Russian Malware 'Patches' Chrome and Firefox To Fingerprint TLS Traffic (zdnet.com)

13

Posted by EditorDavid on Sunday October 06, 2019 @06:44PM from the browsing-bad dept.

An anonymous reader quotes ZDNet:

> A Russian cyber-espionage hacker group has been spotted using a novel technique that involves patching locally installed browsers like Chrome and Firefox in order to modify the browsers' internal components. The end goal of these modifications is to alter the way the two browsers set up HTTPS connections, and add a per-victim fingerprint for the TLS-encrypted web traffic that originates from the infected computers...
>
> According to a Kaspersky report published this week, hackers are infecting victims with a remote access trojan named Reductor, through which they are modifying the two browsers. This process involves two steps. They first install their own digital certificates to each infected host. This would allow hackers to intercept any TLS traffic originating from the host. Second, they modify the Chrome and Firefox installation to patch their pseudo-random number generation (PRNG) functions. These functions are used when generating random numbers needed for the process of negotiating and establishing new TLS handshakes for HTTPS connections.
>
> Turla hackers are using these tainted PRNG functions to add a small fingerprint at the start of every new TLS connection.

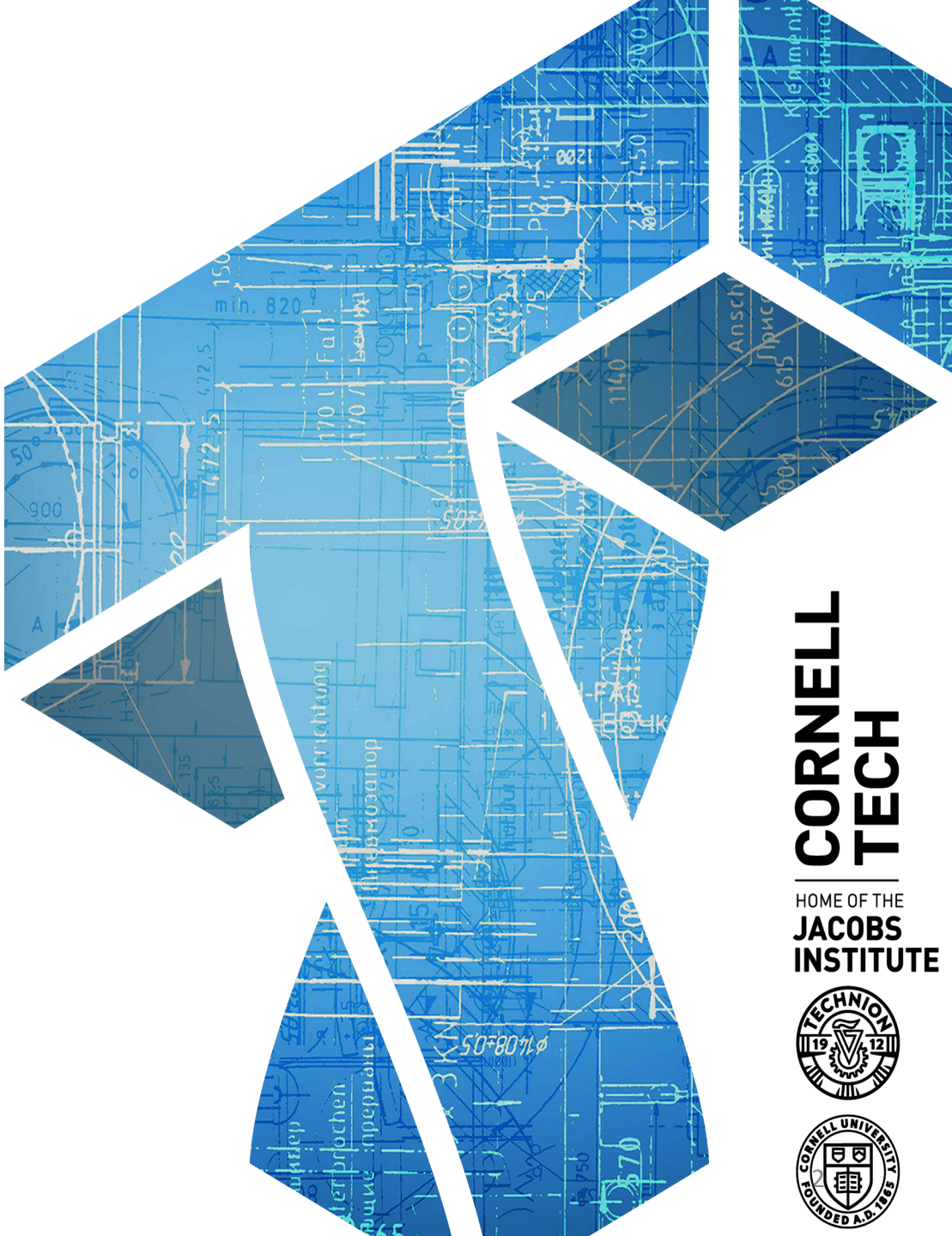
The attack is being attributed to Turla, "a well-known hacker group believed to operate under the protection of the Russian government," ZDNet reports. And though the remote-access trojan already grants full control over a victim's device, one theory is the modified browsers offer "a secondary surveillance mechanism" if that trojan was discovered and removed. Researchers believe the malware is installed during file transfers over HTTP connections, suggesting an ISP had been compromised, according to the article.

