Cryptography Intro Part 2

CS642: Computer Security

Spring 2019

Cryptography

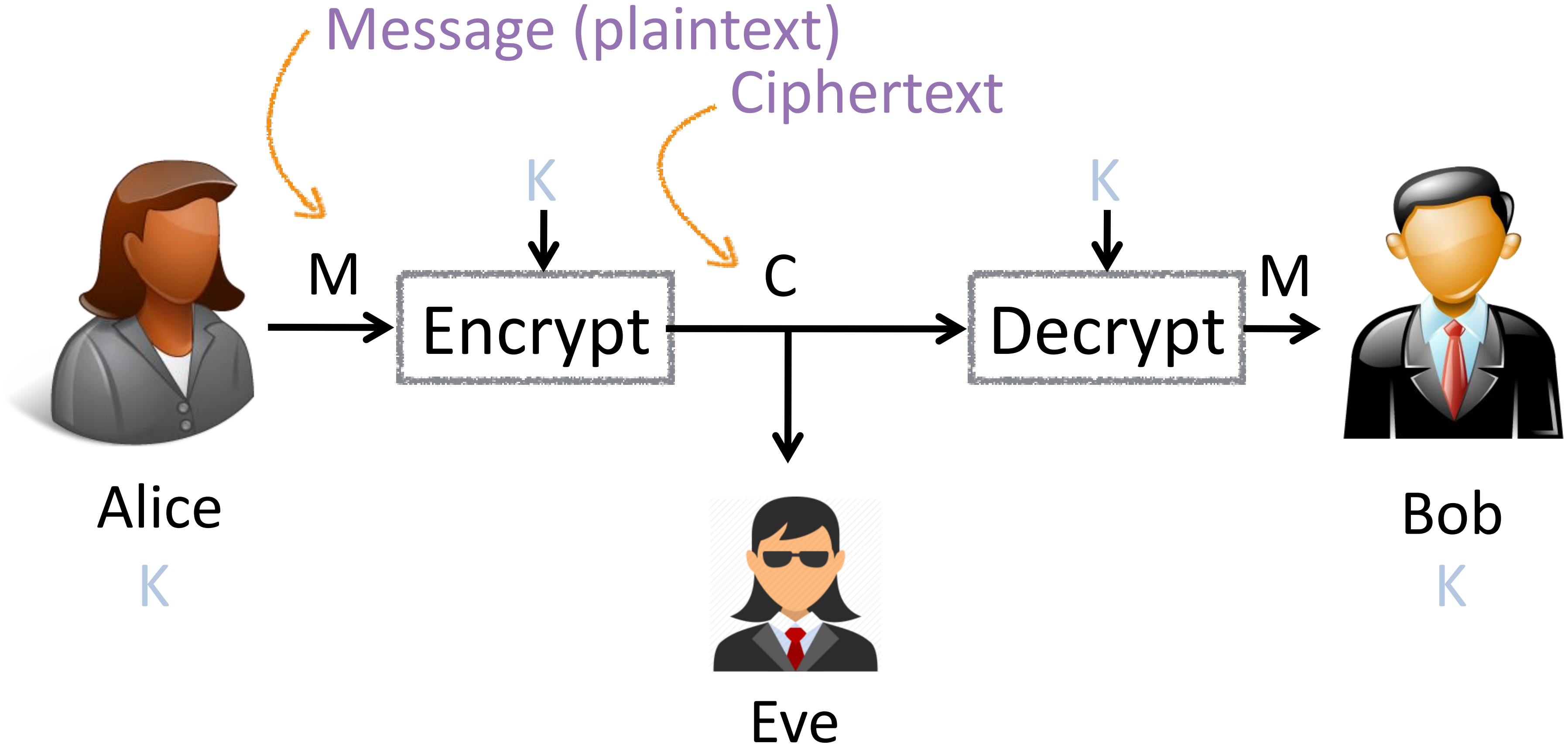
Basic goals and setting

TLS (HTTPS)

Provable security

One time pad

Block ciphers



Message (plaintext)

