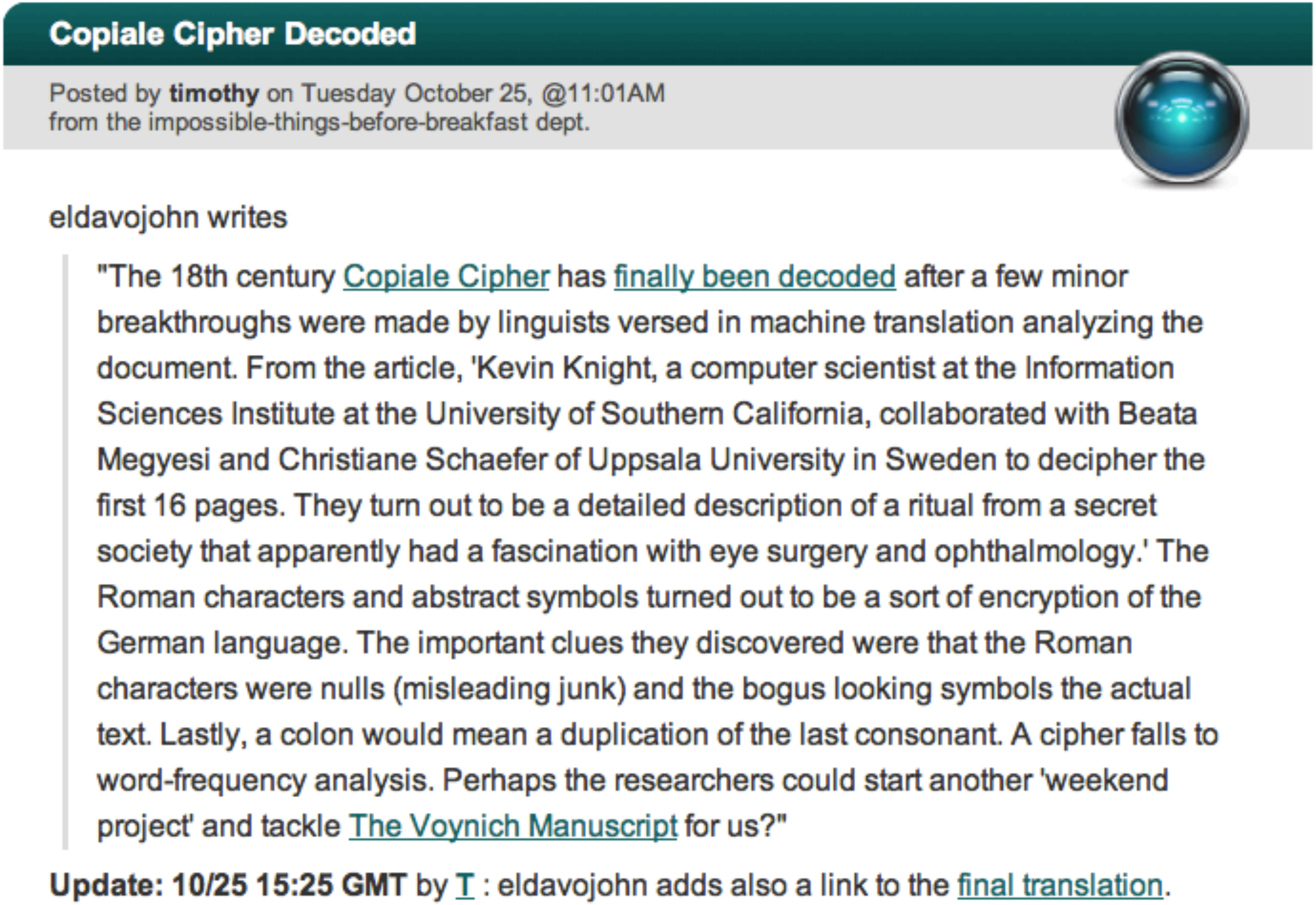
Cryptography Intro

CS642: Computer Security

Professor Ristenpart

http://www.cs.wisc.edu/~rist/

rist at cs dot wisc dot edu



Copiale Cipher Decoded

Posted by **timothy** on Tuesday October 25, @11:01AM
from the impossible-things-before-breakfast dept.

eldavojohn writes

> "The 18th century Copiale Cipher has finally been decoded after a few minor breakthroughs were made by linguists versed in machine translation analyzing the document. From the article, 'Kevin Knight, a computer scientist at the Information Sciences Institute at the University of Southern California, collaborated with Beata Megyesi and Christiane Schaefer of Uppsala University in Sweden to decipher the first 16 pages. They turn out to be a detailed description of a ritual from a secret society that apparently had a fascination with eye surgery and ophthalmology.' The Roman characters and abstract symbols turned out to be a sort of encryption of the German language. The important clues they discovered were that the Roman characters were nulls (misleading junk) and the bogus looking symbols the actual text. Lastly, a colon would mean a duplication of the last consonant. A cipher falls to word-frequency analysis. Perhaps the researchers could start another 'weekend project' and tackle The Voynich Manuscript for us?"

Update: 10/25 15:25 GMT by T : eldavojohn adds also a link to the final translation.