"A January 2018 report from fellow cyber-security firm ESET revealed that Turla had compromised at least four ISPs before, in Eastern Europe and the former Soviet space, also with the purpose of tainting downloads and adding malware to legitimate files."

CS 5435: Cryptography

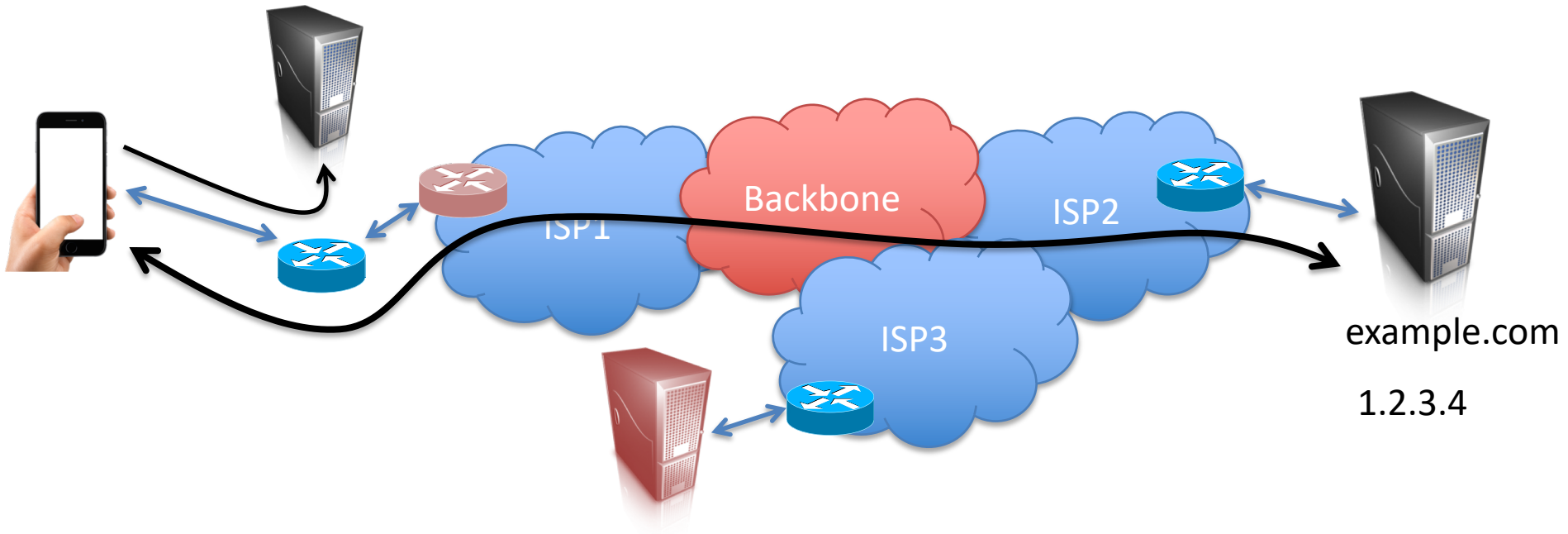
Instructor: Tom Ristenpart

https://github.com/tomrist/cs5435-fall2019



CORNELL TECH
HOME OF THE JACOBS INSTITUTE

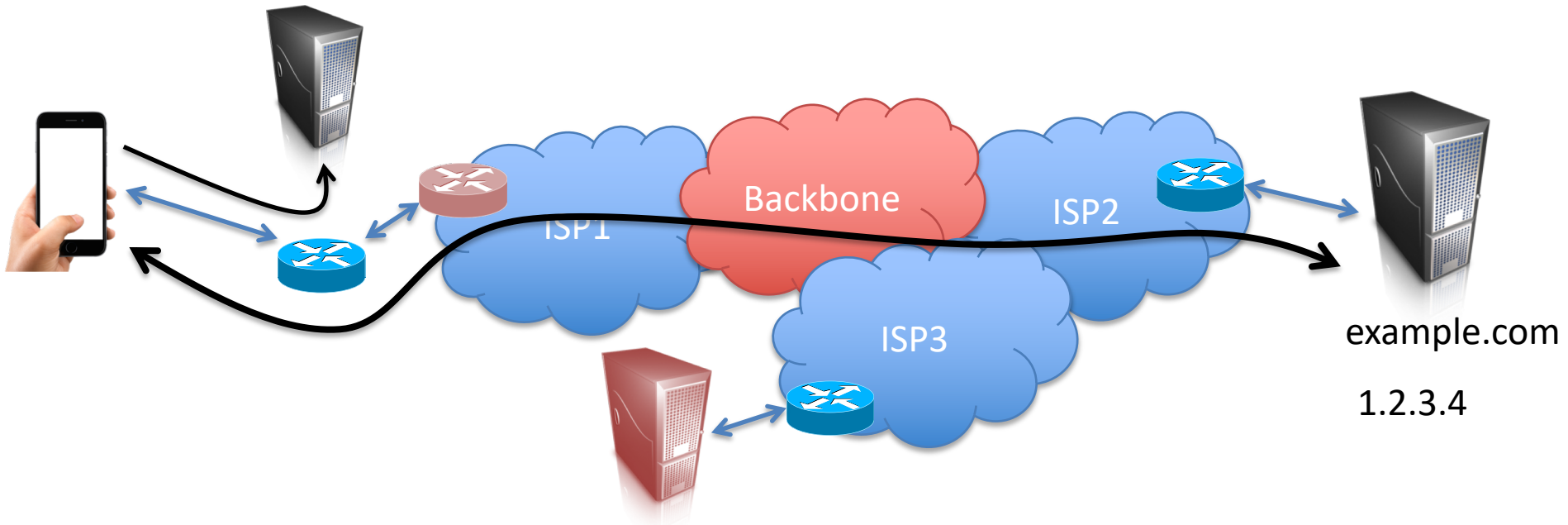
Network threat models



(1) Malicious hosts } Off-path attackers

On-path attackers {
(2) Subverted routers or links
(3) Malicious ISPs or backbone

Network threat models



