CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACRONYMS

PTM probablistic Turing machine

PPTM polynomial probablistic Turing machine

PRF pseudorandom function

PRP pseudorandom permutation

SES symmetric encryption scheme

PKES	public key encryption scheme
CPA	chosen-plaintext attack
CPA+	chosen-plaintext+ attack
CCA	chosen-ciphertext attack
CCA+	chosen-ciphertext+ attack
JPEG	chosen-ciphertext+ attack

1 INTRODUCTION

$\mathrm{SE}^{c\mathcal{C}}(k, m, h)$

Input: key k, message m, history h
  $s = \varnothing$ ▷ initialize the empty state
  for $i = 1, 2, \ldots, \ell(\kappa)$ **do**
    $(d_i, s) \leftarrow \mathrm{SE}^{c\mathcal{C}_{h,nn(\kappa)}}(k, m, h, s)$
    $h := h || d_i$
  end for
  return $d_1, d_2, \ldots, d_{\ell(\kappa)}$

$\mathrm{SE}^{c\mathcal{C}}(k, m, h)$: The run of a stegosystem

We define a *bad value* as

bad value

$$aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa$$

2 PRELIMINARIES

To understand the formal models of steganography presented in the next chapter, we first need to introduce some basic notations concerning probabilities, algorithms and cryptographic primitives. We denote the set of natural numbers, including 0, by $\mathbb{N}$, the set of real numbers by $\mathbb{R}$ and the set of rational numbers by $\mathbb{Q}$. For an alphabet Σ and a string $s \in \Sigma^*$, we denote the length of s by $|s|$ and for two strings $s, s' \in \Sigma^*$, the concatenation of s and s' is written as $s \,\|\, s'$. For a set S, denote by $\mathcal{P}(S)$ the set of all subsets of S.

2.1 PROBABILITIES

As the undetectable embedding of a message into a document is an inherent random process, we will now give a short overview on the probability theory needed in this work. As no continuous probability spaces are used in this thesis, it is sufficient to focus on the discrete case. For a thorough discussion of this subject, see e. g. the textbook of Mitzenmacher and Upfal [MU05]. A *probability distribution* Pr upon a *probability space* Ω – a finite or countable infinite set – is a function $\Pr\colon \mathcal{P}(\Omega) \to [0,1]$ such that $\Pr(\emptyset) = 0$, $\Pr(\Omega) = 1$ and $\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B)$ for all $A, B \subseteq \Omega$. The elements of Ω are called *elementary events* and subsets of Ω are simply called *events*. To simplify notation, we omit the probability space if it is clear from the context or denote it by $\mathrm{dom}(\Pr)$. A very important subset of elementary events is the set of all elementary events that may occur, i. e. those that have probability greater than zero. This set is called the *support* of Pr and we denote it by $\mathrm{supp}(\Pr) = \{\omega \in \mathrm{dom}(\Pr) \mid \Pr(\omega) > 0\}$. The *min-entropy* measures the amount of randomness of a probability distribution Pr and is defined as $H_\infty(\Pr) = \min_{\omega \in \mathrm{supp}(\Pr)}\{-\log \Pr(\omega)\}$. For two events A and B, the *conditional probability* that A occurs given that B occurs is defined as $\Pr(A \mid B) := \frac{\Pr(A \cap B)}{\Pr(B)}$. We say that A and B are *independent events*, if $\Pr(A \mid B) = \Pr(A)$.

probability distribution
probability space
elementary events
events
support
min-entropy
conditional probability
independent events

Example. To describe the throw of a six-sided dice, the elementary events are described by $\Omega = \mathrm{dom}(\Pr) = \{$⚀, ⚁, ⚂, ⚃, ⚄, ⚅$\}$. The probabilities are then given by $\Pr(\{$⚀$\}) = \Pr(\{$⚁$\}) = \Pr(\{$⚂$\}) = \Pr(\{$⚃$\}) = \Pr(\{$⚄$\}) = \Pr(\{$⚅$\}) = 1/6$. The Events $A = \{$⚀, ⚂, ⚄$\}$ (whether the thrown side shows an odd number of eyes) and $B = \{$⚄, ⚅$\}$ (whether the number of eyes is strictly larger than four) are independent, as $\Pr(A) = 1/2$, $\Pr(B) = 1/3$ and $\Pr(A \cap B) = \Pr(\{$⚄$\}) = 1/6$. ◇

Examples always end with ◇

It is often convenient to assign numerical values to the elementary events. This is formally described by the notion of a (real-valued) *ran-*

random variable *dom variable*, which is a mapping from Ω to $\mathbb{R}$. If Pr is a probability distribution on Ω and $X\colon \Omega \to \mathbb{R}$ is a random variable, we define $\Pr[X = x] := \Pr(X^{-1}(x))$ as the probability that X gets the value x. To measure the expected outcome of a random variable X, we define the *expected value* of X as $\mathrm{Exp}[X] := \sum_{x \in X(\Omega)} x \cdot \Pr[X = x]$. Two random variables X and Y are *independent random variables*, if $X^{-1}(x)$ and $Y^{-1}(y)$ are independent events for all $x \in \mathrm{img}(X)$ and all $y \in \mathrm{img}(Y)$.

expected value

independent random variables

Example. Continuing the previous example, the canonical random variable X would assign $X(⚀) = 1$, $X(⚁) = 2$ and so on. But we could also measure whether the number of eyes was odd by using the random variable Y with $Y(⚀) = Y(⚂) = Y(⚄) = 1$ and $Y(⚁) = Y(⚃) = Y(⚅) = 0$. Hence $\Pr[Y = 1] = \Pr(Y^{-1}(1)) = \Pr(\{⚀, ⚁, ⚂\})$. The respective expected values are $\mathrm{Exp}[X] = 7/2$ and $\mathrm{Exp}[Y] = 1/2$. $\diamond$

The simplest random variable one can think of is a binary indicator variable that only takes values 0 and 1. A *Bernoulli random variable* X with parameter p only takes the values 0 and 1 with probability p and $1 - p$ i.e. $\Pr[X = 0] = p$ and $\Pr[X = 1] = 1 - p$. If $X_1, X_2, \dots, X_n$ are independent Bernoulli random variables with parameters $p_1 = p_2 = \dots = p_n = p$, their sum $X = \sum_{i=1}^{n} X_i$ is called a *Binomial random variable* X with parameters p and n. It takes values $0, 1, \dots, n$ with $\Pr[X = k] = \binom{n}{k} \cdot p^k \cdot (1 - p)^{n-k}$ for each $k = 0, 1, \dots, n$.