Ciphertext

M

K

Encrypt

C

K

Decrypt

M

Alice

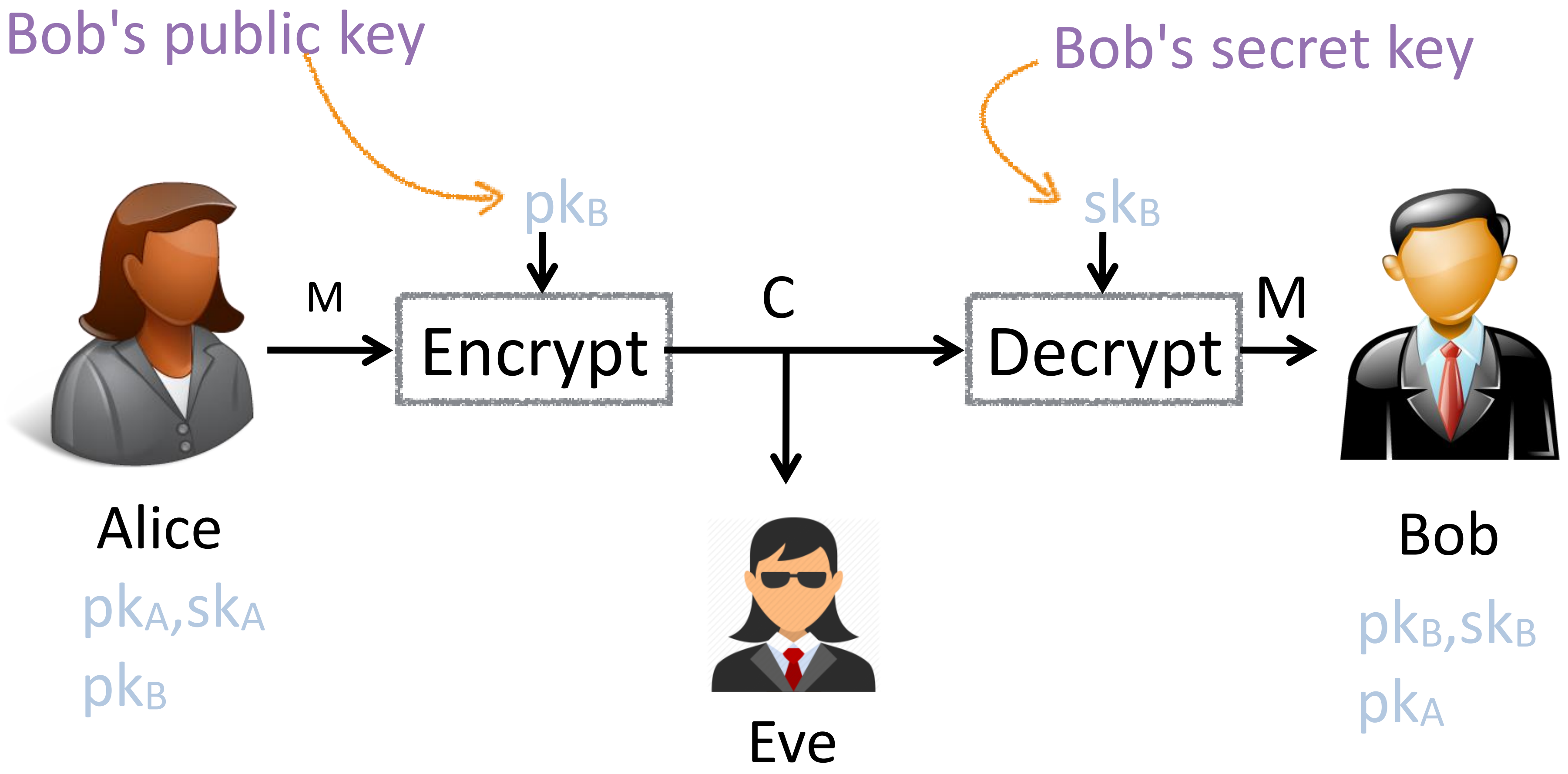
K

Eve

Bob

K

symmetric encryption



Bob's public key

Bob's secret key

pk_B

sk_B

M

Encrypt

C

Decrypt

M

Alice

pk_A, sk_A

pk_B

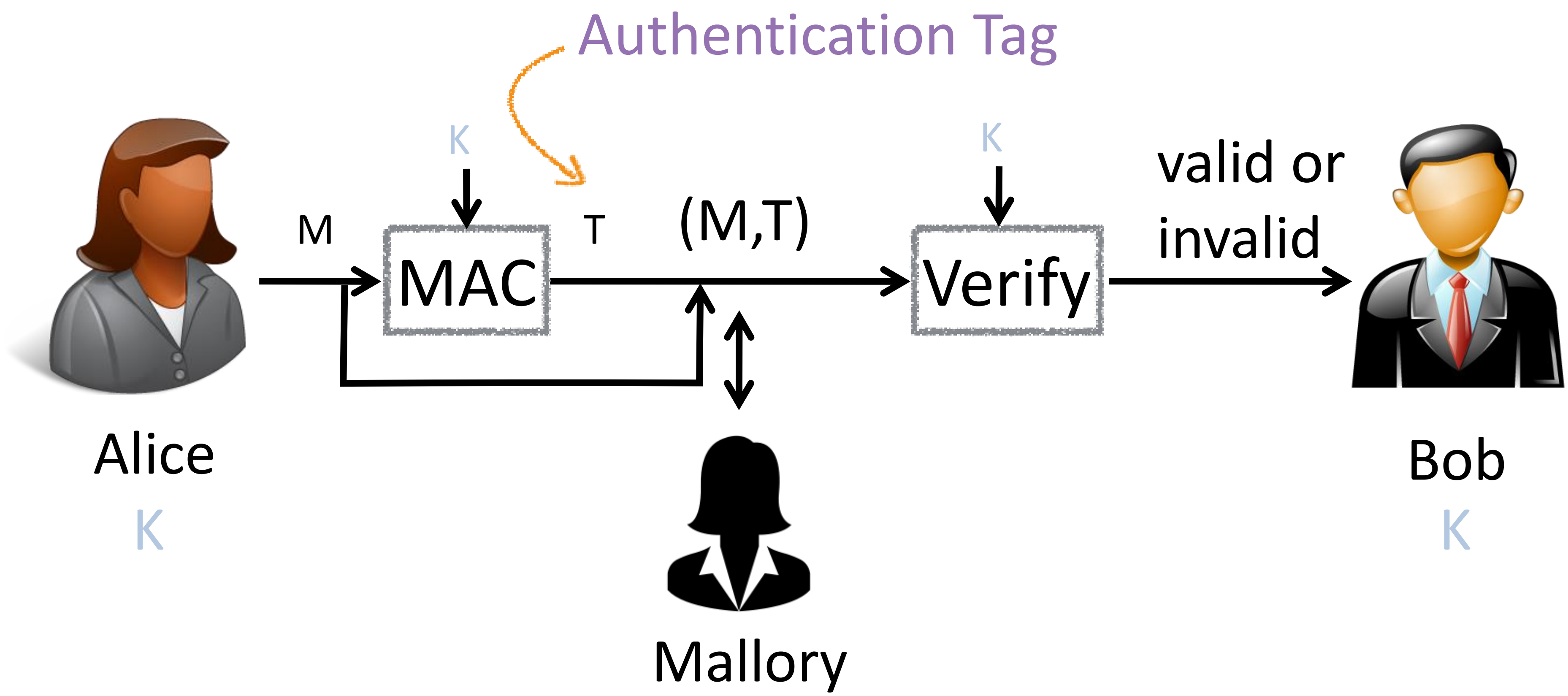
Eve

Bob

pk_B, sk_B

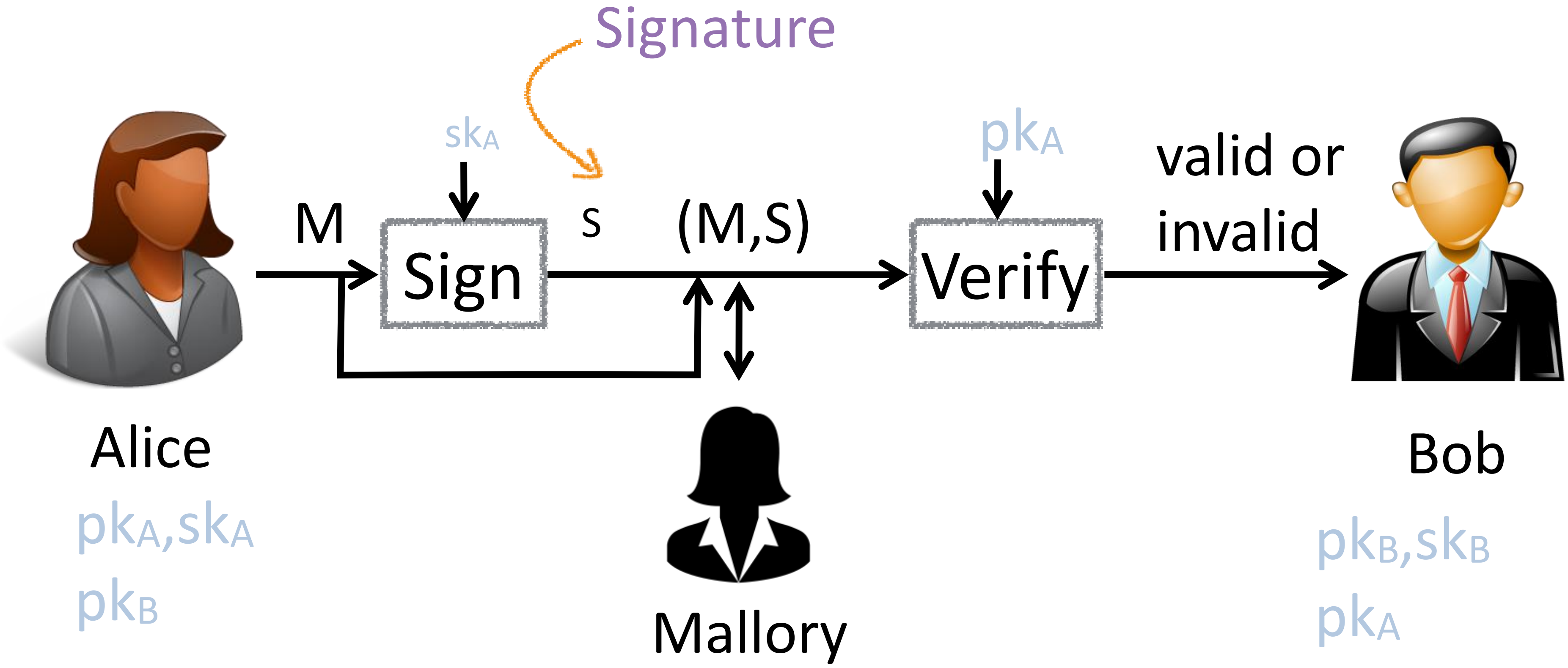
pk_A

asymmetric encryption



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

mac



Signature

sk_A

M

Sign

s

(M,S)

pk_A

Verify

valid or invalid

Alice

pk_A,sk_A

pk_B

Mallory

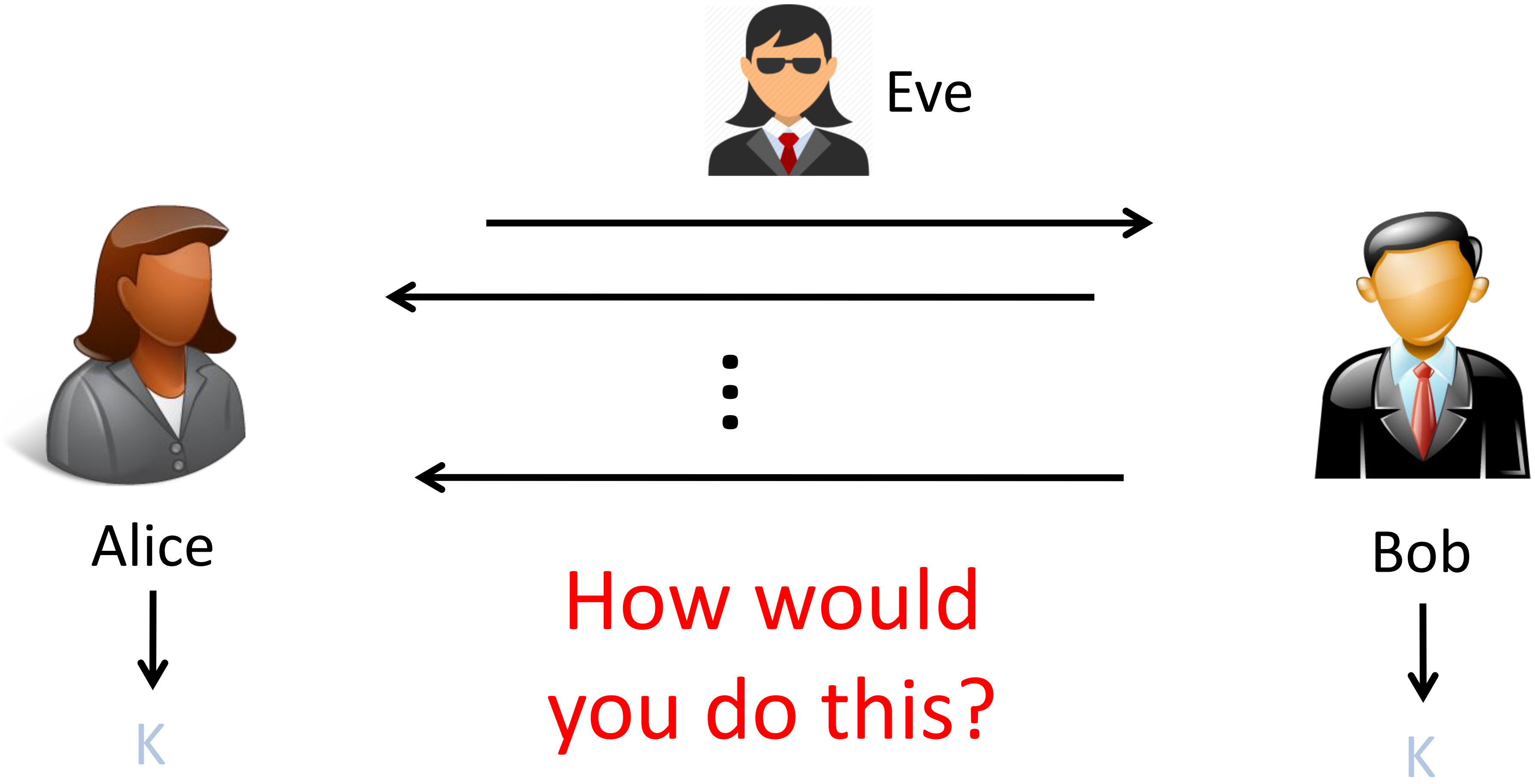
Bob

pk_B,sk_B

pk_A

message integrity & authenticity / asymmetric

digital signatures



Eve

Alice

Bob

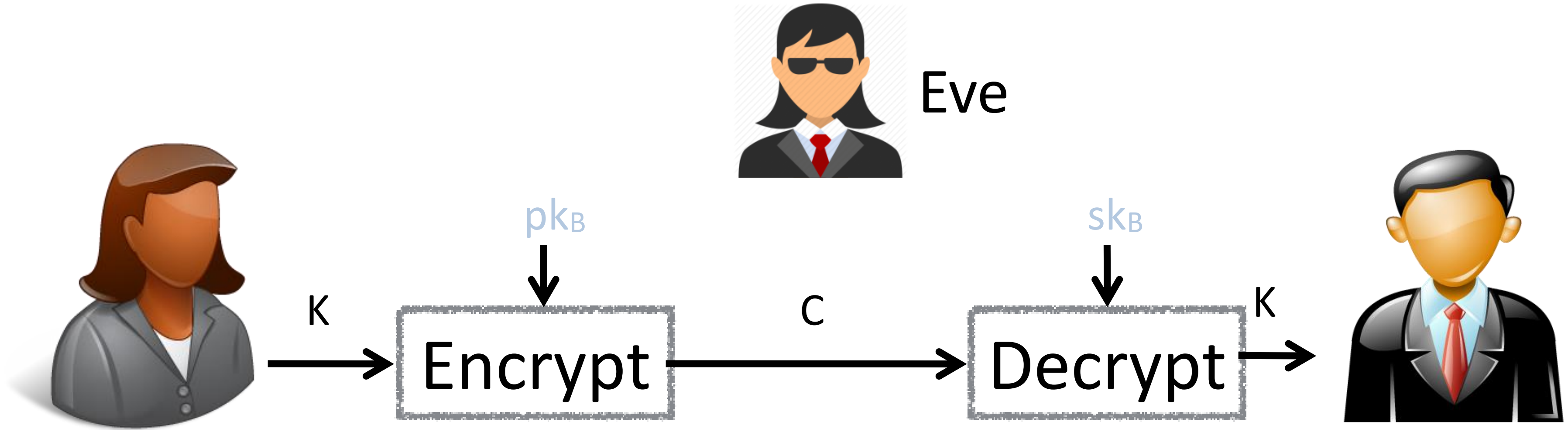
K

K

How would you do this?

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



Eve

K

pk_B

Encrypt

C

sk_B

Decrypt

K

Alice

pk_A, sk_A

pk_B

K := rand()