Client and server want to communicate securely:

- **Confidentiality** (messages are private)
- **Integrity** (accepted messages are as sent)
- **Authenticity** (is it really example.com?)
- Sometimes: anonymity (hide identities)
- Sometimes: steganography (hide that communication took place)

TLS, SSH, IPsec, PGP

Cryptography: “Hidden writing”

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

- 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"

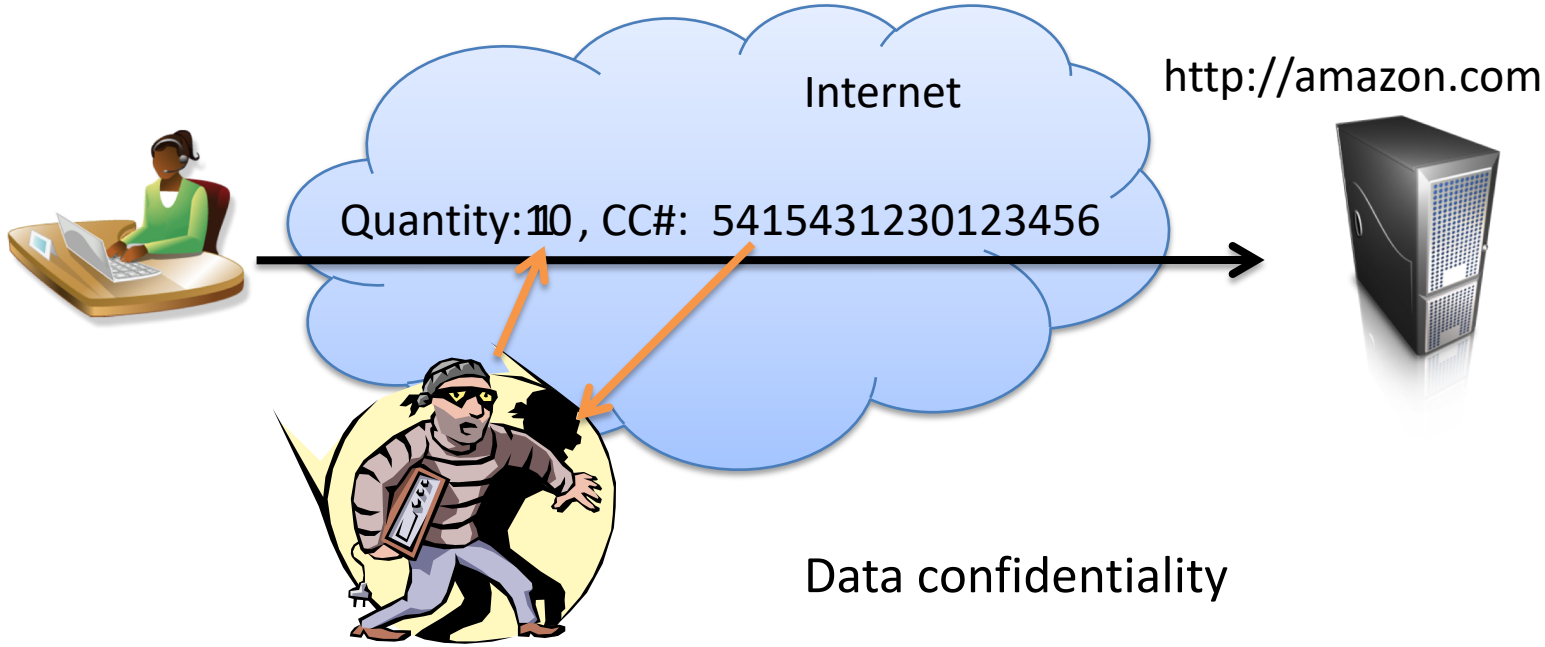
A cryptosystem should be secure even if its algorithms, implementations, configuration, etc. is made public --- the only secret should be a key

Why?