Bernoulli random variable

Binomial random variable

It is often important to rule out some events by proving that they very rarely occur. For the special case of a (generalized) Binomial random variable X, the extremely helpful *Chernoff bound* shows that X very rarely deviates from its expected value.

Theorem 1 (Chernoff Bound). *Let $X_1, \dots, X_n$ be independent Bernoulli random variables with parameters $p_1, p_2, \dots, p_n$ and $X = \sum_{i=1}^{n} X_i$ with expected value $\mu = \mathrm{Exp}[X] = \sum_{i=1}^{n} p_i$. For every $0 < \delta < 1$,*

$$\Pr[|X - \mu| \geqslant \delta \cdot \mu] \leqslant 2 \cdot \exp(-(\mu \cdot \delta^2)/3).$$

Example. The Chernoff bound tells us that after throwing 1,000 fair coins, the probability that at most 100 heads or at least 900 heads occurred, is bounded by $2 \cdot \exp(-(500 \cdot 0.8 \cdot 0.8)/3) \leqslant 10^{-48}$. $\diamond$

By letting the parameter n grow to ∞, while keeping $p = p(n)$ as a bounded function of n, the resulting random variable will also be useful in the later chapters and is easily described by the following theorem (see e. g. [MU05, pp. 98-99] for a proof).

Theorem 2. *Let X_n be a Binomial random variable with parameters n and $p(n)$, where p is a function of n and $\lim_{n \to \infty} n \cdot p(n) = \mu$ is a constant independent of n. Then, for any fixed k,*

$$\lim_{n \to \infty} \Pr[X_n = k] = \frac{\exp(-\mu) \cdot \mu^k}{k!}.$$

This fact leads to the definition of a Poisson random variable. A *Poisson random variable* with parameter μ takes values in $\mathbb{N}$ with probability $\Pr[X = k] = \frac{\exp(-\mu)\cdot\mu^k}{k!}$. These random variable can be used to analyze a variety of *balls into bin* experiments relatively easy and we will make use of them later on.

Poisson random variable

2.2 ALGORITHMS

We use *Turing machines* as our model of computation in this work. For a detailed introduction and formal definitions, see the textbook of Papadimitriou [Pap94]. The Turing machines in this work will also be able to make independent fair coin flips and will thus be called *probablistic Turing machine (PTM)*. We will use the terms PTM and algorithm interchangeably. The output of such a machine is thus a random variable upon the probability space $\{0,1\}^k$, where k is the maximum number of coin flips performed by the machine. Similarly, the *running time* of a PTM is defined as the expected number of steps that the machine performs and is a function of the length of its input. If the running time of a PTM M is bounded by a polynomial, we say that M is a *polynomial probablistic Turing machine (PPTM)* or an efficient algorithm.

PTM

running time

PPTM

Often, the machine M will also be equipped with different *oracles*, that allow us to increase the abilities of M. See the next section for examples of this.

oracles

- For a random variable X (e. g. another machine), the PTM M^X can get a sample x distributed according to X. If X is the uniform distribution on a set S, we simply write M^S. The running time to receive a single sample is simply the encoding length of the sample.
- If $f\colon U \to V$ is a function, M^f can provide an element $u \in U$ and gets back the value $f(u)$. The running time for this operation is the encoding length of u plus the encoding length of $f(u)$.

If M can access several oracles $O_1, O_2, \ldots$, we write $M^{O_1,O_2,\cdots}$. If an algorithm M gets a sample x distributed according to the random variable X, we denotes this as $x \leftarrow X$ and $x \leftarrow M$ for the output of the randomized algorithm. If M is not randomized, i.e. it can only output a single value for fixed inputs, we denote this by $x := M$ to highlight this difference. If S is a finite set, we denote the uniform sampling of a random element s of S by $s \twoheadleftarrow S$.

If Pr is a probability distribution, we say that Pr is a *efficiently sampleable distribution*, if there is a PPTM M that gets no input such that the output distribution of Pr and M is the same. A sequence of probability distributions $\Pr_1, \Pr_2, \ldots$ will also be denoted as $\{\Pr_n\}_{n\in\mathbb{N}}$ and is called an *distribution ensemble*. A distribution ensemble $\{\Pr_n\}_{n\in\mathbb{N}}$ is

efficiently sampleable distribution

distribution ensemble

called an *efficiently sampleable ensemble*, if there is a PPTM M such that its upon the unary encoding of a number n – denoted by 1^n – is the same as $\Pr_n$, i. e. $M(1^n) = \Pr_n$. Similarly, a function $f\colon U \to V$ is *efficiently computable*, if there is a PPTM M such that $M(u) = f(u)$ for all $u \in U$.

efficiently sampleable ensemble

efficiently computable

2.3 CRYPTOGRAPHIC PRIMITIVES

We will make use of a wide range of cryptographic primitives ranging from *one-way functions* to *public-key cryptosystems*. Most of the definitions are taken from or inspired by the excellent textbook of Katz and Lindell [KL07].