Cryptography

Basic goals and setting

TLS (HTTPS)

Provable security

One time pad

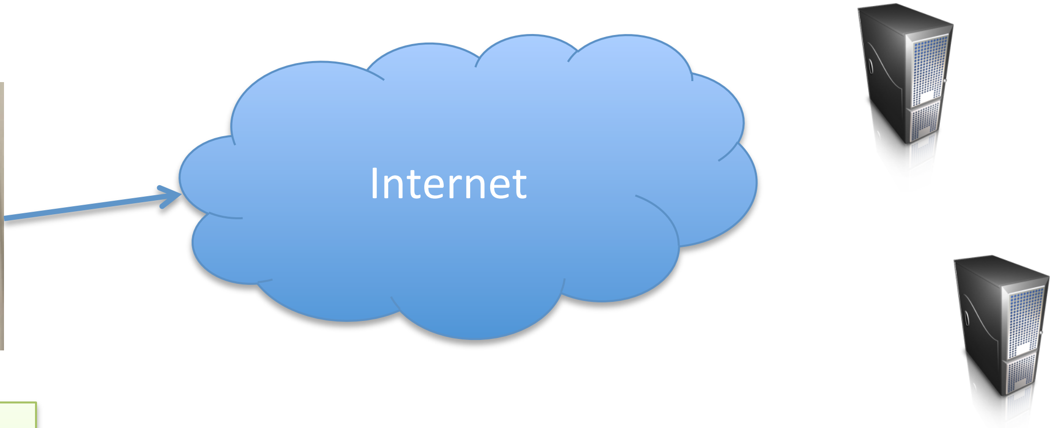
Block ciphers

Cryptography: “Hidden writing”

- Study and practice of building security protocols that resist adversarial behavior
- Blend of mathematics, engineering, computer science

WikiLeaks encrypted cables

Cryptography



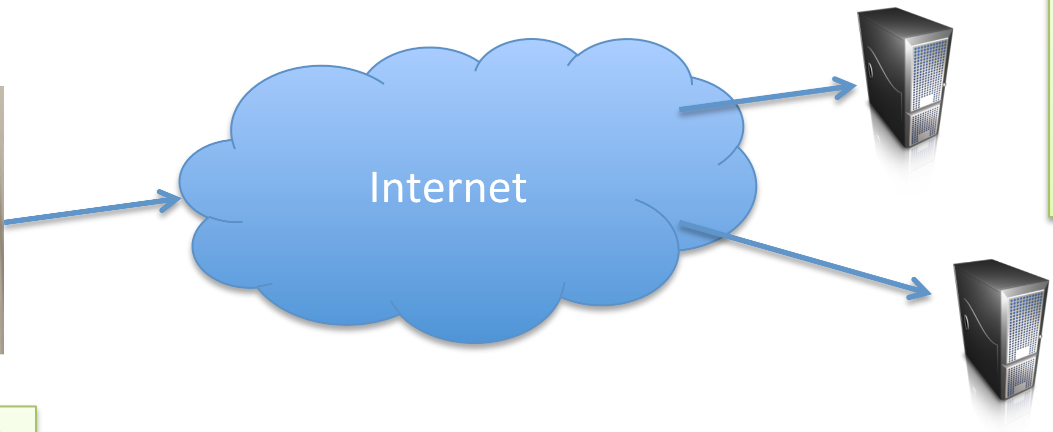
US diplomatic cables

Don’t want to reveal data early

Want to store it in way that it can quickly be revealed later

Cryptography

01101010
10101010
10101010
11111101



01101010
10101010
10101010
11111101

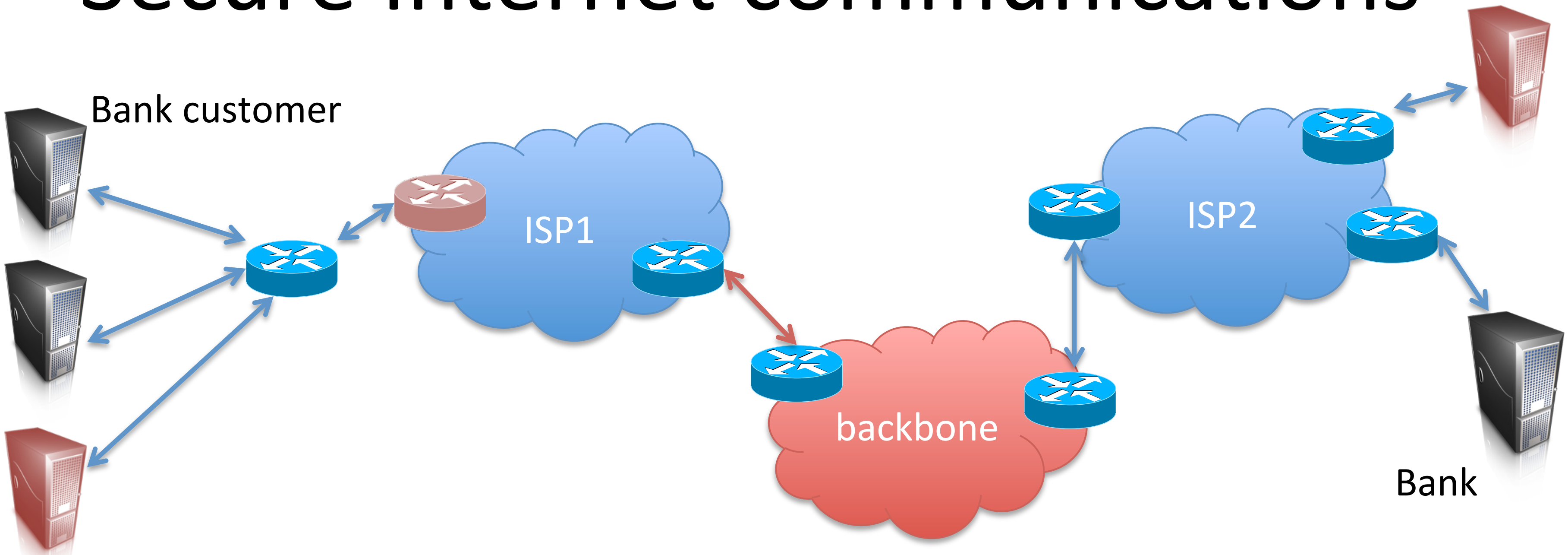
01101010
10101010
10101010
11111101

Don’t want to reveal data early

Want to store it in way that it can quickly be revealed later

Modern cryptography enables this:
- Encrypt file
- Store key in secure place

Secure Internet communications



Customer and bank want to communicate securely:

- **Confidentiality** (messages are private)
- **Integrity** (accepted messages are as sent)
- **Authenticity** (is it the bank? is this the customer?)
- Non-goal: anonymity (hide identities)
- Non-goal: steganography (hide that communication took place)

TLS, SSH, IPsec, PGP

Encrypted hard disks

Company’s intellectual property
Customer records
Your personal diary

Encrypt hard drives (or volumes):

- Confidentiality of data
- Attacker has physical access to device

Bitlocker, Truecrypt, Seagate

Crypto

- Powerful tool for confidentiality, authenticity, and more
- But:
  - must design securely
  - must implement designs securely
  - must use properly (e.g., key management)

Auguste Kerckhoffs' (Second) Principle

"The system must not require secrecy and can be stolen by the enemy without causing trouble"

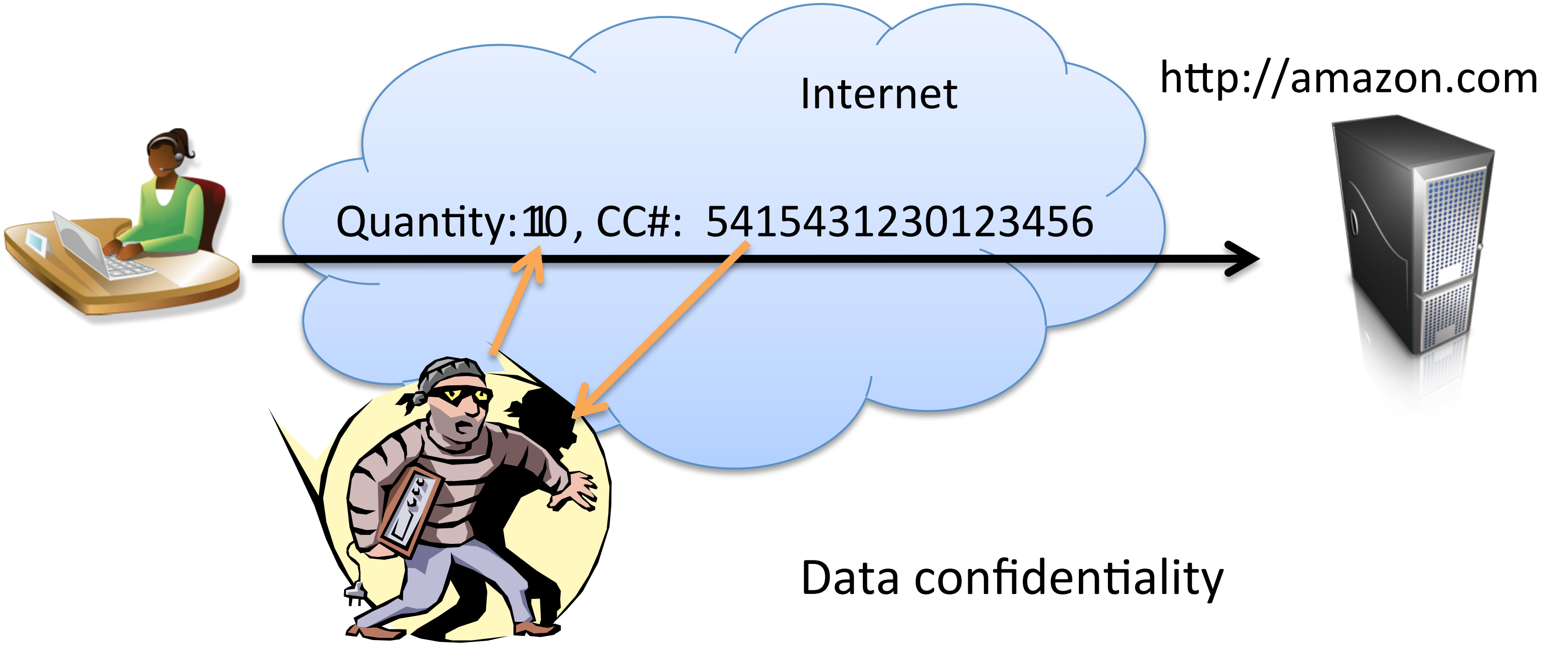
A cryptosystem should be secure even if its algorithms, implementations, configuration, etc. is made public --- the only secret should be a key

Why?

Basic primitives

- Symmetric cryptography (shared key K)
  - encryption & decryption using K
  - message authentication using K
  - pseudorandom functions
- Public-key cryptography (public key pk, secret key sk)
  - encrypt with pk and decrypt with sk
  - digitally sign using sk and verify with pk
- Hash functions (no keys)
  - used to “compress” messages in a secure way

