Network security and all iLabs

Modern cryptography for communications security part 1

Benjamin Hof
hof@in.tum.de

Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München

Cryptography – 16ws

Outline

Cryptography

Symmetric setting

Outline

Cryptography

Symmetric setting

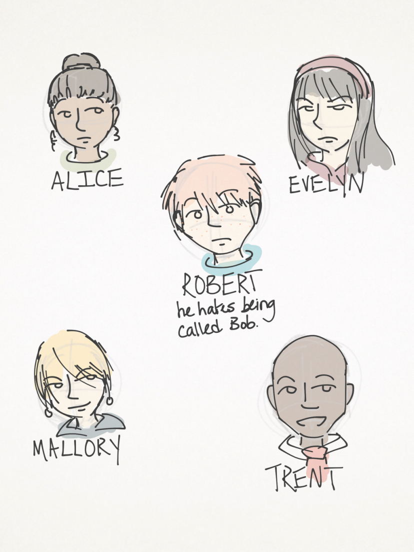
Scope

Focus on:

- modern cryptography
- methods used in communications security

Based on: Introduction to modern cryptography, Katz and Lindell, 2^{nd} edition, 2015.

Communication



by Melissa Elliott

https://twitter.com/0xabad1dea/status/400676797874208768

What we are concerned with

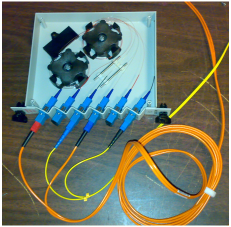
Alice —— “Let’s meet up at 9!” ——→ Bob

What we are concerned with

Alice —— "Let's meet up at 9!" ——→ Bob

↑

BfV



Roens/Wikipedia. CC-by-sa 2.0

What we are concerned with

passive attack: eavesdropping
We want to provide confidentiality!

What we are concerned with

active attack: message modification or forgery
We want to provide message authentication!

Limitations

- cryptography is typically bypassed, not broken
- not applied correctly
- not implemented correctly
- subverted

No protection of information *about* the communication.

- existence
- time
- extent
- partners

Kerckhoffs' principle

Security should only depend on secrecy of the key, not the secrecy of the system.

- key easier to keep secret
- change
- compatibility

No security by obscurity.

- scrutiny
- standards
- reverse engineering

Another principle as a side note

The system should be usable easily.

- Kerckhoffs actually postulated 6 principles
- this one got somewhat forgotten
- considered uncontroversial by Kerckhoffs
- starting to be rediscovered in design of secure applications and libraries

Example

Signal, NaCl

What should secure encryption guarantee?

It should be impossible for the attacker to

What should secure encryption guarantee?

It should be impossible for the attacker to

- recover the key.
- recover the entire plaintext from the ciphertext.
- recover any character of the plaintext from the ciphertext.

What should secure encryption guarantee?

It should be impossible for the attacker to

- recover the key.
- recover the entire plaintext from the ciphertext.
- recover any character of the plaintext from the ciphertext.

Regardless of any information an attacker already has, a ciphertext should leak no additional information about the underlying plaintext.

Modern cryptography

relies on

- formal definitions
- precisely defined assumptions
- mathematical proofs

Reductionist security arguments, the proofs, require to formulate assumptions explicitly.

A definition of security

A scheme is secure, if any *probabilistic polynomial time* adversary succeeds in breaking the scheme with at most *negligible* probability.

Negligible

For every polynomial p and for all sufficiently large values of n:

$$f(n) < \frac{1}{p(n)}$$

e.g., $f(n) = \frac{1}{2^n}$

Church-Turing Hypothesis

We believe polynomial time models all computers.

Our goals

symmetric (secret-key)

- confidentiality
- authenticity
  (as in: message integrity)

asymmetric (public-key)

- confidentiality
- authenticity
- key exchange

Something providing confidentiality generally makes no statement whatsoever about authenticity.

Motivation

What does a perfectly encrypted message look like?