Bob

pk_B, sk_B

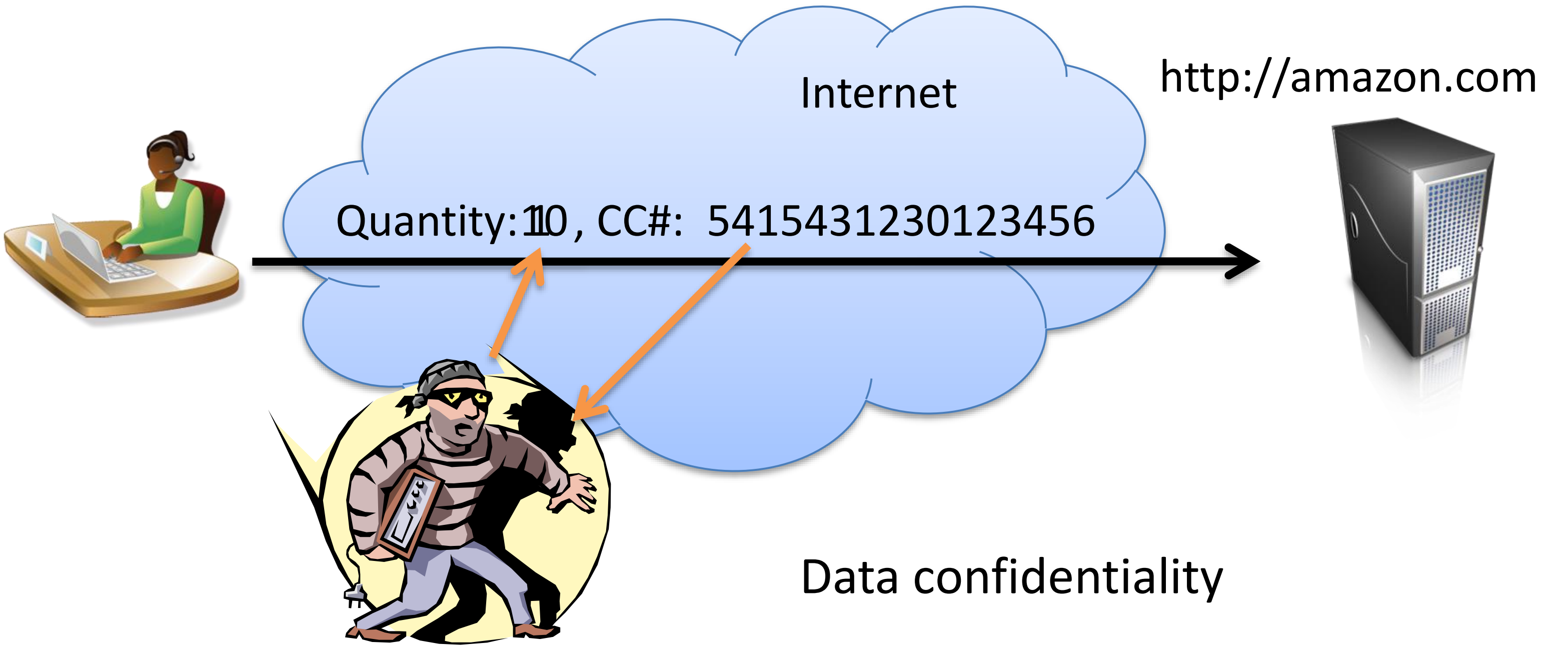
pk_A

Two main techniques for key exchange

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

key transport

An example: Online shopping

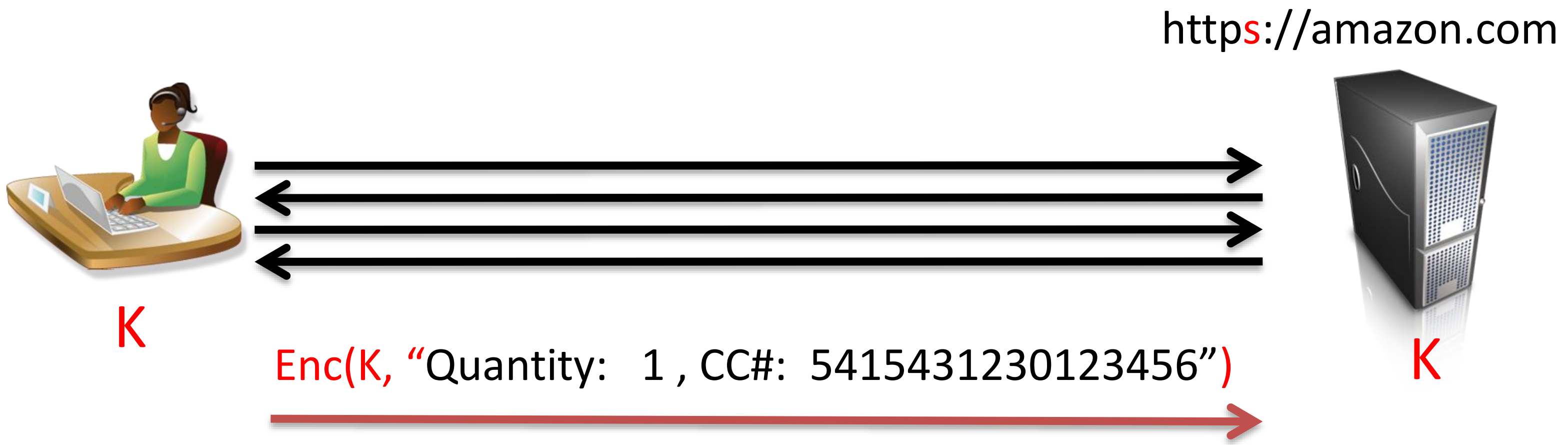


Data confidentiality

Data integrity

We need secure channels for transmitting data

An example: On-line shopping with TLS



https://amazon.com

K

K

Enc(K, “Quantity: 1 , CC#: 5415431230123456”)

Step 1:
Key exchange
protocol to
share secret K

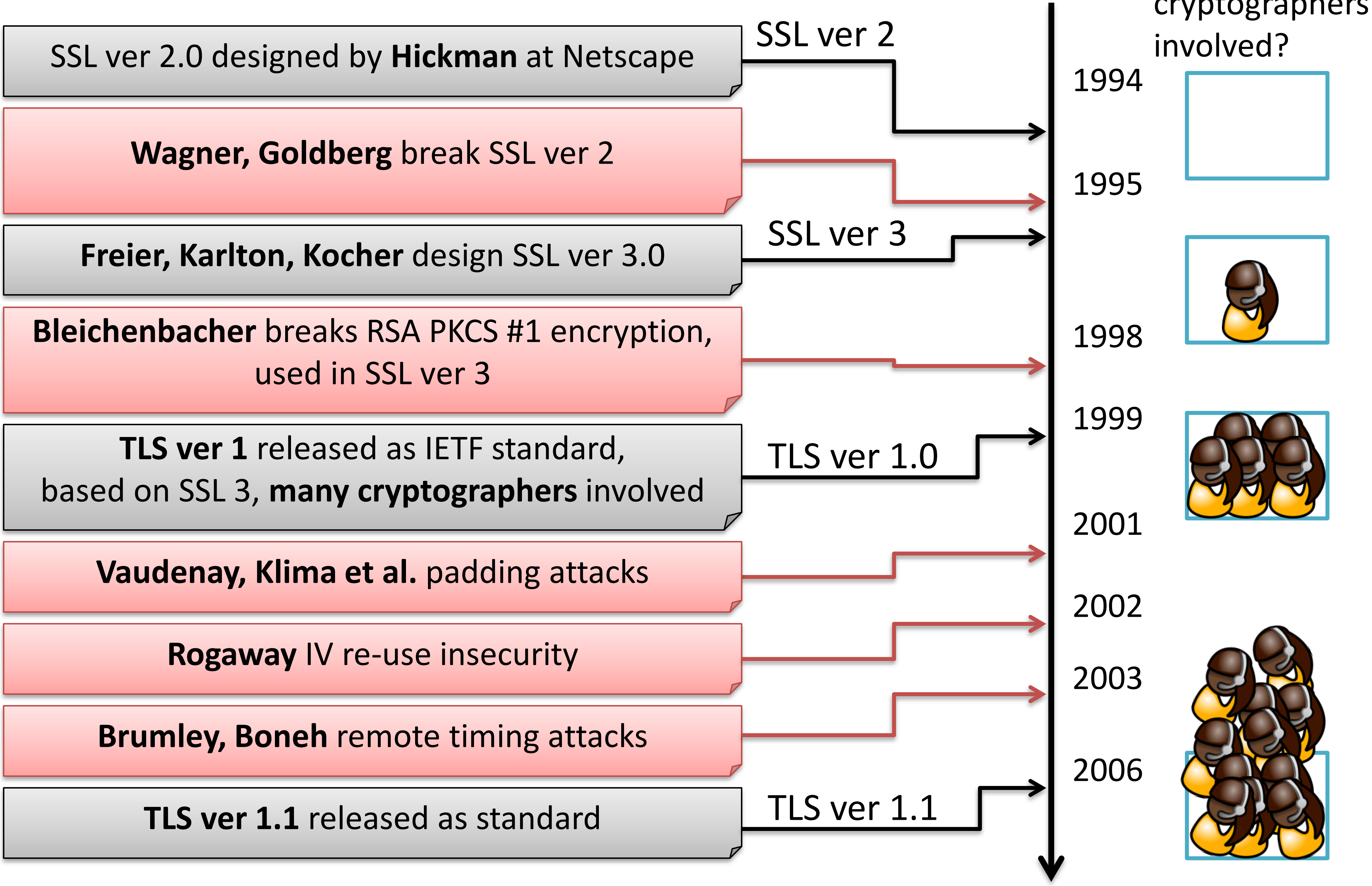
Step 2:
Send data via
secure
channel

TLS uses many cryptographic primitives:

- **key exchange:** hash functions, digital signatures, public key encryption
- **secure channel:** symmetric encryption, message authentication

Mechanisms to resist replay attacks, man-in-the-middle attacks, truncation attacks, etc…

A short history of TLS up to 2009



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

Wagner, Goldberg break SSL ver 2

Freier, Karlton, Kocher design SSL ver 3.0

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

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

Vaudenay, Klima et al. padding attacks

Rogaway IV re-use insecurity

Brumley, Boneh remote timing attacks

TLS ver 1.1 released as standard

How many cryptographers involved?

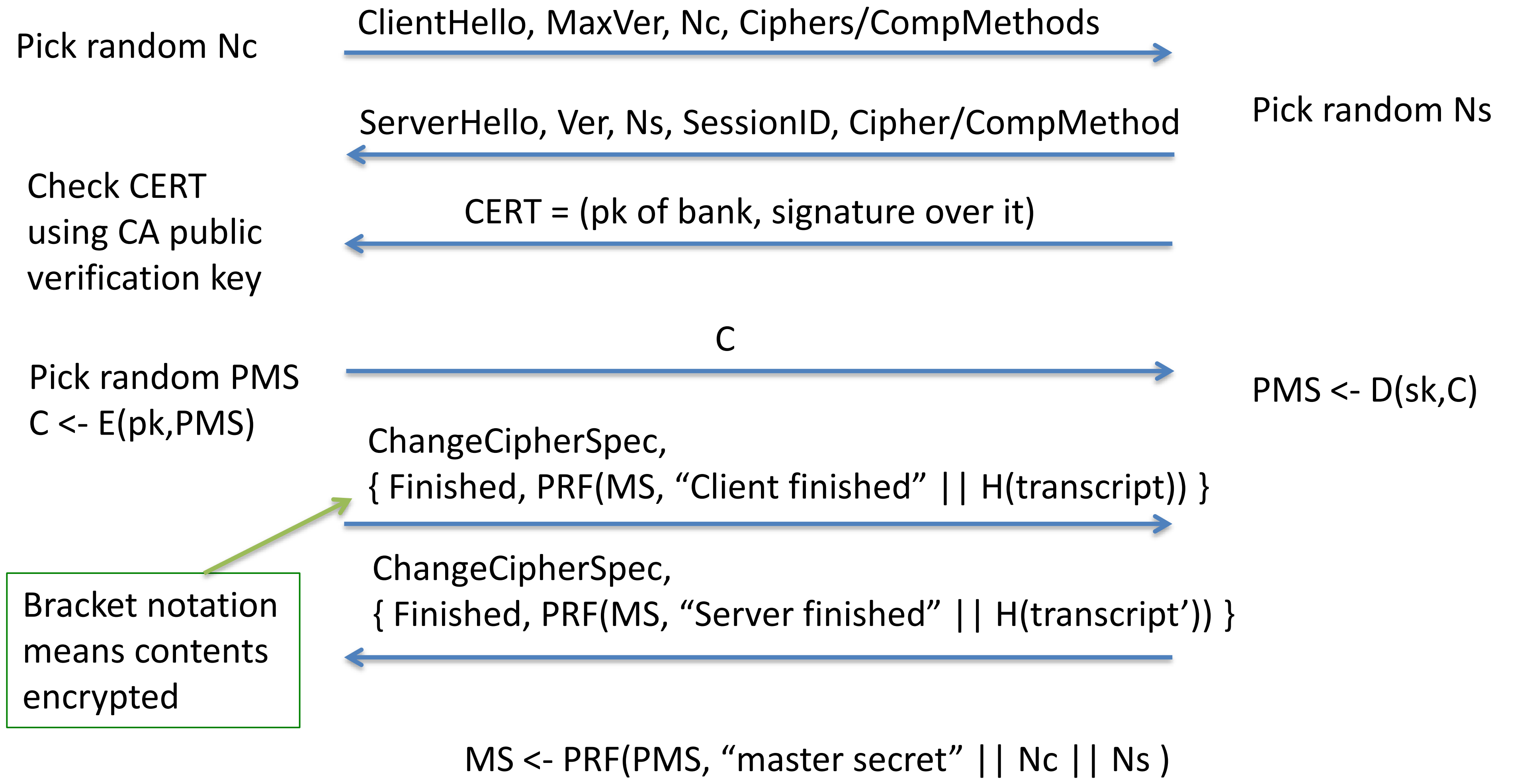
⋮

(more attacks and fixes)

