Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics
Department of Computer Science

Properties of Cryptographic Hash Functions

Michal Rjaško

Advisor: RNDr. Martin Stanek, PhD.

Bratislava 2008

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics
Department of Computer Science

Properties of Cryptographic Hash Functions

Diploma Thesis

Michal Rjaško

Advisor: RNDr. Martin Stanek, PhD.

Bratislava 2008

Acknowledgments

I want to thank my advisor Martin Stanek for his invaluable guidance, the materials he gave me and many useful suggestions during my work on this thesis.

Special thanks goes to my family and friends for their great support and care.

I hereby declare that I wrote this thesis by myself, only with help of the referenced literature, under the careful supervision of my thesis advisor.

.....

Abstract

Cryptographic hash functions are corner-stones of current cryptography. Recently, NIST (National Institute for Standards and Technology) has announced a public competition to develop a new hash standard called AHS (Advanced Hash Standard). We summarize the basic properties that the new cryptographic hash standard should preserve, give formal definitions of them and work out all of the implications or separations among these definitions. Some of the implications/separations have been proven before, others appear to be new. We provide two types of the implication and separation, conventional and provisional. While the conventional implication (separation) carries the usual semantics of the word implication (separation), the strength of the provisional implication or separation depends on a particular hash function. We show that a property pseudo-random oracle introduced by Coron, Dodis, Malinaud and Puniya is (as expected) the strongest one, since it implies almost all of the other properties. We also discuss the practical use of the pseudo-random oracle and multi-property preserving transforms introduced by Bellare and Ristenpart.

Keywords: cryptographic hash function, provable security, hash function properties, collision resistance, pseudo-random oracle

Contents

List of Figures

List of Tables

Introduction

Cryptographic hash functions are basic primitives, widely used in many applications, from which more complex cryptosystems are build. In the last few years many popular hash functions such as MD5 or SHA1 have been broken, also some structural flaws in popular constructions (e.g. Merkle-Damgård construction) of hash functions have been found. These findings caused great activity in the cryptographic community, which in January 2007 escalated into NIST's (National Institute for Standards and Technology) announcement of a public competition for a new hash standard, similar to one when AES was standardized.

By this thesis we try to participate in the development of a new hash standard by summarizing security properties of cryptographic hash functions. We extend the work by Rogaway and Shrimpton [13], where they provide definitions of seven security properties – notions of preimage resistance, second-preimage resistance and collision resistance, and they also give all the relationships among these definitions. To these seven security properties we add five more — definitions of unforgeability, two notions of chosen target forced prefix preimage resistance, pseudo random function and pseudo random oracle. Between each two of all twelve definitions, we provide implication or separation with exact proof, except those implications or separations proven in [13].

The Thesis is divided into two parts. In the first part we introduce some basic notations and definitions, then we give formal definitions of twelve security properties a cryptographic hash function should preserve. At the end of the first part we discuss when a hash function is secure in some sense (i.e. when it preserves some property), and we give formal definitions of implication and separation between the security properties. In the second part of the Thesis we provide the relationships (implication or separation) with exact proofs among the definitions from the first part.

Informally speaking, a hash function is a function that maps messages of an arbitrary length to strings of fixed length. An output of a hash function with some message on its input is called a hash of the message. One of the many applications of hash functions are digital signatures. Digital signatures are used as an electronic replacement of

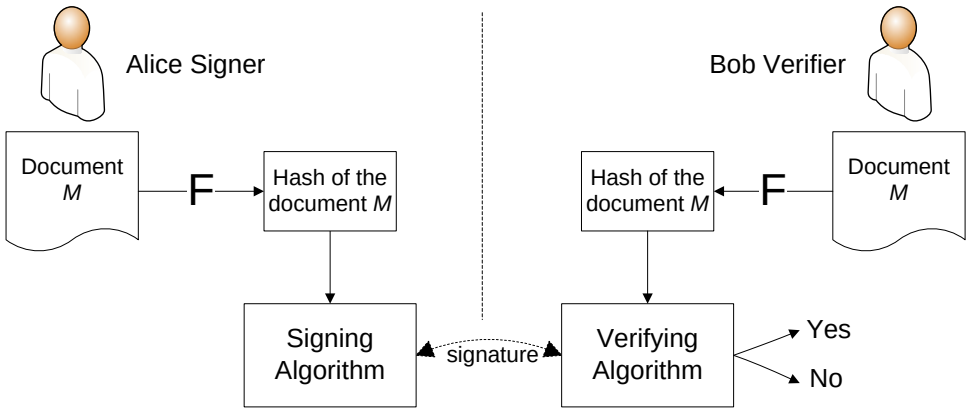

Figure 1: Scheme of digital signature

classical hand-written signatures. Many digital signature schemes have been designed so far, however all of them share the same basic scheme (see Figure 1). Every digital signature scheme is based on an asymmetric cipher. Suppose that Alice wants to send Bob some signed document. To be able to perform the signing process, Alice must have a private key and share some public key. The private key is used by the signing algorithm to sign the document and Bob uses Alice's public key in the verifying algorithm to verify, whether the document was really signed by Alice. As asymmetric ciphers are very computationally demanding, instead of signing the whole document, only hash of the document is signed.

Digital signatures are good example for presenting properties that hash functions should preserve. One important property is efficiency. Hash function algorithm should be very fast, as very long documents can be signed. Thus hash functions based on a computationally hard problem (factorization or discrete logarithm) are out of the question (even if they provide some provable security), as such hash functions are slow.

Fundamental property of digital signatures used in practice is that for some digitally signed document A no one can produce document B that has the same signature as A. Similar property is that no one can produce two different documents that have the same signature. Thus a hash function used in a digital signature scheme must guarantee that for some document A no one can produce another document B that hashes to the same hash as A (otherwise A and B would have the same signature). Similarly, it must guarantee that no one can produce two different documents that hash to the same hash. Thus we have two properties that hash functions should preserve – the first is called *second-preimage resistance* and the latter *collision resistance*. When we are designing a hash function for digital signatures, we would like to know the

relationship between these properties. If we knew that collision resistance implies second-preimage resistance, then instead of proving preservation of each property separately, we would only need to prove, that our hash function is collision resistant and therefore it is automatically second-preimage resistant. In this thesis we try to help designers of hash functions by summarizing all properties a "good" cryptographic hash function should preserve and by giving relationships among these properties.

Chapter 1

Definitions

A cryptographic hash function is a function $F : \mathcal{M} \to \mathcal{Y}$ where $\mathcal{M}$ is a possibly infinite nonempty set of strings, $\mathcal{Y}$ is a finite nonempty set of strings and $|\mathcal{M}| > |\mathcal{Y}|$. Members of the domain $\mathcal{M}$ are called messages, members of the set $\mathcal{Y}$ are called images or hashes. However, not every such function F is a "good" cryptographic hash function. There are three main properties which a "good" cryptographic hash function has to preserve.

- *preimage resistance* – for essentially all hashes y, it is difficult to find message m which is hashed to y.
- *2nd-preimage resistance* – for given message m, it is difficult to find message m', which hashes to the same value as the message m, i.e. $F(m) = F(m')$.
- *collision resistance* – it is difficult to find two different messages m and m' such that $F(m) = F(m')$.

However, the properties above are written informally, what can lead to a lot of ambiguity. In this chapter we discuss which properties should a cryptographic hash function preserve and give formal definitions of them.

In [13] Rogaway and Shrimpton provide definitions for various notions of collision resistance, preimage resistance and second preimage resistance. They also give all the relationships among the definitions. Bellare and Ristenpart in [2] give another three properties: pseudo-random function, pseudo-random oracle and MAC. Finally, Kohno and Kelsey in [8] proposed a new type of attack called "herding attack" and they introduced new property called chosen target forced prefix preimage resistance, which if a hash function preserves, then it is resistant to the herding attack.

Formal definitions of these properties use hash functions in a different setting than we presented. Hash function has one more input, so called *dedicated-key* input, which extends a hash function to a *hash function family*.

Definition 1 (Hash function family)**.** A *hash function family* is a function $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, where $\mathcal{K} = \{0,1\}^k$, $\mathcal{Y} = \{0,1\}^y$ for some integers $k, y > 0$ and $\mathcal{M} = \{0,1\}^*$. Set $\mathcal{K}$ is called *key space*, number y is called *hash length* of H.

The reason why we use the hash function family instead of the hash function is its universality, which leads to easier construction of message authentication codes (MACs), where some secret key is needed to build MAC (more about message authentication codes can be seen in Section 1.2.5). Hash function family has some other benefits, which are discussed in [2], however significant drawback of hash function family is its loss in efficiency (we need k more bits to process every message block).

Now we introduce some notations used in this thesis. We write $M \xleftarrow{\$} \mathcal{S}$ for the experiment of choosing random element from the distribution $\mathcal{S}$. If $\mathcal{S}$ is a finite set, then M is chosen uniformly from $\mathcal{S}$. Concatenation of finite strings M_1 and M_2 we denote by $M_1||M_2$ or simply M_1M_2. Bitwise complement of string M we write as $\overline{M}$. Empty string is denoted by μ. If i is an integer, then $\langle i \rangle_r$ is r-bit string representation of i. Let $Func(D,R)$ represent the set of all functions $\rho : D \to R$ and let $RF_{D,R}$ be a function chosen randomly from the set $Func(D,R)$ (i.e. $RF_{D,R} \xleftarrow{\$} Func(D,R)$). We sometimes write $RF_{d,r}$ when $D = \{0,1\}^d$ and $R = \{0,1\}^r$. By $\text{Prefix}_n(M)$ we denote the n-bit prefix of message M, similarly by $\text{Suffix}_n(M)$ we denote the n-bit suffix of M.

Definition 2 (Adversary)**.** An *adversary* is a *random access machine* (RAM) with any number of inputs (i.e. it can access ith bit of input j in *unit time*) that can toss a coin in unit time (i.e. it can choose a sample from the set $\{0,1\}$ in a unit time). *Running time* of an adversary A on some input is the *average* time needed to compute an output (relative to some fixed RAM model) plus the description size of A (relative to some fixed coding of RAMs).

It is important to include the description size of an algorithm A into the running time of an adversary. For example consider, that we are constructing an adversary A which finds preimages for a hash function $F : \mathcal{M} \to \mathcal{Y}$. We hardwire into A an array P of pairs $[M, Y]; Y = F(M)$ sorted by the second component, such that P includes all possible images $Y = F(M)$. Therefore the size of the array P is at most $|\mathcal{Y}|$ (note that not all of the messages $M \in \mathcal{M}$ are included in P, but P includes all images that F outputs). The adversary A takes as an input image Y and searches in P for one pair (M, Y). As P is sorted, A can use binary search, which runs in logarithmic time.

Thus A finds preimages for a set $\mathcal{Y}$ with cardinality 2^y in time $O(y)$, what is feasible even for a large y. However, it is practically unfeasible to construct such algorithm A because of its complexity. Let us assume that output size of F is 128 bits and maximum message length in P is 256 bits. Therefore $256 \cdot 2^{128}$ bits are needed to store such array P in a memory, what is about 2^{96} terabytes.

Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family. We denote by $\text{Time}_{H,n}$ the running time of an algorithm P (i.e. some random access machine) computing H that has the best worst case running time over all inputs $(K, M); K \in \mathcal{K}; M \in \mathcal{M}; |M| = n$, that is, any other algorithm P' computing H has the worst case running time over all the inputs $(K, M); K \in \mathcal{K}; M \in \mathcal{M}; |M| = n$ greater or equal to P's. Informally speaking, $\text{Time}_{H,n}$ is the time needed to compute H_K on any input of length n.

In this work we will often use the term *random oracle*. It is an abstract function, which we are unable to construct, however, it is widely used in cryptography, mainly due to the so called random oracle model, firstly introduced by Bellare and Rogaway [4], where the security of cryptosystems is proven under the assumption, that any party has access to a random oracle. Instead of proving that some system is secure with the particular hash function F (e.g. F being SHA-1), one assumes, that F is an "ideal" hash function (i.e. random oracle) and proves the security of the system under this assumption. Such formal proof in the random oracle model indicates, that there are no structural flaws in the construction of the system, and therefore we can believe, that no such flaws will appear in the system with a particular well-constructed hash function F.

Definition 3 (Random oracle)**.** A *random oracle* is a function $f : D \to R$ chosen uniformly randomly from the set of all functions from D to R (i.e. from the set $Func(D, R)$), where R is a finite set.

Thus $RF_{D,R}$, already defined, is a random oracle. Based on the definition of random oracle, we can define *ideal hash function*.

Definition 4 (Ideal hash function)**.** A hash function $F : \mathcal{M} \to \mathcal{Y}$ is an *ideal hash function* if every attack against F has the same complexity as against a random oracle (i.e. any adversary performing the attack against F has the running time greater or equal to the running time of the adversary performing attack against random oracle).

We will also use the term *negligible*. A function $f : \mathsf{N} \to \mathsf{R}^+$ is negligible, if it descends faster than any polynomial powered to -1. The formal definition is following.

Definition 5 (Negligible function)**.** A function $f : \mathsf{N} \to \mathsf{R}^+$ is *negligible*, if for every constant $c > 0$, there exists an integer $N_0 \in \mathsf{N}$, such that for all integers $n > N_0$ it

holds

$$f(n) < \frac{1}{n^c}.$$

We say that a constant c is negligible, when it is affected by some security parameter k and the function $c(k)$ is negligible. For example consider, that we have a hash function F, for which any adversary A has the probability of success $\frac{1}{2^k}$, where k is some security parameter (e.g. hash length of F). Thus the probability of A's success is negligible and so we say that F is secure against the attack that adversary A is performing.

1.1 Constructions of hash functions

In this Section we describe the common way of constructing hash functions $F : \mathcal{M} \to \mathcal{Y}$ — the *iterated* construction and its Merkle-Damgård strengthening.

1.1.1 Iterated construction

In order to process messages of an arbitrary length, the iterated hash functions process messages in blocks of fixed length r. However the length of message does not need to be divisible by r. Thus some message preprocessing is needed, which pads the message to multiple of block length and eventually makes some other modifications (e.g. adds binary interpretation of message length to the end). Then the padded message is divided into blocks $m_1, m_2, \ldots, m_t$. The blocks are processed consequently using the so called "compression" function $f : \{0,1\}^{y+r} \to \mathcal{Y}$ and the ongoing hash Y_i is produced:

$$\begin{aligned} Y_0 &= IV, \\ Y_i &= f(m_i, Y_{i-1}), \qquad i = 1, 2, \ldots t, \end{aligned}$$

where IV is some constant initialization vector. The last ongoing hash Y_t is also the output of the iterated hash function F, however sometimes an output transformation g is applied to Y_t.

The iterated construction of hash function is the most common construction of popular hash functions. Mainly it is due the matter, that when a compression function f has some "good" properties, we are able to prove these "good" properties for whole function F, as it is in *Merkle-Damgård strengthening* of iterated hash function.

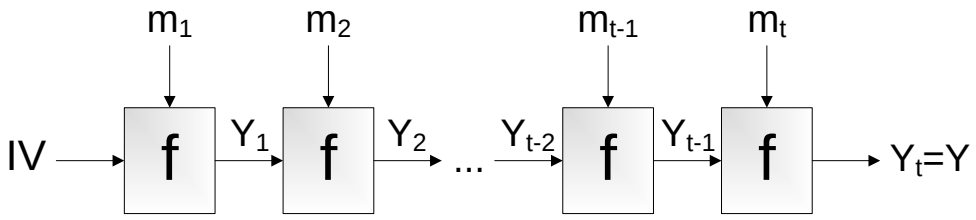

Figure 1.1: Iterated construction of hash function.

1.1.2 Merkle-Damgård strengthening

Merkle-Damgård strengthening extends a collision resistant compression function $f : \{0,1\}^{y+r} \to \mathcal{Y}$ to a *collision resistant* hash function $F : \mathcal{M} \to \mathcal{Y}$. A message M of length l is divided into blocks $x_1, x_2, \ldots, x_t$ of length r bits, where the last block is filled up with zeros if needed. Then an additional block x_{t+1} is added, which contains binary interpretation of length l (if $l \geq 2^r$, then we add more than one block). After that we iterate over all blocks and the ongoing hash value is produced:

$$\begin{aligned} Y_0 &= 0^n, \\ Y_i &= f(Y_{i-1}||m_i), \qquad i = 1, \ldots t+1. \end{aligned}$$

The output of the hash function F is the last ongoing hash Y_{t+1}.

Propositioin 1. *Let $f : \{0,1\}^{y+r} \to \mathcal{Y}$ be a collision resistant function. Then function $F : \mathcal{M} \to \mathcal{Y}$ described above is collision resistant too.*

We do not provide the proof of this proposition, as it is not in our main interest. The proof is quite straightforward and can be found in [10].