Some cryptographic primitives

- Symmetric cryptography (shared key K)
  - encryption & decryption using K
  - message authentication using K
  - pseudorandom functions (PRF)
- 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 with TLS

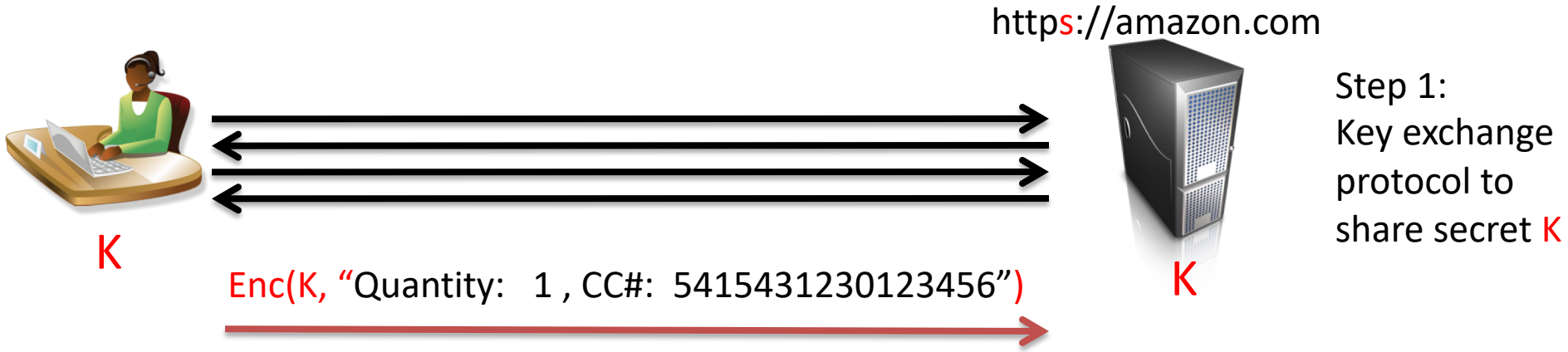


Data confidentiality

Data integrity

We need secure channels for transmitting data

An example: On-line shopping with TLS



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 1.2 handshake (key transport)

Client

Server

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 server, signature over it)

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

C →

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

TLS 1.2 Record layer

Client

Server

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

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

C1 <- Enc(K1,Message)

C1

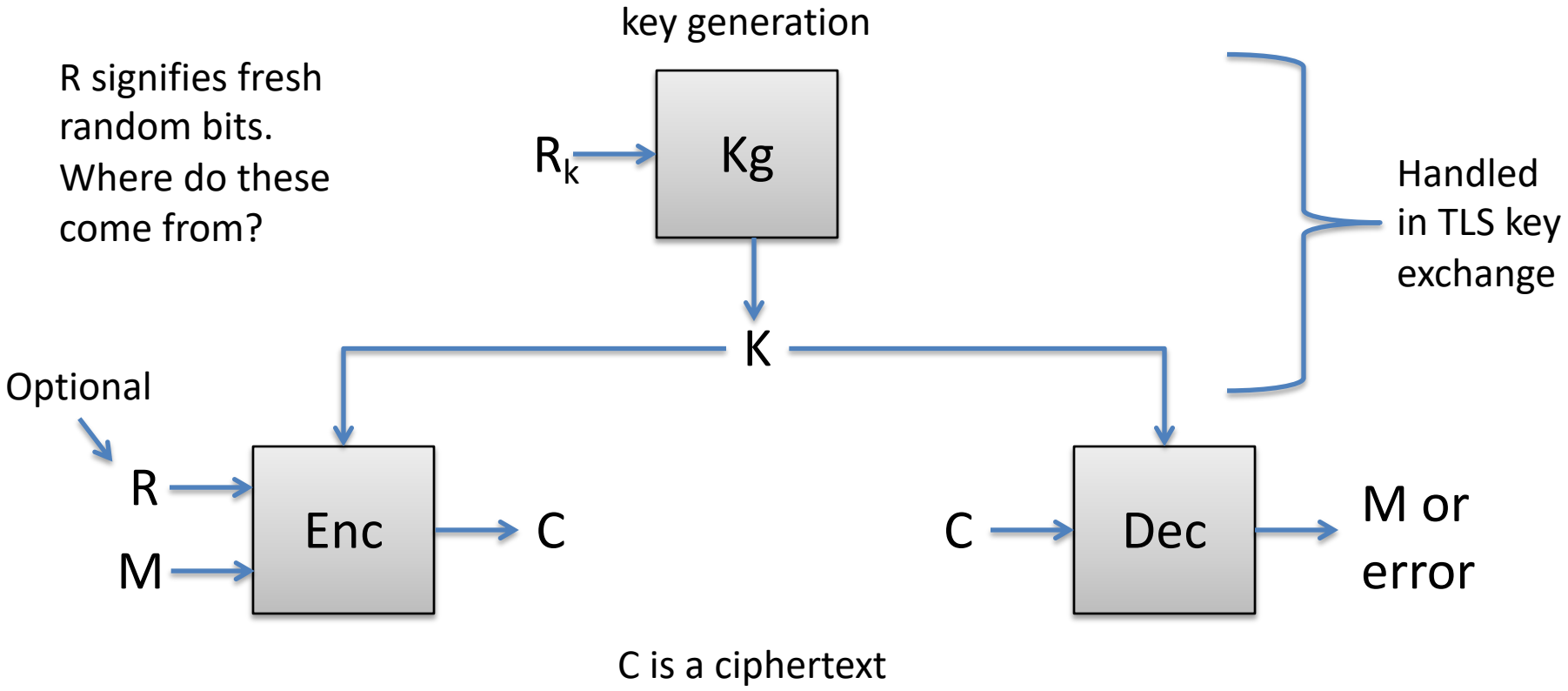
Message <- Dec(K1,C1)

