

The foundations of the Foundations of cryptography

Summer semester 2019

Claus Diem

Some words about complexity theory

Strings and numbers

We use $\{0, 1\}^*$ or **N**:

$$100110 \longleftrightarrow 011001 \longleftrightarrow 1011001 \longleftrightarrow (1011001)_2$$

Strings and numbers

We use $\{0, 1\}^*$ or **N**:

$$100110 \longleftrightarrow 011001 \longleftrightarrow 1011001 \longleftrightarrow (1011001)_2$$

In the following $\{0, 1\}^*$. For a string x we denote by $|x|$ its length.

The “polynomial is fast”-paradigma

- ▶ Only asymptotic statements are made.
- ▶ An algorithm is considered to be **fast** if its running time is polynomially bounded in the input length.

One might say: polynomially bounded running time = qualitatively fast

One says “polynomial running time” instead of polynomially “bounded running time”.

The “polynomial is fast”-paradigma

- ▶ Only asymptotic statements are made.
- ▶ An algorithm is considered to be **fast** if its running time is polynomially bounded in the input length.

One might say: polynomially bounded running time = qualitatively fast

One says “polynomial running time” instead of polynomially “bounded running time”.

Language: $L \subseteq \{0, 1\}^*$, corresponds to a function $f : \{0, 1\}^* \rightarrow \{0, 1\}$.

The “polynomial is fast”-paradigma

- ▶ Only asymptotic statements are made.
- ▶ An algorithm is considered to be **fast** if its running time is polynomially bounded in the input length.

One might say: polynomially bounded running time = qualitatively fast

One says “polynomial running time” instead of polynomially “bounded running time”.

Language: $L \subseteq \{0, 1\}^*$, corresponds to a function

$f : \{0, 1\}^* \rightarrow \{0, 1\}$.

Decision problem for $L \subseteq \{0, 1\}^* / f : \{0, 1\}^* \rightarrow \{0, 1\}$:

The “polynomial is fast”-paradigma

- ▶ Only asymptotic statements are made.
- ▶ An algorithm is considered to be **fast** if its running time is polynomially bounded in the input length.

One might say: polynomially bounded running time = qualitatively fast

One says “polynomial running time” instead of polynomially “bounded running time”.

Language: $L \subseteq \{0, 1\}^*$, corresponds to a function $f : \{0, 1\}^* \rightarrow \{0, 1\}$.

Decision problem for $L \subseteq \{0, 1\}^* / f : \{0, 1\}^* \rightarrow \{0, 1\}$:

Decide if an input $x \in \{0, 1\}^*$ lies in L / if $f(x) = 1$ holds.

Complexity classes

P. Set of languages / 0 – 1-functions / problems decidable in polynomial time by a deterministic Turing machine (TM).

Complexity classes

P. Set of languages / 0 – 1-functions / problems decidable in polynomial time by a deterministic Turing machine (TM).

NP. Set of languages / 0 – 1-functions / problems decidable in polynomial time by a non-deterministic Turing machine.

Complexity classes

P. Set of languages / 0 – 1-functions / problems decidable in polynomial time by a deterministic Turing machine (TM).

NP. Set of languages / 0 – 1-functions / problems decidable in polynomial time by a non-deterministic Turing machine.

BPP. Set of languages / 0 – 1-functions / problems probabilistically decidable with bounded error by a (probabilistic) Turing machine.

The class NP

For $L \subseteq \{0, 1\}^*$:

Possible definitions for $L \in \text{NP}$:

- ▶ There is a non-deterministic TM T with:
 - ▶ T terminates in polynomial time.
 - ▶ For an input x are equivalent:
 - ▶ $x \in L$.
 - ▶ At least one possible output of T applied to x is 1.