Both iterated construction and Merkle-Damgård strengthened iterated construction can be used to build also hash function families $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$. If the size of the key space $\mathcal{K}$ can be equal to the size of the set $\mathcal{Y}$, we can initialize the iteration with the key instead of the initialization vector. That is, the message preprocessing and the whole iteration process are the same as they are in the case of standard hash functions, except the first step of the iteration, where Y_0 is not set to the initialization vector, but to the key K. We note that in this way we can modify most of the popular hash functions such as SHA1, MD5 or SHA2 versions to accept dedicated-key input and therefore to build hash function families.

If the size of the key set has to be different from the size $\mathcal{Y}$, we can modify compression function to accept additional (dedicated-key) input $f : \mathcal{K} \times \{0,1\}^{y+r} \to \mathcal{Y}$ and iterate over all blocks with such compression function:

$$\begin{aligned} Y_{0,K} &= IV, \\ Y_{i,K} &= f(K, Y_{i-1}||m_i), \qquad i = 1, \ldots t. \end{aligned}$$

The output of $H(K, M)$ is then equal to $Y_{t,K}$.

Now we can proceed to the formal definitions of cryptographic hash function security.

1.2 Definitions of hash function security

Here we give the formal definitions of hash function security notions. Notions for *preimage resistance*, *second-preimage resistance* and *collision resistance* were defined in [13] by Rogaway and Shrimpton. Bellare and Ristenpart in [2] defined notions for *MAC*, *pseudo random function* and *pseudo random oracle*. Finally, the *chosen target forced prefix preimage resistance* notion (CTFP) was defined in [8] by Kelsey and Kohno.

1.2.1 Preimage resistance

A hash function is preimage resistant, when it is difficult to find a preimage for a point in the range of the hash function. There are several ways how to formalize this intuition in the sense of hash function family.

Definition 6 (Preimage resistance)**.** Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family and let λ be a number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. Let A be an adversary. Then we define:

$$\begin{aligned} \mathbf{Adv}_H^{\mathrm{Pre}[\lambda]}(A) &= \Pr\left[K \xleftarrow{\$} \mathcal{K}; M \xleftarrow{\$} \{0,1\}^\lambda; Y \leftarrow H_K(M); M' \leftarrow A(K, Y) : H_K(M') = Y\right] \\ \mathbf{Adv}_H^{\mathrm{ePre}}(A) &= \max_{Y \in \mathcal{Y}} \left(\Pr\left[K \xleftarrow{\$} \mathcal{K}; M \leftarrow A(K) : H_K(M) = Y\right]\right) \\ \mathbf{Adv}_H^{\mathrm{aPre}[\lambda]}(A) &= \max_{K \in \mathcal{K}} \left(\Pr\left[M \xleftarrow{\$} \{0,1\}^\lambda; Y \leftarrow H_K(M); M' \leftarrow A(Y) : H_K(M') = Y\right]\right) \end{aligned}$$

We say that H is (t, L, ε)-xxx for xxx $\in \{\mathrm{Pre}, \mathrm{aPre}\}$ if any adversary A running in time at most t and outputting messages of length less than or equal to L has advantage $\mathbf{Adv}_H^{\mathrm{xxx}[\lambda]}(A) \leq \varepsilon$ for all λ such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. We say that H is (t, L, ε)-ePre if any adversary A running in time at most t and outputting messages of length less than or equal to L has advantage $\mathbf{Adv}_H^{\mathrm{xxx}}(A) \leq \varepsilon$.

Note that the parameter $[\lambda]$ is added to the advantage of adversaries to avoid random selection from the possibly infinite set $\mathcal{M}$ and also to bound the length of randomly selected messages.

The first definition (*preimage resistance*) is the standard way how to define preimage resistance for a hash function family. However, a hash function family H, which for every key K maps the message 0 to the image 0^y (i.e. $H_K(0) = 0^y$ for every key K), can be preimage resistant (i.e. advantage of any adversary is negligible), even if we know the preimage for image 0^y. This problem solves *everywhere preimage resistance*, which captures the intuition, that it is infeasible to find a preimage for every image – whatever image is selected, it is difficult to find its preimage. Third definition, *always preimage resistance*, strengthens the first one in the following way. Consider a hash function family H, which for the particular key K_0 maps every message to the image 0^y (i.e. $H_{K_0}(M) = 0^y$ for every message $M \in \mathcal{M}$). The probability of choosing the key K_0 is negligible, therefore the hash function family H can be preimage resistant (if for every key $K \neq K_0$ it is hard to find preimages). However, H isn't always preimage resistant, as trivial adversary, which always returns message 0, would prevail against H in always preimage attack. Thus *always preimage resistance* captures the intuition that it is hard to find preimages for *every* function H_K from a hash function family H.

Note that we do not bound the running time t of adversaries here. However, in order to define the security of a hash function family in some sense (Pre, ePre, aPre), such bounding is necessary. We will discuss this in section 1.3.

1.2.2 Second-preimage resistance

A common way of defining the second-preimage resistance is as follows. We say that a hash function F is second-preimage resistant if for a message M, it is hard to find a different message M', which hashes to the same image, i.e. $F(M) = F(M')$. We call such messages M and M' *partners*. Again, in the hash function family sense, there are few possibilities, how to formally define this intuition.

Definition 7 (Second-preimage resistance)**.** Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family and let λ be a number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. Let A be an adversary. Then we define:

$$\begin{aligned}
\mathbf{Adv}_H^{\mathrm{Sec}[\lambda]}(A) &= \Pr\left[K \xleftarrow{\$} \mathcal{K}; M \xleftarrow{\$} \{0,1\}^\lambda; M' \leftarrow A(K,M) : (M \neq M') \wedge (H_K(M) = H_K(M'))\right] \\
\mathbf{Adv}_H^{\mathrm{eSec}[\lambda]}(A) &= \max_{M \in \{0,1\}^\lambda} \left(\Pr\left[K \xleftarrow{\$} \mathcal{K}; M' \leftarrow A(K) : (M \neq M') \wedge (H_K(M) = H_K(M'))\right]\right) \\
\mathbf{Adv}_H^{\mathrm{aSec}[\lambda]}(A) &= \max_{K \in \mathcal{K}} \left(\Pr\left[M \xleftarrow{\$} \{0,1\}^\lambda; M' \leftarrow A(M) : (M \neq M') \wedge (H_K(M) = H_K(M'))\right]\right)
\end{aligned}$$

We say that H is (t, L, ε)-xxx for xxx $\in \{\mathrm{Sec}, \mathrm{eSec}, \mathrm{aSec}\}$ if any adversary A running

in time at most t and outputting messages of length less than or equal to L has advantage $\mathbf{Adv}_H^{\mathrm{xxx}[\lambda]}(A) \leq \varepsilon$ for all λ such that $\{0,1\}^\lambda \subseteq \mathcal{M}$.

The first definition, *second-preimage resistance*, is the standard way how to define second-preimage resistance. However, it is different from the definition of a second-preimage resistance for a hash function $F : \mathcal{M} \to \mathcal{Y}$, as the hash function F and the hash function family H are syntactically different objects. The definition (Sec) is equivalent to the classical version of second-preimage resistance for a randomly chosen hash function F from the hash function family H. The second definition, *everywhere second-preimage resistance*, captures the intuition, that it is hard to find a partner for *every* message M from a domain set $\mathcal{M}$. Everywhere second-preimage resistance is also known as *target collision resistance* used in [2], or a *universal one-way hash function family* defined in [12]. The third definition, *always second-preimage resistance*, is strengthening of the first one in the way, that for an always second-preimage resistant hash function family H and *every* key K, it is hard to find partner M' for a randomly chosen message M, such that $H_K(M) = H_K(M')$. A second-preimage resistant hash function family can have a "weak" key K_0, such that it is possible to find second-preimages for a hash function H_{K_0}. On the other hand, for any always second-preimage resistant hash function family it *must* be hard to find second-preimages for *all* keys from $\mathcal{K}$.

1.2.3 Collision resistance

Very important property of every "good" cryptographic hash function is collision resistance. A hash function is collision resistant, if it is hard to find two different messages, that hashes to the same image (i.e. it is hard to find two partners). In the hash function family sense, a formal definition is following.

Definition 8 (Collision resistance)**.** Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family. Let A be an adversary. Then we define:

$$\mathbf{Adv}_H^{\mathrm{Coll}}(A) \quad = \quad \Pr\left[K \xleftarrow{\$} \mathcal{K}; (M, M') \leftarrow A(K) : (M \neq M') \wedge (H_K(M) = H_K(M'))\right]$$

We say that H is (t, L, ε)-Coll if any adversary A running in time at most t and outputting messages of length less than or equal to L has advantage $\mathbf{Adv}_H^{\mathrm{Coll}}(A) \leq \varepsilon$.

Thinking of strengthening this definition by maximizing over all $K \in \mathcal{K}$, like it was in the aPre and aSec definitions, does not make much sense here, because for every fixed K and $|\mathcal{M}| > |\mathcal{Y}|$ there exists a trivial adversary that finds two partners (M, M'). Such adversary would have hardwired two different messages M and M', such that

for some key K_0 are $H_{K_0}(M)$ and $H_{K_0}(M')$ equal. However, it can be difficult to find such algorithm in practice.

Maximizing over all messages M (i.e. defining everywhere collision resistance) makes no sense neither, as adversary has no message on input, it has only one input — a chosen key.

1.2.4 Chosen target forced prefix preimage resistance

In [8] John Kelsey and Tadayoshi Kohno developed a new attack on Merkle-Damgård hash functions called *herding attack*. The attack can be described by the following example. One day in early 2006, the following ad appears in a news:

> I, Nostradamus, hereby provide the MD5 hash Y of many important predictions about the future, including the closing prices of all stocks in the S&P500 as of the last business day of 2006.

Few weeks after the last business day of 2006, Nostradamus publishes a message containing in its first block precise closing prices of the S&P500 stocks. The message then continues with many uncertain predictions which haven't come true yet.

The question is, whether Nostradamus can do this, even if he didn't know the predictions before providing the hash Y. As an answer to this question, Kohno and Kelsey proposed the herding attack, which applies to Merkle-Damgård hash functions and reduces time complexity needed to compute the suffix (possibly containing some predictions), which merged with the closing prices of the S&P500 stocks and hashed by MD5 produces the image Y that Nostradamus provided.

Authors in the paper [8] introduced a new property for the hash functions — *Chosen Target Forced Prefix (CTFP) preimage resistance*, which directly connects to the herding attack. When a hash function preserves this property, it is resistant to the herding attack.

Definition 9 (CTFP preimage resistance)**.** Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family and let λ be a number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. Let A be an adversary. Then we define:

$$\mathbf{Adv}_H^{\mathrm{CTFP}[\lambda]}(A) \quad = \quad \Pr\left[K \xleftarrow{\$} \mathcal{K}; (Y,S) \leftarrow A(K); P \xleftarrow{\$} \{0,1\}^\lambda; M \leftarrow A(P,S) : H_K(P||M) = Y\right]$$

We say that H is (t, L, ε)-CTFP if any adversary A running in time at most t and outputting messages of length less than or equal to L has advantage $\mathbf{Adv}_H^{\mathrm{CTFP}[\lambda]}(A) \leq \varepsilon$ for all λ such that $\{0,1\}^\lambda \subseteq \mathcal{M}$.

The variable S in the definition is adversary's state. It is a string of an arbitrary length, where A can store some information (i.e. it's state) for the second phase. For example A can store in S the key K it gets in the first phase, as in the second phase it has no input with the key. The image Y which A chooses in the first phase corresponds to *chosen target* from the name of the security notion (i.e. the hash, which Nostradamus provides). Similarly, P corresponds to the *forced prefix*, that is the precise closing prices of the S&P500 stocks from the example above.

Similarly to the preimage and second preimage resistance, we can define always CTFP (aCTFP) security notion. It does not make sense to define everywhere CTFP, i.e. strengthen the definition by maximizing over all prefixes P, as any adversary returning $(H_K(P_0||M), S)$ in the first step and M in the second step, where M is an arbitrary string and P_0 is some fixed prefix, has advantage 1, if prefix P_0 is chosen. Thus if we maximize the advantage over all prefixes, it can not be smaller than 1.

Definition 10 (aCTFP preimage resistance)**.** Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family and let λ be a number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. Let A be an adversary. Then we define:

$$\mathbf{Adv}_H^{\mathrm{aCTFP}[\lambda]}(A) \quad = \quad \max_{K \in \mathcal{K}} \Big(\Pr \Big[(Y, S) \leftarrow A; P \xleftarrow{\$} \{0,1\}^\lambda; M \leftarrow A(P, S) : H_K(P||M) = Y \Big] \Big)$$

We say that H is (t, L, ε)-aCTFP if any adversary A running in time at most t and outputting messages of length less than or equal to L has advantage $\mathbf{Adv}_H^{\mathrm{aCTFP}[\lambda]}(A) \leq \varepsilon$ for all λ such that $\{0,1\}^\lambda \subseteq \mathcal{M}$.

1.2.5 Message Authentication Code

There are situations, when we need to be sure, that a message we have received was surely produced by the second party, with which we are communicating, and the message was not modified during the transmission. To solve this problem, *Message authentication codes* were designed. Message authentication code (MAC), roughly speaking, is a hash function parametrized by some secret key K (i.e. a hash function family). During the communication, with every message M is also sent its authentication code $H_K(M)$. When a message M with corresponding MAC C are received, receiver verifies, whether $H_K(M) = C$. If so, the receiver can be sure, that the message M was produced by someone, who knows the secret key K and it was not modified during the transmission (see Figure 1.2).

One widely used MAC construction is HMAC [1]. Consider, that we have some hash function $F : \mathcal{M} \to \mathcal{Y}$. Then

$$HMAC_K(M) = F(L \oplus \mathrm{opad}||F(K \oplus \mathrm{ipad}||M))$$

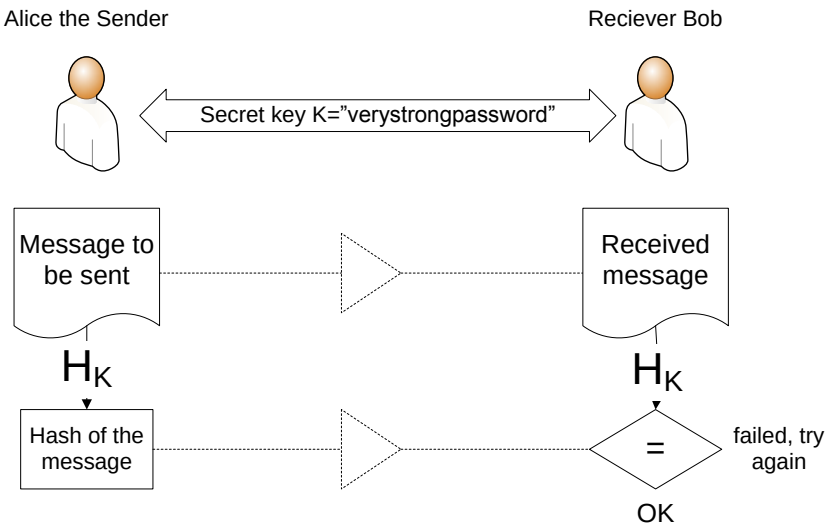

Figure 1.2: Message authentication scheme: Alice and Bob share some secret key K. Alice sends to Bob with every message also its hash. If the received hash and hash of the received message are equal, Bob can be sure, that the message was sent by Alice and it was not modified during the transmission.

where $\oplus$ means XOR operation and opad and ipad are some constant strings.

When we use a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ to build MACs, the following security notion can be useful. The adversary A from the following definition does not have access to the key K. It takes function $H_K : \mathcal{M} \to \mathcal{Y}$ as a black-box and can not output message, that was queried. Otherwise it would be easy to find such adversary for every function family H (it would query some message M and return pair $(M, H_K(M))$).

Definition 11 (MAC)**.** Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family. Let A be an adversary. Then we define:

$$\mathbf{Adv}_H^{\mathrm{MAC}}(A) \quad = \quad \Pr\left[K \xleftarrow{\$} \mathcal{K}; (M, Y) \leftarrow A^{H_K} : H_K(M) = Y \wedge M \text{ not queried}\right]$$

We say that H is (t, q, L, ε)-MAC if any adversary A running in time at most t, outputting or querying messages of length less than or equal to L and making at most q queries to its oracle has advantage $\mathbf{Adv}_H^{\mathrm{MAC}}(A) \leq \varepsilon$.

We note that the security property defined above is also known as *unforgeability* (see [11]).

Consider following situation. Alice sent to Bob n messages $M_1, \ldots, M_n$ with corresponding MACs $C_1, \ldots, C_n$. Attacker Denis intercepted this communication and

wants to send Bob one fake message M_{fake}. The probability of Denis success is given by the advantage $\mathbf{Adv}_H^{\mathrm{MAC}}(\text{Denis})$. However the definition also captures the situation, when Denis has ability to choose some messages for which he wants to get corresponding MACs, i.e. he can get MACs not only for the messages he intercepted.