C2 <- Enc(K2,Message’)

C2

Message’ <- Dec(K2,C2)

Symmetric encryption



Correctness: Dec(K , Enc(K,R,M)) = M with probability 1 over randomness R used

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

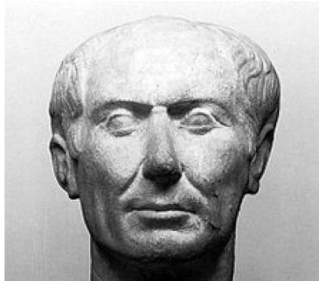
Some attack settings, high level

Given ciphertext C of unknown plaintext, want to recover (information about) plaintext

- Unknown plaintext
  - attacker only sees some ciphertexts
- Known plaintext
  - Attacker gets some plaintext-ciphertext pairs
- Chosen plaintext
  - Attacker can see encryption examples of chosen plaintexts
- Chosen plaintext & ciphertext
  - Attacker can see encryption/decryption examples of chosen plaintexts/ciphertexts

Substitution ciphers

Julius Caeser



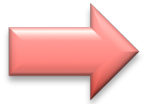
Kg: output randomly chosen permutation of digits

K =

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

WWII Enigma machine built by Germans

Polyalphabetic substitution cipher

- Substitution table changes from character to character
- Rotors control substitutions

Allies broke Enigma (even before the war), significant intelligence impact

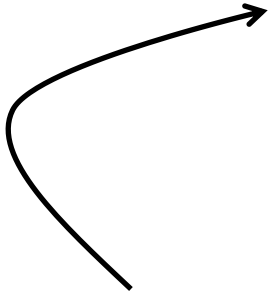
Computers were built to break WWII ciphers, by Alan Turing and others

More modern approach to 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 notions

Security proofs

Show it is mathematically impossible to break security

Analogous to (mathematical or physics) models

- Necessarily abstract view of computers & adversarial capabilities
- We can prove things, but only relative to model
- Must interpret results via utility of model to security in applications

One-time pads

Fix some message length L

Kg: output random bit string K of length L

$$Enc(K,M) = M \oplus K \qquad Dec(K,C) = C \oplus K$$

Exclusive OR operation

Shannon's security notion

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

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

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

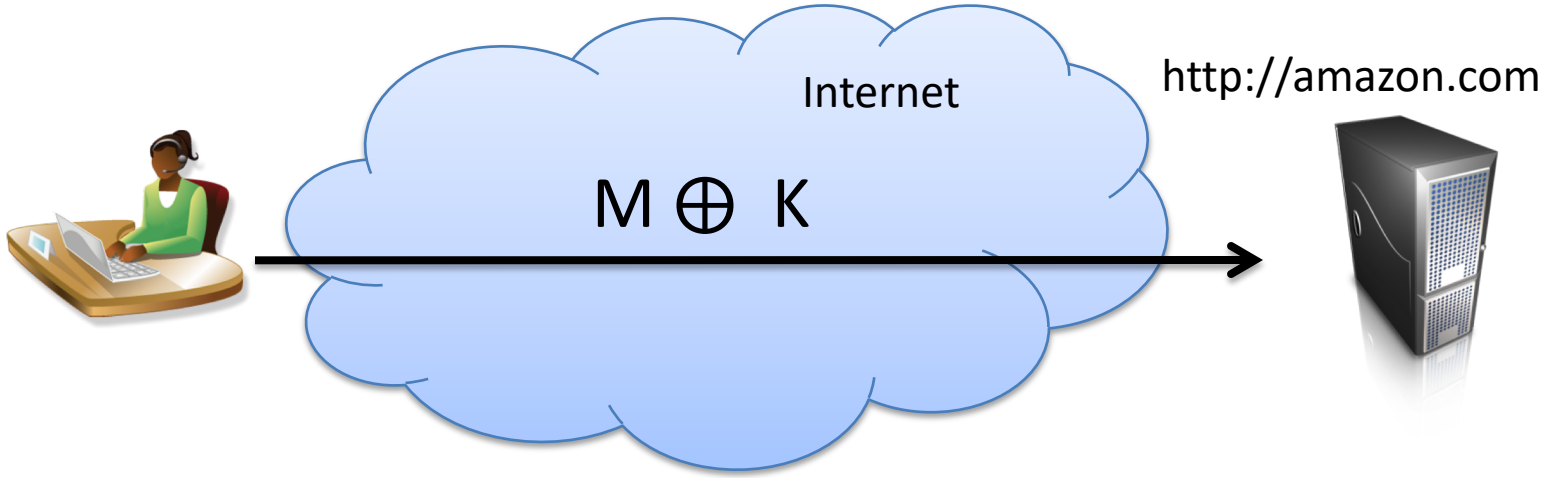
Def. A symmetric encryption scheme is perfectly secure if for all messages M,M' and ciphertexts C

