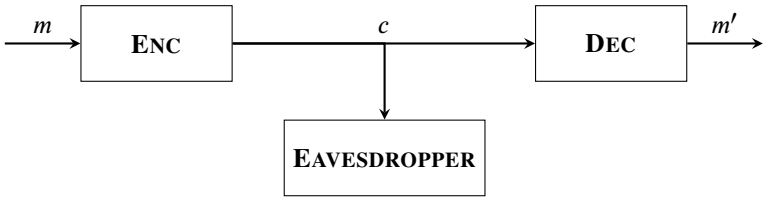
1 One Time Pad

1.1 What is [NOT] CRYPTOGRAPHY

1.1.1 Introduction

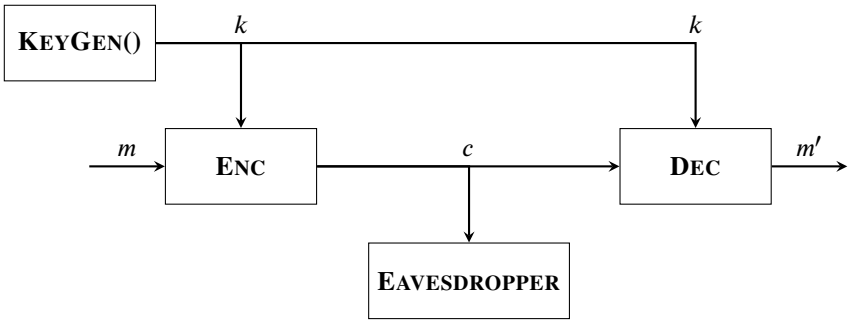
In the idealized model we assume that Alice wants to send a message m (*privately*) to Bob. Alice will modify the message m, also called ***plaintext***, with any method to create a ***ciphertext*** c which will be actually sent to Bob. This transformation is also called encryption (***Enc***). After receiving the ciphertext c, Bob will reverse the step of transforming by using a decription algorithm (***Dec***) to (*hopefully*) get the original plaintext m.
NOTE: We are not trying to hide that a message is sent, so an **EAVESDROPPER** (an attacker between Alice and Bob) can obtain the ciphertext c at any instance. Hiding the existence of communication is called *steganography*.



1.1.2 Kerckhoff's principle

The method must not be required to be secret, and it must be able to fall into the enemy's hands without causing any inconvenience.

So if the algorithm do not need to be secret, there must be additional information in the system, which is kept secret from any **EAVESDROPPER**. This information is called a **(secret) key** k.



1.2 Specifics of ONE-TIME PAD

A *one-time pad* often uses a secret key in the form of a bit string of length λ. The plain- and ciphertexts are also λbit-strings.
The construction of such an one-time pad looks as follows:

KEYGEN()	ENC(k, $m \in \{0,1\}^\lambda$)	DEC(k, $c \in \{0,1\}^\lambda$)
$k \leftarrow \{0,1\}^\lambda$	$c := k \oplus m$	$m' := k \oplus c$
return k	**return** c	**return** m'

Recapture that $k \leftarrow \{0,1\}^\lambda$ means that k is sampled uniformly from the set of λbit-strings.
Further we claim that for all $k, m \in \{0,1\}^\lambda$ it is true, that $Dec(k, Enc(k, m)) = m$.
Otherwise the usage of one-time pad would be silly.
For security reasons we want to say about the encryption scheme that an EAVESDROPPER (who does not know k) cannot leran anything about the message m.
In the end we need to claim that an encryption algorithm is secure if for every $m \in \{0,1\}^\lambda$ the distribution EAVESDROP(m) is the **uniform distribution** of $\{0,1\}^\lambda$, explicitly for every $m, m' \in \{0,1\}^\lambda$ the distributions EAVESDROP(m) and EAVESDROP(m') are identical.

2 The Basics of Proveable Security

2.1 Generalizing and Abstracting One-Time Pad

2.1.1 Syntax & Correctness

A **symmetric key encryption *(SKE)* scheme** consissts of the following algorithms:
- ▷ KEYGEN: a randomized algorithm that outputs a ***key*** $k \in K$
- ▷ ENC: a (*possibly randomized*) algorithm that takes a key $k \in K$ and ***plaintext*** $m \in M$ as input, and ouputs a ***ciphertext*** $c \in C$
- ▷ DEC: a deterministic algorithm that takes a key $k \in K$ and a ciphertexts $c \in C$ as input, and outputs a plaintext $m \in M$

K is called the ***key space***, M the ***message space***, and C the ***ciphertext space*** of the scheme. Often the entire scheme (with all its algorithms) is referred to as Σ. Each component is then referred to as Σ.KEYGEN, Σ.ENC, Σ.DEC, $\Sigma.K$, $\Sigma.M$, and $\Sigma.C$.

Furthermore we define an encryption scheme to be ***correct*** if for all $k \in K$ and all $m \in M$:

$$Pr[\Sigma.\text{DEC}(k, \Sigma.\text{ENC}(k, m)) = m] = 1$$

In other words, decrypting a ciphertext, using the same key used for encryption, ***always*** results in the original plaintext.

2.2 Towards an Abstract Security Definition

With the properties of one-time pad shown in the first chapter a first attempt can be made to define security:
For all $m \in \{0,1\}^\lambda$, the output of the following subroutine is uniformly distributed over $\{0,1\}^\lambda$:

```
EAVESDROP(m ∈ {0,1}^λ):
  k ← {0,1}^λ
  c := k ⊕ m
  return c
```

This property is too specific to one-time pad. To get a more general-purpose security definition, we can write the subroutine using the totally generic encryption scheme Σ:

```
EAVESDROP(m ∈ Σ.M):
  k ← Σ.K
  c := Σ.ENC(k, m)
  return c
```

Such an encryption scheme Σ is secure if, for all $m \in \Sigma.M$, the output of this subroutine is uniformly distributed over $\Sigma.C$.

2.2.1 Adversary as Distinguishers

Filler

2.2.2 Critically Analyzing a Security Definition

Filler

2.2.3 Chosen Plaintext Attack Template

Filler

2.3 Provable Security Fundamentals

2.3.1 Libraries & Interfaces

Filler

2.3.2 Semantics & Scopes

Filler

2.3.3 Interchangeability

Filler

2.3.4 Security Definition, Using New Terminology

Filler

2.4 How to Prove Security with the Hybrid Technique

2.4.1 Chaining Several Components

Filler

2.4.2 One-Time Secrecy of One-Time Pad

Filler

2.5 How to Demonstarte Insecurity with Attacks

Filler