It makes no sense to think about strengthening this definition by maximizing over all K (i.e. defining always MAC), as for a given function $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ we can construct an adversary A always returning pair $(M, H_{K_0}(M))$ for some fixed K_0. The advantage of such adversary, if the key K_0 is chosen, is 1, thus if we maximize the advantage over all keys, it can not be smaller than 1.

1.2.6 Pseudo random function and Pseudo random oracle

Hash functions are often used as a basic primitive, from which more complex cryptosystems are build. To prove the security of such cryptosystem $\mathcal{C}(F)$ with hash function F, one first proves that $\mathcal{C}(I)$ is secure with some idealized hash function[1] I. Then, one proves the following relation between I and F: For every cryptosystem $\mathcal{C}'(\cdot)$, its security is not affected, when I is replaced by F.

Such relation between I and F (that we can replace I with F without affecting the security of the system) is called *indistinguishability*. Two systems I and F are indistinguishable if no (efficient) algorithm D connected to either I or F, is able to decide, whether it is interacting with I or F. More formally, I and F are indistinguishable, if for any *efficient* adversary D (called distinguisher), the advantage

$$\left| \Pr\left[1 \leftarrow D(I)\right] - \Pr\left[1 \leftarrow D(F)\right] \right|$$

is negligible. We note that the discussion, about what the *efficient* adversary means, is in the Section 1.3. If I and F are indistinguishable, then the following proposition holds.

Propositioin 2. *If and only if I and F are indistinguishable, then, for every cryptosystem $\mathcal{C}(I)$, the cryptosystem $\mathcal{C}(F)$ obtained from $\mathcal{C}(I)$ by replacing I with F is at least as secure as $C(I)$.*

Here we permit some inconsistency, as we do not formally define what *at least as secure as* means. Intuitively, system $\mathcal{C}(F)$ is at least as secure as $\mathcal{C}(I)$, when every successful attack on $\mathcal{C}(F)$ is successful also on $\mathcal{C}(I)$.

Thus, the following notion appears as useful for a hash function family. A hash function family is a *pseudo random function*, when a randomly chosen hash function from the family is indistinguishable from the random oracle.

[1] e.g. random oracle

Definition 12 (Pseudo random function)**.** Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family. Let A be an adversary. Then we define:

$$\mathbf{Adv}_H^{\mathrm{Prf}}(A) \quad = \quad \left| \Pr\left[K \xleftarrow{\$} \mathcal{K}; 1 \leftarrow A^{H_K(\cdot)}\right] - \Pr\left[f \xleftarrow{\$} Func(\mathcal{M}, \mathcal{Y}); 1 \leftarrow A^{f}\right] \right|$$

We say that H is (t, q, L, ε)-Prf if any adversary A running in time at most t and making at most q queries to its oracle each of length less than or equal to L has advantage $\mathbf{Adv}_H^{\mathrm{Prf}}(A) \leq \varepsilon$.

However, Proposition 2 holds only if each component a cryptosystem is based on belongs to one specific party which have exclusive access to it and no one else can directly access its behavior or obtain information about its randomness. When speaking of hash functions, it means, that the Proposition 2 holds only if a hash function F, which is replacing the ideal hash function I in the cryptosystem $\mathcal{C}(\cdot)$, is known only by C and no one else knows the algorithm computing F or can query F (i.e. F is "hidden" to the other world).

To be more specific, let R be a random oracle and H be a hash function family (which is known to the other world). Canetti, Goldreich, and Halevi in [6] proved, that there exists a cryptosystem $\mathcal{C}(\cdot)$, where $\mathcal{C}(R)$ is secure, but security of $\mathcal{C}(H_K)$ for some particular key K (where the key K is public) is lost, even if the hash function family H is indistinguishable from the random oracle R. This work was extended by J. Black in [5], where Black presents a block-cipher based hash function F (i.e. a hash function build from a block-cipher), which is provably secure in the ideal-cipher model, but trivially insecure when instantiated by any block-cipher.

Thus indistinguishability does not work with cryptosystems, which have some public components. In order to extend the definition of indistinguishability to capture such systems with public parameters, Maurer, Renner and Holenstein in [9] proposed new concept, called *indifferentiability*. Indifferentiability does the same as indistinguishability, but it applies to more general settings. More formally, let S^1 denote private components (i.e. known only tho the cryptosystem S) of a system S and let S^2 denote public (i.e. known to the other world) components of S. Then cryptosystem H is indifferentiable from I, if for any *efficient* adversary D (called distinguisher) there is a simulator $\mathcal{S}$ such that

$$\left| \Pr\left[1 \leftarrow D(H^1, H^2)\right] - \Pr\left[1 \leftarrow D(I^1, \mathcal{S}(I^2))\right] \right|$$

is negligible. The simulator $\mathcal{S}$ is an algorithm (i.e. a RAM), which simulates the public component H^2 to make distinguishing H^1 and H^2 from I^1 and I^2 more difficult. Note, that indifferentiability is, unlike indistinguishability, asymmetric — we can not commute H and I, as we could in the case of indistinguishability.

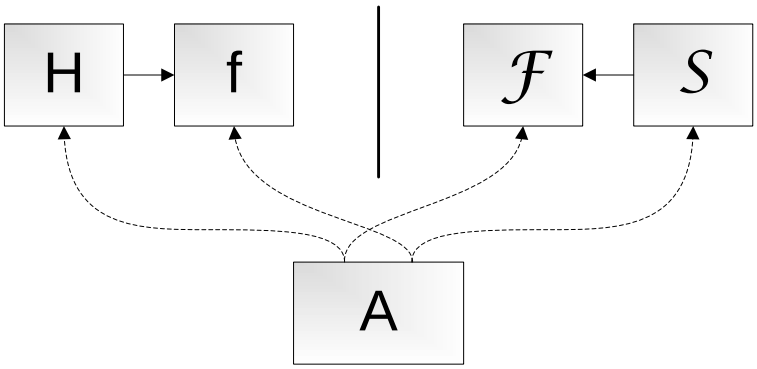

Figure 1.3: The pseudo-random oracle notion: the adversary A (distinguisher) can either interact with a hash function family H and its ideal compression function f or with a random function $\mathcal{F}$ and simulator $\mathcal{S}$. The hash function family H has oracle access to f and the simulator $\mathcal{S}$ to $\mathcal{F}$.

If H is indifferentiable from I, then the following proposition holds.

Propositioin 3. *If and only if I and H are indifferentiable, then, for every cryptosystem $\mathcal{C}(I)$, the cryptosystem $\mathcal{C}(H)$ obtained from $\mathcal{C}(I)$ by replacing I with H is at least as secure as $C(I)$.*

Based on indifferentiability framework, Coron, Dodis, Malinaud and Puniya defined in [7] *pseudo-random oracle* notion. A hash function family H^f with access to an ideal hash function $f : \{0,1\}^{y+d} \to \{0,1\}^y$ (i.e. f is a compression function from which H is build) is a pseudo-random oracle, if it is indifferentiable from a random oracle.

Definition 13 (Pseudo-random oracle)**.** Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family. Let A be an adversary, $f = RF_{y+d,y}$ for some integer $d > 0$ and let $\mathcal{S}$ be a simulator. Then we define:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{Pro}}_{H,f,\mathcal{S}}(A) \quad = \quad & \Big|\Pr\left[K \xleftarrow{\$} \mathcal{K}; 1 \leftarrow A^{H^f_K(\cdot),f(\cdot)}(K)\right] - \\ & - \Pr\left[K \xleftarrow{\$} \mathcal{K}; \mathcal{F} \xleftarrow{\$} Func(\mathcal{M},\mathcal{Y}); 1 \leftarrow A^{\mathcal{F}(\cdot),\mathcal{S}^{\mathcal{F}}(K,\cdot)}(K)\right]\Big|\end{aligned}$$

We say that H is $(t_A, t_{\mathcal{S}}, q_1, q_2, L, \varepsilon)$-Pro if for any adversary A running in time at most t_A and making at most q_1 (q_2) queries to its first (second) oracle each of length less than or equal to L, there exists a simulator $\mathcal{S}$ running in time $t_{\mathcal{S}}$ such that the advantage $\mathbf{Adv}^{\mathrm{Pro}}_{H,f,\mathcal{S}}(A) \leq \varepsilon$.

The role of the simulator $\mathcal{S}$ in the definition is to simulate the ideal primitive (compression function) f, so that no distinguisher can tell, whether it is interacting with H_K and f or with $\mathcal{F}$ and $\mathcal{S}^{\mathcal{F}}$. The output of $\mathcal{S}$ thus has to be "consistent" with that,

what can distinguisher obtain from $\mathcal{F}$ (see Figure 1.3). The simulator does not see the queries made by distinguisher to $\mathcal{F}$ but it has oracle access to $\mathcal{F}$, thus it can call it directly, when needed.

Note, that it makes no sense to think about strengthening the definitions of Prf and Pro by maximizing over all keys K (i.e. defining aPrf and aPro), as trivial adversary, which has hardwired pair $(M, H_{K_0}(M))$ for some fixed key K_0 and an arbitrary message M, and returns 1, if response of its first oracle to query M is equal to $H_{K_0}(M)$, has significant advantage, when the key K_0 is chosen. Therefore when we maximize advantage over all the keys, it can not be smaller than the one for the key K_0.

Finally we note, that an advantage in Prf sense of any adversary A attacking some hash function family H cannot be equal to 1, as there is always a nonzero probability (even very little) that a randomly chosen function f from the second component (in the definition of Prf) is the same as H_K for some key K from the first component. It means that there is always a nonzero probability that A returns the same output when its oracle is f as when its oracle is H_K. Therefore if the first component $\Pr[K \xleftarrow{\$} \mathcal{K}; 1 \leftarrow A^{H_K(\cdot)}]$ is equal to 1, then the second one $\Pr[f \xleftarrow{\$} Func(\mathcal{M}, \mathcal{Y}); 1 \leftarrow A^f]$ can not be equal to 0 (as with a nonzero probability is f equal to H_K for some key K and A^{H_K} outputs 1), and vice-versa. In fact, if A makes at most q queries $M_1, M_2, \ldots, M_q$, then the probability that it returns the same output when its oracle is a random function f as when its oracle is H_K is at least $\frac{1}{|\mathcal{Y}|^q}$, what is the probability that f maps messages $M_1, M_2, \ldots, M_q$ to the same value as H_K does. Similar situation is for advantage in Pro sense.

1.3 Security of a hash function family

In this section, we discuss, what it means, when we say that a hash function family is xxx *secure* for xxx $\in$ {Pre, aPre, ePre, Sec, aSec, eSec, Coll, CTFP, aCTFP, MAC, Prf, Pro}. To be more succinct, let *Atks* temporarily denote the set {Pre, aPre, ePre, Sec, aSec, eSec, Coll, CTFP, aCTFP, MAC, Prf, Pro}.

Consider the following example. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family. We can construct an adversary A attacking H in Pre sense. Adversary $A(K, Y)$ runs through all of the messages M from the set $\mathcal{M}$ and checks, whether $H_K(M)$ is equal to Y. If so, it returns message M as the preimage for Y. Such adversary works against an arbitrary hash function family, however, its time complexity is $O(|\mathcal{M}|)$. When $\mathcal{M} = \{0,1\}^{256}$, i.e. the hash function H_K can process only messages of length 256 bits, the adversary A would need in average 2^{128} hash operations (i.e. to compute $H_K(M)$ for some key K and message M) to find the preimage. Running such adversary on a

computer which can handle 1 billion hash operations per second would last $\frac{1}{10^9} \cdot 2^{128}$ seconds, what is about 2^{82} days. So it would last unfeasibly long until A finishes its work.

Such kind of attack exists against all security notions we defined in Section 1.2, we call them *brute force* attacks. Brute force attacks can be performed against all hash function families. However significant disadvantage of such attacks is their infeasible time complexity.

Thus we can see, that in order to define xxx (xxx $\in$ Atks) security for a hash function family, it is necessary to bound the time complexity of adversaries (as we do not want to rate all hash function families as insecure in Pre sense). We consider some hash function family H as xxx *secure* for xxx $\in$ Atks, when every *efficient* adversary has *negligible* advantage against H in xxx sense. In the sequel we will try to give the formal definition of *efficient* adversary.

We say that an adversary A attacking a hash function family $H : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^y$ runs in a *polynomially bounded time*, when the running time t of the adversary is a polynomial of $(k + y + l)$, where l is a length of the adversary's input, that is, there exists a polynomial P, that for every k, y, l the running time of the adversary attacking H is $P(k + y + l)$. We say that an adversary A is *polynomially bounded*, if it is running in a polynomially bounded time.

Definition 14. We say that a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ is xxx *secure* for xxx $\in$ Atks if any adversary running in a polynomially bounded time has a negligible advantage in xxx sense.

We note, that for Prf, Pro and MAC, the polynomially bounded adversary can make at most polynomial number of queries. Similarly, the polynomially bounded adversary can produce (or query) messages of at most polynomial length.

Polynomial limitation is the standard way how to define *efficiency*. However in the case of hash functions it may deliver some ambiguity. Consider the popular hash function MD5, which processes messages of length at most 2^{64} bits and produces images of length 128 bits and consider, that we want to prove, that MD5 is Pre secure. Here we fall into the problem, what is the polynomial adversary attacking MD5. Even the brute force attack described above performed on MD5 has constant time complexity.

On the other hand suppose, that we have a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ and we have proved that H is Pre secure (i.e. any polynomial adversary has negligible Pre-advantage against H). Then someone finds an adversary against H running in exponential time (i.e. its running time is exponential function of $(k+y+l)$) with time

complexity $2^{\frac{y}{1024}}$. Thus, to maintain desired security properties, we will need to use H with hash length y much greater than 1024 bits, what makes H incompatible with the practical use (due to great computing and memory requirements).

Thus it seems that defining efficiency for adversaries attacking hash functions and hash function families does not have simple solution. In the rest of this Thesis we will understand xxx-security (xxx $\in$ Atks) as defined in Definition 14, however some intuition behind term *efficiency* will be needed too.

1.3.1 Implication and separation

In the Chapter 2 we discuss relationships among the definitions of security notions. Among all of the definitions we give implications or separations. Informally, when we say that *xxx implies yyy*, it means that if a hash function family H is xxx secure, then it is also yyy secure. Saying that *xxx nonimplies yyy* means, that some hash function family H is xxx secure, but it is not yyy secure.

Let $\mathbf{Adv}_H^{\mathrm{xxx}}(\mathcal{R})$ be the maximal advantage over all adversaries A in xxx sense (xxx $\in$ Atks) that uses resources bounded by $\mathcal{R}$. For our consideration it is sufficient to think only about resource t, the running time of the adversary. Thus $\mathbf{Adv}_H^{\mathrm{xxx}}(t)$ is the maximal advantage in xxx sense over all adversaries running in time bounded by t.

The formal definition of an implication was proposed in [13] and can be found in the following Definition 15. We note that by $\mathrm{Time}_{H,n}$ we denote (speaking informally) the time needed by the fastest algorithm to compute an output of a hash function H on an input of length n. By $\mathbf{Adv}_H^{\mathrm{xxx}[\cdot]}(A)$ we denote the advantage of an adversary A attacking a hash function family H in xxx sense (xxx $\in$ Atks).

We note that in the following definition, and later, $[\cdot]$ is a placeholder which is either $[\lambda]$ (for Pre, aPre, Sec, aSec, eSec, CTFP, aCTFP) or empty (for ePre, Coll, Prf, Pro).

Definition 15 (xxx $\rightarrow$ yyy to ε)**.** Let $\mathcal{K} = \{0,1\}^k$, $\mathcal{M} = \{0,1\}^*$ and $\mathcal{Y} = \{0,1\}^y$ for some fixed k and y, let $\{0,1\}^\lambda \subseteq \mathcal{M}$ for some fixed λ and suppose, that xxx, yyy $\in$ Atks. We say that the definition of security notion xxx *implies* security notion yyy to ε (shortly xxx $\rightarrow$ yyy to ε), if for any hash function family $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$ and any adversary A running in time t, outputting messages of length less than or equal to L and with advantage $\mathbf{Adv}_H^{\mathrm{yyy}[\cdot]}(A)$, there exists an adversary A' such that A' runs in time t' and has advantage $c_1 \mathbf{Adv}_H^{\mathrm{xxx}[\cdot]}(A') + \varepsilon \geq \mathbf{Adv}_H^{\mathrm{yyy}[\cdot]}(A)$, where $t' = t + c_2 \mathrm{Time}_{H,L+\lambda}$ and c_1 and c_2 are absolute constants (i.e. their values do not depend on k, y or λ).

The adversary A' in the definition above can run in the time $t' = t + c_2 \mathrm{Time}_{H,L+\lambda}$,

what means that A' can perform only constantly more hash operations (and therefore also simple operations) than A can. The value $L + \lambda$ in $\text{Time}_{H,L+\lambda}$ means, that A' can process messages that A outputs, and eventually can add to them some more bits, but maximally λ. Informally, the definition above says that xxx implies yyy, if any adversary A attacking in yyy sense can be converted (without significant loss of performance) to another adversary A', which performs an attack in xxx sense and its advantage is in the worst case only a little bit smaller than the advantage of A in yyy sense.