An example: On-line shopping



Internet

http://amazon.com

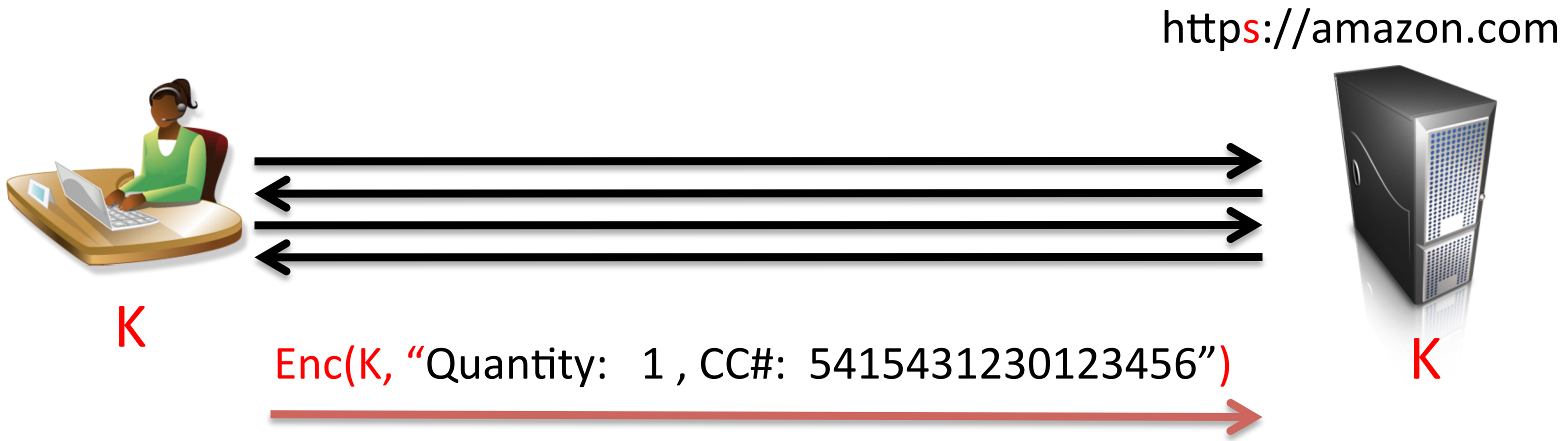
Quantity:110 , CC#: 5415431230123456

Data confidentiality

Data integrity

We need secure channels for transmitting data

An example: On-line shopping with TLS



https://amazon.com

K

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

K

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

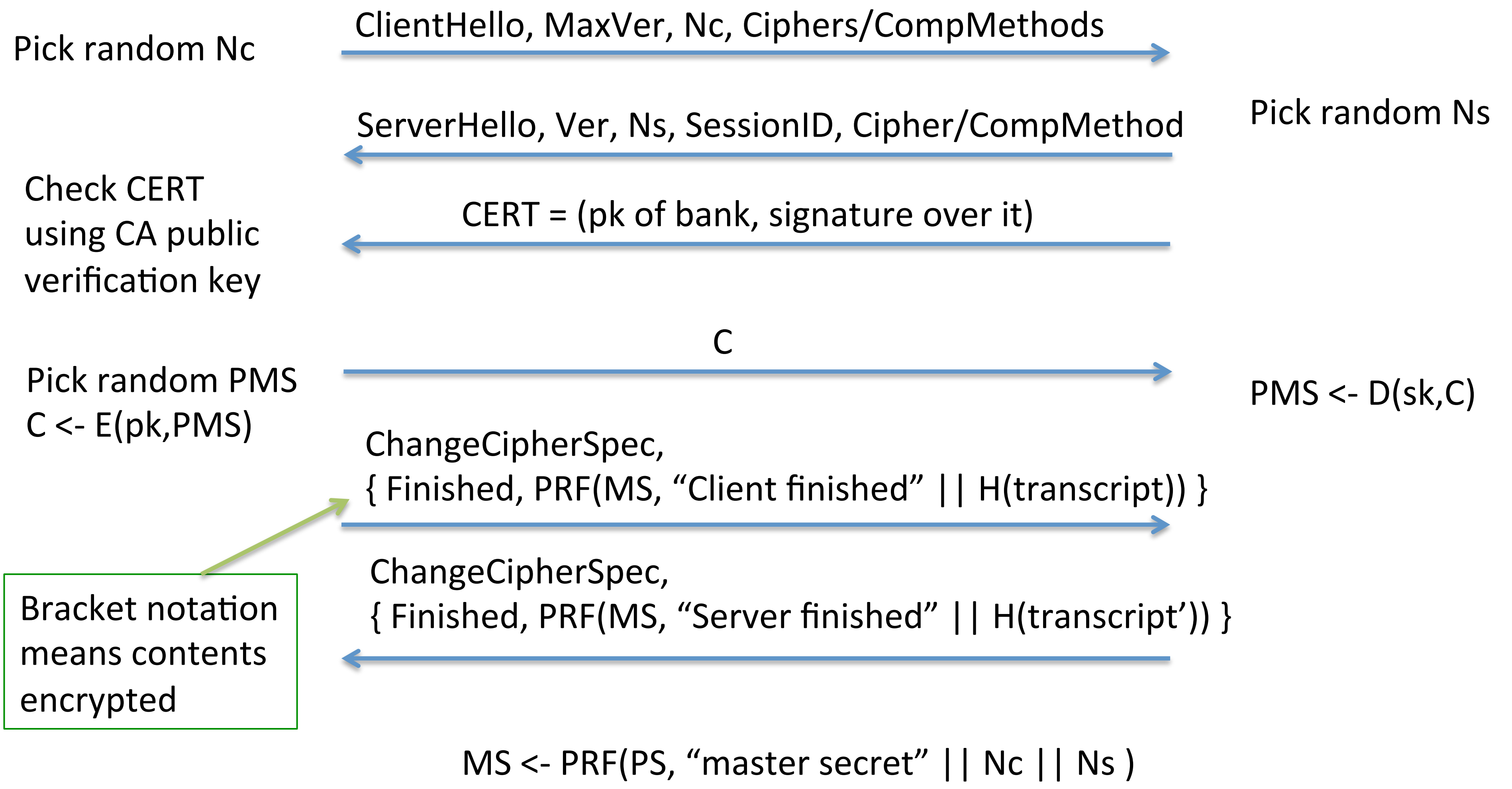
secure channel: symmetric encryption, message authentication

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

TLS handshake for RSA transport

Bank customer

Bank



Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Pick random Ns

Check CERT using CA public verification key

CERT = (pk of bank, signature over it)

C

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

PMS <- D(sk,C)