Uniform distribution

$$P : U \to [0, 1]$$

$$\sum_{x \in U} P(x) = 1$$

$$\forall x \in U : P(x) = \frac{1}{|U|}$$

Randomness

- required to do any cryptography at all
- somewhat difficult to get in a computer (deterministic!)
- required to be cryptographically secure: indistiguishable from truly random
- not provided in programming languages

Example

used to generate keys or other information unkown to any other parties

Collecting unpredictable bits

- physical phenomena
  - time between emission of particles during radioactive decay
  - thermal noise from a semiconductor diode or resistor
- software-based
  - elapsed time between keystrokes or mouse movement
  - packet interarrival times
- attacker must not be able to guess/influence the collected values

1. collect pool of high-entropy data
2. process into sequence of nearly independent and unbiased bits

Pseudo-random generator

$$G : \{0,1\}^s \to \{0,1\}^n, \quad n \gg s$$

Outline

Symmetric encryption scheme

1. $k \leftarrow Gen(1^n)$, security parameter 1^n
2. $c \leftarrow Enc_k(m), m \in \{0,1\}^*$
3. $m := Dec_k(c)$

- provide confidentiality
- definition of security: chosen-plaintext attack (CPA)

Cryptography uses theoretical attack games to analyze and formalize security.

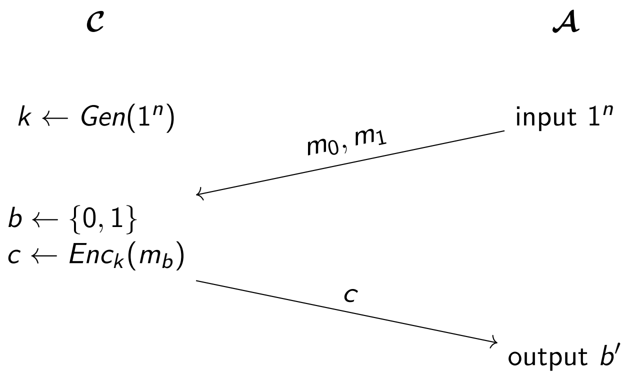
$\mathcal{C}$: challenger,
$\mathcal{A}$: adversary

$\leftarrow$ means non-deterministic,
$:=$ means deterministic

The eavesdropping experiment

$\mathcal{C}$	$\mathcal{A}$
$k \leftarrow Gen(1^n)$	input 1^n

The eavesdropping experiment



- $\mathcal{A}$ succeeds, iff $b = b'$

Discussion of the eavesdropping experiment

- $|m_0| = |m_1|$
- probabilistic polynomial time algorithms
- success probability should be $0.5 + \textit{negligible}$
- if so, *Enc* has indistinguishable encryptions in the presence of an eavesdropper

Pseudorandom permutation

$$F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$$

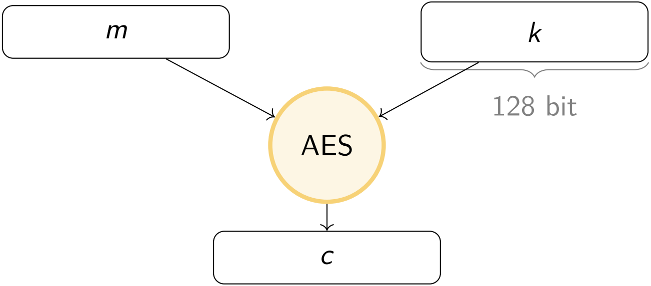
- $F_k(x)$ and $F_k^{-1}(y)$ efficiently computable
- F_k be indistinguishable from uniform permutation
- adversary may have access to F^{-1}

We can assume that all inputs and the output have the same length.

A block cipher

Example

- fixed key length and block length
- chop m into 128 bit blocks

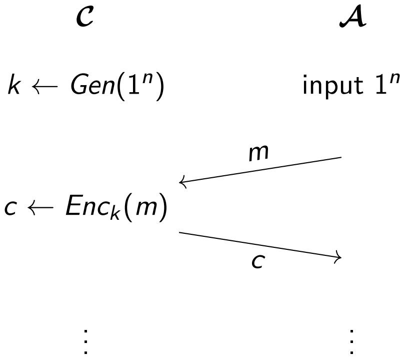


Does this function survive the eavesdropping experiment?