The strength of an implication depends on the value of ε, if $\varepsilon = 0$, we speak about *conventional* implication and we omit writing "to ε", if $\varepsilon > 0$ we rather speak about *provisional* implication. The provisional implication carries the usual semantics of the word *implication* only if ε is negligible with respect to k, y or λ or some other parameter of a particular hash function family.

If we treat the time t in the definition as polynomial time, then the definition says, what we intuitively wanted, that xxx implies yyy when for any hash function H holds: if H is xxx secure, then it is *yyy* secure too. However, the definition 15 is more general and applies also to non-polynomial adversaries.

On the other hand, the definition of implication above (introduced by Rogaway and Shrimpton in [13]) can be too strict in some cases, since the adversary A' can perform only constantly more hash operations than A can. For example A' can not simulate the adversary A twice. Therefore we introduce the new definition of implication between two security notions, xxx $\rightsquigarrow$ yyy, where we try to be more general.

Definition 16 (xxx $\rightsquigarrow$ yyy)**.** Let $\mathcal{K} = \{0,1\}^k$, $\mathcal{M} = \{0,1\}^*$ and $\mathcal{Y} = \{0,1\}^y$ for some fixed k and y, let $\{0,1\}^\lambda \subseteq \mathcal{M}$ for some fixed λ and suppose, that xxx, yyy $\in$ Atks. We say that the definition of security notion xxx *implies* security notion yyy (shortly xxx $\rightsquigarrow$ yyy), if for any hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ and any adversary A running in time t, outputting messages of length less than or equal to L and with non-negligible advantage (with respect to k, y or λ) in yyy sense, there exists an adversary A' such that A' runs in time t' and has non-negligible advantage in xxx sense, where $t' = p(k, y, \lambda).(t + \text{Time}_{H,L+\lambda})$ and $p(k, y, \lambda)$ is some polynomial of k, y and λ.

Thus the adversary A' in the definition above can perform polynomially more operations than A can. Note that our definition does not have provisional part, i.e. "to ε" statement, since the only condition on the advantage of adversary A' in xxx sense is its non-negligibility. If we consider only polynomial adversaries, the definition 16 captures our intuition of implication — if H is xxx secure, then it is *yyy* secure too.

We note that most of proofs of implications between security notions in the Chapter

2 satisfy the conditions of Definition 15, whereas we utilize the Definition 16 only in one proof. It is clear that if xxx $\rightarrow$ yyy to ε, where ε is negligible, then also xxx $\rightsquigarrow$ yyy.

We can also formally define the separation of two security notions xxx and yyy, however here we have two different possibilities.

The first definition, *conventional separation*, informally says, that if H is a hash function family secure in xxx sense, then we can convert H into another hash function family H', which is also secure in xxx sense, but completely insecure in yyy sense.

Definition 17 (xxx $\not\rightarrow$ yyy to ε)**.** Let $\mathcal{K} = \{0,1\}^k$, $\mathcal{M} = \{0,1\}^*$ and $\mathcal{Y} = \{0,1\}^y$ for some fixed k and y, let $\{0,1\}^\lambda \subseteq \mathcal{M}$ for some fixed λ and suppose, that xxx, yyy $\in$ Atks. We say that the definition of security notion xxx *nonimplies* security notion yyy to ε, in the conventional case (shortly *xxx* $\not\rightarrow$ *yyy to* ε), if for any hash function family $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$ there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$, such that $\mathbf{Adv}^{\mathrm{xxx}[\cdot]}_{H'}(t) \leq c_1 \, \mathbf{Adv}^{\mathrm{xxx}[\cdot]}_{H}(t') + \varepsilon$ and $\mathbf{Adv}^{\mathrm{yyy}[\cdot]}_{H'}(t'') \geq 1 - \varepsilon$, where t is an arbitrary running time, $t' = t + c_2 \, \mathrm{Time}_{H,L+\lambda}$, $t'' = c_3 \, \mathrm{Time}_{H,\lambda}$ and c_1, c_2 and c_3 are absolute constants and L is a maximum message length that an adversary running in the time t can output.

The time $t'' = c_3 \, \mathrm{Time}_{H,\lambda}$ in the definition (c_3 is an absolute constant) represents what we intuitively call "constant" time, i.e. the adversary running in such time is able to perform only constant number of hash operations on a messages it gets on an input. Constant c_3 does not depend on k, m, y or λ – it is given by the particular adversary that performs the attack against yyy and has advantage greater or equal to $1 - \varepsilon$. In a proof of some particular separation we do not need to know the exact value of this constant, its existence will be sufficient (as its value can vary among different RAM models). Note that the time t'' also covers the situations where an adversary does not perform any hash operations, however it returns some output (or operates on messages) of length that is constant multiple of k, y or λ (as $\mathrm{Time}_{H,\lambda}$ is a multiple of k, y and λ).

If xxx nonimplies yyy to 0, we simply write xxx non implies yyy or shortly xxx $\not\rightarrow$ yyy and we call such separation *conventional separation*. If $\varepsilon > 0$, we call it *provisional separation*.

The second definition, *unconditional separation*, says that, there exists a hash function family H, which is secure in xxx sense, but it is completely insecure in yyy sense. Thus the conventional separation needs a xxx secure hash function family H in order to separate xxx from yyy, while the unconditional separation does not.

Definition 18 (Unconditional separation)**.** Let $\mathcal{K} = \{0,1\}^k$, $\mathcal{M} = \{0,1\}^*$ and $\mathcal{Y} =$

$\{0,1\}^y$ for some fixed k and y, let $\{0,1\}^\lambda \subseteq \mathcal{M}$ for some fixed λ and suppose, that xxx, yyy $\in$ Atks. We say that the definition of security notion xxx *non implies* security notion yyy to ε, in the unconditional case (shortly *xxx* $\not\rightarrow$ *yyy to* ε), if there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, such that $\mathbf{Adv}_H^{\mathrm{xxx}[\cdot]}(t) \leq \varepsilon$ for all t and $\mathbf{Adv}_H^{\mathrm{yyy}[\cdot]}(t') \geq 1 - \varepsilon$, where $t' = c\,\mathrm{Time}_{H,\lambda}$ and c is an absolute constant.

An unconditional separation between two notions can be consequence of the matter, that for some domains $\mathcal{M}$ and ranges $\mathcal{Y}$ secure hash functions trivially exist, for example identity function $H_K(M) = M$ is trivially collision resistant[2], however it is definitely not preimage resistant.

Note that a separation is not negation of an implication. Both a separation and an implication can exists between two notions xxx and yyy, their relative strength depends on a provisional part of the implication/separation. Such example of coexistence can be found in [13], however implication and separation can not coexist with arbitrary provisional parts. Intuitively, when xxx implies yyy to ε_1 and xxx nonimplies yyy to ε_2 and if ε_1 is negligible, then ε_2 can't be negligible (otherwise both provisional implication and separation would be "strong", what is in contrast with our intuition). Similarly, when ε_2 is negligible then ε_1 can not be negligible. In the following lemma we prove the relationship between ε_1 and ε_2 and we will see, that our intuition is good. We note that if xxx implies yyy to ε, then xxx implies yyy to $\varepsilon + \alpha$ for any $\alpha \geq 0$. Similar holds for separation.

Lemma 1. *Let xxx and yyy be some security notions from the set Atks,* $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ *be a hash function family and* $\{0,1\}^\lambda \subseteq \mathcal{M}$ *for some fixed* λ*. Let* t *be a "constant" running time, such that* $t = a\,\mathrm{Time}_{H,\lambda}$*, where* a *is an absolute constant. If xxx* $\rightarrow$ *yyy to* ε_1 *and xxx* $\not\rightarrow$ *yyy to* ε_2*, then* $\varepsilon_1 \geq 1 - (1+c)\varepsilon_2 - c\varepsilon$*, where* $\varepsilon = \mathbf{Adv}_H^{xxx[\cdot]}(t)$*.*

Proof. From the assumption that xxx $\rightarrow$ yyy to ε_1 we have:

$$\forall H, \forall A, \exists B : \mathbf{Adv}_H^{\mathrm{yyy}[\cdot]}(A) \leq c_1\,\mathbf{Adv}_H^{\mathrm{xxx}[\cdot]}(B) + \varepsilon_1 \tag{1.1}$$

where H represents some hash function family and A and B some adversaries such that if A runs in time t, then B runs in time $t' = t + c_2\,\mathrm{Time}_{H,L+\lambda}$, where c_1 and c_2 are absolute constants and L is the maximum message length that A can output. From the assumption that xxx $\not\rightarrow$ yyy to ε_2 we have:

$$\forall H, \exists H', \quad \forall A, \exists B : \quad \mathbf{Adv}_{H'}^{\mathrm{xxx}[\cdot]}(A) \leq c_3\,\mathbf{Adv}_H^{\mathrm{xxx}[\cdot]}(B) + \varepsilon_2 \tag{1.2}$$

$$\wedge \qquad \exists C : \qquad \mathbf{Adv}_{H'}^{\mathrm{yyy}[\cdot]}(C) \geq 1 - \varepsilon_2 \tag{1.3}$$

[2] actually such H does not satisfy our definition of hash function family in the Definition 1, but we find it as a simple example sufficient to explain how unconditional separation works

where H and H' represent some hash function families and A, B and C represent some adversaries, such that if A runs in time t then B runs in time $t' = t + c_4 \operatorname{Time}_{H,L+\lambda}$ and C runs in time $c_5 \operatorname{Time}_{H,\lambda}$. Now let H be a hash function family, such that any adversary running in a "constant" time $(c_5 + c_2 + c_4) \operatorname{Time}_{H,\lambda}$ has advantage at most ε. From (1.3) we have that for the hash function family H there exists a hash function family H' and an adversary C running in time $c_5 \operatorname{Time}_{H,\lambda}$ such that

$$\mathbf{Adv}_{H'}^{\mathrm{yyy}[\cdot]}(C) \geq 1 - \varepsilon_2.$$

From (1.1) we have that for H' and C there exists an adversary C' running in time $c_5 \operatorname{Time}_{H,\lambda} + c_2 \operatorname{Time}_{H,\lambda}$ such that

$$\mathbf{Adv}_{H'}^{\mathrm{yyy}[\cdot]}(C) \leq c_1 \mathbf{Adv}_{H'}^{\mathrm{xxx}[\cdot]}(C') + \varepsilon_1,$$

thus

$$1 - \varepsilon_2 \leq c_1 \mathbf{Adv}_{H'}^{\mathrm{xxx}[\cdot]}(C') + \varepsilon_1.$$

However from (1.2) we know that

$$\mathbf{Adv}_{H'}^{\mathrm{xxx}[\cdot]}(C') \leq c_3 \mathbf{Adv}_{H}^{\mathrm{xxx}[\cdot]}(C'') + \varepsilon_2,$$

for some adversary C'' that runs in time $c_5 \operatorname{Time}_{H,\lambda} + c_2 \operatorname{Time}_{H,\lambda} + c_4 \operatorname{Time}_{H,\lambda}$, therefore

$$1 - \varepsilon_2 \leq c_1(c_3 \mathbf{Adv}_{H}^{\mathrm{xxx}[\cdot]}(C'') + \varepsilon_2) + \varepsilon_1.$$

When we put ε_1 on the left side and anything else on the right we get:

$$\varepsilon_1 \geq 1 - c_1 c_3 \mathbf{Adv}_{H}^{\mathrm{xxx}[\cdot]}(C'') - (c_1 + 1)\varepsilon_2.$$

We can see that C'' runs in a "constant" time $(c_5 + c_2 + c_4) \operatorname{Time}_{H,\lambda}$, therefore $\mathbf{Adv}_{H}^{\mathrm{xxx}[\cdot]}(C'') \leq \varepsilon$. Thus

$$\varepsilon_1 \geq 1 - (1 + c)\varepsilon_2 - c\varepsilon$$

for a constant $c = \max\{c_1 c_2, c_1\}$. □

Note that in the Lemma above we assume the existence of the hash function family H, which is secure against any adversary running in the "constant" time $(c_5 + c_2 + c_4) \operatorname{Time}_{H,\lambda}$. We need this assumption, as the conventional separation assumes the existence of xxx secure hash function family. The similar lemma can be proved for unconditional separation too, however there we do not need to have such hash function family H and therefore the relationship between ε_1 and ε_2 would be $\varepsilon_1 \geq 1 - (1 + c)\varepsilon_2$.

When we assume that ε and ε_2 are negligible, then from $\varepsilon_1 \geq 1 - (1 + c)\varepsilon_2 - c\varepsilon$ we have that ε_1 can not be negligible. Similarly when we assume that ε and ε_1 are negligible, then ε_2 can not be.

In the following lemma we prove that the implication from the Definition 15 is transitive in some cases.

Lemma 2. *Let xxx, yyy and zzz be some security notions from the set Atks. If xxx $\rightarrow$ yyy to ε_1 and yyy $\rightarrow$ zzz to ε_2, then xxx $\rightarrow$ zzz to $\varepsilon_1 + c\varepsilon_2$, where c is an absolute constant.*

Proof. Let $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$ be an arbitrary hash function family and A be an arbitrary adversary attacking H in zzz sense, running in time t and outputting messages of length at most L. Fix some λ such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. From the assumption, that yyy $\rightarrow$ zzz to ε_2 and Definition 15 we have, that there exists an adversary A', running in time $t' = t + c_2 \operatorname{Time}_{H,L+\lambda}$ and with advantage $c_1 \mathbf{Adv}_H^{\mathrm{yyy}[\cdot]}(A') + \varepsilon_1 \geq \mathbf{Adv}_H^{\mathrm{zzz}[\cdot]}(A)$. Similarly from the assumption that xxx $\rightarrow$ yyy to ε_2 we have, that for the adversary A' there exists an adversary A'' running in time $t'' = t' + c_4 \operatorname{Time}_{H,L+\lambda}$ and with advantage $c_3 \mathbf{Adv}_H^{\mathrm{xxx}[\cdot]}(A'') + \varepsilon_2 \geq \mathbf{Adv}_H^{\mathrm{yyy}[\cdot]}(A')$. Thus we showed, that for the hash function family H and the adversary A there exists the adversary A'' running in time $t'' = t + (c_2 + c_4) \operatorname{Time}_{H,L+\lambda}$ and with advantage $c_1 c_3 \mathbf{Adv}_H^{\mathrm{xxx}[\cdot]}(A'') + c_1\varepsilon_2 + \varepsilon_1 \geq \mathbf{Adv}_H^{\mathrm{zzz}[\cdot]}(A)$, what means that xxx $\rightarrow$ zzz to $\varepsilon_1 + c_1\varepsilon_2$. □

Finally we note, that two definitions of security notions xxx and yyy are equivalent, if xxx implies yyy and yyy implies xxx.

1.4 Equivalent definitions with a two stage adversary

In definitions of aPre, ePre, aSec, eSec, aCTFP we maximize over some quantity (over all keys or messages). However, there exist equivalent definitions to these already mentioned, where the specific value (key or message) is chosen by an adversary. That is, in the "first phase" the adversary choses that value, then a random choice is made by the environment and in the "second phase" the adversary continues, where it ended, but with given that randomly chosen value.

Definition 19 (two stage versions of aPre, ePre, aSec, eSec, aCTFP)**.** Let $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$ be a hash function family, and let λ be a number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$.

Let A be an adversary. Then we define:

$$\begin{aligned}
\mathbf{Adv}_H^{\mathrm{aPre}[\lambda]}(A) &= \Pr\Big[(K,S)\leftarrow A; M\xleftarrow{\$}\{0,1\}^\lambda; Y\leftarrow H_K(M); M'\leftarrow A(Y,S): \\
&\qquad H_K(M')=H_K(M)\Big] \\
\mathbf{Adv}_H^{\mathrm{ePre}}(A) &= \Pr\Big[(Y,S)\leftarrow A; K\xleftarrow{\$}\mathcal{K}; M'\leftarrow A(K,S): H_K(M')=Y\Big] \\
\mathbf{Adv}_H^{\mathrm{aSec}[\lambda]}(A) &= \Pr\Big[(K,S)\leftarrow A; M\xleftarrow{\$}\{0,1\}^\lambda; M'\leftarrow A(M,S): \\
&\qquad (M\neq M')\wedge(H_K(M)=H_K(M'))\Big] \\
\mathbf{Adv}_H^{\mathrm{eSec}}(A) &= \Pr\Big[(M,S)\leftarrow A; K\xleftarrow{\$}\mathcal{K}; M'\leftarrow A(K,S): \\
&\qquad (M\neq M')\wedge(H_K(M)=H_K(M'))\Big] \\
\mathbf{Adv}_H^{\mathrm{aCTFP}[\lambda]}(A) &= \Pr\Big[(Y,K,S)\leftarrow A; P\xleftarrow{\$}\{0,1\}^\lambda; M\leftarrow A(P,S): H_K(P||M)=Y\Big]
\end{aligned}$$

We say that H is (t,L,ε)-xxx for xxx $\in$ {aPre,aSec,aCTFP} if any adversary A running in time at most t and outputting messages of length at most L has advantage $\mathbf{Adv}_H^{\mathrm{xxx}[\lambda]}(A)\leq\varepsilon$ for all λ such that $\{0,1\}\subseteq\mathcal{M}$. We say that H is (t,L,ε)-yyy for yyy $\in$ {ePre,eSec} if any adversary A running in time at most t and outputting messages of length at most L has advantage $\mathbf{Adv}_H^{\mathrm{yyy}}(A)\leq\varepsilon$.