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

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

Bracket notation means contents encrypted

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

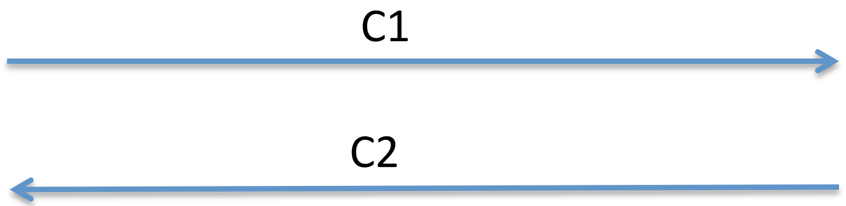
TLS Record layer

Bank customer

Bank

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

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



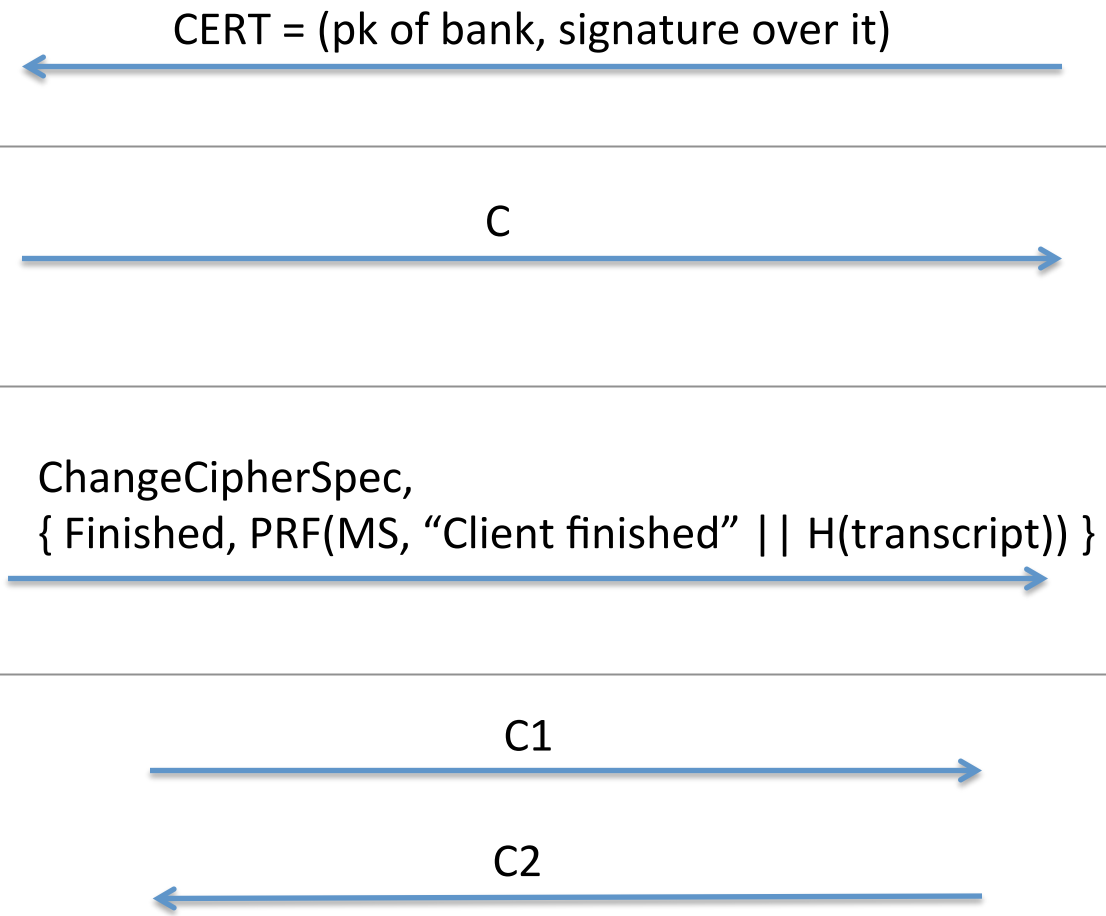
C1 <- E(K1,Message)