TLS handshake for RSA transport

Bank customer

Bank



Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods

Pick random Ns

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Check CERT using CA public verification key

CERT = (pk of bank, signature over it)

Pick random PMS
C <- E(pk,PMS)

C

PMS <- D(sk,C)

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

Bracket notation means contents encrypted

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

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

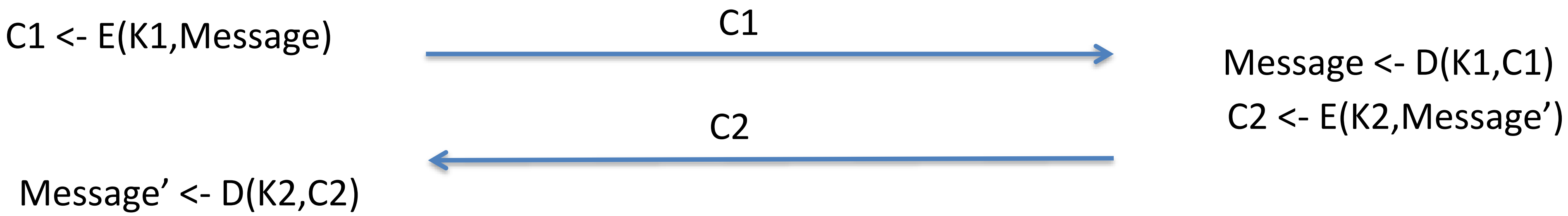
TLS Record layer

Bank customer

Bank

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

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



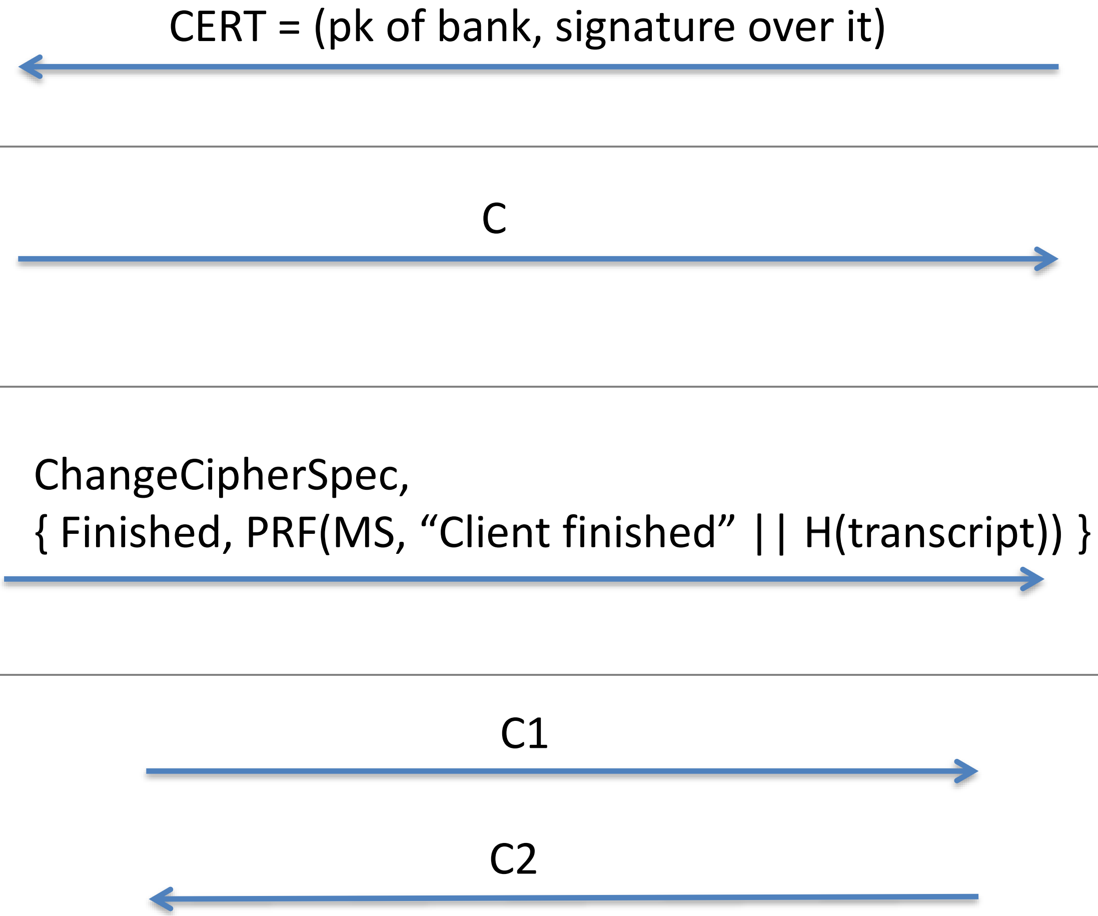
C1 <- E(K1,Message)

Message <- D(K1,C1)

C2 <- E(K2,Message’)

Message’ <- D(K2,C2)

Primitives used by TLS



CERT = (pk of bank, signature over it)

Digital signatures

C

Public-key encryption (RSA)

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

PRF
Hash function

C1

C2

Symmetric encryption

TLS was built via "design-break-redesign-break…"

We're now at TLS ver 1.2
~~No (publicly) known attacks~~

Did the TLS designers get it right?

In last few years host of attacks that affect TLS 1.2 as well have been discovered

[Paterson, Ristenpart, Shrimpton 2011]
Lucky 13 attack [AlFardan, Paterson 2013]
…

Even for "simple" applications (secure channels), secure cryptography is **really hard to design**. The problems are rarely in primitives.

Many other tools have similar story:

SSH, IPSec, Kerberos, WEP/WPA (WiFi security), GSM (cell phone networks), …

Provable security cryptography

Supplement “design-break-redesign-break…” with a more mathematical approach

1. Design a cryptographic scheme
2. Provide proof that no one is able to break it

} Shannon 1949

<u>Formal definitions</u>

Scheme semantics

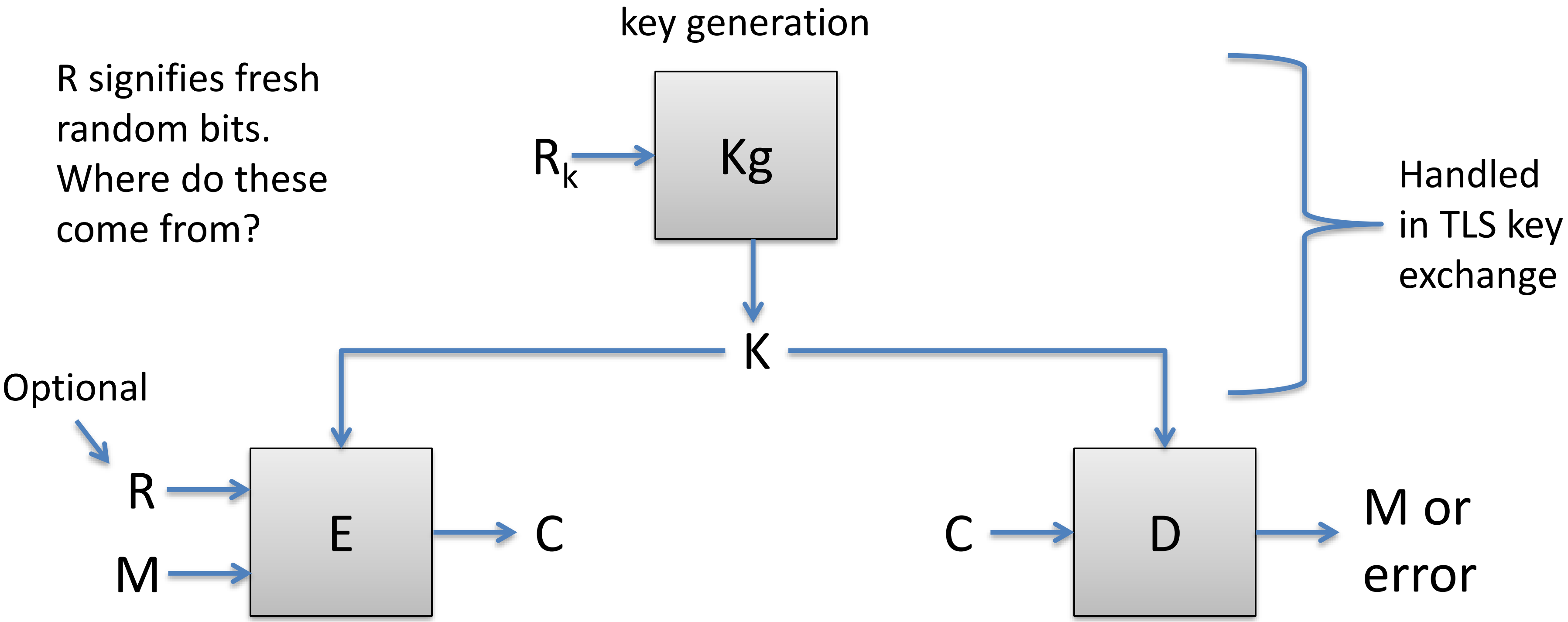
Security

<u>Security proofs</u>

Show it is mathematically impossible to break security

Symmetric encryption

R signifies fresh random bits.
Where do these come from?



key generation

R_k

Kg

K

Optional

R

M

E

C

C

D

M or error

Handled in TLS key exchange

C is a ciphertext

Correctness: D(K , E(K,M,R)) = M with probability 1 over randomness used

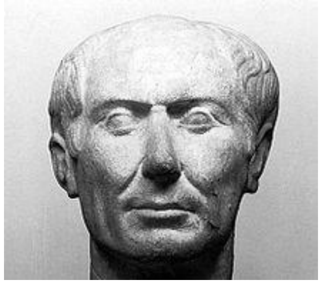
Kerckhoffs' principle: what parts are public and which are secret?

Some attack settings

- Attacker goal: decrypt ciphertext or obtain key
- Unknown plaintext
  - attacker only sees ciphertexts
- Known plaintext
  - attacker knows some plaintext-ciphertext pairs
- Chosen plaintext
  - attacker can choose some plaintexts and receive encryptions of them
- Chosen ciphertext
  - Attacker can get someone to decrypt a message of their choosing,

Substitution ciphers

Julius Caeser



Kg: output randomly chosen permutation of digits

	0	1	2	3	4	5	6	7	8	9	plaintext digit
K =	8	2	7	4	1	6	0	5	9	3	ciphertext digit

E(K, 2321-4232-1340-1410) = 7472-1747-2418-2128

Jane Doe	2414-2472-2742-7428
Michael Swift	3612-4260-2478-7243
John Jones	6020-7412-7412-2728
Eve Judas	7472-1747-2418-2128

1343-1321-1231-2310

Knowing one plaintext, ciphertext pair leaks key material!