We prove the equivalence for aPre and aPre2, where aPre2 temporarily denotes the two stage version of aPre, in the following lemma. Equivalence of the other definitions is proven similarly.

Lemma 3. *(aPre $\leftrightarrow$ aPre2) The definitions of security notions aPre and aPre2 are equivalent.*

Proof. Consider an adversary A attacking H in aPre sense and let K be the key, for which A has the maximum advantage α (i.e. $\alpha=\mathbf{Adv}_H^{\mathrm{aPre}[\lambda]}(A)$). We construct an adversary B, which in the first phase returns pair (K,S), where $S=K$, and in the second phase it does the same as A, that is $B(Y,S)$ returns the same value as $A(Y)$. Adversary B runs in the time that is only constantly greater than running time of A, thus if t is a running time of A and L is the maximum message length that A can output, then B runs in time, which is not greater than $t+c\,\mathrm{Time}_{H,L+\lambda}$ for some absolute constant c. Advantage of B in aPre2 sense is equal to α, what is equal to $\mathbf{Adv}_H^{\mathrm{aPre}[\lambda]}(A)$, so $\mathbf{Adv}_H^{\mathrm{aPre2}[\lambda]}(B)\geq\mathbf{Adv}_H^{\mathrm{aPre}[\lambda]}(A)$. Thus aPre2 implies aPre.

Conversely, consider, that the advantage of an adversary B attacking H in aPre2 sense is α. Consider an adversary A, which simulates B. Suppose that B returns (K,S) in the first phase, then $A(Y)$ returns the same as $B(Y,S)$ in the second phase. Advantage of A in aPre sense is at least α (it can not be smaller than α, as the probability that A finds preimage when using key K is α). Therefore $\mathbf{Adv}_H^{\mathrm{aPre}[\lambda]}(A)\geq\mathbf{Adv}_H^{\mathrm{aPre2}[\lambda]}(B)$.

At last we note, that running time of A is equal to the running time of B, thus aPre implies aPre2. $\square$

Chapter 2

Relationships among the definitions

Here we provide relationships among the definitions from the Section 1.2. Relationships between the definitions of preimage resistance, second-preimage resistance and collision resistance were proven by Rogaway and Shrimpton in [13]. Relationship between MAC and Prf can be found in [11]. Other relations are work of authors, we are not aware of any other work, where these relations occur. We give an overview over all of the relations in the Table 2.1. In Figure 2.1 we provide all constructions used in the proofs of separations.

In the rest of this Chapter we will assume, that $\mathcal{M} = \{0,1\}^*$, $\mathcal{K} = \{0,1\}^k$ and $\mathcal{Y} = \{0,1\}^y$ for some fixed k and y and also that $\{0,1\}^\lambda \subseteq \mathcal{M}$ for some fixed λ.

Some of the implications and separations are conventional, others are provisional. In this thesis no unconditional separation is proven, however some unconditional separations can be found in [13]. In the Table 2.1 we make no difference between conventional and provisional implications, as all provisional implications/separations have their provisional part negligible, if hash function families with standard domains and ranges (i.e. such as $\mathcal{K}$, $\mathcal{M}$ and $\mathcal{Y}$ defined above) are used.

2.1 Message authentication codes

2.1.1 Coll vs. MAC

In this section we prove the separations between Coll and MAC. The proof of Theorem 1 (MAC nonimplies Coll) is based on the fact, that an adversary attacking in MAC sense does not have access to a key K chosen by the environment, thus some

	Pre	aPre	ePre	Sec	aSec	eSec	Coll	MAC	CTFP	aCTFP	Prf	Pro
Pre	x	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ 5	$\not\to$ 7	$\not\to$ 16	$\not\to$ 21	$\not\to$ 26
aPre	$\to$ [13]	x	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ 5	$\not\to$ 7	$\not\to$ 16	$\not\to$ 21	$\not\to$ 26
ePre	$\to$ [13]	$\not\to$ [13]	x	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ 5	$\not\to$ 8	$\not\to$ 16	$\not\to$ 21	$\not\to$ 26
Sec	$\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	x	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ 3	$\not\to$ 7	$\not\to$ 16	$\not\to$ 22	$\not\to$ 27
aSec	$\to$ [13]	$\to$ [13]	$\not\to$ [13]	$\to$ [13]	x	$\not\to$ [13]	$\not\to$ [13]	$\not\to$ 3	$\not\to$ 7	$\not\to$ 16	$\not\to$ 22	$\not\to$ 27
eSec	$\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	$\to$ [13]	$\not\to$ [13]	x	$\not\to$ [13]	$\not\to$ 3	$\not\to$ 8	$\not\to$ 16	$\not\to$ 22	$\not\to$ 27
Coll	$\to$ [13]	$\not\to$ [13]	$\not\to$ [13]	$\to$ [13]	$\not\to$ [13]	$\to$ [13]	x	$\not\to$ 2	$\rightsquigarrow$ 9	$\not\to$ 17	$\not\to$ 19	$\not\to$ 27
Mac	$\not\to$ 6	$\not\to$ 6	$\not\to$ 6	$\not\to$ 4	$\not\to$ 4	$\not\to$ 4	$\not\to$ 1	x	$\not\to$ 13	$\not\to$ 16	$\not\to$ [11]	$\not\to$ 30
CTFP	$\not\to$ 11	$\not\to$ 11	$\not\to$ 11	$\not\to$ 10	$\not\to$ 10	$\not\to$ 10	$\not\to$ 10	$\not\to$ 12	x	$\not\to$ 15	$\not\to$ 24	$\not\to$ 32
aCTFP	$\not\to$ 18	$\not\to$ 18	$\not\to$ 18	$\not\to$ 18	$\not\to$ 18	$\not\to$ 18	$\not\to$ 18	$\not\to$ 18	$\to$ 14	x	$\not\to$ 24	$\not\to$ 32
Prf	$\not\to$ 23	$\not\to$ 23	$\not\to$ 23	$\not\to$ 23	$\not\to$ 23	$\not\to$ 23	$\not\to$ 20	$\to$ [11]	$\not\to$ 25	$\not\to$ 25	x	$\not\to$ 34
Pro	$\to$ 28	$\not\to$ 29	$\to$ 28	$\to$ 28	$\not\to$ 29	$\to$ 28	$\to$ 28	$\to$ 31	$\to$ 33	$\not\to$ 33	$\to$ 35	x

Table 2.1: Relationships among the definitions. Numbers in brackets [·] are citations, other numbers are numbers of theorems, where the proof of the corresponding relation can be found.

$$
\begin{aligned}
H_K^{(1)}(M) &= \begin{cases} H_K(M) & \text{if } M \neq K \\ H_K(0^k) & \text{if } M = K \end{cases} \\
H_K^{(2)}(M) &= \begin{cases} 0^y & \text{if } M = 0 \\ H_K(M) & \text{if } M \neq 0 \text{ and } H_K(M) \neq 0^y \\ H_K(0) & \text{otherwise} \end{cases} \\
H_K^{(3)}(M) &= \begin{cases} Y & \text{if } \mathrm{Prefix}_{(k+1+y)}(M) = K||b||Y \text{ for some } b \in \{0,1\} \\ H_K(M) & \text{otherwise} \end{cases} \\
H_K^{(4)}(M) &= H_K(M[1 \ldots |M|-1]||0) \\
H_K^{(5)}(M) &= \begin{cases} K[1 \ldots \min\{k,y\}] & \text{if } \mathrm{Suffix}_k(M) = K \\ H_K(M) & \text{otherwise} \end{cases} \\
H_K^{(6)}(M) &= \begin{cases} M & \text{if } |M| = y \\ H_K(M) & \text{otherwise} \end{cases} \\
H_K^{(7)}(M) &= \begin{cases} 0^y & \text{if } M = 0 \\ H_K(M) & \text{otherwise} \end{cases} \\
H_K^{(8)}(M) &= \begin{cases} \mathrm{Suffix}_y(M) & \text{if } K = K_0 \\ H_K(M) & \text{otherwise} \end{cases} \\
H_K^{(9)}(M) &= \begin{cases} H_K(M) & \text{if } K \neq K_0 \\ 0^y & \text{if } K = K_0 \end{cases}
\end{aligned}
$$

Figure 2.1: Constructions of hash function families used in proofs of separations.

information which helps in finding collisions can be bundled by the key K.

Theorem 1 (MAC $\not\to$ Coll to $q/|\mathcal{K}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is (t, q, L, ε)-MAC, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-MAC, but there exists an adversary C attacking H' in Coll sense running in time $c\,Time_{H,\lambda}$ for some absolute constant c and with advantage $\mathbf{Adv}^{Coll}_{H'}(C) = 1$.*

Proof. Suppose, that we have a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is (t, q, L, ε)-MAC and consider the construction $H^{(1)}$ from the Figure 2.1

$$H^{(1)}_K(M) = \begin{cases} H_K(M) & \text{if } M \neq K \\ H_K(0^k) & \text{if } M = K \end{cases}$$

Thus $H^{(1)}$ differs from H only in one point for every key $K \in \mathcal{K}$. We show, that $H^{(1)}$ is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-MAC. Let A be any adversary, which runs in time t, outputs or queries messages of length at most L and makes q queries to its oracle f. From the assumption we have, that $\mathbf{Adv}^{\mathrm{MAC}}_H(A) \leq \varepsilon$. Now consider $\mathbf{Adv}^{\mathrm{MAC}}_{H^{(1)}}(A)$. Adversary A has no access to the key K, so it can only guess, thus the probability, that the adversary A with oracle $H^{(1)}$ queries $H^{(1)}_K(K)$ is $\frac{q}{|\mathcal{K}|}$. This means the probability, that A with oracle $H^{(1)}_K$ returns different output than A with oracle H_K is at most $\frac{q}{|\mathcal{K}|}$. Thus

$$\mathbf{Adv}^{MAC}_{H^{(1)}}(A) \leq \mathbf{Adv}^{\mathrm{MAC}}_H(A) + \frac{q}{|\mathcal{K}|} \leq \varepsilon + \frac{q}{|\mathcal{K}|}.$$

Therefore $H^{(1)}$ is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-MAC.

Now we show that there exists an adversary C attacking $H^{(1)}$ in Coll sense running in time $c\,\mathrm{Time}_{H,\lambda}$. For a randomly chosen key K, $C(K)$ returns pair $(0^k, K)$. From the definition of $H^{(1)}$ we can see, that $H^{(1)}_K(0^k) = H^{(1)}_K(K)$. Thus C attacks $H^{(1)}$ in Coll sense with advantage 1 and runs in time $c\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c. The constant c is determined by the time needed by the adversary C to return the pair $(0^k, K)$ on a particular RAM model (note that c need not to depend on k, since $\mathrm{Time}_{H,\lambda}$ depends on k). □

We note, that if we permit only polynomial adversaries, then $\frac{q}{|\mathcal{K}|}$ is negligible and the following statement holds: if H is MAC secure, then so is $H^{(1)}$, but $H^{(1)}$ is completely not Coll secure.

In the following theorem we use the construction $H^{(2)}$, which was designed in [13], where can be found also the proof, that if H is Coll secure then also $H^{(2)}$ is. For completeness we provide this proof, but it is slightly adjusted to be more understandable.

Theorem 2 (Coll $\not\to$ MAC)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is (t, L, ε)-Coll, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t + c_1\, Time_{H,L+\lambda}, L, \varepsilon)$-Coll, but there exists an adversary C running in time $c_2\, Time_{H,\lambda}$, making no queries to its oracle and with advantage $\mathbf{Adv}^{MAC}_{H'}(C) = 1$, where c_1 and c_2 are absolute constants.*

Proof. Suppose, that we have a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is (t, L, ε)-Coll and consider the construction $H^{(2)}$ from the Figure 2.1:

$$H^{(2)}_K(M) = \begin{cases} 0^y & \text{if } M = 0 \\ H_K(M) & \text{if } M \neq 0 \text{ and } H_K(M) \neq 0^y \\ H_K(0) & \text{otherwise} \end{cases}$$

We show that if H is (t, L, ε)-Coll, then $H^{(2)}$ is $(t + c_1 \,\mathrm{Time}_{H,L+\lambda}, L, \varepsilon)$. Thus let A be an adversary running in time t, producing messages of length at most L and with advantage $\mathbf{Adv}^{\mathrm{Coll}}_{H^{(2)}}(A) = \varepsilon$. Consider the following adversary B:

Adversary $B(K)$
1 $(M, M') \leftarrow A(K)$
2 **if** $M = 0$ and $H_K(M') = 0^y$ **then return** (M, M')
3 **if** $M \neq 0$ and $H_K(M) \neq 0^y$ and
  $M' \neq 0$ and $H_K(M') = 0^y$ **then return** $(M, 0)$
4 **if** $M \neq 0$ and $H_K(M) = 0^y$ and $M' = 0$ **then return** (M, M')
5 **if** $M \neq 0$ and $H_K(M) = 0^y$ and
  $M' \neq 0$ and $H_K(M') \neq 0^y$ **then return** $(0, M')$
6 **else return** (M, M')

The running time of such adversary B is at most $t' = t + c_1 \,\mathrm{Time}_{H,L+\lambda}$ for some absolute constant c_1. Now consider messages M and M', that adversary A returns in the first line, are partners for $H^{(2)}_K$ and consider following situations based on the line number, where B returns:

[B **returns in the** 2^{nd} **line.**] Then $M = 0$ and $H_K(M') = 0^y$, thus $H^{(2)}_K(M') = H_K(0)$. And so $0^y = H^{(2)}_K(0) = H^{(2)}_K(M') = H_K(0)$, thus $H_K(0) = 0^y = H_K(M')$, therefore M and M' are partners for H_K.

[B **returns in the** 3^{rd} **line.**] Then $H^{(2)}_K(M) = H_K(M)$ and $H^{(2)}_K(M') = H_K(0)$. However we know that $H^{(2)}_K(M) = H^{(2)}_K(M')$, thus $H_K(M) = H_K(0)$. And so M and 0 are partners for H_K.

[B **returns in the** 4^{th} **line.**] Then $H^{(2)}_K(M) = H_K(0)$ and $H^{(2)}_K(M') = 0^y$. From the assumption that $H^{(2)}_K(M) = H^{(2)}_K(M')$ we have, that $H_K(0) = 0^y$. As $H_K(M) = 0^y$ and $M' = 0$, we know that M and M' are partners for H_K.

[B **returns in the** 5^{th} **line.**] Then $H_K^{(2)}(M) = H_K(0)$ and $H_K^{(2)}(M') = H_K(M')$. Thus $H_K(0) = H_K(M')$, as $H_K^{(2)}(M) = H_K^{(2)}(M')$. Therefore 0 and M' are partners for H_K.

[B **returns in the** 6^{th} **line.**] Then we have several possibilities:

- $M = 0$ and $H_K(M') \neq 0^y$. Thus $H_K^{(2)}(M) = 0^y$ and $H_K^{(2)}(M') = H_K(M')$. However $H_K^{(2)}(M) = H_K^{(2)}(M')$, thus $H_K(M') = 0^y$, what is a contradiction. Therefore this possibility can not occur.
- $M \neq 0$ and $H_K(M) = 0^y$ and $M' \neq 0$ and $H_K(M') = 0^y$. However then M and M' are partners for H_K, as $H_K(M) = H_K(M') = 0^y$.
- $M \neq 0$ and $H_K(M) \neq 0^y$ and $M' = 0$. Then $H_K^{(2)}(M) = H_K(M)$ and $H_K^{(2)}(M') = 0^y$, what leads to a contradiction, as M and M' are not partners for $H_K^{(2)}$ ($H_K(M) \neq 0^y$). Therefore this possibility cannot occur.
- $M \neq 0$ and $H_K(0) \neq 0^y$ and $M' \neq 0$ and $H_K(M') \neq 0$. Then $H_K^{(2)}(M) = H_K(M)$ and $H_K^{(2)}(M') = H_K(M')$. Therefore M and M' are partners also for H_K.

Thus wherever $B(K)$ returns, it always returns partners for H_K, if A returns partners for $H_K^{(2)}$. Therefore $H_K^{(2)}$ is (t', L, ε)-Coll.