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)) }

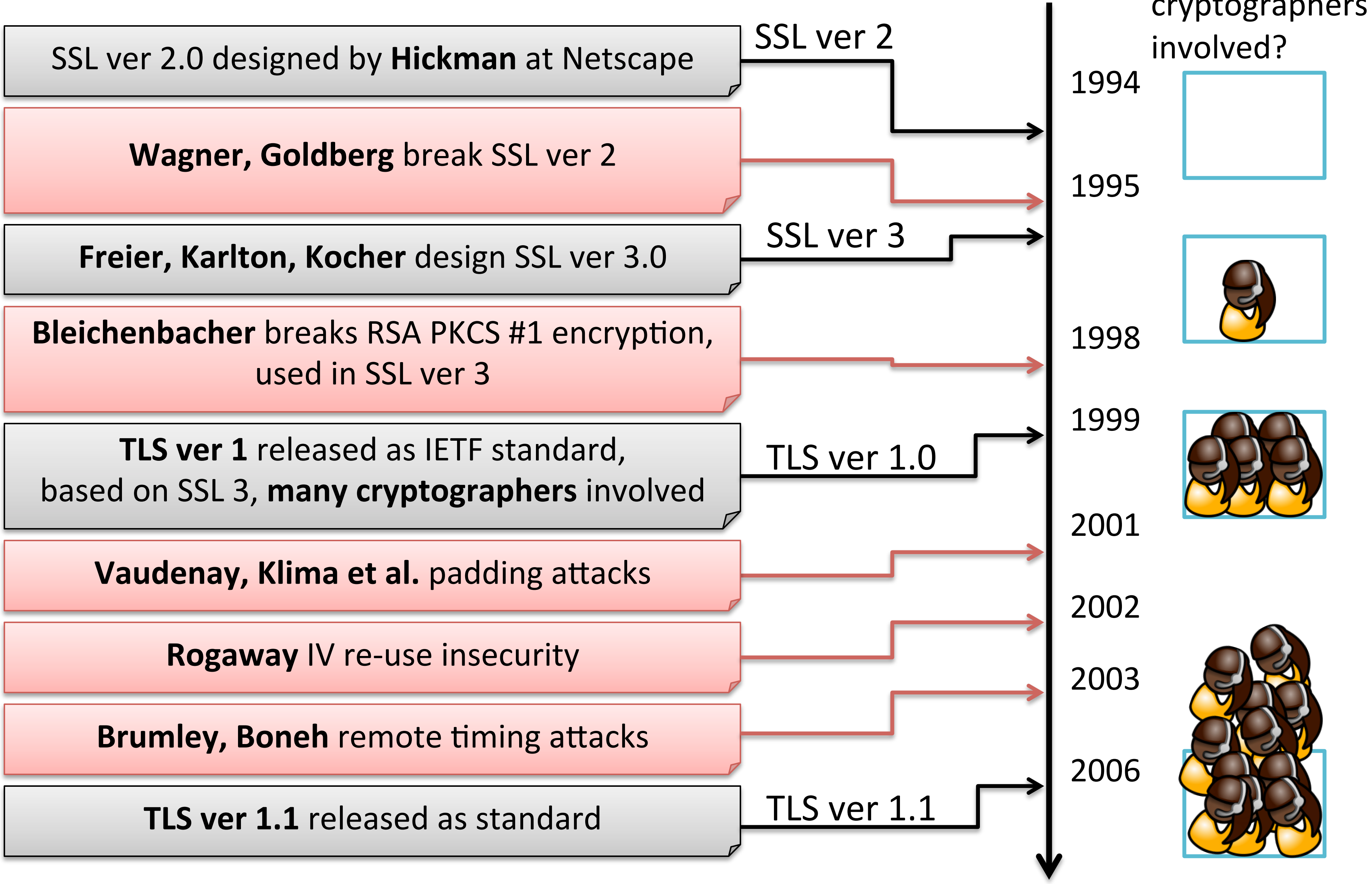
PRF
Hash function

C1

C2

Symmetric encryption

A short history of TLS up to 2009



How many cryptographers involved?

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

⋮

(more attacks and fixes)

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?

We recently showed some new attacks against TLS record layer (Paterson, Ristenpart, Shrimpton 2011)

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

Mistakes are costly. Finding them requires rare expertise.

Many other examples of tools produced by "design-break-redesign-break…"

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

"Those who cannot remember the past are condemned to repeat it"

[Santayana 1905]

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

Formal definitions

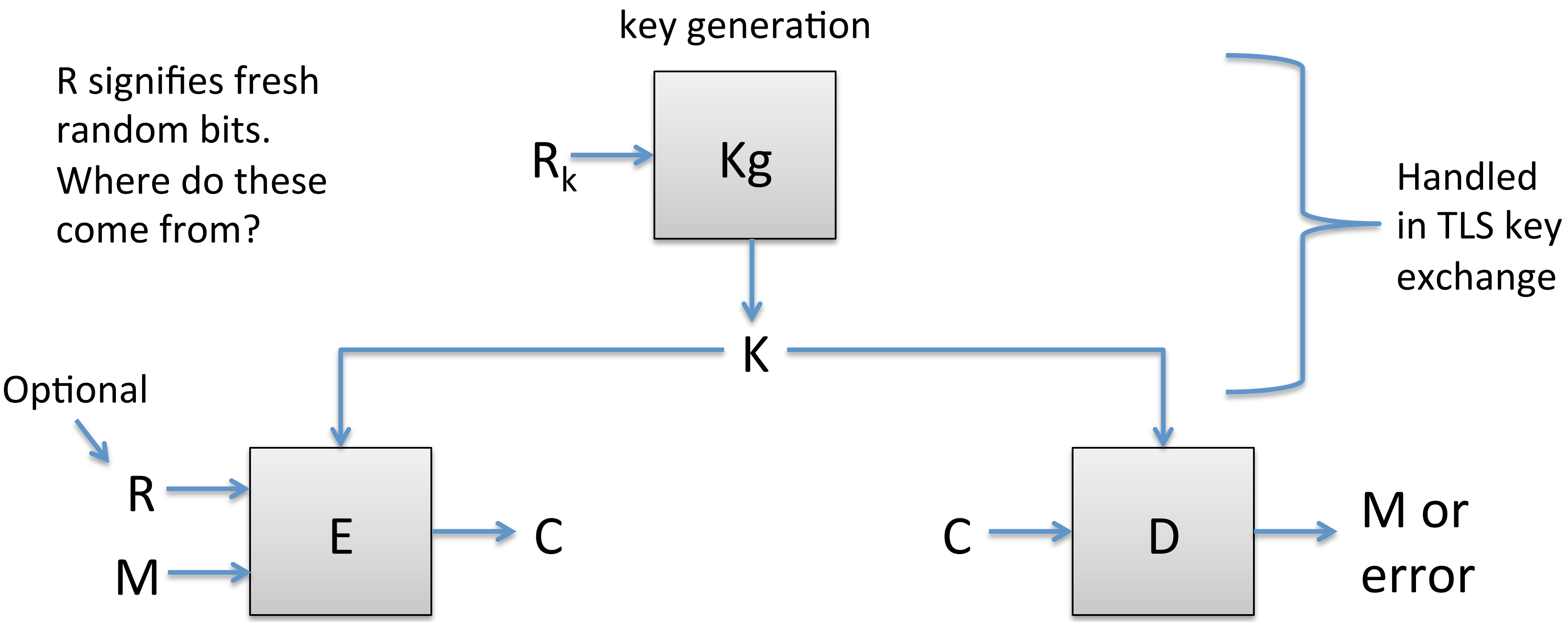
Scheme semantics

Security

Security proofs

Show it is mathematically impossible to break security

Symmetric encryption



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

Handled in TLS key exchange

C is a ciphertext

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

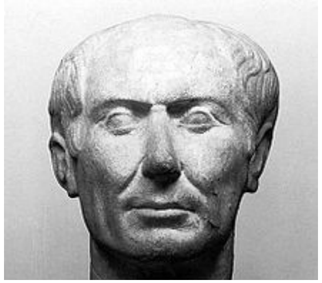
Kerckhoffs’ principle: what parts are public and which are secret?