Two main approaches for the definition of cryptographic primitives exist in the literature. In the first approach, the length of the key (also called the *security parameter*) is treated as a constant. Consequently, the running time of all involved algorithms are also a constant. The typical assumption in this model is that a primitive is (t, ϵ)-secure, i. e. the advantage of every attacker with running time t against the primitive is at most ϵ. This line of work was first introduced by Bellare et al. and is commonly called *concrete security* [Bel+97]. The second approach – the *asymptotic security* – treats the security parameter as a variable and analyzes the security of the primitives in an asymptotic way, i. e. for large enough values. We typically denote this variable with κ and the corresponding key $k \in \{0,1\}^\kappa$ with k. To define security in this setting, the notion of negligible functions is needed. A function $\mathsf{negl}\colon \mathbb{N} \to [0,1]$ is called *negligible* if for every polynomial p, there is $n_0 \in \mathbb{N}$ such that $\mathsf{negl}(n) < p(n)^{-1}$ for every $n \geqslant n_0$. Hence, a negligible function decreases faster than the inverse of every polynomial.

security parameter

concrete security

asymptotic security

negligible

Example. Typical examples for negligible functions are $n \mapsto 2^{-n}$, $n \mapsto 1.01^{-n}$, $n \mapsto 2^{-0.1n}$, but also $n \mapsto n^{-\log n}$. ◇

The typical assumption in the asymptotic security setting is now that upon security parameter κ, every attacker that runs in time $p(\kappa)$ for a polynomial p only has a negligible advantage of $\mathsf{negl}(\kappa)$ to break the primitive.

While the concrete approach gives more concrete bounds, the analysis of the security in the asymptotic approach is often more helpful in understanding the underlying arguments. Additionally, it is unclear for which parameters t and ϵ we can treat a primitive as "secure". As one can typically easily translate asymptotic bounds into concrete bound and as we want to emphasize upon the arguments rather than those concrete bounds, we have decided to use the asymptotic approach in this work. For a more thorough discussion of this models, see the textbook of Katz and Lindell [KL07, pp. 49-52]. For an example of the concrete approach in steganography see e. g. [BR16]. Throughout this chapter, let ℓ, ℓ' and L be polynomials.

One-Way Functions

A function $F: \{0,1\}^* \to \{0,1\}^*$ is called a *one-way function*, if the following properties hold: *one-way function*

- For all $n \in \mathbb{N}$ and all $x \in \{0,1\}^n$, we have $\ell(n) \leqslant |F(x)| \leqslant \ell'(n)$.
- The function F is efficiently computable.
- For every PPTM Inv (the *inverting algorithm*), there exists a negligible function negl such that for all sufficiently large $n \in \mathbb{N}$,

  $$\Pr_{x \twoheadleftarrow \{0,1\}^n}[\mathsf{Inv}(F(x)) \in F^{-1}(F(x))] \leqslant \mathsf{negl}(n),$$

  where the probability is taken over the random choice of x and the internal coin flips of Inv.

A wide range of works shows that the existence of one-way functions is the minimal assumption needed for cryptography, as most of the following primitives can be constructed out of them [KL07, pp. 181-225].

Hash Functions

In the following, we will often use *keyed functions* $f: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. The first parameter of f is then called the *key* of the function. To simplify notation, for each key $k \in \{0,1\}^*$, we define the function $f_k: \{0,1\}^* \to \{0,1\}^*$ with $f_k(x) = f(k,x)$. *keyed functions*

A cryptographic primitive typically consists of a keyed function f and a *generator* algorithm Gen_f that upon input 1^κ produces a suitable key k of sufficient size larger than κ for f. *generator*

A *hash function* (H, Gen_H) is a pair of PPTMs such that Gen_H upon input 1^κ produces a key $k \in \{0,1\}^*$ with $|k| \geqslant \kappa$. The keyed function H takes the key $k \leftarrow \mathsf{Gen}_H(1^\kappa)$ and a string $x \in L(\kappa)$ and produces a string $H_k(x)$ of length $\ell(\kappa) < L(\kappa)$. *hash function*

In order to define the "security" of this function, we first need to define a corresponding *experiment*. This is a typical approach in cryptography and steganography: For a primitive Π, we define an experiment $E(\Pi)$, that takes an "attacker" A. Whenever the probability that A passes (its *advantage* $\mathbf{Adv}_{A,\pi}(\kappa)$) is negligible we say that Π is secure. *advantage*

As it should be hard for an adversary to find two different elements $x \neq x'$ such that $H_k(x) = H_k(x')$, we need to find a corresponding experiment. A *collision finder* Fi for a hash function (H, Gen_H) is a PPTM that upon input $k \in \mathrm{supp}(\mathsf{Gen}_H(1^\kappa))$ outputs two strings $x, x' \in \{0,1\}^{L(\kappa)}$. Its goal is to pass the following experiment: *collision finder*

$\mathsf{Coll}_{\mathsf{Fi},(\mathsf{H},\mathsf{Gen}_\mathsf{H})}(\kappa)$

Input: Hash function $(\mathsf{H}, \mathsf{Gen}_\mathsf{H})$, Finder Fi, key length κ
$k \leftarrow \mathsf{Gen}_\mathsf{H}(1^\kappa)$
$(x, x') \leftarrow \mathsf{Fi}(k)$
if $x \neq x'$ and $\mathsf{H}_k(x) = \mathsf{H}_k(x')$ **then return** 1
else return 0
end if

$\mathsf{Coll}_{\mathsf{Fi},(\mathsf{H},\mathsf{Gen}_\mathsf{H})}(\kappa)$: Collision-Finding Experiment

collision resistant A hash function $\Pi = (\mathsf{H}, \mathsf{Gen}_\mathsf{H})$ is called *collision resistant*, if for all collision finders Fi, there is a negligible function negl such that

$$\mathbf{Adv}^{\text{hash}}_{\mathsf{Fi},(\mathsf{H},\mathsf{Gen}_\mathsf{H})}(\kappa) := \Pr[\mathsf{Coll}_{\mathsf{Fi},\Pi}(\kappa) = 1] \leqslant \mathsf{negl}(\kappa).$$

As collision resistant hash functions compresses an input of length $L(\kappa)$ into a smaller value of length $\ell(\kappa) < L(\kappa)$, they are often used to create short signatures of a longer bitstring.

