Network Security

Modern cryptography for communications security

Benjamin Hof
hof@in.tum.de

Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München

Cryptography – 15ws

Outline

Cryptography

Private-key setting

Outline

Cryptography

Private-key setting

Scope

Focus on:

- modern cryptography
- methods used in communications security

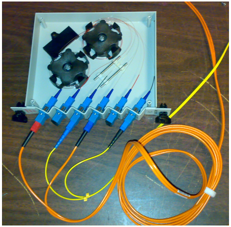
Based on: Introduction to modern cryptography, Katz and Lindell, 2^{nd} edition, 2015.

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



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
We want to provide message authentication!

Limitations

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

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
- starting to be rediscovered in design of secure applications and libraries

Example

Signal, NaCl

Modern cryptography

relies on

- formal definitions
- precisely defined assumptions
- mathematical proofs

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

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

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

Pseudo-random generator

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

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

private-key (symmetric)

- confidentiality
- authenticity
  (as in: message integrity)

public-key (asymmetric)

- confidentiality
- authenticity
- key exchange

Something providing confidentiality generally makes no statement whatsoever about authenticity.

Outline

Private-key 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.

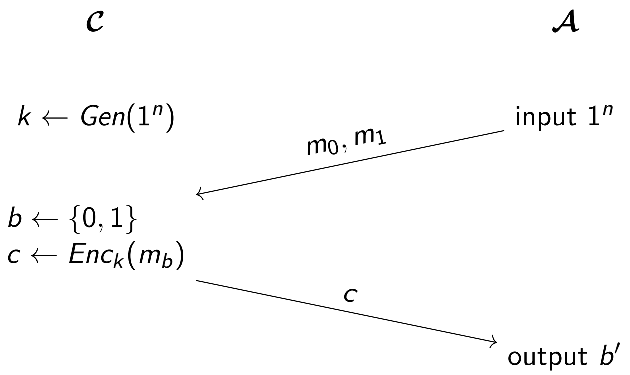
$\mathcal{C}$: challenger,
$\mathcal{A}$: adversary

$\leftarrow$ 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

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

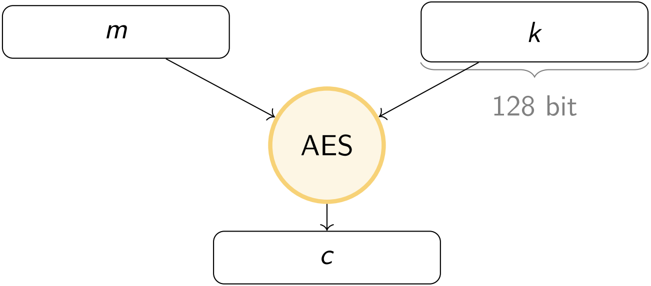
- $F_k(x)$ and $F_k^{-1}(y)$ efficiently computable
- F_k be indistinguishable from uniform permutation
- adversary may have access to F^{-1}