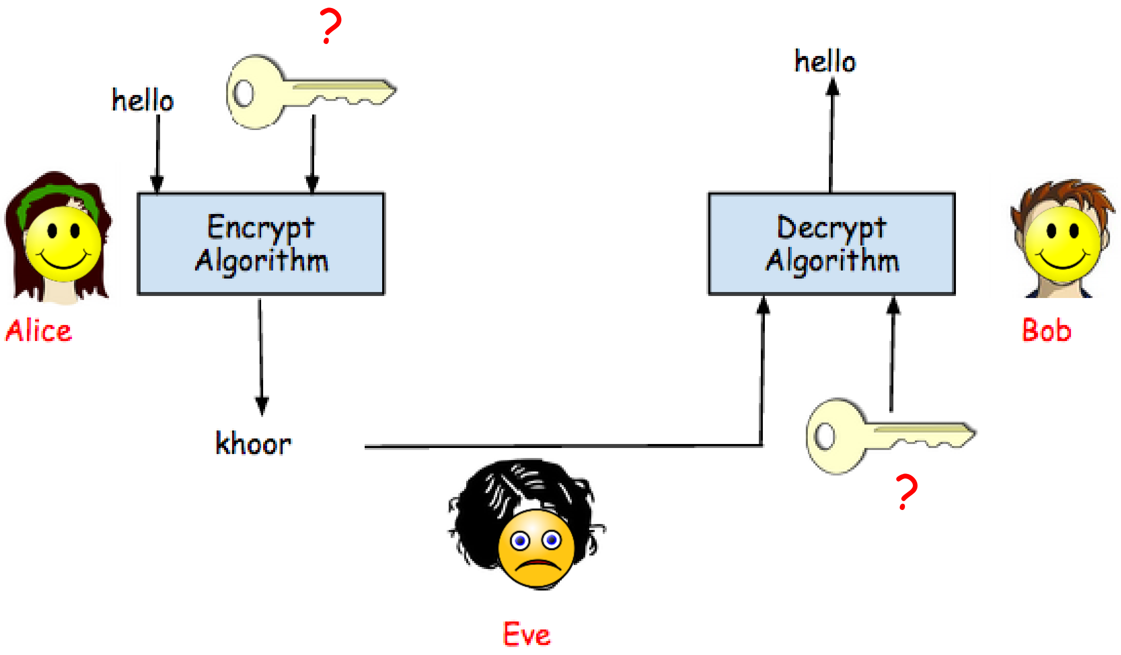
Attacker knows 2321-4232-1340-1410

7472-1747-2418-2128

0	1	2	3	4	5	6	7	8	9
?	?	?	?	?	?	?	?	?	?

Cracking Simple Substitution

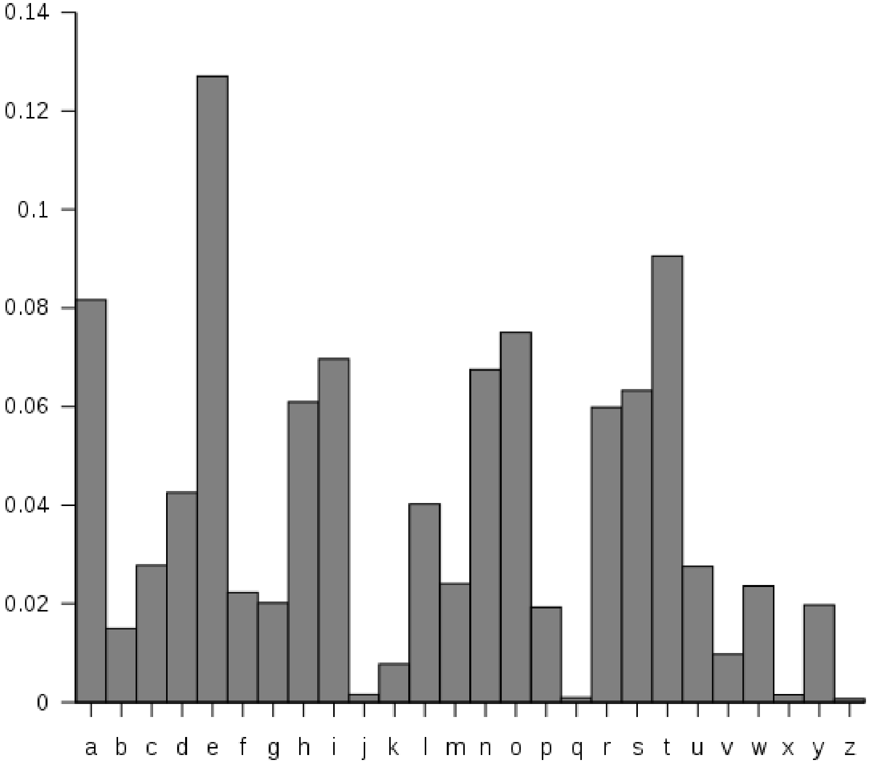
- *Brute force attack:* Eve would need 26! keys.
- That’s 4.0329146e+26 keys. Too hard!