The class NP

For $L \subseteq \{0,1\}^*$:

Possible definitions for $L \in \text{NP}$:

- ▶ There is a non-deterministic TM T with:
 - ▶ T terminates in polynomial time.
 - ▶ For an input x are equivalent:
 - ▶ $x \in L$.
 - ▶ At least one possible output of T applied to x is 1.
- ▶ There is a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, a DTM T and a positive polynomial $p(n)$ with:
 - ▶ T computes R : $T(x, y) = 1 \leftrightarrow x \sim_R y$ (i.e., $(x, y) \in R$)
 - ▶ $x \in L$ if and only if there is a y with $|y| \leq p(|x|)$ and $x \sim_R y$ (i.e., $(x, y) \in R$).
 - ▶ T terminates in polynomial time.

The class NP

For $L \subseteq \{0,1\}^*$:

Possible definitions for $L \in \text{NP}$:

- ▶ There is a non-deterministic TM T with:
 - ▶ T terminates in polynomial time.
 - ▶ For an input x are equivalent:
 - ▶ $x \in L$.
 - ▶ At least one possible output of T applied to x is 1.
- ▶ There is a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, a DTM T and a positive polynomial $p(n)$ with:
 - ▶ T computes R : $T(x, y) = 1 \leftrightarrow x \sim_R y$ (i.e., $(x, y) \in R$)
 - ▶ $x \in L$ if and only if there is a y with $|y| \leq p(|x|)$ and $x \sim_R y$ (i.e., $(x, y) \in R$).
 - ▶ T terminates in polynomial time.

Given x , a y with $x \sim_R y$ is called a **witness** or a **proof**.

The class BPP

For $L \subseteq \{0,1\}^*$:

The class BPP

For $L \subseteq \{0,1\}^*$:

Definition of $L \in \text{BPP}$:

The class BPP

For $L \subseteq \{0,1\}^*$:

Definition of $L \in \text{BPP}$:

There is a probabilistic TM T with:

- ▶ T terminates in polynomial time.
- ▶ For $x \in L$, T outputs 1 with a probability of $\geq \frac{2}{3}$.
- ▶ For $x \notin L$, T outputs 0 with a probability of $\leq \frac{1}{3}$.

The class BPP

For $L \subseteq \{0,1\}^*$:

Definition of $L \in \text{BPP}$:

There is a probabilistic TM T with:

- ▶ T terminates in polynomial time.
- ▶ For $x \in L$, T outputs 1 with a probability of $\geq \frac{2}{3}$.
- ▶ For $x \notin L$, T outputs 0 with a probability of $\leq \frac{1}{3}$.

Note: This must hold for all x !

The class BPP

For $L \subseteq \{0,1\}^*$:

Definition of $L \in \text{BPP}$:

There is a probabilistic TM T with:

- ▶ T terminates in polynomial time.
- ▶ For $x \in L$, T outputs 1 with a probability of $\geq \frac{2}{3}$.
- ▶ For $x \notin L$, T outputs 0 with a probability of $\leq \frac{1}{3}$.

Note: This must hold for all x !

So: The error probability is always $\leq \frac{1}{3}$. This can be substituted by any bound $c > 0$.

Complexity classes

It is obviously $P \subseteq NP$, $P \subseteq BPP$.

Complexity classes

It is obviously $P \subseteq NP$, $P \subseteq BPP$.

All other relationships are unknown.

Complexity classes

It is obviously $P \subseteq NP$, $P \subseteq BPP$.

All other relationships are unknown.

Is $P = NP$, $P = BPP$, $NP \subseteq BPP$, $BPP \subseteq NP$?

The notion of algorithm

In the following:

Algorithm = (randomized) Turing machine or
= informel description of a computation.

The notion of algorithm

In the following:

Algorithm = (randomized) Turing machine or
= informel description of a computation.

Attacker are modeled with (randomized) algorithms.

Fast attackers are polynomial time algorithms (also called
PPT-algorithms)

Negligible

For cryptographic protocols,
we define rigorous notions of “secure”.

Negligible

For cryptographic protocols,
we define rigorous notions of “secure”.

This is always defined like this:

For a fast attackers (= polynomial time Turing machines) some computational goal (**details!**) is only achieved with a negligible probability.