Some attack settings (not security definitions)

- 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

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
Thomas Ristenpart	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
?	?	?	?	?	?	?	?	?	?

Th
STE
wiring
ENIGMA
machine.

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

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

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?

Shannon's security notion

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

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

where probabilities are over choice of K

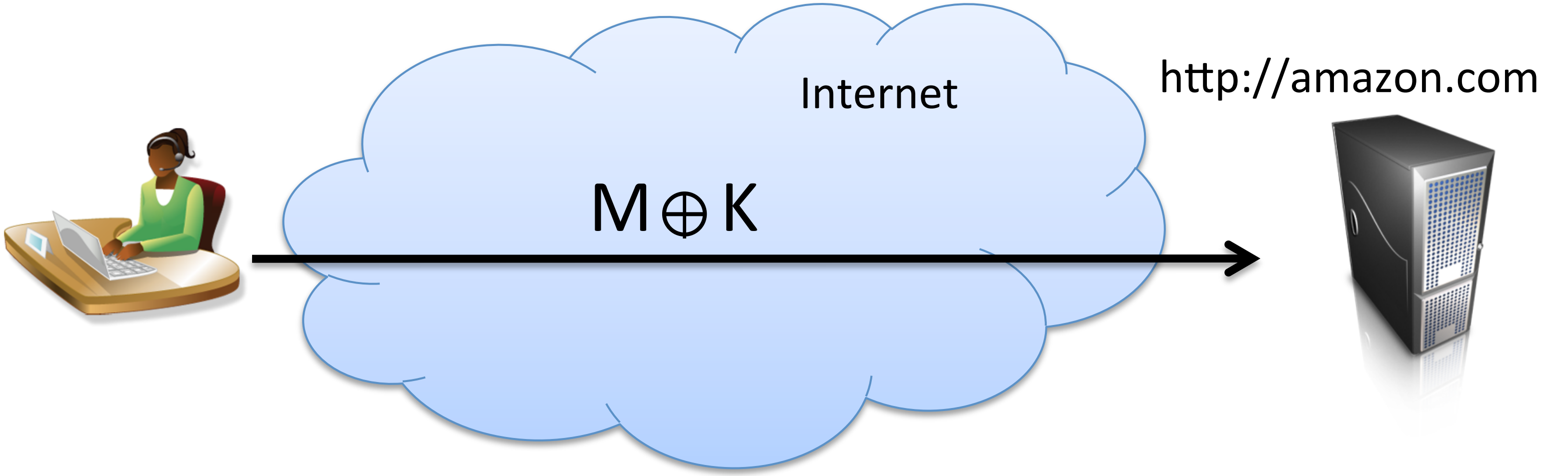
Thm. OTP is perfectly secure

For any C and M of length L bits

$$\Pr[\ K \oplus M = C\ ] \quad = \quad 1 / 2^L$$

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

Back to our application



Does OTP provide a secure channel?

Integrity easily violated

Reuse of K for messages M,M’ leaks $M \oplus M'$

Encrypting same message twice leaks the fact

K must be as large as message message length revealed

Message length revealed

Cryptography as computational science

Use computational intractability as basis for confidence in systems

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

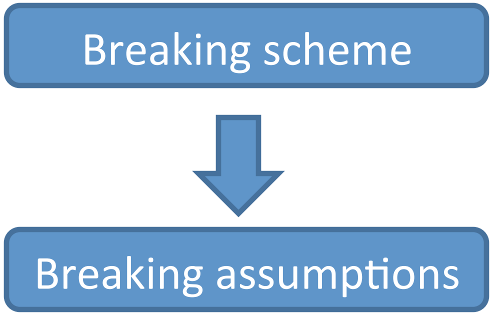
Goldwasser, Micali and Blum circa 1980's

Formal definitions

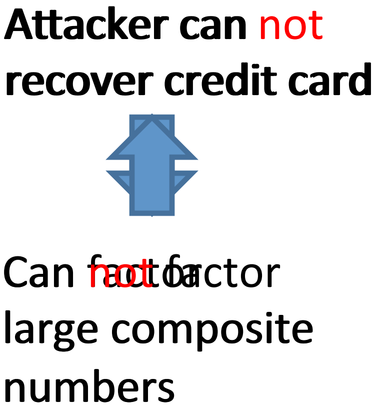
Scheme semantics

Security

Security proofs (reductions)



Example:



Attacker can not recover credit card
Can not factor large composite numbers

As long as assumptions holds
we believe in security of scheme!

But no one knows how to do this. It's been studied for a very long time!