Pseudorandom Functions

set of all function For two numbers $\ell, \ell' \in \mathbb{N}$, denote the *set of all function* from $\{0,1\}^\ell$ to $\{0,1\}^{\ell'}$ by $\mathrm{Fun}(\ell, \ell')$. Clearly, in order to specify a random element of $\mathrm{Fun}(\ell, \ell')$, one needs $2^\ell \times \ell'$ bits and we can thus not use completely random functions in an efficient setting. We will thus use efficient functions that are indistinguishable from completely random function. A *pseudorandom function (PRF)* $(\mathsf{F}, \mathsf{Gen}_\mathsf{F})$ is a pair of PPTMs such that Gen_F upon input 1^κ produces a key $k \in \{0,1\}^*$ with $|k| \geqslant \kappa$. The keyed function F takes the key $k \leftarrow \mathsf{Gen}_\mathsf{F}(1^\kappa)$ and a string $x \in \{0,1\}^{\ell(\kappa)}$ and produces a string $\mathsf{F}_k(x)$ of length $\ell'(\kappa)$. An attacker, called *distinguisher* Dist upon input 1^κ gets oracle access to a function that is either completely random or equals F_k for a randomly chosen key k and its goal is to distinguish between those cases. Hence, for every distinguisher Dist there is a negligible function negl such that

PRF

distinguisher

$$\mathbf{Adv}^{\text{prf}}_{\mathsf{Dist},(\mathsf{F},\mathsf{Gen}_\mathsf{F})}(\kappa) := \left|\Pr[\mathsf{Dist}^{\mathsf{F}_k}(1^\kappa) = 1] - \Pr[\mathsf{Dist}^{f}(1^\kappa)]\right| \leqslant \mathsf{negl}(\kappa),$$

where $k \leftarrow \mathsf{Gen}_\mathsf{F}(1^\kappa)$ and $f \leftarrow \mathrm{Fun}(\ell(\kappa), \ell'(\kappa))$.

PRP Furthermore, if $\ell(\kappa) = \ell'(\kappa)$ and if every F_k is a permutation we say that the pair $(\mathsf{F}, \mathsf{Gen}_\mathsf{F})$ is a *pseudorandom permutation (PRP)*.

Note that due to the definition of PRFs, they share *all* properties of totally random functions that one can test in polynomial time (up to a negligible probability). A typical security analysis of a protocol that uses PRFs thus starts with the analysis of the protocol if one replaces the PRF with a totally random function. This modified protocol is then examined with probability- or information-theoretic means to conclude something about the behaviour of the modified protocol. By replacing the totally random function with a PRF, one can conclude that the behaviour of the modified protocol and the behaviour of the original protocol are very similar. This allows one to also use the results of the modified protocol for the original protocol.

In the chapters concerned with non-efficient steganography, we will drop the requirement that F is efficient computable and say that such a pair $(\mathsf{F}, \mathsf{Gen}_\mathsf{F})$ is a *non-efficient PRF*. *non-efficient PRF*

Signature Schemes

A *signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is a triple of PPTMs such that the algorithm $\mathsf{Gen}(1^\kappa)$ produces a pair (pk, sk) of keys with $|pk| \geqslant \kappa$ and $|sk| \geqslant \kappa$. We call pk a *public key* and sk a *secret/private key*. The *signing algorithm* Sign takes as input the secret key sk, a *message* $m \in \{0,1\}^{\ell(\kappa)}$ and outputs a *signature* $\sigma \in \{0,1\}^*$. The *verifying algorithm* Vrfy takes as input the public key pk, a message $m \in \{0,1\}^{\ell(\kappa)}$ and a signature $\sigma \in \{0,1\}^*$. It outputs a bit b with $b = 1$ iff $\sigma \in \mathrm{supp}(\mathsf{Sign}(pk, m))$. We will typically treat Sign and Vrfy as keyed functions and will thus also write Sign_{sk} and Vrfy_{pk} for the corresponding function, where the key is fixed. A *forger* Fo is a PPTM that upon input pk and oracle access to Sign_{sk} tries to produce a pair (m, σ) such that $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$. Formally, this is defined via the following experiment $\mathsf{Sig\text{-}Forge}$.

signature scheme
signing algorithm
signature
verifying algorithm
forger

$\mathsf{Sig\text{-}Forge}_{\mathsf{Fo},(\mathsf{Gen},\mathsf{Sign},\mathsf{Vrfy})}(\kappa)$

Input: Signature Scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$, Forger Fo, length κ
$(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$(m, \sigma) \leftarrow \mathsf{Fo}^{\mathsf{Sign}_{sk}}(pk)$
Let Q be the set of messages given to Sign_{sk} by Fo
if $m \notin Q$ and $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$ **then return** 1
else return 0
end if

$\mathsf{Sig\text{-}Forge}_{\mathsf{Fo},(\mathsf{Gen},\mathsf{Sign},\mathsf{Vrfy})}(\kappa)$: Signature-Forging Experiment

A signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is called *existentially unforgeable* if for every forger Fo, there is a negligible function negl such that *existentially unforgeable*

$$\mathbf{Adv}^{\mathrm{sig}}_{\mathsf{Fo},(\mathsf{Gen},\mathsf{Sign},\mathsf{Vrfy})}(\kappa) := \Pr[\mathsf{Sig\text{-}Forge}_{\mathsf{Fo},\Pi}(\kappa) = 1] \leqslant \mathsf{negl}(\kappa).$$

Note that this definition of security implies that a existentially unforgeable signature scheme is *publicly verifiable* and has the property of *non-repudiation* [KL07], two important aspects that we will also make use of.