Cracking Simple Substitution

- *But, wait a minute...*

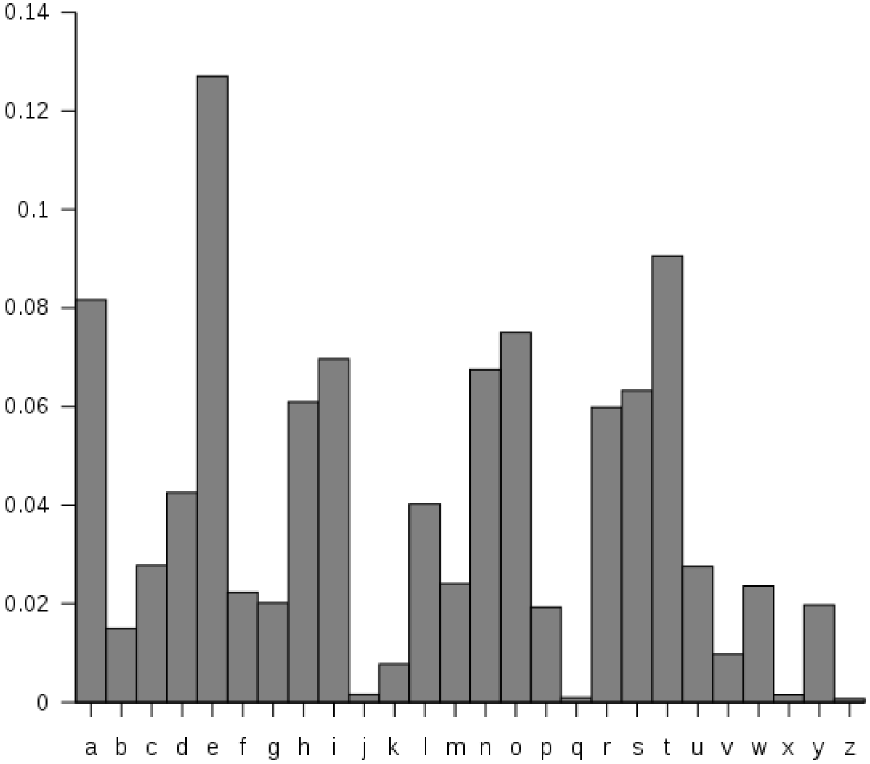
English plaintext
letter frequencies



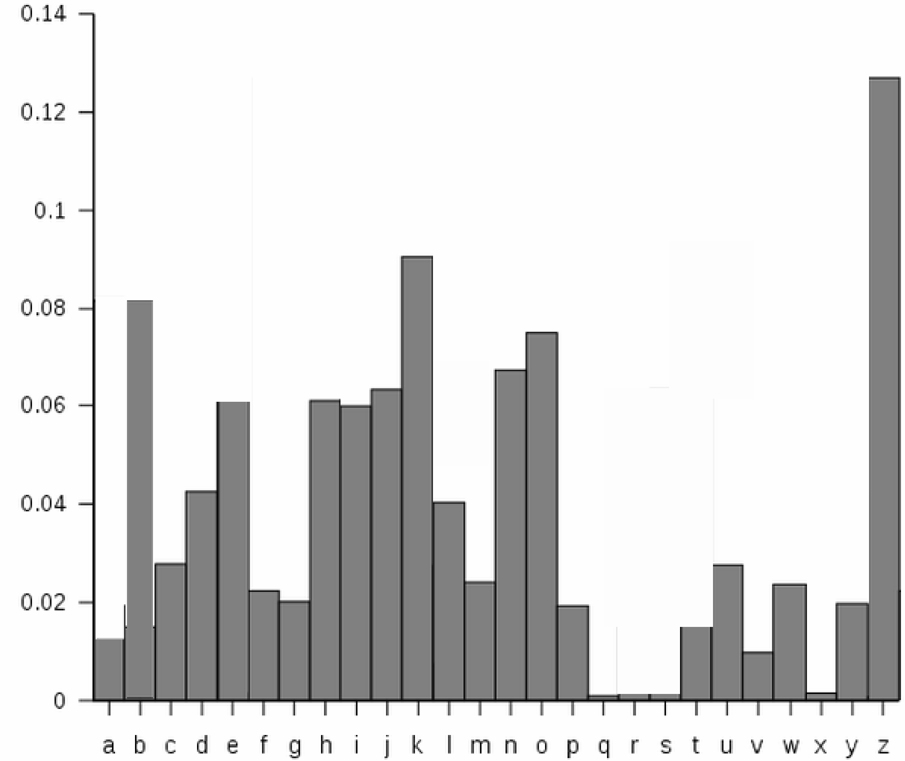
Cracking Simple Substitution

- *But, wait a minute...*

English plaintext
letter frequencies



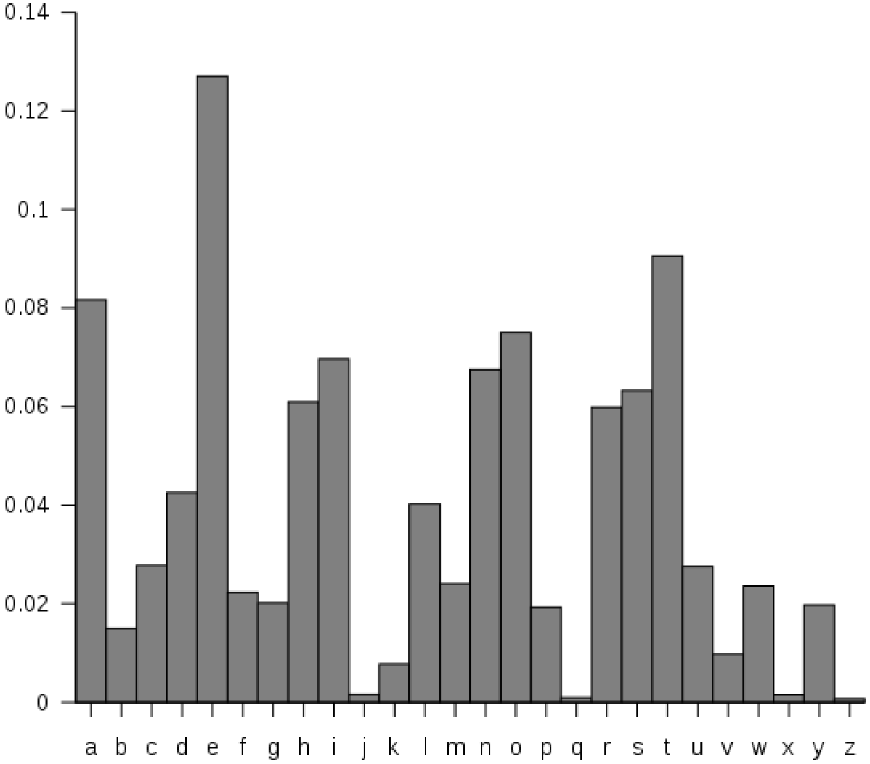
Ciphertext
letter frequencies



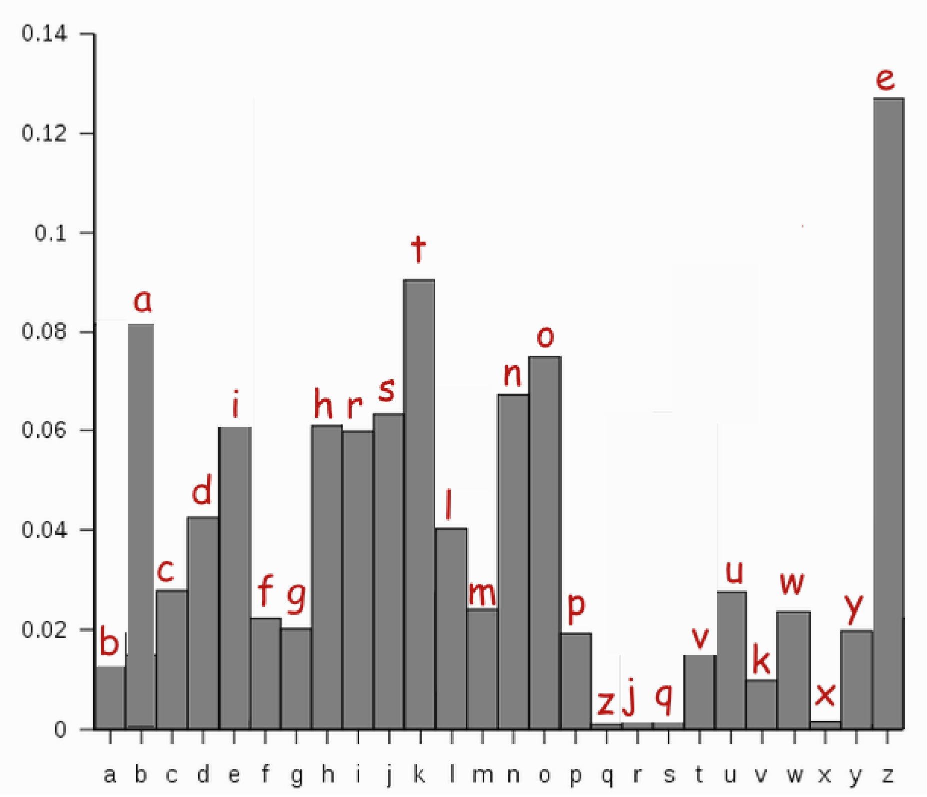
Cracking Simple Substitution

- *But, wait a minute… frequency analysis works!*

English plaintext
letter frequencies

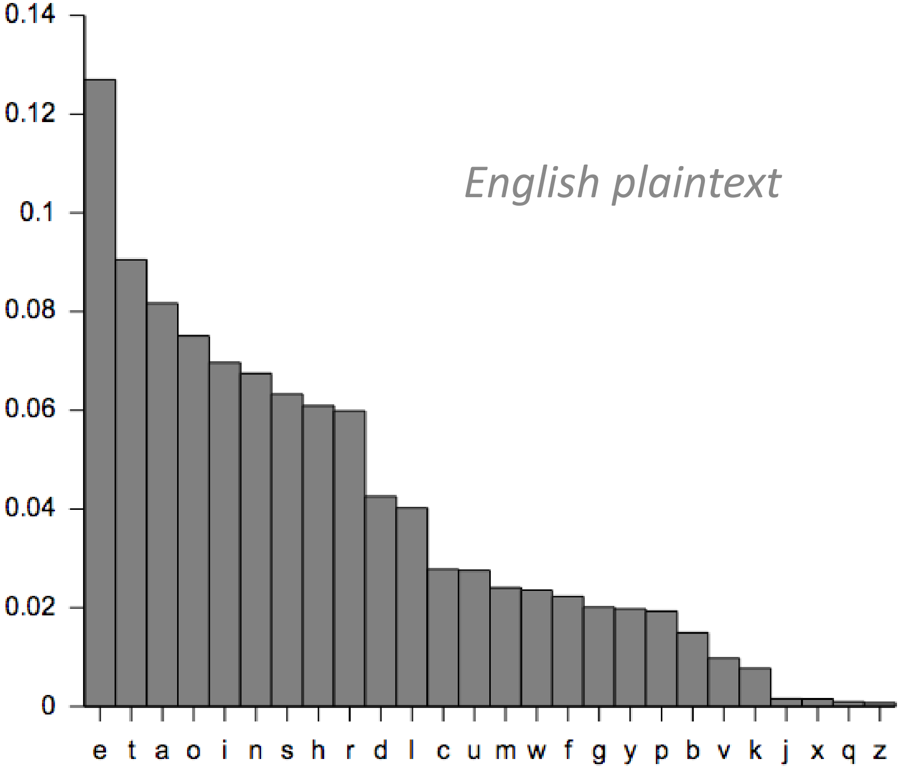


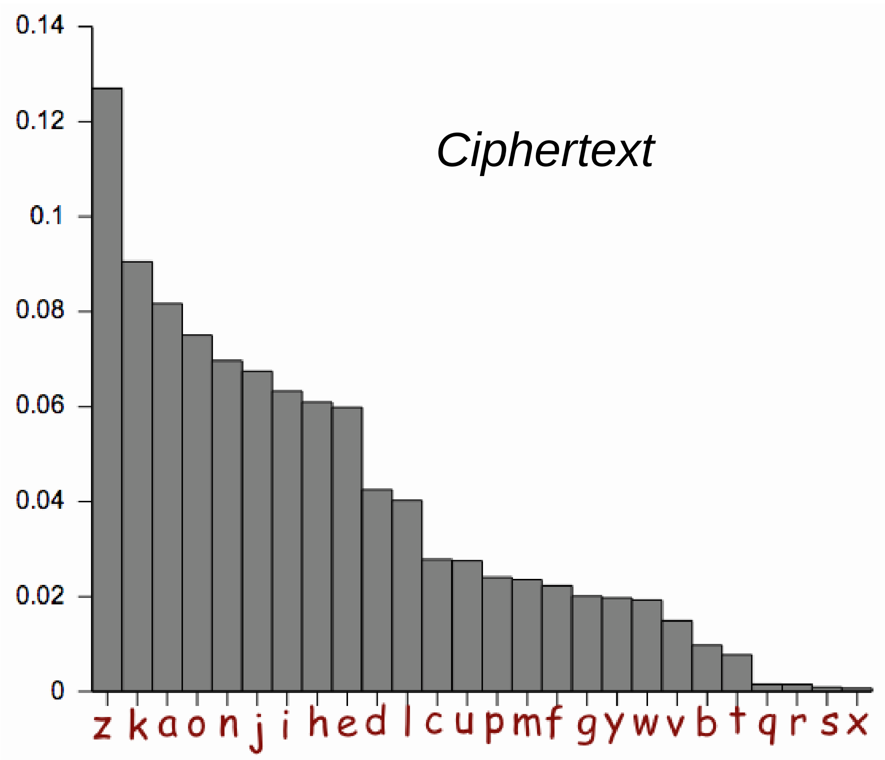
Ciphertext
letter frequencies



Cracking Simple Substitution

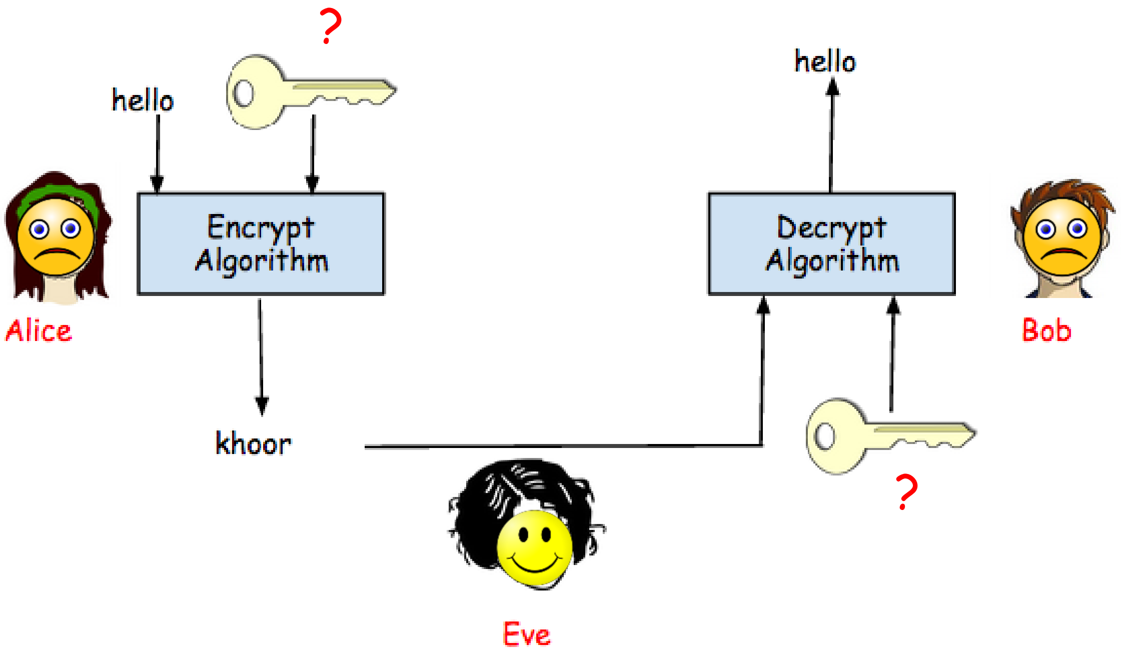
- *Can sort by frequencies*





Cracking Simple Substitution

- Eve wins … you don't need brute force
- *Frequency analysis* will break simple substitution



enigma

- Enigma was state of the art cryptography developed by the Germans
- Broken by the Allies
- Raised theoretical questions about cryptography

One-time pads

Fix some message length L

Kg: output random bit string K of length L

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

Shannon’s security notion

Def. A symmetric encryption scheme 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

In words:
each message is equally likely to map to a given ciphertext

In other words:
seeing a ciphertext leaks nothing about what message was encrypted

Does a substitution cipher meet this definition? No!