$$\Pr[\,\mathrm{Enc}(K,M) = C\,] = \Pr[\,\mathrm{Enc}(K,M') = C\,]$$

where probabilities are over choice of K

Thm. OTP is perfectly secure

Thm. To be perfectly secure, encryption key must be as long as message



In-class exercise:
OTP is perfectly secure. Does OTP suffice for protecting Internet communications?

Integrity easily violated

Reuse of K for messages M,M' leaks M ⊕ M'

Encrypting same message twice under K leaks the message equality

K must be as large as message

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)

Breaking scheme

Breaking assumptions

Example:

Attacker can not recover credit card

Can not factor large composite numbers

As long as assumptions hold and security model is good we believe in security of scheme!

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

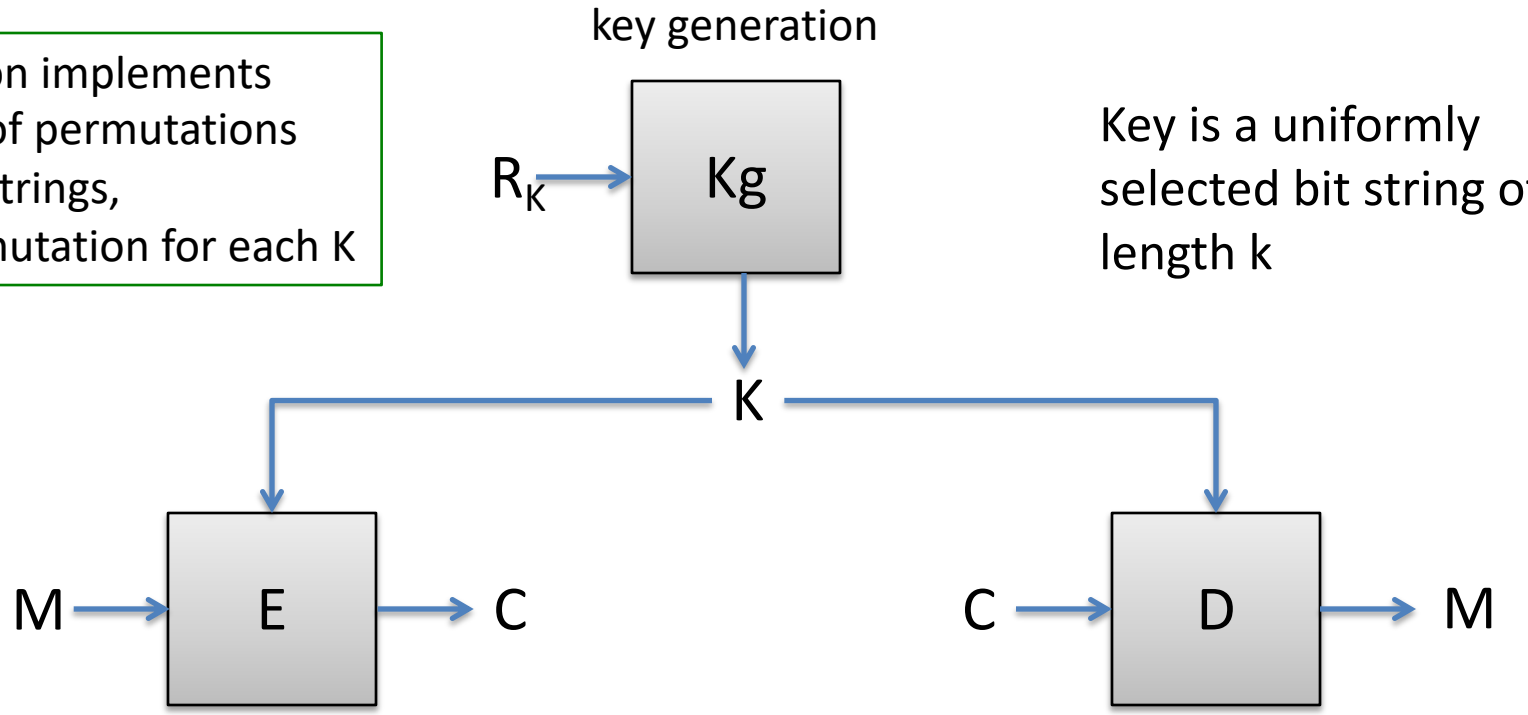
Typical assumptions

- Basic primitives that we believe are hard to break:
  - Large composites hard to factor
  - RSA permutation hard-to-invert
  - Discrete log of elliptic curve group points hard to recover
  - Block ciphers (AES, DES) are good pseudorandom permutations (PRPs)
  - Hash functions are collision resistant