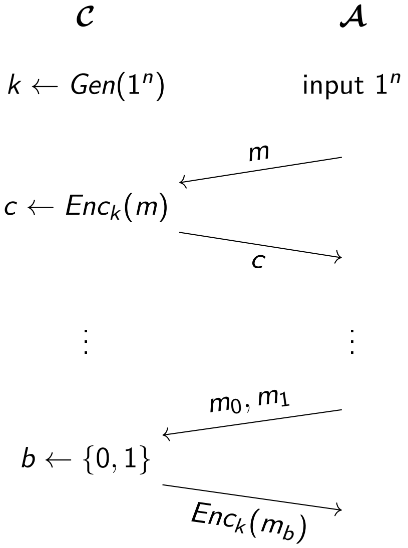
Chosen-plaintext attack

$\mathcal{C}$	$\mathcal{A}$
$k \leftarrow Gen(1^n)$	input 1^n

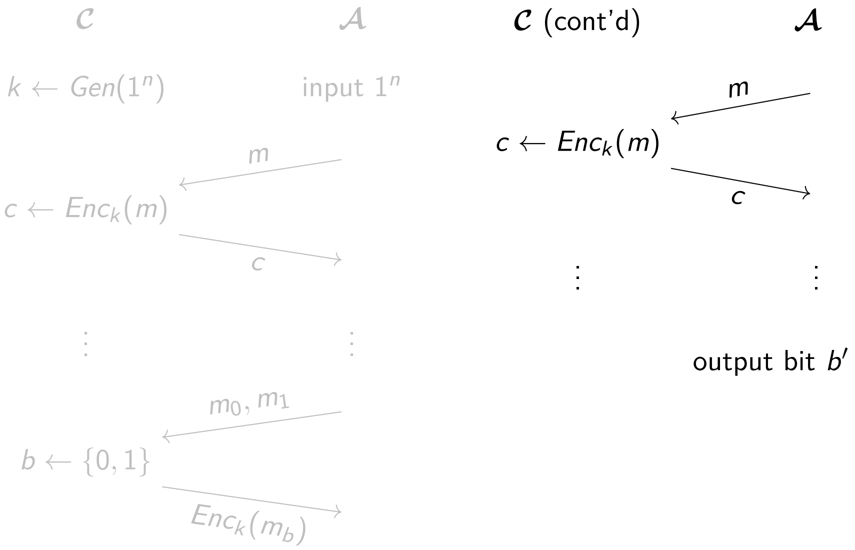
Chosen-plaintext attack



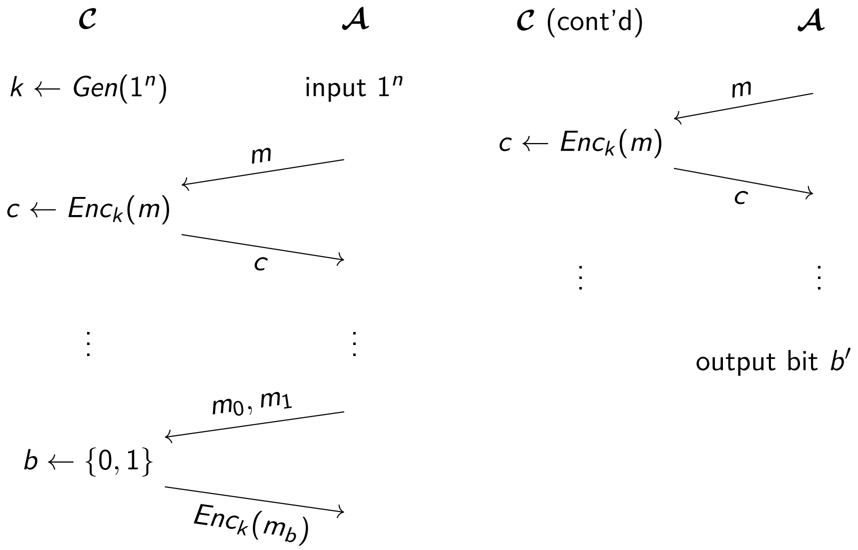
Chosen-plaintext attack



Chosen-plaintext attack



Chosen-plaintext attack



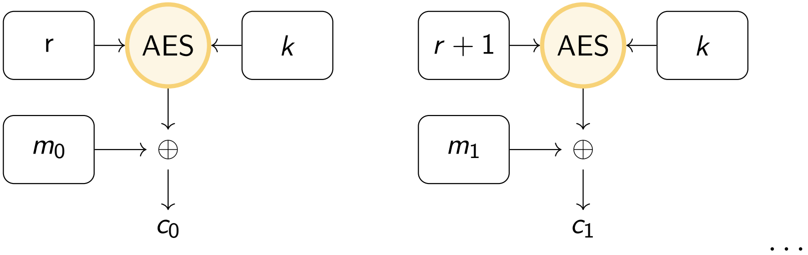
Discussion of CPA

- *Enc* is secure under chosen-plaintext attack
- again, messages must have same length
- multiple-use key
- non-deterministic (e. g. random initialization vector) or state
- block cipher requires *operation mode*, e. g.: counter (CTR), output-feedback (OFB), ...