SES A *symmetric encryption scheme (SES)* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a triple of PPTMs such that $\mathsf{Gen}(1^\kappa)$ produces a key $k \in \{0,1\}^*$ with $|k| \geqslant \kappa$. The *encryption algorithm* *encryption algorithm* Enc takes as input the key k and a *plaintext* $m \in \{0,1\}^{\ell(\kappa)}$ and outputs a *ciphertext* $c \in \{0,1\}^*$. The *decryption algorithm* *decryption algorithm* Dec takes as input the key k and a ciphertext c and outputs a plaintext $m \in \{0,1\}^{\ell(\kappa)}$. In order to make sure that the decryption is successful, we assume that there exists a negligible function negl such that the probability $\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) \neq m] \leqslant \mathsf{negl}(\kappa)$, with $k \leftarrow \mathsf{Gen}(1^\kappa)$.

attacker An *attacker* (A_1, A_2) on the encryption scheme is a pair of PPTMs. In the *first round*, the algorithm A_1 produces upon input of 1^κ and with oracle access to Enc_k two messages $m_0, m_1 \in \{0,1\}^{\ell(\kappa)}$. In the *second round*, A_2 is given the encryption of m_b and should decide whether $b = 0$ or $b = 1$. This security notion is known as security against chosen-plaintext attacks (CPAs). Formally, this is defined via the following experiment CPA-Dist.

$\mathsf{CPA\text{-}Dist}_{(A_1,A_2),(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})}(\kappa)$

Input: SES $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, Attacker (A_1, A_2), length κ
$k \leftarrow \mathsf{Gen}(1^\kappa)$
$(m_0, m_1, s) \leftarrow A_1^{\mathsf{Enc}_k}(1^\kappa)$ ▷ s contains *state information*
$b \twoheadleftarrow \{0,1\}$
$c \leftarrow \mathsf{Enc}_k(m_b)$
$b' \leftarrow A_2^{\mathsf{Enc}_k}(1^\kappa, c, s)$
if $b = b'$ **then return** 1
else return 0
end if

$\mathsf{CPA\text{-}Dist}_{(A_1,A_2),(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})}(\kappa)$: CPA-Experiment

CPA-secure A SES $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *CPA-secure* if for every attacker (A_1, A_2), there is a negligible function negl such that

$$\mathbf{Adv}^{\mathrm{cpa}}_{(A_1,A_2),(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})}(\kappa) := \left|\Pr[\mathsf{CPA\text{-}Dist}_{(A_1,A_2),(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})}(\kappa) = 1] - \frac{1}{2}\right| \leqslant \mathsf{negl}(\kappa).$$

An even stronger security notion is the notion of security against chosen-plaintext+ attacks (CPA+s), where the attacker A_1 outputs a single message m and the string c is either $\mathsf{Enc}_k(m)$ ($b = 0$) or a completely random bitstring of length $|\mathsf{Enc}_k(m)|$ ($b = 1$). The goal of A_2 is to reconstruct the bit b from c. Denote this modification of CPA-Dist by $\mathsf{CPA}+\text{-Dist}$. Informally, this implies that the ciphertext constructed by the SES are indistinguishable from random strings. A *CPA+-secure* symmetric encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *CPA+-secure* if for every attacker (A_1, A_2), there is a negligible function negl such that

$$\mathbf{Adv}^{\text{cpa+}}_{(A_1,A_2),(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})}(\kappa) :=$$
$$\left|\Pr[\mathsf{CPA+\text{-}Dist}_{(A_1,A_2),(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})}(\kappa) = 1] - \frac{1}{2}\right| \leqslant \mathsf{negl}(\kappa).$$

Clearly, CPA+-security implies CPA-security, but the other implication is not true: For example, the encryption algorithm Enc may always appends a certain string at the end of each ciphertext. Fortunately, most known CPA-secure SES are also CPA+-secure.

Random Counter Mode

We will sometimes use simple, yet incredibly useful SES called the *random counter mode*. Let (F, Gen_F) be a PRF that maps input strings of length $\ell(\kappa)$ into output strings of the same length $\ell(\kappa)$. The following algorithms then yields a SES $(\mathsf{Gen}^F, \mathsf{Enc}^F, \mathsf{Dec}^F)$:

random counter mode

- The generator algorithm Gen^F simply uses Gen_F to create a symmetric key k, i. e. $\mathsf{Gen}^F(1^\kappa) = \mathsf{Gen}_F(1^\kappa)$.
- The encryption algorithm works as follows for messages $m = m_1 m_2 \dots m_{n(\kappa)}$ with $m_i \in \{0,1\}^{\ell(\kappa)}$:

Enc^F

```
Input: key k, m = m_1 m_2 ... m_{n(κ)} ∈ {0,1}^{n(κ)·ℓ(κ)}
  κ := |k|
  r ← {0,1}^{ℓ(κ)}                ▷ r is treated as string and number
  for i = 1,...,n(κ) do
      c_i := F_k(r + i mod 2^{ℓ(κ)}) ⊕ m_i
  end for
  return r, c_1, c_2, ..., c_{n(κ)}
```

Enc^F: Random Counter Mode Encryption

- Similarly, the decryption inverts the encryption:

Dec^F

Input: key k, $c = c_0 c_1 \dots c_{n(\kappa)} \in \{0,1\}^{(n+1)\cdot\ell(\kappa)}$
  $\kappa := |k|$
  $r := c_0$
  for $i = 1, \dots, n(\kappa)$ **do**
    $m_i := F_k(r + i \bmod 2^{\ell(\kappa)}) \oplus c_i$
  end for
  return $m_1, m_2, \dots, m_{n(\kappa)}$

Dec^F: Random Counter Mode Decryption

Clearly, every ciphertext $c = \mathsf{Enc}^F(k, m)$ is decoded correctly. Concerning the security, Bellare et al. already proved the following theorem in [Bel+97], where they called this construction the *XOR-Scheme*.