Shannon's security notion

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

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

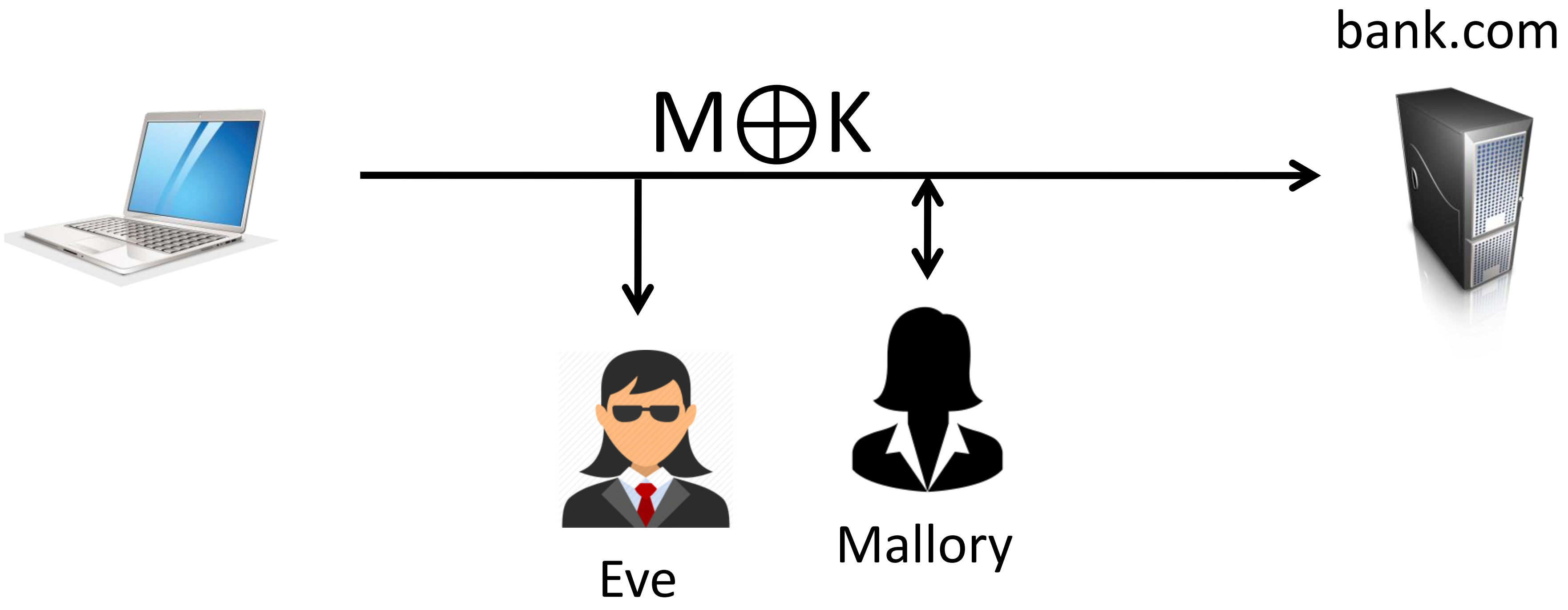
where probabilities are over choice of K

Thm. OTP is perfectly secure

For any C and M of length L bits

$$\Pr[\, K \oplus M = C \,] = 1 / 2^{L}$$

$$\Pr[\, K \oplus M = C \,] = \Pr[\, K \oplus M' = C \,]$$



bank.com

$M \oplus K$

Eve

Mallory

K must be as large as M

Reusing K for M,M' leaks $M \oplus M'$

Message length is obvious

Mallory can make undetected (unknown) modifications

OTP limitations

provable security

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

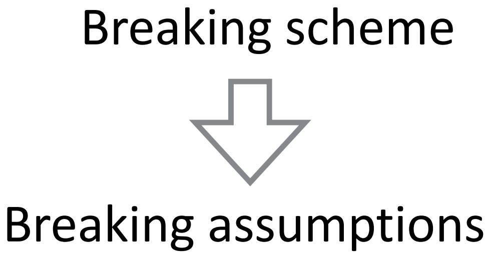
1. Design a cryptographic scheme
2. Provide a **proof** that no attacker with bounded computational resources can break it

[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

- Basic atomic primitives are hard to break:
  - Factoring of large composites intractable
  - RSA permutation hard-to-invert
  - Block ciphers (AES, DES) are good pseudorandom permutations (PRPs)
  - Hash functions are collision resistant

Confidence in atomic primitives is gained by cryptanalysis, public design competitions

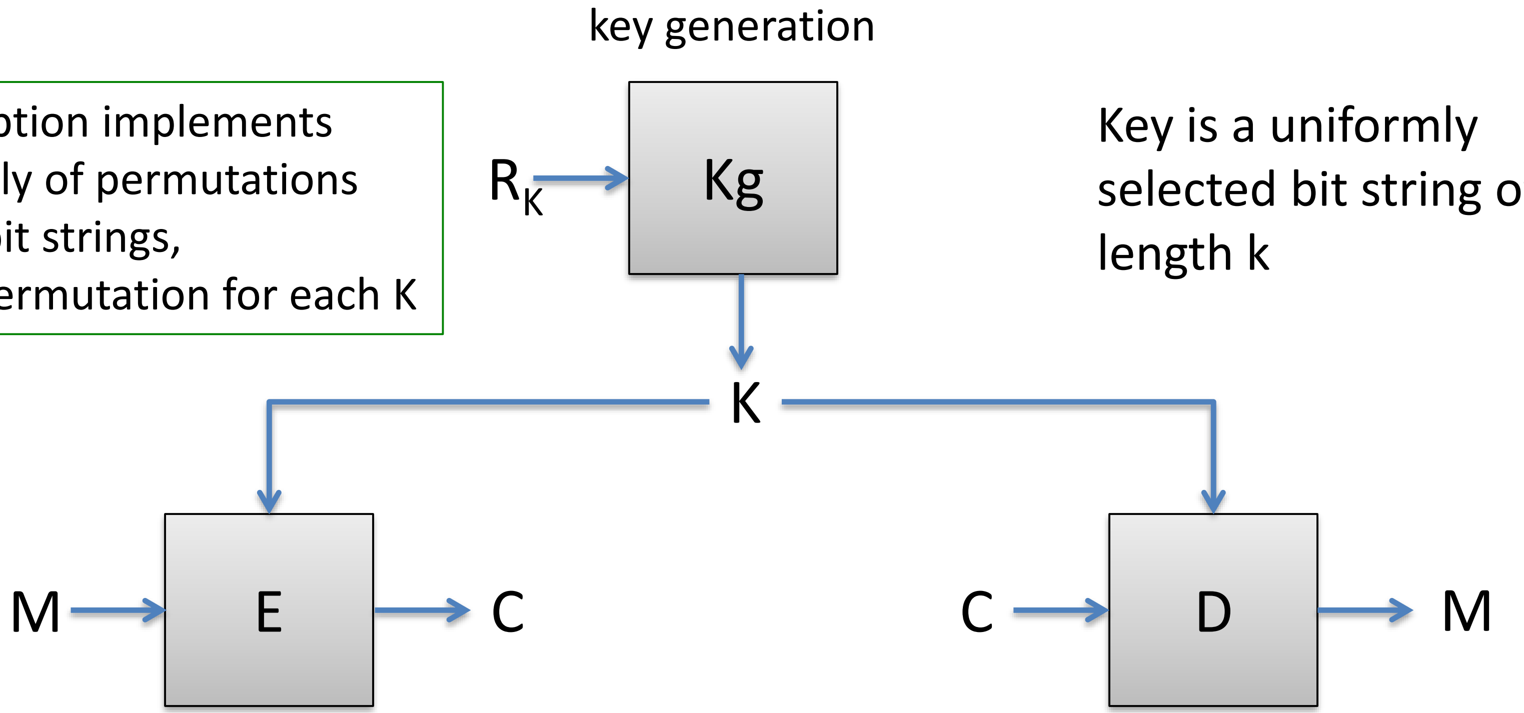
SHA-3 competition, AES competition

recap

- 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

Block ciphers

Encryption implements a family of permutations on n bit strings, one permutation for each K

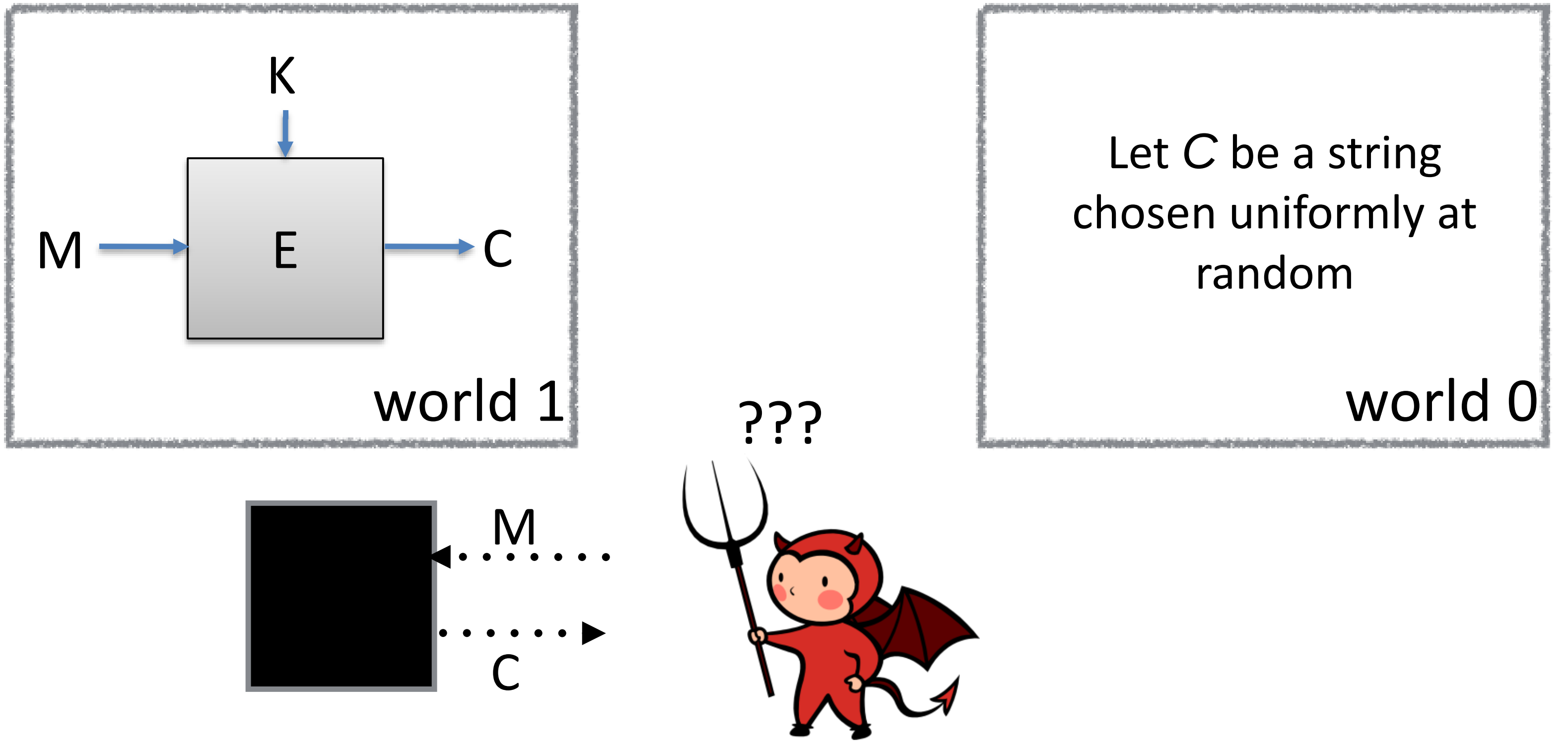


Key is a uniformly selected bit string of length k

E: $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$

Security goal: *E*(*K*,*M*) is indistinguishable from a random *n*-bit string for anyone that doesn't know *K*

E: $\{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$



Can adversary distinguish between World 0 and World 1?