Negligible

For cryptographic protocols,
we define rigorous notions of “secure”.

This is always defined like this:

For a fast attackers (= polynomial time Turing machines) some computational goal (**details!**) is only achieved with a negligible probability.

Motivated by polynomial time paradigm:

Definition. A function $\epsilon : \mathbf{N} \rightarrow \mathbf{R}$ is called **negligible**, if for every $e > 0$ it holds:

$$|\epsilon(n)| \leq \frac{1}{n^e}$$

for all n with $n \gg 0$.

One-way functions

One-way functions

Let $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ be efficiently computable, that is, in polynomial time on a DTM.

One-way functions

Let $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ be efficiently computable, that is, in polynomial time on a DTM.

We want that preimages are hard to compute.

One-way functions

Let $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ be efficiently computable, that is, in polynomial time on a DTM.

We want that preimages are hard to compute.

Let n be the input length.

One-way functions

Let $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ be efficiently computable, that is, in polynomial time on a DTM.

We want that preimages are hard to compute.

Let n be the input length.

Idea. The portion of $x \in \{0, 1\}^n$ for which given $f(x)$ one can compute efficiently some x' with $f(x) = f(x')$ is negligible.

One-way functions

Let $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ be efficiently computable, that is, in polynomial time on a DTM.

We want that preimages are hard to compute.

Let n be the input length.

Idea. The portion of $x \in \{0, 1\}^n$ for which given $f(x)$ one can compute efficiently some x' with $f(x) = f(x')$ is negligible.

One cannot define this “portion”.

One-way functions

Let $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ be efficiently computable, that is, in polynomial time on a DTM.

We want that preimages are hard to compute.

Let n be the input length.

Idea. The portion of $x \in \{0, 1\}^n$ for which given $f(x)$ one can compute efficiently some x' with $f(x) = f(x')$ is negligible.

One cannot define this “portion”.

For “efficiently” and “negligible” one has to fix an algorithm.

One-way functions

Let $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ be efficiently computable, that is, in polynomial time on a DTM.

We want that preimages are hard to compute.

Let n be the input length.

Idea. The portion of $x \in \{0, 1\}^n$ for which given $f(x)$ one can compute efficiently some x' with $f(x) = f(x')$ is negligible.

One cannot define this “portion”.

For “efficiently” and “negligible” one has to fix an algorithm.

We want to consider all (randomized) algorithms.

One-way functions

Better idea for a definition. A **one-way function** is an efficiently computable function $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ with:

One-way functions

Better idea for a definition. A **one-way function** is an efficiently computable function $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ with:

For all PPT-algorithms \mathcal{A} ,

$$\mathbf{P}[f(\mathcal{A}(f(x))) = f(x)] ,$$

where $x \in \{0, 1\}^n$ is uniform, is negligible in $n = |x|$.

One-way functions

Better idea for a definition. A **one-way function** is an efficiently computable function $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ with:

For all PPT-algorithms \mathcal{A} ,

$$\mathbf{P}[f(\mathcal{A}(f(x))) = f(x)] ,$$

where $x \in \{0, 1\}^n$ is uniform, is negligible in $n = |x|$.

Problem. Like this the function $f : x \mapsto |x|$ is a one-way function.

One-way functions

Better idea for a definition. A **one-way function** is an efficiently computable function $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ with:

For all PPT-algorithms \mathcal{A} ,

$$\mathbf{P}[f(\mathcal{A}(f(x))) = f(x)] ,$$

where $x \in \{0, 1\}^n$ is uniform, is negligible in $n = |x|$.

Problem. Like this the function $f : x \mapsto |x|$ is a one-way function.

One cannot efficiently compute x from the length of x , because the output size is exponential in the input size.

One-way functions

Better idea for a definition. A **one-way function** is an efficiently computable function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ with:

For all PPT-algorithms \mathcal{A} ,

$$\mathbf{P}[f(\mathcal{A}(f(x))) = f(x)] ,$$

where $x \in \{0, 1\}^n$ is uniform, is negligible in $n = |x|$.