Theorem 3 (Theorem 13 in the full version of [Bel+97]). *If the pair* (F, Gen_F) *is a pseudorandom function, the symmetric encryption scheme* $(\mathsf{Gen}^F, \mathsf{Enc}^F, \mathsf{Dec}^F)$ *is CPA+-secure.*

Public-Key Encryption Schemes

While SESs are very useful, the problem of the key management remains complicated. If n parties want to communicate via a SES, each pair $i, j \in \{1, \dots, n\}$ needs to share a key $k_{i,j}$. Hence, $\binom{n}{2}$ keys are needed if every party wants to communicate with every other party. And furthermore, those $\binom{n}{2}$ somehow need to be exchanged over a secure communication channel before the actual communication may take part. In order to remedy these problems, Diffie and Hellman introduced the notion of *public-key cryptography* in their groundbreaking work [DH76].

PKES

public key

secret key

public-key encryption algorithm

public-key decryption algorithm

A *public key encryption scheme (PKES)* $(\mathsf{Gen}, \mathsf{PKEnc}, \mathsf{PKDec})$ is a triple of PPTMs such that $\mathsf{Gen}(1^\kappa)$ produces a pair of keys (pk, sk) with $|pk| \geqslant \kappa$ and $|sk| \geqslant \kappa$. The key pk is called the *public key* and the key sk is called the *secret key* (or *private key*). While pk will be publicly available to all parties, the secret key sk must be kept private by its owner. The *public-key encryption algorithm* PKEnc takes as input the public key pk and a plaintext $m \in \{0,1\}^{\ell(\kappa)}$ and outputs a ciphertext $c \in \{0,1\}^*$. The *public-key decryption algorithm* PKDec takes as input the secret key sk and the ciphertext c and produces a plaintext $m \in \{0,1\}^{\ell(\kappa)}$. In order to make sure that the decryption is successful, we assume that there exists a negligible function negl such that the probability $\Pr[\mathsf{PKDec}(sk, \mathsf{PKEnc}(pk, m)) \neq m] \leqslant \mathsf{negl}(\kappa)$, with $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$.

While an attacker against a SES was given oracle access to the encryption algorithm, this is not needed in the public-key setting: Everyone knows the public key pk needed to encrypt messages. On the other hand, research has shown that the security requirements for PKESs are much higher. Informally, we will allow an attacker to first choose a message that should be encrypted. In the next step, the attacker is allowed to insert arbitrary ciphertexts into the communication and watch Bob's behaviour upon receiving those texts. Formally we equip an attacker with a *decryption oracle* in order to perform this kind of attack.

An *public-key attacker* (A_1, A_2) on the PKES is a pair of PPTMs. In the *first round*, the algorithm A_1 produces upon input *pk* and with oracle access to PKDec_{sk} two messages $m_0, m_1 \in \{0,1\}^{\ell(\kappa)}$. A random bit $b \twoheadleftarrow \{0,1\}$ is then chosen and in the *second round*, A_2 is given the encryption c of m_b and should decide whether $b = 0$ or $b = 1$. While we still allow A_2 to have oracle access to the decoding algorithm PKDec_{sk}, clearly we must forbid that it uses it to decrypt c. This security notion is known as security against chosen-ciphertext attacks (CCAs). Formally, this is defined via the following experiment CCA-Dist.

public-key attacker

$\mathsf{CCA\text{-}Dist}_{(A_1,A_2),(\mathsf{Gen},\mathsf{PKEnc},\mathsf{PKDec})}(\kappa)$

Input: PKES $(\mathsf{Gen}, \mathsf{PKEnc}, \mathsf{PKDec})$, Attacker (A_1, A_2), length κ
$(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$(m_0, m_1, s) \leftarrow A_1^{\mathsf{PKDec}_{sk}}(pk)$ ▷ s contains *state information*
$b \twoheadleftarrow \{0,1\}$
$c \leftarrow \mathsf{PKEnc}_{pk}(m_b)$
$b' \leftarrow A_2^{\mathsf{PKDec}_{sk}}(pk, c, s)$
if A_2 queries $\mathsf{PKDec}_{sk}(c)$ or $b \neq b'$ **then return** 0
else return 1
end if

$\mathsf{CCA\text{-}Dist}_{(A_1,A_2),(\mathsf{Gen},\mathsf{PKEnc},\mathsf{PKDec})}(\kappa)$: CCA-Experiment

A PKES $(\mathsf{Gen}, \mathsf{PKEnc}, \mathsf{PKDec})$ is called *CCA-secure*, if for every attacker (A_1, A_2), there is a negligible function negl such that

CCA-secure

$$\mathbf{Adv}^{\mathrm{cca}}_{(A_1,A_2),(\mathsf{Gen},\mathsf{PKEnc},\mathsf{PKDec})}(\kappa) := \left|\Pr[\mathsf{CCA\text{-}Dist}_{(A_1,A_2),(\mathsf{Gen},\mathsf{PKEnc},\mathsf{PKDec})}(\kappa) = 1] - \frac{1}{2}\right| \leqslant \mathsf{negl}(\kappa).$$

As in the symmetric key, this notion of security can also be strengthened to security against chosen-ciphertext+ attacks (CCA+s), where the attacker needs to distinguish the ciphertext of a chosen message ($b = 0$) from a completely random bitstring ($b = 1$) of corresponding length $|\mathsf{PKEnc}_{pk}(m)|$. Denote this modification of CCA-Dist by $\mathsf{CCA}+$-Dist. This implies that the output of the PKES is indistinguishable from random strings. A PKES $(\mathsf{Gen}, \mathsf{PKEnc}, \mathsf{PKDec})$ is *CCA+-secure*