Example constructions: counter mode

Example

- randomised AES counter mode (AES-CTR$)
- choose nonce $r \leftarrow \{0,1\}^{128}$, key $k \leftarrow \{0,1\}^{128}$
- great if you have dedicated circuits for AES, else vulnerable to timing attacks

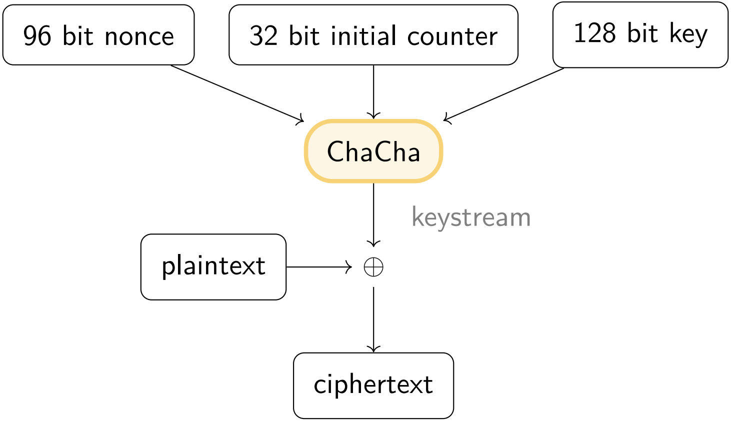


complete ciphertext $c := (r, c_0, c_1, \cdots)$

Example constructions: stream ciphers

Example

A modern stream cipher, fast in software:



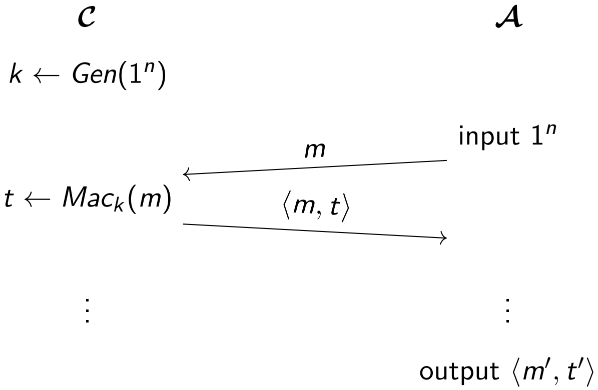
Message authentication code (MAC)

1. $k \leftarrow Gen(1^n)$, security parameter 1^n
2. $t \leftarrow Mac_k(m), m \in \{0,1\}^*$
3. $b := Vrfy_k(m,t)$

$b = 1$ means valid, $b = 0$ invalid

- transmit $\langle m, t \rangle$
- tag t is a short authenticator
- message authenticity $\Leftrightarrow$ integrity
- detect tampering
- no protection against replay
- "existentially unforgeable"
- security definition: adaptive chosen-message attack

Adaptive chosen-message attack



- let $\mathcal{Q}$ be the set of all queries m
- $\mathcal{A}$ succeeds, iff $Vrfy_k(m', t') = 1$ and $m' \notin \mathcal{Q}$

Used in practice

Example

- HMAC based on hash functions
- CMAC based on cipher block chaining mode (CBC)
- authenticated encryption modes

Example: side-channel attack

How does tag verification work and how to implement tag comparison correctly?

Recap: secret-key cryptography

- attacker power: probabilistic polynomial time
- confidentiality defined as IND-CPA:
  encryption, e. g. AES-CTR$
- message authentication defined as existentially unforgeable under adaptive chosen-message attack:
  message authentication codes, e. g. HMAC-SHA2
- authenticated encryption modes

Combining confidentiality and authentication

- encrypt-then-authenticate is generally secure:
  $c \leftarrow Enc_{k_1}(m), t \leftarrow Mac_{k_2}(c)$
  transmit: $\langle c, t \rangle$
- authenticated encryption is also a good choice:
  e. g. offset codebook (OCB), Galois counter mode (GCM)
  $c, t \leftarrow AEAD_k^{enc}(ad, m)$
  $m := AEAD_k^{dec}(ad, c, t)$ or verification failure