If this holds for all polynomial time adversaries, then *E* is called a secure pseudorandom function (PRF)

block cipher security

Data encryption standard (DES)

Originally called Lucifer

- team at IBM
- input from NSA
- standardized by NIST in 1976

n = 64

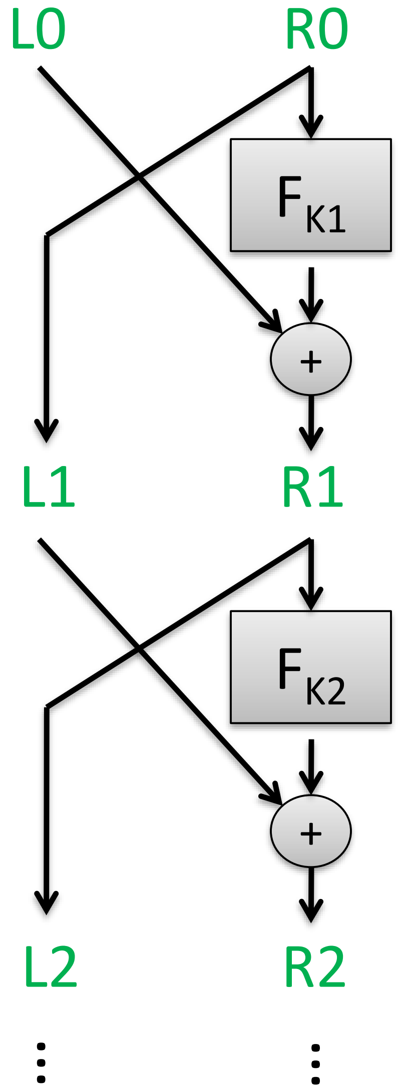
k = 56

Number of keys:
72,057,594,037,927,936

Split 64-bit input into L0,R0 of 32 bits each

Repeat Feistel round 16 times

Each round applies function F using separate round key



Best attacks against DES

Attack	Attack type	Complexity	Year
Biham, Shamir	Chosen plaintexts, recovers key	2^{47} plaintext, ciphertext pairs	1992
DESCHALL	Unknown plaintext, recovers key	$2^{56/4}$ DES computations 41 days	1997
EFF Deepcrack	Unknown plaintext, recovers key	~4.5 days	1998
Deepcrack + DESCHALL	Unknown plaintext, recovers key	22 hours	1999

- DES is still used in some places
- 3DES (use DES 3 times in a row with more keys) expands keyspace and still used widely in practice

Advanced Encryption Standard (AES)

Response to 1999 attacks:

- NIST has design competition for new block cipher standard
- 5 year design competition
- 15 designs, Rijndael design chosen

Advanced Encryption Standard (AES)

Rijndael (Rijmen and Daemen)

n = 128
k = 128, 192, 256

Number of keys for k=128:
340,282,366,920,938,463,463,374,607,431,768,211,456

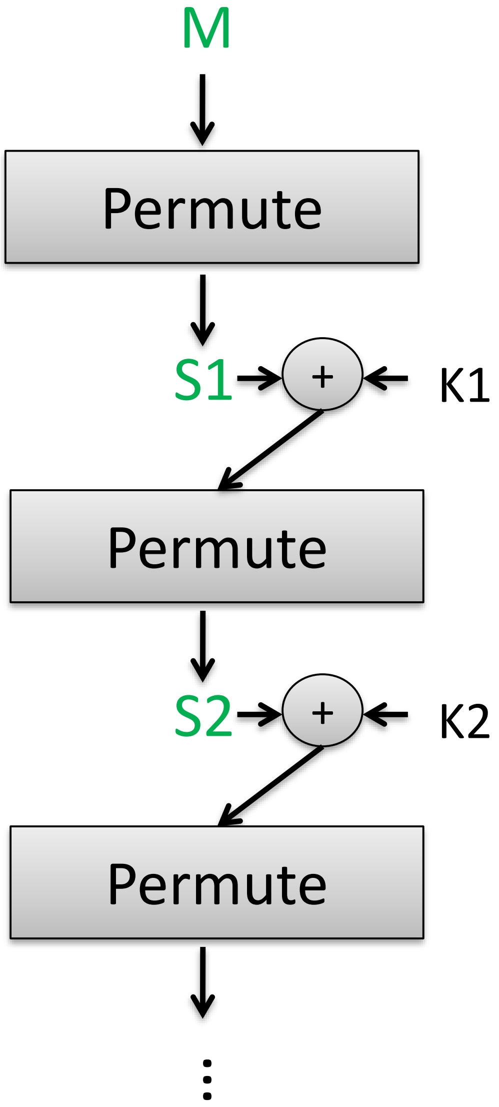
Substitution-permutation design.
For k=128 uses 10 rounds of:

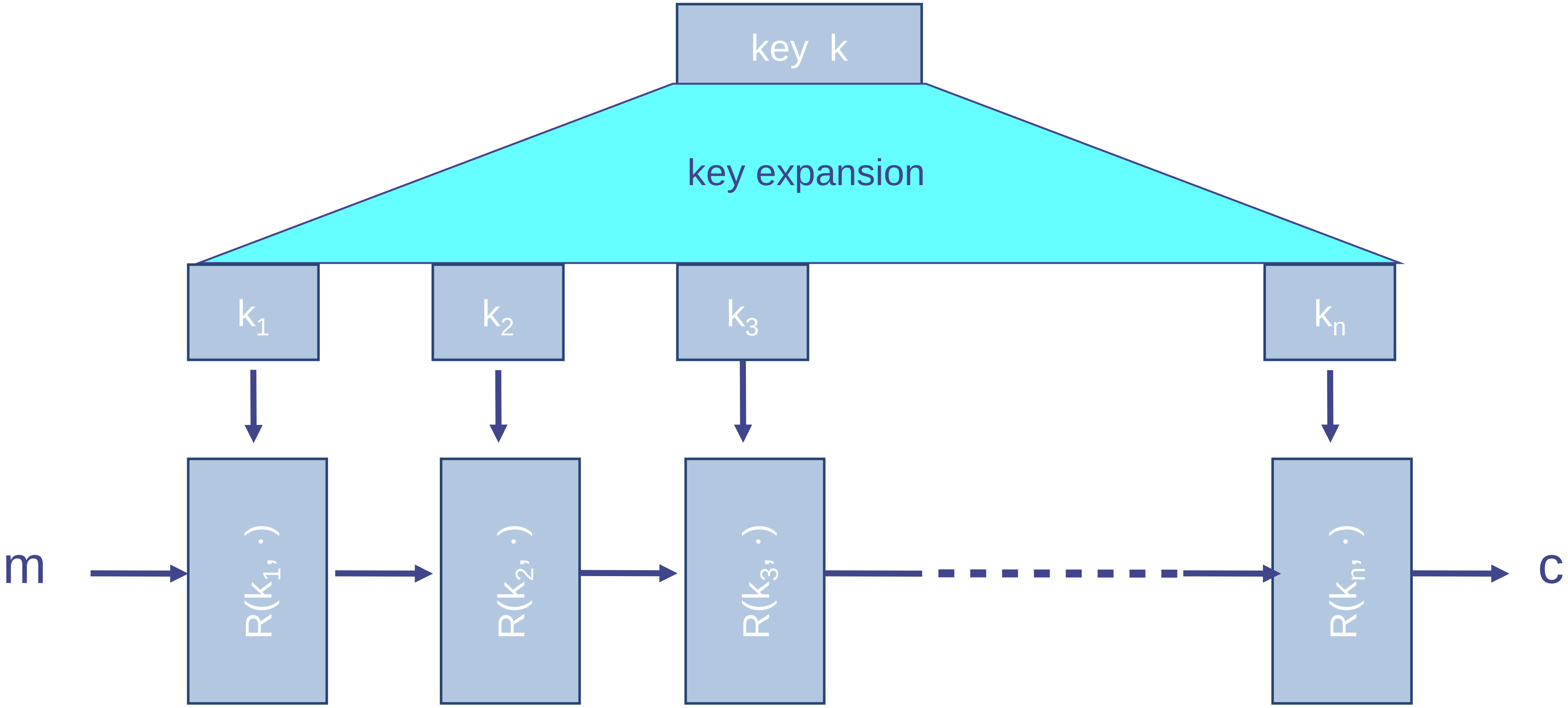
1) Permute:

SubBytes (non-linear S-boxes)
ShiftRows + MixCols (invertible linear transform)

2) XOR in a round key derived from K

(Actually last round skips MixCols)



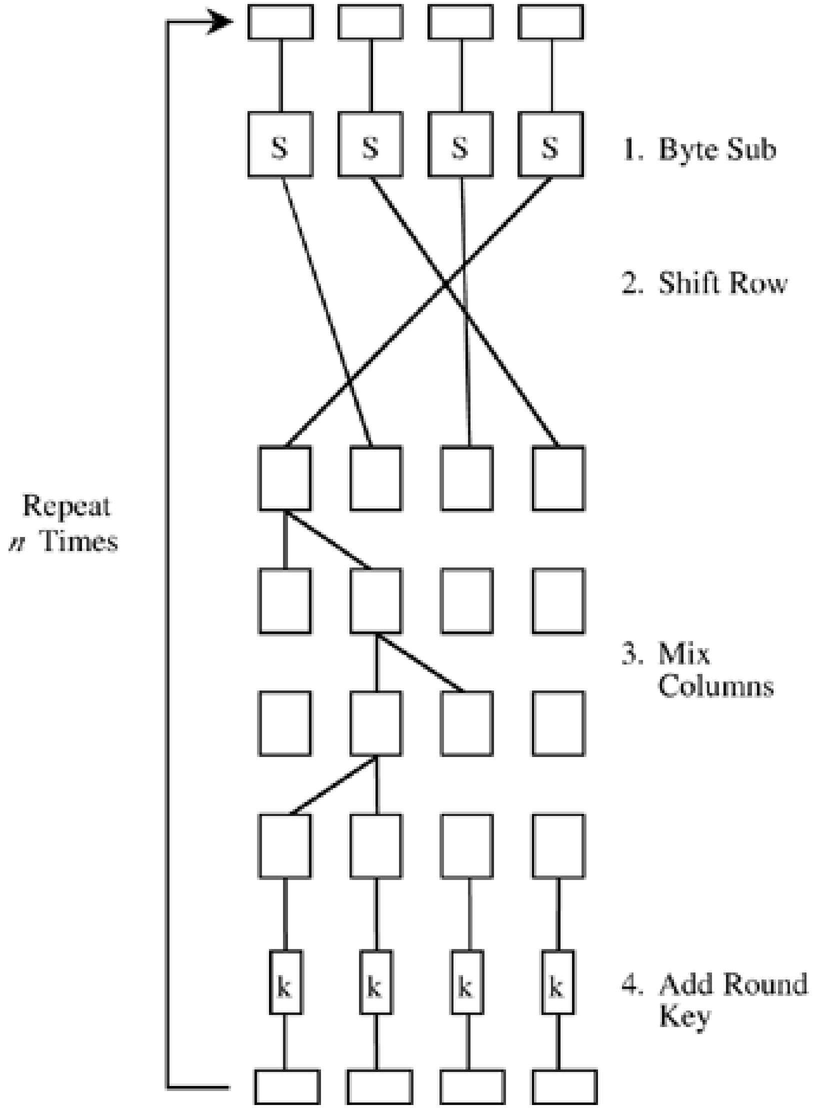


R(k,m): round function

AES-128 n=10

building a block cipher

[slide credit: Dan Boneh, CS155]



Designing good block ciphers is a dark art

Must resist subtle attacks: differential attack, linear attacks, others

Chosen through public design contests

Use build-*break*-build-*break* iteration

aes round function

Best attacks against AES

Attack	Attack type	Complexity	Year
Bogdanov, Khovratovich, Rechberger	chosen ciphertext, recovers key	$2^{126.1}$ time + some data overheads	2011

- Brute force requires time 2^{128}
- Approximately factor 4 speedup

Summary and next time

- Crypto as computational science
- Overview of TLS
- Symmetric encryption and block ciphers introduced