CCA+-secure

if for every attacker $(\mathsf{A}_1, \mathsf{A}_2)$, there is a negligible function negl such that

$$\mathbf{Adv}^{\mathrm{cca+}}_{(\mathsf{A}_1,\mathsf{A}_2),(\mathsf{Gen},\mathsf{PKEnc},\mathsf{PKDec})}(\kappa) := \left|\Pr[\mathsf{CCA+}\text{-}\mathsf{Dist}_{(\mathsf{A}_1,\mathsf{A}_2),(\mathsf{Gen},\mathsf{PKEnc},\mathsf{PKDec})}(\kappa) = 1] - \frac{1}{2}\right| \leqslant \mathsf{negl}(\kappa).$$

3 MODELS OF STEGANOGRAPHY

After the previous chapter introduced all necessary notions concerning cryptography, this chapter deal with the formal definitions of *provably secure steganography*. Throughout this thesis, we will use multiple different models of steganography, that mainly differ in three aspects:

EFFICIENCY: The first formal definition of provably secure steganography was given by Hopper, von Ahn, and Langford in [HvL09], the running time of a stegosystem was not necessarily efficient, i. e. not bounded by a polynomial in the security parameter κ. While some subsequent works defined efficiency as a requirement (see e. g. [BC05; Ded+09]), Hopper, von Ahn, and Langford make use of the fact that their stegosystems may run for a long time to obtain their results. We thus distinguish between the original definition – which we will call *non-efficient stegosystems* – and the updated notion *efficient stegosystems*.

non-efficient stegosystems

efficient stegosystems

APPLICABILITY: A typical problem that arises when one designs a stegosystem concerns their *applicability*: On which kind of channels should or stegosystem work? One could for example design a stegosystem that works for a concrete channel where the documents are 200 JPEG pictures of size 600×600 pixels that we know in beforehand. Such a stegosystem is called a *white-box stegosystem*, as the stegosystem has complete knowledge of the channel. Typically one wants to design more general stegosystems. For example, it might be appropriate to design a stegosystem that works for all channels that contain JPEG pictures of size 600×600 pixels. As the stegosystem still has some knowledge about the documents, such a system is called a *grey-box stegosystem*. The most general form of a stegosystem is a stegosystem that works *on every channel* (containing sufficiently many documents). We call such a system a *universal stegosystem* or a *black-box stegosystem*. As we try to give as general results as possible in this thesis, we will develop grey-box or black-box stegosystems for our positive results and rule out white-box stegosystems for our negative results.

This nomenclature is taken from [LRW13].

white-box stegosystem

grey-box stegosystem

universal stegosystem

black-box stegosystem

KEY-SYMMETRY: As in the cryptographic setting, the stegoencoder needs a key k to encode the message into the channel and the stegodecoder also needs a key k'. If $k = k'$ we speak of a *symmetric-key stegosystem* or *secret-key stegosystem*. In contrast, if $k \neq k'$ and k is publicly known and k' is kept secret, we call such a system a *public-key stegosystem*. Furthermore, we denote

symmetric-key stegosystem

secret-key stegosystem

public-key stegosystem

the publicly known key k as *pk* (for public key) and the secret key k' as *sk* (for secret key). Depending on the setting we will also analyze different security notions.

To help the reader to keep track which of these $2 \cdot 3 \cdot 2 = 12$ configurations we currently use, the names of the chapters typically contain all information about the notions used in the chapter. We will also always give a short description about these aspects in the first few sentences of the chapter.

3.1 UNSUSPICIOUS COMMUNICATION

In order to formalize that the output of a secure stegosystem is indistinguishable from unsuspicious communication, we first need a mean to define this unsuspicious communication. We will do this via the notion of a *channel*. We will think of this unsuspicious communication as the unidirectional transmission of *documents* from Alice to Bob and will model this as a probability distribution upon those documents. This distribution indicates the probability that Alice sends a certain document to Bob. There are two more things we need to consider to make this model realistic and useful for us. First, the probabilities may change over the time depending on the already sent documents. If Alice sends Bob a postcard from the beach, it is quite unlikely (though not impossible) that the next postcard that Bobs get will come from the Antarctic. This change of the probability distribution will be reflected by something we call the *history* – the sequence of already transmitted documents. Second, larger security parameters typically allow us to send larger messages. Hence, the amount of information needed to hide those messages also grows. To hide those messages, there are to approaches to handle the need for more information:

- In the first approach used by Hopper, von Ahn, and Langford in [HvL09], it is assumed that the size of the documents is independent from the security parameter and thus treated as a constant. In order to have a large enough entropy to handle larger messages, Hopper, von Ahn, and Langford do not deal with single documents, but rather with sequences of documents of sufficient length. This model was critized by Lysyanskaya and Meyerovich in [LM06] as one should only be able to look at the distribution with history h containing the document d *after* the document d was transmitted to Bob especially if the the size of documents is very small.

- In the second approach that we will use, we assume that the size of the document depends on the security parameter, i. e. the entropy of a single document is high enough already. This approach is more general then the first one as we will simply interpret a sequence of constant-sized objects as a single document. This simplifies the analysis and our notation as we can always directly talk about documents and not about sequences.

Example. Let us look at the example that Alice send Bob pictures from her holiday. Suppose that every picture is encoded in JPEG and of size 600×600 pixels. Denote the set of all such pictures by Pics. Furthermore suppose that on security parameter κ, we want to embed messages of length $m(\kappa)$.

In the first approach, a document d would consist of a *single* picture, i. e. $d \in$ Pics and we would only deal with sequences of pictures of length $\approx m(\kappa)$. Hence, our channel would be a probability distribution on Pics, but our stegosystem would only deal with sequences taken from this distribution.