The adversary C, which returns pair $(0, 0^y)$ has advantage $\mathbf{Adv}^{\mathrm{MAC}}_{H^{(2)}}(C) = 1$. We can see that C makes no queries to its oracle and runs in time $c_2\,\mathrm{Time}_{H,\lambda}$, where c_2 is some absolute constant determined by the time needed to return the pair $(0, 0^y)$ on a particular RAM model. □

2.1.2 Sec vs. MAC

In this section, the proofs for separations between Sec and MAC, eSec and MAC and aSec and MAC can be found. As the proofs for separations Sec nonimplies MAC, eSec nonimplies MAC and aSec nonimplies MAC are very similar, we prove these separations in one theorem — Theorem 3. In the proof we will use the same construction $H^{(2)}$ as in Theorem 2.

Theorem 3 (Sec, eSec, aSec $\not\to$ MAC)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is (t, L, ε)-Sec (eSec, aSec), then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t + c_1\,Time_{H,L+\lambda}, L, \varepsilon)$-Sec (eSec, aSec), but there exists an adversary C running in time $c_2\,Time_{H,\lambda}$, making no queries to its oracle and with advantage $\mathbf{Adv}^{MAC}_{H'}(C) = 1$, where c_1 and c_2 are absolute constants.*

Proof. We use the construction $H^{(2)}$ from the Figure 2.1. Assume, that $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ is (t, L, ε)-Sec (eSec, aSec) and consider the following hash function family:

$$H_K^{(2)}(M) = \begin{cases} 0^y & \text{if } M = 0 \\ H_K(M) & \text{if } M \neq 0 \text{ and } H_K(M) \neq 0^y \\ H_K(0) & \text{otherwise} \end{cases}$$

We need to show, that if H is (t, L, ε)-xSec, then $H^{(2)}$ is $(t + c_1 \operatorname{Time}_{H,L+\lambda}, L, \varepsilon)$-xSec (where xSec represents one of the notions Sec, eSec or aSec). Let A_{xSec} be an adversary attacking $H^{(2)}$, running in time t, outputting messages of length at most L and with advantage $\mathbf{Adv}^{\mathrm{xSec}[\lambda]}_{H^{(2)}}(A_{\mathrm{xSec}}) = \varepsilon$. We construct an adversary B_{xSec} attacking H as follows:

Adversary $B_{Sec}(K, M)$
1 $(M') \leftarrow A_{\mathrm{Sec}}(K, M)$
2 **if** $M = 0$ and $H_K(M') = 0^y$ **then return** (M')
3 **if** $M \neq 0$ and $H_K(M) \neq 0^y$ and $M' \neq 0$ and $H_K(M') = 0^y$ **then return** (0)
4 **if** $M \neq 0$ and $H_K(M) = 0^y$ and $M' = 0$ **then return** (M')
5 **if** $M \neq 0$ and $H_K(M) = 0^y$ and $M' \neq 0$ and $H_K(M') \neq 0^y$ **then return** (M')
6 **else return** (M')

Adversary B_{aSec}
[1^{st} **phase**]
$(K, S) \leftarrow A_{\mathrm{aSec}}$
return (K, S)
[2^{nd} **phase**]
1 $(M') \leftarrow A_{\mathrm{aSec}}(M, S)$
2 **if** $M = 0$ and $H_K(M') = 0^y$ **then return** (M')
3 **if** $M \neq 0$ and $H_K(M) \neq 0^y$ and $M' \neq 0$ and $H_K(M') = 0^y$ **then return** (0)
4 **if** $M \neq 0$ and $H_K(M) = 0^y$ and $M' = 0$ **then return** (M')
5 **if** $M \neq 0$ and $H_K(M) = 0^y$ and $M' \neq 0$ and $H_K(M') \neq 0^y$ **then return** (M')
6 **else return** (M')

Adversary $B0_{eSec}$
[1^{st} **phase**]
$(M, S) \leftarrow A_{\mathrm{eSec}}$
return (M, S) (*)
[2^{nd} **phase**]
1 $(M') \leftarrow A_{\mathrm{eSec}}(K, S)$
2 **if** $M = 0$ and $H_K(M') = 0^y$ **then return** (M')
3 **if** $M \neq 0$ and $H_K(M) \neq 0^y$ and $M' \neq 0$ and $H_K(M') = 0^y$ **then return** (0)
4 **if** $M \neq 0$ and $H_K(M) = 0^y$ and $M' = 0$ **then return** (M')
5 **if** $M \neq 0$ and $H_K(M) = 0^y$ and $M' \neq 0$ and $H_K(M') \neq 0^y$ **then return** (M')
6 **else return** (M')

Let $B1_{\text{eSec}}$ be an adversary constructed as $B0_{\text{eSec}}$ but the line marked with (*) replaced by "**return** $(0, S)$". From $B0_{\text{eSec}}$ and $B1_{\text{eSec}}$ we construct an adversary B_{eSec}, which simulates both $B0_{eSec}$ and $B1_{eSec}$ and if one of them wins then B_{eSec} returns the same as the winning adversary (we note that B_{eSec} need to simulate the adversary A_{eSec} only once). Note that running time of all the adversaries above is $t' = t + c_1 \,\text{Time}_{H,L+\lambda}$ for some absolute constant c_1.

Now consider that the messages M and M', that are either on input of adversaries B_{xSec} or returned by adversaries A_{xSec}, are partners for $H^{(2)}$. We analyze situations based on the line number where adversary B_{Sec} (B_{aSec}, $B0_{\text{eSec}}$, $B1_{\text{eSec}}$) returns. Let B represent one of the adversaries B_{Sec}, B_{aSec}, $B0_{\text{eSec}}$, $B1_{\text{eSec}}$.

[B **returns in the** 2^{nd} **line.**] Then $M = 0$ and $H_K(M') = 0^y$, thus $H_K^{(2)}(M') = H_K(0)$. And so $0^y = H_K^{(2)}(0) = H_K^{(2)}(M') = H_K(0)$, thus $H_K(0) = 0^y = H_K(M')$, therefore M and M' are partners for H_K.

[B **returns in the** 3^{rd} **line.**] Then $H_K^{(2)}(M) = H_K(M)$ and $H_K^{(2)}(M') = H_K(0)$. However we know that $H_K^{(2)}(M) = H_K^{(2)}(M')$, thus $H_K(M) = H_K(0)$. And so M and 0 are partners for H_K.

[B **returns in the** 4^{th} **line.**] Then $H_K^{(2)}(M) = H_K(0)$ and $H_K^{(2)}(M') = 0^y$. From the assumption that $H_K^{(2)}(M) = H_K^{(2)}(M')$ we have, that $H_K(0) = 0^y$. As $H_K(M) = 0^y$ and $M' = 0$, we know that M and M' are partners for H_K.

[B **returns in the** 5^{th} **line.**] Then $H_K^{(2)}(M) = H_K(0)$ and $H_K^{(2)}(M') = H_K(M')$. Thus $H_K(0) = H_K(M')$, as $H_K^{(2)}(M) = H_K^{(2)}(M')$. Therefore 0 and M' are partners for H_K.

[B **returns in the** 6^{th} **line.**] Then we have several possibilities:

- $M = 0$ and $H_K(M') \neq 0^y$. Thus $H_K^{(2)}(M) = 0^y$ and $H_K^{(2)}(M') = H_K(M')$. However $H_K^{(2)}(M) = H_K^{(2)}(M')$, thus $H_K(M') = 0^y$, what is contradiction. Therefore this possibility can not occur.
- $M \neq 0$ and $H_K(M) = 0^y$ and $M' \neq 0$ and $H_K(M') = 0^y$. However then M and M' are partners for H_K, as $H_K(M) = H_K(M') = 0^y$.
- $M \neq 0$ and $H_K(M) \neq 0^y$ and $M' = 0$. Then $H_K^{(2)}(M) = H_K(M)$ and $H_K^{(2)}(M') = 0^y$, what leads to contradiction, as M and M' are not partners for $H_K^{(2)}$ ($H_K(M) \neq 0^y$). Therefore this possibility cannot occur.
- $M \neq 0$ and $H_K(0) \neq 0^y$ and $M' \neq 0$ and $H_K(M') \neq 0$. Then $H_K^{(2)}(M) = H_K(M)$ and $H_K^{(2)}(M') = H_K(M')$. Therefore M and M' are partners also for H_K.

Thus we showed that wherever adversary $B_{\mathrm{Sec}}(K, M)$ returns, it returns M's partner for H_K, if $A_{\mathrm{Sec}}(K, M)$ returns M's partner for $H_K^{(2)}$. Similarly if $A_{\mathrm{aSec}}(M)$ returns M's partner for $H_K^{(2)}$, then wherever $B_{\mathrm{aSec}}(M)$ returns it always returns M's partner for H_K, where K is the key B_{aSec} choses in the first phase. For $B0_{\mathrm{eSec}}$ and $B1_{\mathrm{eSec}}$ the following holds: if A_{eSec} returns M in the first phase and M's partner for $H^{(2)}$ in the second phase, then one of the adversaries $B0_{\mathrm{eSec}}$ and $B1_{\mathrm{eSec}}$ returns two messages (one in the first phase, one in the second phase) that are partners for H_K and therefore B_{eSec} returns partners for H_K. Thus we proved that if A_{xSec} wins against $H^{(2)}$, then B_{xSec} wins against H and therefore $H^{(2)}$ is $(t + c_1 \operatorname{Time}_{H,n}, L, \varepsilon)$-Sec (eSec, aSec).

Now we only need to show that $H^{(2)}$ is not MAC secure. However we showed that in the proof of Theorem 2. The adversary C returning pair $(0, 0^y)$ and running in time $c_2 \operatorname{Time}_{H,\lambda}$ has advantage $\mathbf{Adv}^{\mathrm{MAC}}_{H^{(2)}}(C) = 1$ for some absolute constant c_2 determined by the time needed to return the pair $(0, 0^y)$ on a particular RAM model. □

In the following theorem we show the separation between MAC and notions of second-preimage resistance. The proof for the separation is very similar for each notion, so we give only one proof, in which we try to cover all Sec notions.

One can think that if we prove that MAC $\not\to$ Sec to ε, where ε is negligible, then other separations (MAC $\not\to$ aSec, eSec) come from the fact that eSec $\to$ Sec and aSec $\to$ Sec proven in [13]. However, this is not so easy, as our definitions of implication and separation are not contrary (see Section 1.3.1). For example consider, that we have proven MAC $\not\to$ Sec to ε_1 (ε_1 is negligible). If we assume that MAC $\to$ eSec to ε_2, where ε_2 is negligible, then from the Lemma 2 we have that MAC $\to$ Sec to $\varepsilon_1 + c\varepsilon_2$ for some absolute constant c. Since ε_1 and ε_2 are negligible, also $\varepsilon_1 + c\varepsilon_2$ is negligible, what is a contradiction, as from Lemma 1 we know, that if MAC $\not\to$ Sec to ε_1 and MAC $\to$ Sec to $\varepsilon_1 + c\varepsilon_2$ then both ε_1 and $\varepsilon_1 + c\varepsilon_2$ can not be negligible. Thus we know that implications between MAC and eSec or aSec are *not* strong, however we do not know anything about the strength of the separations.

Since adversaries attacking in MAC sense does not have access to a key K, randomly chosen by the environment, we need to somehow bundle the information needed to find second preimages with the key K, so that an adversary attacking in MAC sense can not find that information (because it does not know the key).

Theorem 4 (MAC $\not\to$ Sec, eSec, aSec to $q/|\mathcal{K}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, q, L, ε)-MAC, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-MAC, but there exists an adversary C_{xSec} running in time $c\,\mathit{Time}_{H,\lambda}$ with advantage $\mathbf{Adv}_H^{xSec[\lambda]}(C_{xSec}) = 1$, where c is some absolute constant and xSec*

$\in \{Sec,\ aSec,\ eSec\}$.

Proof. We use the construction $H^{(3)}$ from the Figure 2.1. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a (t, q, L, ε)-MAC hash function family. Then we define $H^{(3)}$ as follows:

$$H_K^{(3)}(M) = \begin{cases} Y & \text{if } \mathrm{Prefix}_{(k+1+y)}(M) = K||b||Y \text{ for some } b \in \{0,1\} \\ H_K(M) & \text{otherwise} \end{cases}$$

Let A be an adversary running in time t, outputting or querying messages of length at most L and making q queries to its oracle. From the assumption we have, that the advantage $\mathbf{Adv}_H^{\mathrm{MAC}}(A) \leq \varepsilon$. Now consider the advantage $\mathbf{Adv}_{H^{(3)}}^{\mathrm{MAC}}(A)$. Adversary A with oracle $H_K^{(3)}$ can notice some difference, from the case its oracle is H_K, only when it queries some message of length at least $(k + 1 + y)$ and with prefix K. As A has no access to the key K, it can only guess. Thus the probability that A queries in one query a message of length at least $(k + 1 + y)$ with prefix K is at most $\frac{1}{|\mathcal{K}|}$ (when A queries messages only of length at least $(k+1+y)$, then it is equal to $\frac{1}{|\mathcal{K}|}$, otherwise it is smaller). As A can make at most q queries, the probability that A queries a message with prefix K is at most $\frac{q}{|\mathcal{K}|}$. Thus the advantage $\mathbf{Adv}_{H^{(3)}}^{\mathrm{MAC}}(A) \leq \mathbf{Adv}_H^{\mathrm{MAC}}(A) + \frac{q}{|\mathcal{K}|}$, so $H^{(3)}$ is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-MAC.

Now consider the following adversaries:

Adversary $C_{Sec}(M, K)$
  if $|M| \geq k + 1 + y$ and $\mathrm{Prefix}_k(M) = K$ **then**
    let $b := M[k + 1]$
    return $K||\overline{b}||M[(k + 1) \ldots |M|]$
  else return $K||0||H_K(M)$

Adversary C_{aSec}
[1^{st} **phase**]
  return $1^k, 1^k$
[2^{nd} **phase with input** (M, S)]
  let $K := S$
  if $|M| \geq k + 1 + y$ and $\mathrm{Prefix}_k(M) = K$ **then**
    let $b := M[k + 1]$
    return $K||\overline{b}||M[(k + 1) \ldots |M|]$
  else return $K||0||H_K(M)$

Adversary C_{eSec}
[1^{st} **phase**]
  return $1, 1$
[2^{nd} **phase with input** (K, S)]
  let $M := S$
  else return $K||0||H_K(M)$

We can see that there exists an absolute constant c, such that adversaries above run in time at most $c\,\mathrm{Time}_{H,\lambda}$ (since λ is the length of their input). Their advantage is $\mathbf{Adv}^{\mathrm{xSec}[\lambda]}_{H^{(3)}}(C_{\mathrm{xSec}}) = 1$, where xSec $\in$ {Sec, aSec, eSec}. □

2.1.3 Pre vs. MAC

Here we give the separations between notions of preimage resistance and MAC. Similarly to the theorems above, we give only one proof for all of the preimage resistance notions. The construction $H^{(4)}$ used in Theorem 5 to prove the separation Pre (aPre, ePre) nonimplies MAC was proposed in [13]. The construction $H^{(4)}$ was used in [13] to prove the relationship between preimage resistance and notions of second-preimage resistance.

Theorem 5 (Pre, aPre, ePre $\not\to$ MAC)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is (t, L, ε)-Pre (aPre, ePre), then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t + c_1\ \mathit{Time}_{H,L+\lambda}, L, \varepsilon)$-Pre (aPre, ePre), but there exists an adversary C running in time $c_2\ \mathit{Time}_{H,\lambda}$, making one query to its oracle and with advantage $\mathbf{Adv}^{MAC}_{H'}(C) = 1$, where c_1 and c_2 are absolute constants.*

Proof. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family, which is (t, L, ε)-Pre. We use the construction $H^{(4)}$ from the Figure 2.1 defined as follows:

$$H^{(4)}_K(M) = H_K(M[1 \ldots |M| - 1]||0)$$

We show that if H is secure in Pre (aPre, ePre) sense, then so is $H^{(4)}$. Let A_{xPre} be an adversary attacking $H^{(4)}$ in xPre sense (xPre $\in$ {Pre, aPre, ePre}), running in time t, outputting messages of length at most L and with advantage $\mathbf{Adv}^{\mathrm{xPre}[\cdot]}_{H^{(4)}}(A_{\mathrm{xPre}}) = \varepsilon$. Then consider the following adversaries B_{xPre}:

Adversary $B_{\mathrm{Pre}}(Y, K)$
  $M \leftarrow A_{\mathrm{Pre}}(Y, K)$
  if $H_K(M) = Y$ **then return** M
  else let $b := M[|M|]$**; return** $M[1 \ldots |M| - 1]||\overline{b}$

Adversary B_{aPre}
[1^{st} **phase**]
  $(K, S) \leftarrow A_{\mathrm{aPre}}$
  return (K, S)
[2^{nd} **phase with input** (Y, S)]
  $M \leftarrow A_{\mathrm{aPre}}(Y, S)$
  if $H_K(M) = Y$ **then return** M
  else let $b := M[|M|]$**; return** $M[1 \ldots |M| - 1]||\overline{b}$

Adversary B_{ePre}
[1^{st} **phase**]
  $(Y, S) \leftarrow A_{\mathrm{ePre}}$
  return (Y, S)
[2^{nd} **phase with input** (K, S)]
  $M \leftarrow A_{\mathrm{ePre}}(K, S)$
  if $H_K(M) = Y$ **then return** M
  else let $b := M[|M|]$**; return** $M[1 \ldots |M| - 1]||\overline{b}$

Note that running time of such adversaries is $t' = t + c_1 \operatorname{Time}_{H,L+\lambda}$ for some absolute constant c_1. If the adversary $A_{\mathrm{Pre}}(K, Y)$ returns a message M, such that $H_K^{(4)}(M) = Y$, then either $H_K(M) = Y$ or $H_K(M') = Y$, where the message M' is equal to the message M with the last bit inverted. A similar statement holds for A_{aPre} and A_{ePre}. Thus if A_{xPre} wins (i.e. it finds preimage) for $H^{(4)}$, then B_{xPre} wins for H. Therefore $\mathbf{Adv}^{xPre[\cdot]}_{H^{(4)}}(A_{\mathrm{xPre}}) \leq \mathbf{Adv}^{xPre[\cdot]}_{H}(B_{\mathrm{xPre}})$.