Confidence in low-level primitives gained by cryptanalysis, public design competitions (AES & SHA-3 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

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

Data encryption standard (DES)

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

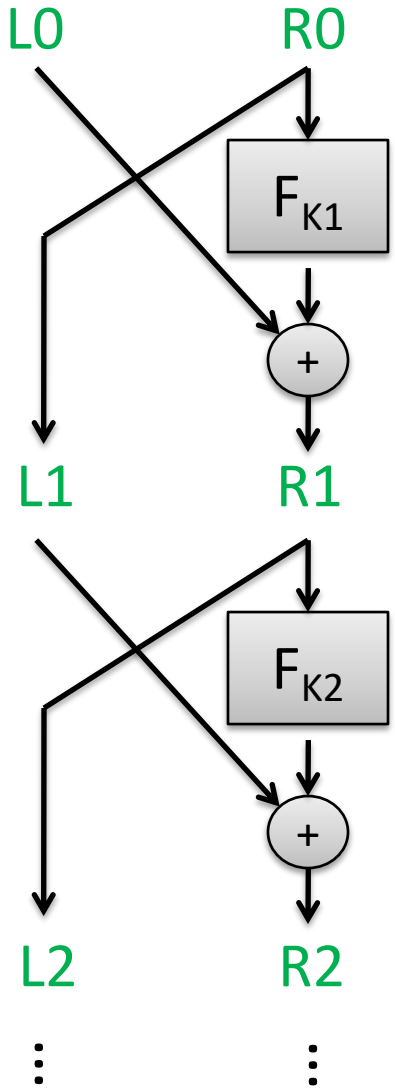
n = 64
k = 56

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

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

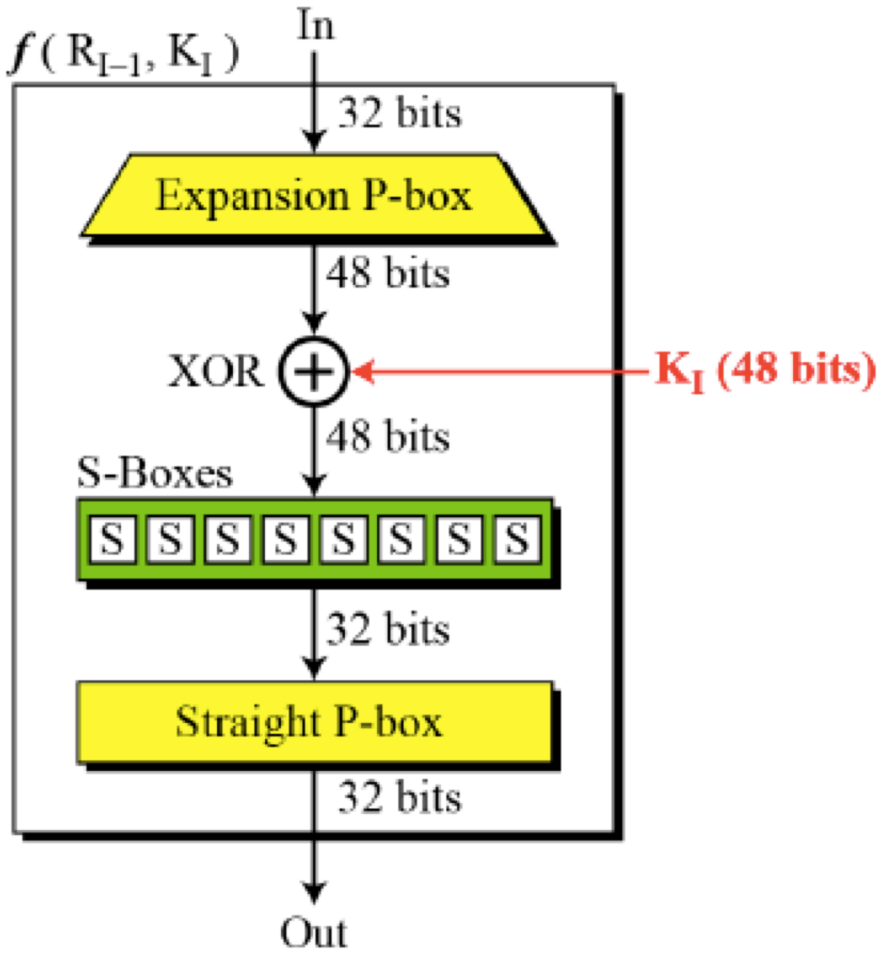
Repeat Feistel round 16 times

Each round applies function F using separate round key



DES round functions

- P-box expands 32 bits to 48 bits and permutes
- S-boxes: 6-bit to 4-bit lookup tables
- XOR in round key
  - 16 48-bit round keys derived via key schedule from 56 bit key deterministically
- How S-boxes chosen? Why particular permutations?
  - Resist cryptanalytic attacks known to NSA at the time (discovered later in 1990s)
  - Differential cryptanalysis