Problem. Like this the function $f : x \mapsto |x|$ is a one-way function.

One cannot efficiently compute x from the length of x , because the output size is exponential in the input size.

This suggests that the definition should be modified.

One-way functions

Better idea for a definition. A **one-way function** is an efficiently computable function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ with:

For all PPT-algorithms \mathcal{A} ,

$$\mathbf{P}[f(\mathcal{A}(f(x))) = f(x)] ,$$

where $x \in \{0, 1\}^n$ is uniform, is negligible in $n = |x|$.

Problem. Like this the function $f : x \mapsto |x|$ is a one-way function.

One cannot efficiently compute x from the length of x , because the output size is exponential in the input size.

This suggests that the definition should be modified.

Possible solution. We say “negligible in n ”.

This is alright, but does not correspond to the standard “framework” of complexity theory.

One-way functions

Better idea for a definition. A **one-way function** is an efficiently computable function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ with:

For all PPT-algorithms \mathcal{A} ,

$$\mathbf{P}[f(\mathcal{A}(f(x))) = f(x)] ,$$

where $x \in \{0, 1\}^n$ is uniform, is negligible in $n = |x|$.

Problem. Like this the function $f : x \mapsto |x|$ is a one-way function.

One cannot efficiently compute x from the length of x , because the output size is exponential in the input size.

This suggests that the definition should be modified.

Possible solution. We say “negligible in n ”.

This is alright, but does not correspond to the standard “framework” of complexity theory.

Solution. We also give $1^n = \overbrace{1 \cdots 1}^{n\text{-mal}}$ as input.

One-way functions

Definition. A (strong) one-way function is a polynomial time computable function $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ with:

One-way functions

Definition. A (strong) one-way function is a polynomial time computable function $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ with:

For all PPT-algorithms \mathcal{A} ,

$$\mathbf{P}[f(\mathcal{A}(1^n, f(x))) = f(x)] ,$$

with $x \in \{0, 1\}^n$ uniform is negligible

One-way functions

Definition. A (strong) one-way function is a polynomial time computable function $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ with:

For all PPT-algorithms \mathcal{A} ,

$$\mathbf{P}[f(\mathcal{A}(1^n, f(x))) = f(x)] ,$$

with $x \in \{0, 1\}^n$ uniform is negligible

The “inversion problem” can clearly be solved efficiently on a non-deterministic Turing machine.

One-way functions

Definition. A (strong) one-way function is a polynomial time computable function $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ with:

For all PPT-algorithms \mathcal{A} ,

$$\mathbf{P}[f(\mathcal{A}(1^n, f(x))) = f(x)] ,$$

with $x \in \{0, 1\}^n$ uniform is negligible

The “inversion problem” can clearly be solved efficiently on a non-deterministic Turing machine.

Therefore: The “inversion problem” can be reduced to a decision problem which is in NP.

If there is a one-way function, this decision problem is not in BPP.
So then $\text{NP} \not\subseteq \text{BPP}$.

One-way functions

Conjecture. The function

$$\{ (m, n) \in \mathbf{N} \times \mathbf{N} \mid \lceil \log_2(m) \rceil = \lceil \log_2(n) \rceil \} \longrightarrow \mathbf{N} ,$$

$$(m, n) \mapsto m \cdot n$$

is / leads to a one-way function.

Families of one-way functions

Idea. Given a security parameter, one chooses first a **parameter**. Then for a given parameter, one considers a function with finite input and output.

Families of one-way functions

Idea. Given a security parameter, one chooses first a **parameter**. Then for a given parameter, one considers a function with finite input and output.