On the other hand consider an advantage in MAC sense of an adversary C attacking $H^{(4)}$, which firstly queries message 00, gets output Y and returns the pair $(01, Y)$. It is clear, that the advantage of such adversary is $\mathbf{Adv}^{\mathrm{MAC}}_{H^{(4)}}(C) = 1$. Running time of C is $c_2 \operatorname{Time}_{H,\lambda}$, where c_2 is an absolute constant determined by the time needed to query the message 00 and to return the pair $(01, Y)$ on a particular RAM model. □

In the following theorem we use the same construction $H^{(3)}$ as in Theorem 4. Thus the proof of the following theorem is only slightly different from the one of Theorem 4.

Theorem 6 (MAC $\not\to$ Pre, ePre, aPre to $\frac{q}{|\mathcal{K}|}$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is (t, q, L, ε)-MAC, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-MAC, but there exists an adversary C_{xPre} running in time $c\,Time_{H,\lambda}$ with advantage $\mathbf{Adv}^{xPre[\cdot]}_{H'}(C_{xPre}) = 1$, where c is an absolute constant and $xPre \in \{Pre, aPre, ePre\}$.*

Proof. Assume that we have a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is

(t, q, L, ε)-MAC. Consider the construction $H^{(3)}$ from the Figure 2.1 defined as follows:

$$H_K^{(3)}(M) = \begin{cases} Y & \text{if } \mathrm{Prefix}_{(k+1+y)}(M) = K||b||Y \text{ for some } b \in \{0,1\} \\ H_K(M) & \text{otherwise} \end{cases}$$

In Theorem 4 we proved that $H^{(3)}$ is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-MAC. Thus we only need to show that $H^{(3)}$ is not Pre (ePre, aPre) secure.

Consider the following adversaries C_{xPre} attacking $H^{(3)}$ in xPre sense (xPre $\in$ {Pre, aPre, ePre}).

Adversary $C_{Pre}(Y, K)$	**Adversary** C_{aPre}	**Adversary** C_{ePre}
return $K\|\|0\|\|Y$	[1^{st} **phase**]	[1^{st} **phase**]
	return $1^k, 1^k$	**return** $1^y, 1^y$
	[2^{nd} **phase, input** (Y, S)]	[2^{nd} **phase, input** (K, S)]
	let $K := S$	**let** $Y := S$
	return $K\|\|0\|\|Y$	**return** $K\|\|0\|\|Y$

Running time of such adversaries is $c\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c and their advantage $\mathbf{Adv}^{\mathrm{xPre}[\cdot]}_{H^{(3)}}(C_{\mathrm{xPre}}) = 1$, what completes the proof. □

2.2 CTFP preimage resistance

2.2.1 Pre, Sec, Coll vs. CTFP

In this section we analyze relationships between notions of preimage resistance, second preimage resistance, collision resistance and chosen target forced prefix preimage resistance.

Theorem 7 (Pre, aPre, Sec, aSec $\not\to$ CTFP to $1/|\mathcal{K}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is (t, L, ε)-xxx for xxx $\in$ {Pre, aPre, Sec, aSec}, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is $(t, L, \varepsilon + 1/|\mathcal{K}|)$-xxx but there exists an adversary C running in time $c\,Time_{H,\lambda}$ and with advantage $\mathbf{Adv}^{CTFP[\lambda]}_{H'}(C) = 1$, where c is an absolute constant.*

Proof. Consider the construction $H^{(5)}$ from the Figure 2.1 defined as follows:

$$H_K^{(5)}(M) = \begin{cases} K[1 \ldots \min\{k, y\}] & \text{if } \mathrm{Suffix}_k(M) = K \\ H_K(M) & \text{otherwise} \end{cases}$$

Let A_{xxx} be an adversary attacking H in xxx sense (xxx $\in$ {Pre, aPre, Sec, aSec}), running in time t, outputting messages of length at most L and with advantage

$\mathbf{Adv}_{H}^{\mathrm{xxx}[\lambda]}(A_{\mathrm{xxx}}) = \varepsilon$. Consider A_{xxx}'s advantage against $H^{(5)}$ in xxx sense. Let K be a key chosen randomly by the environment (or chosen by the adversary in the first phase, for xxx $\in$ {aPre, aSec}). If suffix of randomly chosen message M is different from K, then A can win against $H^{(5)}$ with the same probability as against H (as $H_K^{(5)}(M) = H_K(M)$). However when randomly chosen message M has suffix K, then in the worst case, A's chance to win against $H^{(5)}$ is 1. Thus

$$\mathbf{Adv}_{H^{(5)}}^{\mathrm{xxx}[\lambda]}(A) \leq \mathbf{Adv}_{H}^{\mathrm{xxx}[\lambda]}(A) + \Pr[\text{message } M \text{ with suffix } K \text{ is chosen}].$$

If a randomly chosen message has length at least k, then it has suffix K with probability $\frac{1}{|\mathcal{K}|}$. If a randomly chosen message has length smaller that k, then it can not have suffix K. Thus the probability that randomly chosen message (of arbitrary length) has suffix K is at most $\frac{1}{|\mathcal{K}|}$. Therefore if H is (t, L, ε)-xxx, then $H^{(5)}$ is $(t, L, \varepsilon + \frac{1}{|\mathcal{K}|})$-xxx. Now we show that $H^{(5)}$ is not CTFP secure. Consider the following adversary:

Adversary C
[1^{st} **phase with input** K]
  return $(K[1 \ldots \min\{y, k\}], K)$
[2^{nd} **phase with input** (P, S)]
  let $K := S$
  return K

Running time of such adversary is $c\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c determined by the time needed to return the pair $(K[1 \ldots \min\{y, k\}], K)$ in the first phase and K in the second phase on a particular RAM model. From the definition of $H^{(5)}$ we can see, that C's advantage in CTFP sense is 1, what completes the proof. □

For *everywhere* versions of preimage resistance and second preimage resistance, the proof above does not work, as the message is not chosen by the environment, but adversary chooses it in the first phase. However, the separation between everywhere preimage resistance, everywhere second preimage resistance and chosen target forced prefix preimage resistance holds. In fact, we can use the same construction $H^{(5)}$.

Theorem 8 (ePre, eSec $\not\to$ CTFP to $1/|\mathcal{K}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^{\lambda} \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is (t, L, ε)-xxx for xxx $\in$ {ePre, eSec}, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is $(t, L, \varepsilon + 1/|\mathcal{K}|)$-xxx but there exists an adversary C running in time $c\,\mathrm{Time}_{H,\lambda}$ and with advantage $\mathbf{Adv}_{H'}^{CTFP[\lambda]}(C) = 1$, where c is an absolute constant.*

Proof. Consider the hash function family $H^{(5)}$ from the Figure 2.1.

$$H_K^{(5)}(M) = \begin{cases} K[1 \ldots \min\{k, y\}] & \text{if } \mathrm{Suffix}_k(M) = K \\ H_K(M) & \text{otherwise} \end{cases}$$

Let A be a two stage adversary attacking H in ePre (eSec) sense, running in time t, outputting messages of length at most L and with advantage ε. Consider A's advantage against $H^{(5)}$. In the first phase, A has no access to a key K, as the key is chosen by the environment after the first phase. Let M be a message A chooses in the first phase, the probability that randomly chosen key K is suffix of M is $1/|\mathcal{K}|$, if $|M| \geq k$, otherwise it is 0. Therefore we have several possibilities:

[$|M| < k$] — then A wins against $H^{(5)}$ with the same probability as against H.

[$|M| \geq K$ **and** K **is suffix of** M] — in the worst case A wins with probability 1.

[$|M| \geq K$ **and** K **is not suffix of** M] — then A's probability of winning against $H^{(5)}$ is the same as against H.

Thus A has better chance to win against $H^{(5)}$ as against H only when a key K is chosen, which is suffix of M. As A can not affect the selection of the key, the following holds:

$$\mathbf{Adv}^{\mathrm{xxx}[\cdot]}_{H^{(5)}}(A) \leq \mathbf{Adv}^{\mathrm{xxx}[\cdot]}_{H}(A) + \frac{1}{|\mathcal{K}|},$$

what we wanted to prove, where xxx $\in$ {ePre,eSec}.

The proof that $H^{(5)}$ is not CTFP resistant is given in the proof of Theorem 7. □

The following proof is the only one, where we utilize our (more general) definition of implication (the Definition 16) between security notions. From an adversary A performing attack in CTFP sense we can construct an adversary B performing attack in Coll sense (both have non-negligible advantage), however B need to simulate A twice, what the Definition 15 does not allow.

Theorem 9 (Coll $\rightsquigarrow$ CTFP)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family. If there exists an adversary A running in time t, outputting messages of length at most L and with advantage $\mathbf{Adv}^{CTFP[\lambda]}_{H}(A) = \varepsilon$, then there exists an adversary B attacking H in Coll sense, running in time $2t + c\,Time_{H,\lambda}$ and with advantage $\mathbf{Adv}^{Coll}_{H}(B) \geq \varepsilon^2$, where c is an absolute constant.*

Proof. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family and let A be an adversary attacking H in CTFP sense, running in time t, outputting messages of length at most L and with advantage $\mathbf{Adv}^{CTFP[\lambda]}_{H}(A) = \varepsilon$. Consider the following adversary B attacking H in Coll sense:

Adversary $B(K)$
1 $(Y, S) \leftarrow A(K)$
2 **let** $P_1 \xleftarrow{\$} \{0,1\}^\lambda$
3 $M_1 \leftarrow A(P_1, S)$
4 **let** $P_2 \xleftarrow{\$} (\{0,1\}^\lambda - \{P_1\})$
5 $M_2 \leftarrow A(P_2, S)$
6 **return** (M_1, M_2)

Running time of such adversary is $t' = 2t + c\,\mathrm{Time}_{H,L+\lambda}$ for some absolute constant c. The advantage of such adversary in Coll sense is given by the probability that $A(P_1, S)$ returns a message M_1 that with the prefix P_1 hashes to Y and $A(P_2, S)$ returns a message M_2 that with the prefix P_2 hashes to Y. From the assumption we know, that both these probabilities are equal to ε. Thus the probability that $B(K)$ returns partners for H_K is ε^2, what means that if the advantage of A in CTFP sense is non-negligible, then so is advantage of B in Coll sense. Therefore Coll $\rightsquigarrow$ CTFP. □

Theorem 10 (CTFP $\not\rightarrow$ Sec, eSec, aSec, Coll)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is (t, L, ε)-CTFP, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ which is $(t + c_1\, Time_{H,L+\lambda}, L, \varepsilon)$-CTFP but there exists an adversary C_{xxx} running in time $c_2\, Time_{H,\lambda}$ and with advantage $\mathbf{Adv}^{xxx[\cdot]}_{H'}(C_{xxx}) = 1$, where c is an absolute constant and $xxx \in \{Sec, eSec, aSec, Coll\}$.*

Proof. Consider the construction $H^{(4)}$ from the Figure 2.1:

$$H^{(4)}_K(M) = H_K(M[1 \ldots |M| - 1]||0)$$

We prove that if H is secure in CTFP sense, then so is $H^{(4)}$. For that reason consider an adversary A, which attacks $H^{(4)}$ in CTFP sense, runs in time t, outputs messages of length at most L and has advantage ε. We construct an adversary B attacking H as follows:

Adversary B
[1^{st} **phase with input** K]
  $Y \leftarrow A(K)$
  return (Y, K)
[2^{nd} **phase with input** (P, S)]
  $M \leftarrow A(P, S)$
  let $K := S$
  if $H_K(P||M) = Y$ **then return** M
  else let $b := M[|M|]$**; return** $M[1 \ldots |M| - 1]||\overline{b}$

Running time of adversary B is $t + c_1\,\mathrm{Time}_{H,L+\lambda}$ for some absolute constant c_1. From

the definition of $H^{(4)}$ we can see, that if A wins against $H^{(4)}$ in CTFP sense, then B wins in CTFP sense against H. Thus $H^{(4)}$ is $(t + c_1 \operatorname{Time}_{H,L+\lambda}, L, \varepsilon)$-CTFP.

Now we need to prove that $H^{(4)}$ is not xxx resistant for xxx $\in$ {Sec, eSec, aSec, Coll}. Consider the following adversaries:

Adversary C_{aSec} [1^{st} **phase**] **return** $1^k, 1^k$ [2^{nd} **phase, input** (M, S)] **let** $b = M[\|M\|]$ **return** $M[1 \dots \|M\| - 1]\|\overline{b}$	**Adversary** C_{eSec} [1^{st} **phase**] **return** $11, 11$ [2^{nd} **phase, input** (K, S)] **let** $M := S$; $b = M[\|M\|]$ **return** $M[1 \dots \|M\| - 1]\|\overline{b}$
Adversary $C_{Sec}(K, M)$ **let** $b := M[\|M\|]$ **return** $M[1 \dots \|M\| - 1]\|\overline{b}$	**Adversary** $C_{Coll}(K)$ **return** $(11, 10)$

Running time of such adversaries is $c_2 \operatorname{Time}_{H,\lambda}$ for some absolute constant c_2 and their advantage is $\mathbf{Adv}^{\text{xxx}[\cdot]}_{H^{(4)}}(C_{\text{xxx}}) = 1$ for xxx $\in$ {Sec, aSec, eSec, Coll}. □

The construction $H^{(4)}$ can not be used to prove the separation *CTFP nonimplies Pre*. In fact, in Theorem 5 we proved, that $H^{(4)}$ is preimage resistant, if H is preimage resistant. Thus we need to find another construction — $H^{(6)}$ is suitable.

Theorem 11 (CTFP $\not\to$ Pre, ePre, aPre to $(\lambda+1).2^{-\lambda}$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, L, ε)-CTFP, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, L, \varepsilon + \frac{\lambda+1}{2^\lambda})$-CTFP, but there exists an adversary C_{xxx} running in time c $Time_{H,\lambda}$ and with advantage $\mathbf{Adv}^{xxx[\cdot]}_{H'}(C_{xxx}) = 1$, where c is an absolute constant and xxx $\in$ {Pre, ePre, aPre}.*

Proof. As mentioned before, we use the construction $H^{(6)}$ from the Figure 2.1.

$$H^{(6)}_K(M) = \begin{cases} M & \text{if } |M| = y \\ H_K(M) & \text{otherwise} \end{cases}$$

We prove, that if H is secure in CTFP sense, then $H^{(6)}$ is too. Let A be an arbitrary adversary attacking H in CTFP sense, running in time t, outputting messages of length at most L and with advantage ε. Consider the advantage of A when attacking $H^{(6)}$. Let Y be an image that A chooses in the first phase. If the prefix P chosen randomly by the environment is prefix of Y, then A's advantage can be in the worst case 1 (A can win by returning the remaining bits of Y, i.e. $Y[|P| + 1 \dots |Y|]$), since $H^{(6)}_K(Y) = Y$ for all $Y \in \mathcal{Y}$. On the other hand, if P is not prefix of Y, then A's

advantage against $H^{(6)}$ is the same as against H. Therefore

$$\mathbf{Adv}^{\mathrm{CTFP}[\lambda]}_{H^{(6)}}(A) \leq \mathbf{Adv}^{\mathrm{CTFP}[\lambda]}_{H}(A) + \Pr[\text{forced prefix } P \text{ is prefix of chosen image } Y]$$

Thus we only need to count the probability that P is prefix of Y. The forced prefix P is uniformly selected from the set $\{0,1\}^\lambda$. If $\lambda > y$, only $y+1$ (including empty string) members of $\{0,1\}^\lambda$ are prefixes of Y, if $\lambda \leq y$ then $\lambda + 1$ members of $\{0,1\}^\lambda$ are prefixes of y. Therefore

$$\mathbf{Adv}^{\mathrm{CTFP}[\lambda]}_{H^{(6)}}(A) \leq \mathbf{Adv}^{\mathrm{CTFP}[\lambda]}_{H}(A) + \frac{\lambda+1}{2^\lambda},$$

what we wanted to prove.