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

A block cipher

Example

- fixed key lenght 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

$\mathcal{C}$ $\qquad\qquad$ $\mathcal{A}$

$k \leftarrow Gen(1^n)$ $\qquad\qquad$ input 1^n

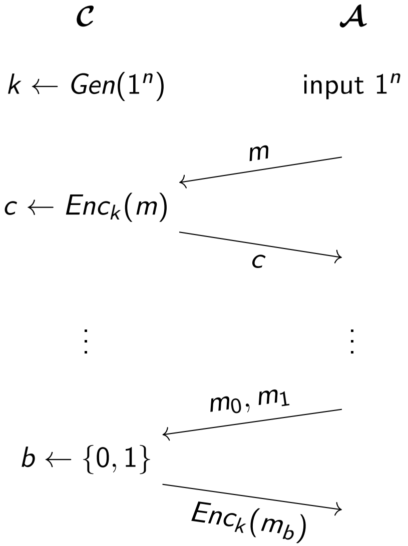
$\xleftarrow{\quad m \quad}$

$c \leftarrow Enc_k(m)$

$\xrightarrow[\quad c \quad]{}$

$\vdots$ $\qquad\qquad$ $\vdots$

Chosen-plaintext attack

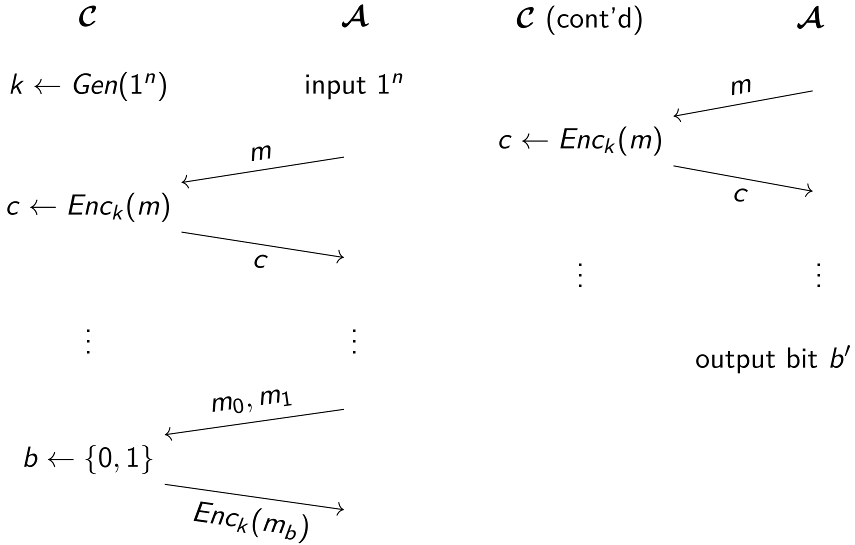


Chosen-plaintext attack

$\mathcal{C}$		$\mathcal{A}$
$k \leftarrow Gen(1^n)$		input 1^n
	$\xleftarrow{\ m\ }$	
$c \leftarrow Enc_k(m)$		
	$\xrightarrow{\ c\ }$	
$\vdots$		$\vdots$
	$\xleftarrow{\ m_0, m_1\ }$	
$b \leftarrow \{0,1\}$		
	$\xrightarrow{\ Enc_k(m_b)\ }$	

$\mathcal{C}$ (cont'd)		$\mathcal{A}$
	$\xleftarrow{\ m\ }$	
$c \leftarrow Enc_k(m)$		
	$\xrightarrow{\ c\ }$	
$\vdots$		$\vdots$
		output bit b'

Chosen-plaintext attack



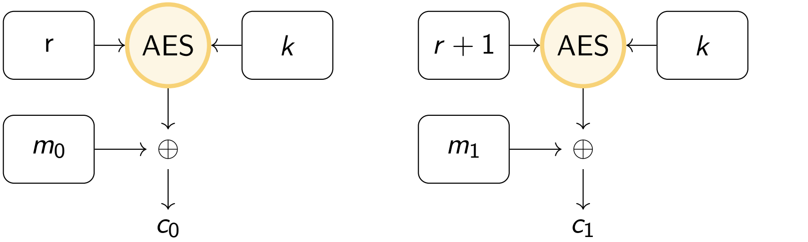
Discussion of CPA

- *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*: 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

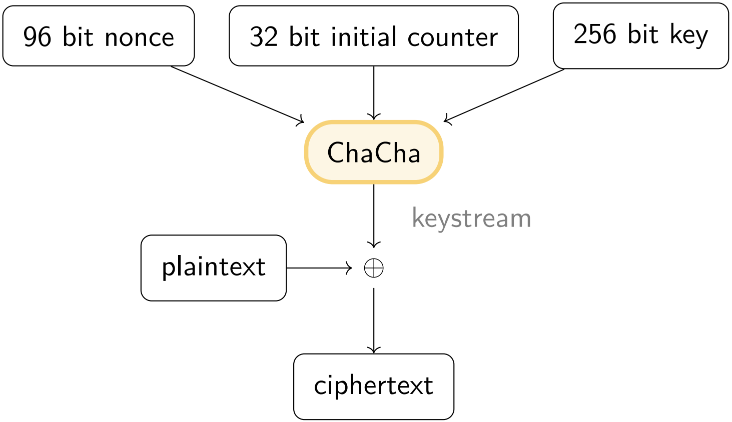


complete ciphertext $c := (r, c_0, c_1, \cdots)$

Example constructions: stream ciphers

Example

A modern stream cipher, fast in software:



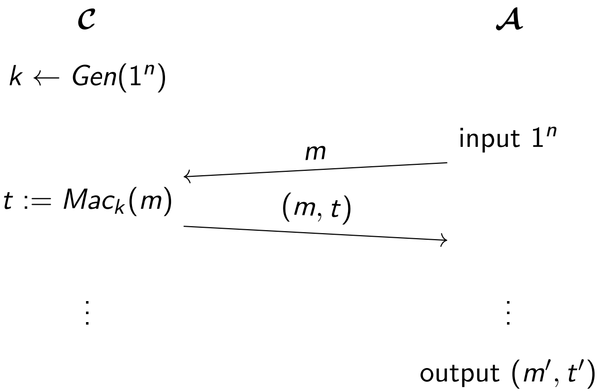
Message authentication code

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 CBC mode
- authenticated encryption modes

Side-channel attacks

How to implement tag comparison correctly?

Cryptographic hash functions

private-key

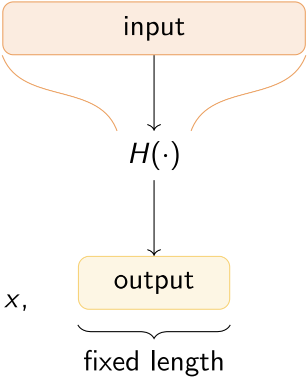
- encryption
- message authentication codes
- *hash functions*

public-key

. . .

Hash functions

- variable length input
- fixed length output



provide:

1. pre-image resistance
   given $H(x)$ with a randomly chosen x,
   cannot find x' s. t. $H(x') = H(x)$
   "H is one-way"
2. second pre-image resistance
   given x, cannot find $x' \neq x$ s. t. $H(x') = H(x)$
3. collision resistance
   cannot find $x \neq x'$ s. t. $H(x) = H(x')$