumption
- Security

Security Proofs (reductions)

Breaking scheme

Breaking assumptions

provable security

* Provable security yields
  / well-defined assumptions and security goals
  / designers (and attackers) can focus on assumptions

* As long as assumptions hold, we can be confident in security of a cryptographic scheme

typical assumptions

* Underlying primitives are hard to break
  / Factoring of large composite numbers is intractable
  / RSA permutation is hard to invert
  / Block ciphers (AES,DES) are good pseudorandom permutations (PRPs)
  / Hash functions are collision resistant

* Confidence in primitives is gained by cryptanalysis, public design competitions
  / design-break-redesign-break over the years

* Symmetric vs asymmetric cryptography

* Primitives
  / symmetric/asymmetric encryption
  / message authentication codes
  / digital signatures
  / key exchange

* Provable security

* Shannon's one-time pad
  / security guarantees and limitations

* Exit slips
  / 1 thing you learned
  / 1 thing you didn't understand

recap