In the second approach, a document d would already consists of sequence of pictures, i. e. $d \in \text{Pics}^{m(\kappa)}$. Hence, our channel would be a probability distribution on $\text{Pics}^{m(\kappa)}$ and our stegosystem would also directly deal with elements of this distribution. $\diamond$

Formally, a *channel* $\mathcal{C}$ on the alphabet Σ is a function that maps an element $h \in \Sigma^*$ – the *history* – and a number $n \in \mathbb{N}$ – the *document length* – to a probability distribution on Σ^n. We will denote this probability distribution by $\mathcal{C}_{h,n}$ instead of $\mathcal{C}(h,n)$. Typically, we will implicitly assume that $\Sigma = \{0,1\}$ to simplify the following analysis concerning the amount of information that is present in the channel $\mathcal{C}$. The *min-entropy of a channel* $H_\infty(\mathcal{C},n)$ for a channel $\mathcal{C}$ and a natural number $n \in \mathbb{N}$ is defined as $H_\infty(\mathcal{C},n) = \min_{h \in \Sigma^*}\{H_\infty(\mathcal{C}_{h,n})\}$. As demonstrated in [HvL09], the number of bits embeddable in a *single document* is bounded by $H_\infty(\mathcal{C},n)$.

channel
history
document length
min-entropy of a channel

3.2 STEGOSYSTEMS

We are now able to finally describe the notion of a stegosystem. As discussed in the beginning of the section, we will follow the definition of [HvL09] and will not assume that a stegosystem needs to run in polynomial time. In order to reduce the redundancy of this work, we will only define secret-key stegosystems and then explain the (relatively minor) differences to public-key systems later on. Let μ, n and ℓ be polynomials throughout this chapter that will model that the stegoencoder upon security parameter κ takes a message of length $\mu(\kappa)$ and embedds it into $\ell(\kappa)$ documents of length $n(\kappa)$. We thus call μ the *message length*, n the *document length* and ℓ the *output length* of the stegosystem.

A *stegosystem* (Gen, SEnc, SDec) is a triple of PTMs such that the algorithm $\text{Gen}(1^\kappa)$ produces a key $k \in \{0,1\}^*$ with $|k| \geqslant \kappa$. The *stegoencoder* SEnc takes as input the key k, a message $m \in \{0,1\}^{\mu(\kappa)}$, a history $h \in \left(\Sigma^{n(\kappa)}\right)^*$ and some state informations $s \in \{0,1\}^*$ and

message length
document length
output length
stegosystem
stegoencoder

outputs a *single document* $d \in \Sigma^{n(\kappa)}$ and updated state information $s' \in \{0,1\}^*$. Its goal is to embed a piece of m into the document d. It will also have access to samples of the probability distribution $\mathcal{C}_{h,n(\kappa)}$. The complete output of the run of the stegoencoder is denoted by $\mathsf{SEnc}^{\mathcal{C}}(k, m, h)$ and defined by the following scheme:

$\mathsf{SEnc}^{\mathcal{C}}(k, m, h)$

Input: Key k, message m, history h
$s := \varnothing$ ▷ initialize the empty state
for $i = 1, 2, \ldots, \ell(\kappa)$ **do**
$(d_i, s) \leftarrow \mathsf{SEnc}^{\mathcal{C}_{h,n(\kappa)}}(k, m, h, s)$
$h := h \parallel d_i$
end for
return $d_1, d_2, \ldots, d_{\ell(\kappa)}$

$\mathsf{SEnc}^{\mathcal{C}}(k, m, h)$: Complete run of stegoencoder SEnc

3.3 SECURITY NOTIONS

4 UNIVERSAL NON-EFFICIENT SECRET-KEY STEGANOGRAPHY

5 UNIVERSAL EFFICIENT SECRET-KEY STEGANOGRAPHY

6 UNIVERSAL EFFICIENT PUBLIC-KEY STEGANOGRAPHY

7 EFFICIENT PRIVATE-KEY STEGANOGRAPHY FOR PATTERN CHANNELS

8 CONCLUSION

[BC05] Michael Backes and Christian Cachin. "Public-Key Steganography with Active Attacks." In: *TCC*. Vol. 3378. Lecture Notes in Computer Science. Springer, 2005, pp. 210–226.

[Bel+97] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. "A Concrete Security Treatment of Symmetric Encryption." In: *FOCS*. Full version available under http://web.cs.ucdavis.edu/~rogaway/papers/sym-enc.pdf. IEEE Computer Society, 1997, pp. 394–403.

[BR16] Sebastian Berndt and Rüdiger Reischuk. "Steganography Based on Pattern Languages." In: *LATA*. Vol. 9618. Lecture Notes in Computer Science. Springer, 2016, pp. 387–399.

[Ded+09] Nenad Dedic, Gene Itkis, Leonid Reyzin, and Scott Russell. "Upper and Lower Bounds on Black-Box Steganography." In: *J. Cryptology* 22.3 (2009), pp. 365–394.

[DH76] Whitfield Diffie and Martin E. Hellman. "New directions in cryptography." In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654.

[HvL09] Nicholas J. Hopper, Luis von Ahn, and John Langford. "Provably Secure Steganography." In: *IEEE Trans. Computers* 58.5 (2009), pp. 662–676.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.

[LRW13] Maciej Liśkiewicz, Rüdiger Reischuk, and Ulrich Wölfel. "Grey-box steganography." In: *Theor. Comput. Sci.* 505 (2013), pp. 27–41.

[LM06] Anna Lysyanskaya and Mira Meyerovich. "Provably Secure Steganography with Imperfect Sampling." In: *Public Key Cryptography*. Vol. 3958. Lecture Notes in Computer Science. Springer, 2006, pp. 123–139.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005. ISBN: 978-0-521-83540-4.

[Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994. ISBN: 978-0-201-53082-7.

INDEX

DECLARATION

Put your declaration here.

Lübeck, March 2016

Sebastian Berndt

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/