Best attacks against DES

Attack	Attack type	Complexity	Year
Biham, Shamir	Chosen plaintexts, recovers key	2^{47} plaintext, ciphertext pairs	1992
Matsui	Known plaintext, ciphertext pairs, recovers key	2^{42} plaintext, ciphertext pairs, ~2^{41} DES computations	1993
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)

A form of key-alternating cipher

n = 128

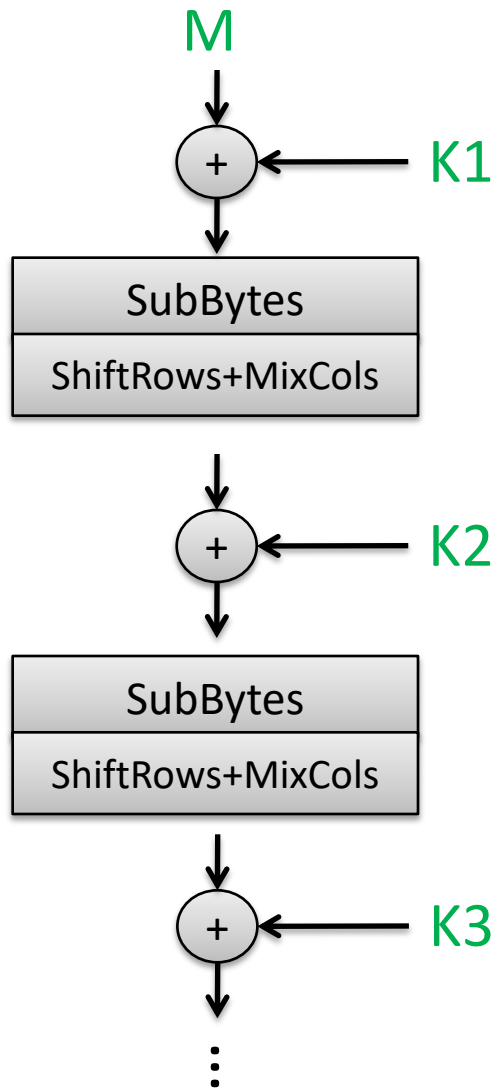
k = 128, 192, 256

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) SubBytes (non-linear 8-bit S-boxes)
2) ShiftRows & MixCols (linear permutation)
3) XOR'ing in a round key

(Last round skips MixCols)

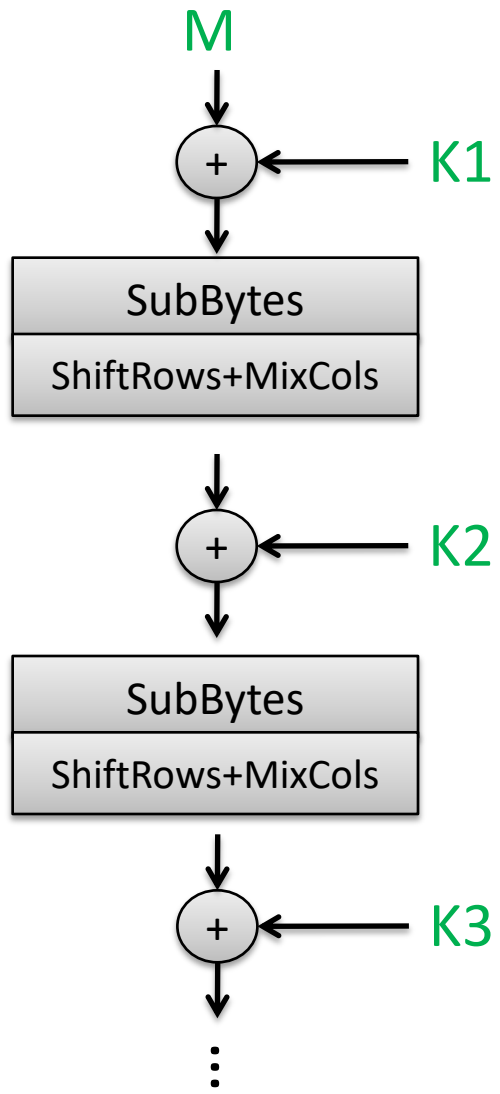


Advanced Encryption Standard (AES)

Designed to resist linear & differential cryptanalysis

"Wide-trail" strategy

- Ensure large # of Sboxes involved in multi-round trail (sequence of intermediate state bits)
- Use coding theory viewpoint to build permutations to ensure rapid ***diffusion***



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 at most 2^{128}
- Approximately factor 4 speedup

Summary and next time

- Crypto as computational science
- Overview of TLS
- Symmetric encryption and block ciphers introduced
- Next time:
  - Symmetric encryption from block ciphers
  - Need for active attack security