Important conjectured example: modulo exponentiation

- ▶ Parameter: A prime p and a generator g of $(\mathbf{Z}/p\mathbf{Z})^\times$.
- ▶ Parameter choice: Given n choose a prime p of size n (how?).
- ▶ Function: $\{0, \dots, p-2\} \longrightarrow (\mathbf{Z}/p\mathbf{Z})^\times, x \mapsto g^x$
- ▶ Inversion: Computation of x
= discrete logarithm

Families of one-way functions

More general, for example for $G = E(\mathbf{F}_q)$:

- ▶ Parameter: A finite group $G = (G, \cdot)$ with efficient arithmetic $a \in G$.
- ▶ Function: $G \longrightarrow G, x \mapsto a^x$

Families of one-way functions

More general, for example for $G = E(\mathbf{F}_q)$:

- ▶ Parameter: A finite group $G = (G, \cdot)$ with efficient arithmetic $a \in G$.
- ▶ Function: $G \longrightarrow G, x \mapsto a^x$
- ▶ Parameter: A finite abelian group $G = (G, +), a \in G$.
- ▶ Function: $G \longrightarrow G, x \mapsto x \cdot a$

Hardcore bits

Idea for Hardcore bits

Let f be a one-way function. Then only for a negligible amount of x one can compute efficiently from $f(x)$ a preimage.

Idea for Hardcore bits

Let f be a one-way function. Then only for a negligible amount of x one can compute efficiently from $f(x)$ a preimage.

But: It could be that nonetheless one can extract information on x from $f(x)$.

Idea for Hardcore bits

Let f be a one-way function. Then only for a negligible amount of x one can compute efficiently from $f(x)$ a preimage.

But: It could be that nonetheless one can extract information on x from $f(x)$.

For example: The first bit of x could be encoded in $f(x)$.

Idea for Hardcore bits

Let f be a one-way function. Then only for a negligible amount of x one can compute efficiently from $f(x)$ a preimage.

But: It could be that nonetheless one can extract information on x from $f(x)$.

For example: The first bit of x could be encoded in $f(x)$.

Then the first bit would **not** be a hardcore bit.

Hardcore bits

Definition. Let $f : \{0, 1\}^* \longrightarrow \{0, 1\}$ be an efficiently computable function (\star). Then a **hardcore bit** for f is a function $b : \{0, 1\}^* \longrightarrow \{0, 1\}$ with:

Hardcore bits

Definition. Let $f : \{0, 1\}^* \rightarrow \{0, 1\}$ be an efficiently computable function (\star). Then a **hardcore bit** for f is a function $b : \{0, 1\}^* \rightarrow \{0, 1\}$ with:

For all PPT-algorithms \mathcal{A} the **success**

$$\mathbf{P}[\mathcal{A}(1^n, f(x)) = b(x)] - \frac{1}{2},$$

where $x \in \{0, 1\}^n$ is uniform, is negligible (in n).

Hardcore bits

Definition. Let $f : \{0, 1\}^* \rightarrow \{0, 1\}$ be an efficiently computable function (\star). Then a **hardcore bit** for f is a function $b : \{0, 1\}^* \rightarrow \{0, 1\}$ with:

For all PPT-algorithms \mathcal{A} the **success**

$$\mathbf{P}[\mathcal{A}(1^n, f(x)) = b(x)] - \frac{1}{2},$$

where $x \in \{0, 1\}^n$ is uniform, is negligible (in n).

(\star) Often “one-way” is required, but we don't do this.

Hardcore bits

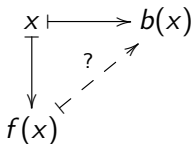
Definition. Let $f : \{0, 1\}^* \rightarrow \{0, 1\}$ be an efficiently computable function (\star). Then a **hardcore bit** for f is a function $b : \{0, 1\}^* \rightarrow \{0, 1\}$ with:

For all PPT-algorithms \mathcal{A} the **success**

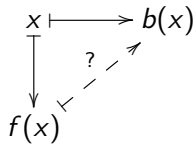
$$\mathbf{P}[\mathcal{A}(1^n, f(x)) = b(x)] - \frac{1}{2},$$

where $x \in \{0, 1\}^n$ is uniform, is negligible (in n).

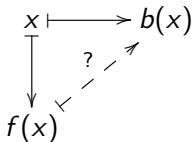
(\star) Often “one-way” is required, but we don't do this.