Provable security yields

1) well-defined assumptions and security goals

2) attackers (cryptanalysts) can focus on assumptions

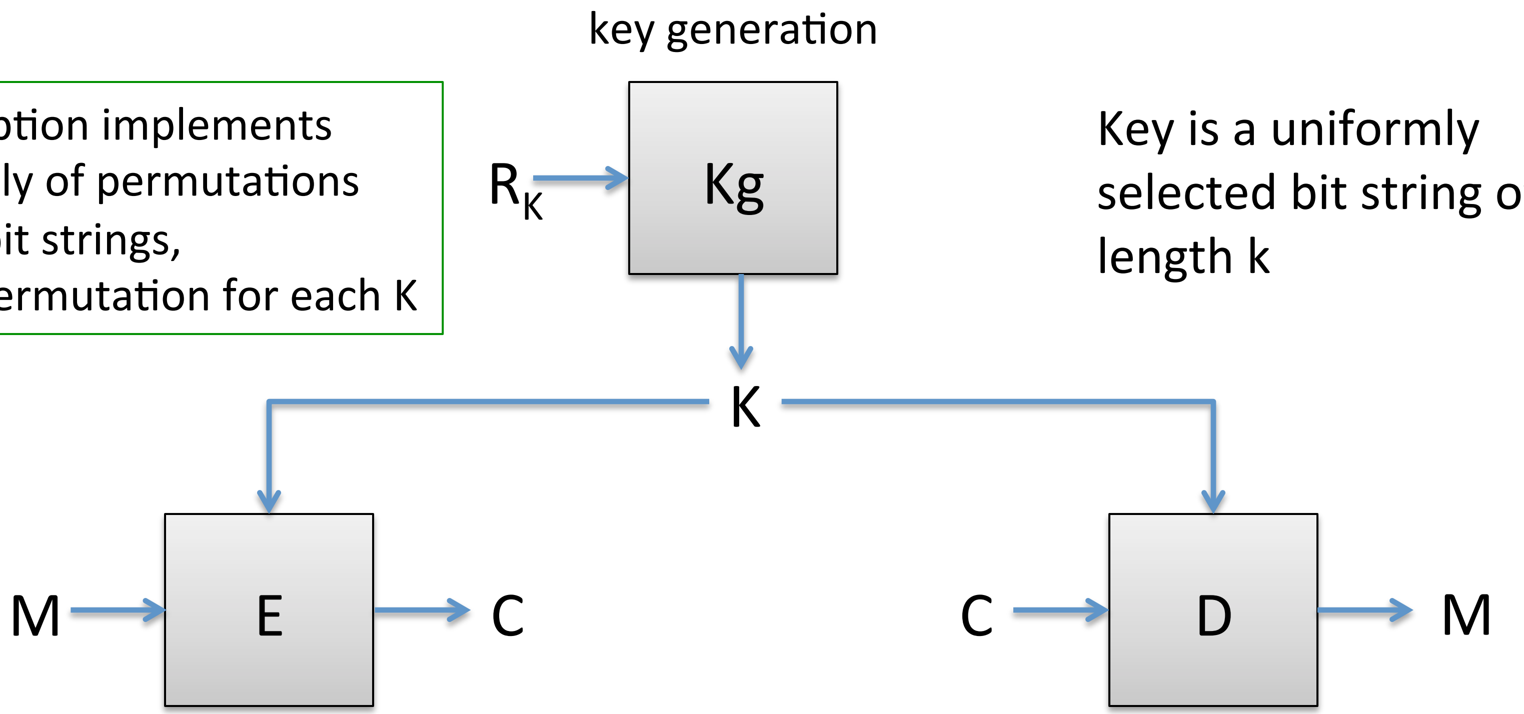
Typical assumptions

- 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

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

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

Data encryption standard (DES)

Originally called Lucifer

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

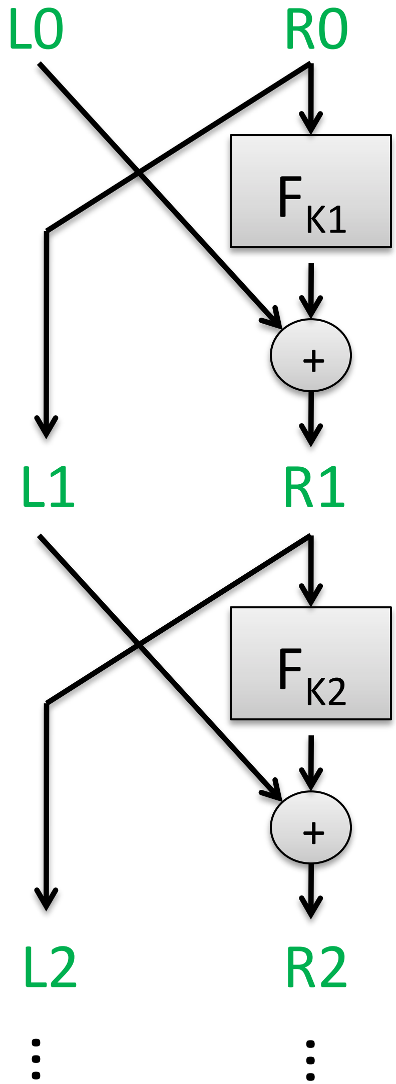
n = 64
k = 56

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.
k=128 has 10 rounds of:

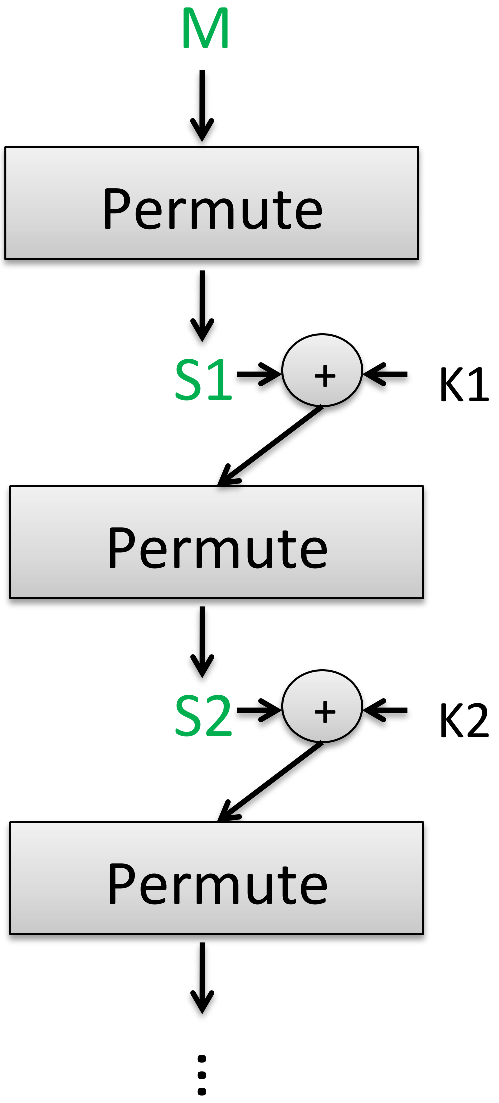
1) Permute:

SubBytes (non-linear S-boxes)

ShiftRows + MixCols (invertible linear transform)

2) XOR in a round key derived from K

(Actually last round skips MixCols)



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