Hardcore bits

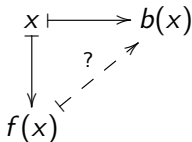


Hardcore bits



Example. Let $f(x_1 \cdots x_n) := x_2 \cdots x_n$, $b(x_1 \cdots x_n) := x_1$.
Then b is a hardcore bit for f .

Hardcore bits

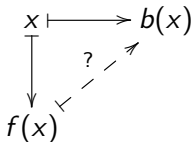


Example. Let $f(x_1 \cdots x_n) := x_2 \cdots x_n$, $b(x_1 \cdots x_n) := x_1$.

Then b is a hardcore bit for f .

But: f is not injective.

Hardcore bits



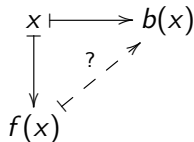
Example. Let $f(x_1 \cdots x_n) := x_2 \cdots x_n$, $b(x_1 \cdots x_n) := x_1$.

Then b is a hardcore bit for f .

But: f is not injective.

Lemma. Let f be injective. If now f has a hardcore bit, then f is a one-way function.

Hardcore bits



Example. Let $f(x_1 \cdots x_n) := x_2 \cdots x_n$, $b(x_1 \cdots x_n) := x_1$.

Then b is a hardcore bit for f .

But: f is not injective.

Lemma. Let f be injective. If now f has a hardcore bit, then f is a one-way function.

Expressed differently: If f is injective and not a one-way function, then it does not have a hardcore bit.

Results on hardcore bits

Theorem. (Blum & Micali, 1984) Let g be a generator of $(\mathbb{Z}/p\mathbb{Z})^\times$.

Results on hardcore bits

Theorem. (Blum & Micali, 1984) Let g be a generator of $(\mathbf{Z}/p\mathbf{Z})^\times$.

If $\{1, \dots, p-1\} \longrightarrow (\mathbf{Z}/p\mathbf{Z})^\times \simeq \{1, \dots, p-1\}$, $x \mapsto g^x$ is a one-way function, then

$$b : x \mapsto \begin{cases} 0, & \text{falls } x \leq \frac{p-1}{2} \\ 1, & \text{falls } x > \frac{p-1}{2} \end{cases}$$

is a hardcore bit thereof.

Results on hardcore bits

Theorem. (Blum & Micali, 1984) Let g be a generator of $(\mathbf{Z}/p\mathbf{Z})^\times$.

If $\{1, \dots, p-1\} \longrightarrow (\mathbf{Z}/p\mathbf{Z})^\times \simeq \{1, \dots, p-1\}$, $x \mapsto g^x$ is a one-way function, then

$$b : x \mapsto \begin{cases} 0, & \text{falls } x \leq \frac{p-1}{2} \\ 1, & \text{falls } x > \frac{p-1}{2} \end{cases}$$

is a hardcore bit thereof.

Theorem. (Goldreich & Levin, 1989) Let f be a one-way funktion. Then a random linear combination of x is a hardcore bit of f .

Results on hardcore bits

Theorem. (Blum & Micali, 1984) Let g be a generator of $(\mathbf{Z}/p\mathbf{Z})^\times$.

If $\{1, \dots, p-1\} \longrightarrow (\mathbf{Z}/p\mathbf{Z})^\times \simeq \{1, \dots, p-1\}$, $x \mapsto g^x$ is a one-way function, then

$$b : x \mapsto \begin{cases} 0, & \text{falls } x \leq \frac{p-1}{2} \\ 1, & \text{falls } x > \frac{p-1}{2} \end{cases}$$

is a hardcore bit thereof.

Theorem. (Goldreich & Levin, 1989) Let f be a one-way funktion. Then a random linear combination of x is a hardcore bit of f .

This means: $(x, u) \mapsto (f(x), u)$ mit $|x| = |u|$ is a one-way function and $b : (x, u) \mapsto x_1 u_1 + \dots + x_n u_n$ is a hardcore bit thereof.