The hash function family $H^{(6)}$ is evidently not secure in Pre (ePre, aPre) sense. The adversary $C_{\mathrm{Pre}}(Y,K)$ attacking $H^{(6)}$ in Pre sense copies its input Y to output and wins. Similarly does the adversary $C_{\mathrm{aPre}}(Y)$ in the second phase. The adversary C_{ePre} in the first phase chooses image 1^y and in the second phase, it returns the same, that is 1^y. The advantage of all these adversaries is $\mathbf{Adv}^{\mathrm{xxx}[\cdot]}_{H^{(6)}}(C_{\mathrm{xxx}}) = 1$ for xxx $\in$ {Pre, ePre, aPre} and their running time is $c\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c determined by the time needed to copy an input to output (in the case of C_{Pre} and C_{aPre}) or to return 1^y (in C_{ePre} case) on a particular RAM model. □

2.2.2 MAC vs. CTFP

Theorem 12 (CTFP $\not\to$ MAC to $2^{1-\lambda}$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, L, ε)-CTFP then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, L, \varepsilon + \frac{2}{2^\lambda})$-CTFP, but there exists an adversary C attacking H' in MAC sense, running in time $c\,\mathrm{Time}_{H,\lambda}$ and with advantage $\mathbf{Adv}^{MAC}_{H'}(C) = 1$, where c is an absolute constant.*

Proof. Let H be a (t, L, ε)-CTFP hash function family. Consider the construction $H^{(7)}$ from the Figure 2.1.

$$H^{(7)}_K(M) = \begin{cases} 0^y & \text{if } M = 0 \\ H_K(M) & \text{otherwise} \end{cases}$$

Let A be an arbitrary adversary attacking H in CTFP sense, running in time at most t, outputting messages of length at most L and with advantage less than or equal to ε. We need to find A's advantage against $H^{(7)}$. When A in the first step chooses image different from 0^y, then A's chance to win against $H^{(7)}$ is the same as against

H. When A chooses in the first step image 0^y, then its chance to win is given by the probability that environment chooses a prefix P, which is prefix of 0. However only two strings are prefixes of 0 (empty string and 0), thus the following holds:

$$\mathbf{Adv}^{\mathrm{CTFP}[\lambda]}_{H^{(7)}}(A) \leq \mathbf{Adv}^{\mathrm{CTFP}[\lambda]}_{H}(A) + \frac{2}{2^\lambda}$$

Therefore $H^{(7)}$ is $(t, L, \varepsilon + \frac{2}{2^\lambda})$-CTFP.

The adversary C, which makes no queries to its oracle and always returns the pair $(0, 0^y)$ has advantage $\mathbf{Adv}^{\mathrm{MAC}}_{H^{(7)}}(C) = 1$ and its running time is $c\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c determined by the time needed to return the pair $(0, 0^y)$ on a particular RAM model. □

Theorem 13 (MAC $\not\to$ CTFP to $q/|\mathcal{K}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, q, L, ε)-MAC, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-MAC, but there exists an adversary C attacking H' in CTFP sense, running in time $c\,Time_{H,m}$ and with advantage $\mathbf{Adv}^{CTFP[\lambda]}_{H'}(C) = 1$, where c is an absolute constant.*

Proof. Assume that H is a (t, q, L, ε)-MAC hash function family and consider the hash function family $H^{(5)}$ from the Figure 2.1.

$$H^{(5)}_K(M) = \begin{cases} K[1 \ldots \min\{k, y\}] & \text{if } \mathrm{Suffix}_k(M) = K \\ H_K(M) & \text{otherwise} \end{cases}$$

Let A be an adversary performing attack in MAC sense, running in time at most t, outputting or querying messages of length at most L, making at most q queries to its oracle and with advantage $\mathbf{Adv}^{\mathrm{MAC}}_H(A) \leq \varepsilon$. As the adversary A has no access to a key K chosen randomly by the environment, A's advantage against $H^{(5)}$ is $\mathbf{Adv}^{\mathrm{MAC}}_H(A)$ plus the probability, that A queries the message with suffix K. The probability that A queries the message with suffix K is $\frac{q}{|\mathcal{K}|}$ (see the proof of Theorem 7). Thus

$$\mathbf{Adv}^{\mathrm{MAC}}_{H^{(5)}}(A) \leq \mathbf{Adv}^{\mathrm{MAC}}_H(A) + \frac{q}{|\mathcal{K}|}.$$

The proof that there exists an adversary running in time $c\,\mathrm{Time}_{H,\lambda}$ and with advantage against $H^{(5)}$ in CTFP sense equal to 1, for some absolute constant c, is in the proof of Theorem 7. □

2.2.3 CTFP vs. aCTFP

If a hash function family is secure in aCTFP sense, i.e. whatever key we choose, an efficient adversary has negligible advantage of success, then it must be secure in CTFP sense, where we permit insecurity of a hash function family for some small number of keys. In this Section we give a formal proof of this intuition. We also show, that opposite implication does not hold.

Theorem 14 (aCTFP $\rightarrow$ CTFP)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. Let $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$ be a hash function family. If there exists an adversary A running in time t, outputting messages of length at most L and with advantage $\mathbf{Adv}_H^{CTFP[\lambda]}(A) = \varepsilon$, then there exists an adversary B attacking H in aCTFP sense, running in time $t + c\,Time_{H,L+\lambda}$ and with advantage $\mathbf{Adv}_H^{aCTFP[\lambda]}(B) \geq \varepsilon$, where c is an absolute constant.*

Proof. Let $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$ be a hash function family and let A be an adversary attacking H in CTFP sense, running in time t, outputting messages of length at most L and with advantage ε. Thus there must exists a key $K_0 \in \mathcal{K}$, such that when the key K_0 is chosen by the environment, then A's chance to win is at least ε, otherwise A's advantage would be smaller than ε. Thus we can construct a two-stage version of adversary B attacking H in aCTFP sense as follows:

Adversary B
[1^{st} **phase**]
  $(Y, S) \leftarrow A(K_0)$
  return (Y, K_0, S)
[2^{nd} **phase with input** (P, S)]
  $M \leftarrow A(P, S)$
  return M

Note that running time of B is $t + c\,\mathrm{Time}_{H,L+\lambda}$ for some absolute constant c. From the assumption that K_0 is the key, where A's chance to win against H in CTFP sense is at least ε we have, that B's chance to win against H in aCTFP sense is at least ε too. □

Theorem 15 (CTFP $\not\rightarrow$ aCTFP to $1/|\mathcal{K}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$, which is (t, L, ε)-CTFP, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$, which is $(t, L, \varepsilon + \frac{1}{|\mathcal{K}|})$-CTFP, but there exists an adversary C attacking H' in aCTFP sense, running in time $c\,Time_{H,\lambda}$ and with advantage $\mathbf{Adv}_{H'}^{aCTFP[\lambda]}(C) = 1$, where c is an absolute constant.*

Proof. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a (t, L, ε)-CTFP hash function family and consider the construction $H^{(8)}$ from the Figure 2.1.

$$H_K^{(8)}(M) = \begin{cases} \text{Suffix}_y(M) & \text{if } K = K_0 \\ H_K(M) & \text{otherwise} \end{cases}$$

Let A be an adversary attacking in CTFP sense, running in time at most t, outputting messages of length at most L and with advantage against H smaller than or equal to ε. If a key $K \neq K_0$ is chosen by the environment, then A's chance to win against $H^{(8)}$ is the same as against H. On the other hand, if the key K_0 is chosen by the environment, then A's chance to win against $H^{(8)}$ can be in the worst case 1. Thus

$$\mathbf{Adv}_{H^{(8)}}^{CTFP[\lambda]}(A) \leq \mathbf{Adv}_{H}^{CTFP[\lambda]}(A) + \frac{1}{|\mathcal{K}|}.$$

Now we need to show, that $H^{(8)}$ is not secure in aCTFP sense. For that reason consider the following adversary:

Adversary C
[1^{st} **phase**]
  return $(0^y, K_0, K_0)$
[2^{nd} **phase with input** (P, S)]
  return 0^y

Running time of the adversary C is $c\,\text{Time}_{H,\lambda}$ and its advantage against $H^{(8)}$ in aCTFP sense is 1, where c is an absolute constant determined by the time needed to return triple $(0^y, K_0, K_0)$ in the first phase and 0^y in the second phase on a particular RAM model. □

2.2.4 Pre, Sec, Coll, MAC vs. aCTFP

For briefer presentation let *Atks* temporarily denote the set {Pre, aPre, ePre, Sec, aSec, eSec, MAC}. In the Sections 2.2.1 and 2.2.2 we showed relationships between CTFP and members of set Atks. The same relations holds when CTFP is replaced by aCTFP and the proofs of these relationships are very similar to ones between CTFP and Atks. For that reason we provide these relations compacted together in the following two theorems without full proof, but with references where a similar proof can be found. Different situation is between Coll and aCTFP, in Theorem 9 we showed, that Coll implies CTFP. On the other hand, Coll can not imply aCTFP, as the definition of collision resistance permits insecurity in some small numbers of keys, what is in contrast with the meaning of "always" notions. The exact proof that Coll nonimplies aCTFP we provide in Theorem 17.

Theorem 16 (Pre, Sec, Coll, MAC $\not\to$ aCTFP)**.**

(1) Pre, aPre, Sec, aSec, Coll $\not\to$ aCTFP to $\frac{1}{|\mathcal{K}|}$

(2) ePre, eSec $\not\to$ aCTFP to $\frac{1}{|\mathcal{K}|}$

(3) MAC $\not\to$ aCTFP to $\frac{q+1}{|\mathcal{K}|}$

Proof. We do not provide the exact proof for this theorem, as the proof for (1) is similar to the one of Theorem 7, the proof for (2) is similar to the proof of Theorem 8 and finally the proof for (3) is similar to the one of Theorem 13. All these proofs use the same construction $H^{(5)}$ from the Figure 2.1. The proofs for (1), (2), (3) differ from their CTFP versions in Theorems 7, 8, 13 only in the last part, where we need to modify the adversary C to perform attack in aCTFP sense as follows:

Adversary C
[1^{st} **phase**]
return $(1^y, 1^k, 1^k)$
[2^{nd} **phase with input** (P, S)]
return 1^k

□

Theorem 17 (Coll $\not\to$ aCTFP to $1/|\mathcal{K}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, L, ε)-Coll, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, L, \varepsilon + \frac{1}{|\mathcal{K}|})$-Coll, but there exists an adversary C attacking H' in aCTFP sense, running in time $c\,Time_{H,\lambda}$ and with advantage $\mathbf{Adv}_{H'}^{aCTFP[\lambda]}(C) = 1$, where c is an absolute constant.*

Proof. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a (t, L, ε)-Coll hash function family and consider the construction $H^{(8)}$ from the Figure 2.1.

$$H_K^{(8)}(M) = \begin{cases} \text{Suffix}_y(M) & \text{if } K = K_0 \\ H_K(M) & \text{otherwise} \end{cases}$$

Suppose that A is an adversary attacking H in Coll sense, running in time t, outputting messages of length at most L and with advantage $\mathbf{Adv}_H^{\text{Coll}}(A) = \varepsilon$. Consider A's advantage against $H^{(8)}$. If a key K chosen by the environment is different from K_0, then A's chance to win against $H^{(8)}$ is the same as against H. However, when the key K_0 is chosen by the environment, then in the worst case A can win with probability 1. Thus

$$\mathbf{Adv}_{H^{(8)}}^{Coll}(A) \leq \mathbf{Adv}_H^{Coll}(A) + \frac{1}{|\mathcal{K}|},$$

and therefore $H^{(8)}$ is $(t, L, \varepsilon + \frac{1}{|\mathcal{K}|})$-Coll. The second part of the proof, that $H^{(8)}$ is insecure in aCTFP sense can be found in the proof of Theorem 15. □

Theorem 18 (aCTFP $\not\to$ Pre, Sec, Coll, MAC)**.**

(1) aCTFP $\not\to$ Pre, aPre, ePre to $\frac{y}{|\mathcal{M}|}$

(2) aCTFP $\not\to$ Sec, aSec, eSec, Coll

(3) aCTFP $\not\to$ MAC to $2^{1-\lambda}$

Proof. Similarly to Theorem 16, we do not provide the full proof for this theorem. The proof for (1) is identical to one of Theorem 11 (we just need to replace CTFP with aCTFP), similarly the proof for (2) is the same as in Theorem 10 and finally the proof for (3) is the same as the proof of Theorem 12. The security of constructions $H^{(6)}$, $H^{(4)}$ and $H^{(7)}$ used in the proofs does not depend on selection of the key, thus these constructions are also aCTFP secure (if a hash function family H is aCTFP secure). □

2.3 Pseudo-random function

Adversary attacking in Prf sense does not have access to a key K, chosen randomly by the environment. Thus we have similar situation here, as it was in MAC case (Section 2.1). In fact we use the same constructions as we used in MAC case, however we need to slightly adopt the proofs as Prf and MAC security notions are different.

In the Section 1.2.6 we showed, that any adversary attacking in Prf sense cannot have advantage 1, what causes that we have only provisional implications and separations here.

2.3.1 Coll vs. Prf

Theorem 19 (Coll $\not\to$ Prf to $1/|\mathcal{Y}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, L, ε)-Coll, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t + c_1\, Time_{H,L+\lambda}, L, \varepsilon)$-Coll, but there exists an adversary C attacking H' in Prf sense, running in time $c_2\, Time_{H,\lambda}$, making one query to its oracle and with advantage $\mathbf{Adv}^{Prf}_{H'}(A) = 1 - \frac{1}{|\mathcal{Y}|}$, where c_1 and c_2 are absolute constants.*

Proof. We use the construction $H^{(2)}$ from the Figure 2.1. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a (t, L, ε)-Coll hash function family, then we define:

$$H_K^{(2)}(M) = \begin{cases} 0^y & \text{if } M = 0 \\ H_K(M) & \text{if } M \neq 0 \text{ and } H_K(M) \neq 0^y \\ H_K(0) & \text{otherwise} \end{cases}$$

in Theorem 2 we proved, that if H is secure in Coll sense, then $H^{(2)}$ is too, what completes the first part of the proof. Thus we only need to prove, that $H^{(2)}$ is not secure in Prf sense. For that purpose consider the following adversary C^f with oracle access to some function f.

Adversary C^f
  if $f(0) = 0^y$ **then return** 1
  otherwise return 0

Running time of the adversary C is $c\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c. Let K be a key chosen randomly by the environment. The probability that C returns 1 if its oracle is $H_K^{(2)}$ is 1. The probability that C returns 1 if its oracle is a function chosen randomly by the environment from the set $Func(\mathcal{M}, \mathcal{Y})$ is $\frac{1}{|\mathcal{Y}|}$, as the number of all functions mapping from $\mathcal{M}$ to $\mathcal{Y}$ is $|\mathcal{Y}|^{|\mathcal{M}|}$ and the number of functions from $\mathcal{M}$ to $\mathcal{Y}$ that maps 0 to 0^y is $|\mathcal{Y}|^{|\mathcal{M}|-1}$. Thus

$$\mathbf{Adv}_{H^{(2)}}^{Prf}(C) = 1 - \frac{|\mathcal{Y}|^{|\mathcal{M}|-1}}{|\mathcal{Y}|^{|\mathcal{M}|}} = 1 - \frac{1}{|\mathcal{Y}|},$$

what we wanted to prove. □

Theorem 20 (Prf $\not\to$ Coll to $q/|\mathcal{K}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, q, L, ε)-Prf, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-Prf, but there exists an adversary C attacking H' in Coll sense, running in time $c\,Time_{H,\lambda}$ and with advantage $\mathbf{Adv}_{H'}^{Coll}(C) = 1$, where c is an absolute constant.*

Proof. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family, which is (t, q, L, ε)-Prf and consider the construction $H^{(1)}$ from the Figure 2.1 defined as follows:

$$H_K^{(1)}(M) = \begin{cases} H_K(M) & \text{if } M \neq K \\ H_K(0^k) & \text{if } M = K \end{cases}$$

Let A be any adversary performing attack in Prf sense, running in time at most t and making at most q queries of length at most L. From the assumption we have, that A's advantage against H is smaller than or equal to ε. When A is attacking $H^{(1)}$, it can notice some difference (i.e. return different output) from the case when attacking

H only, when it queries the message M_0 equal to the key K chosen randomly by the environment. However, the adversary A has no access to the key, therefore its probability of querying the message M_0 is $\frac{q}{|\mathcal{K}|}$. Thus

$$\mathbf{Adv}^{\mathrm{Prf}}_{H^{(1)}}(A) \leq \mathbf{Adv}^{\mathrm{Prf}}_{H}(A) + \frac{q}{|\mathcal{K}|} \leq \varepsilon + \frac{q}{|\mathcal{K}|},$$

what we wanted to prove.

The adversary C, which returns pair $(0^k, K)$ has advantage $\mathbf{Adv}^{Coll}_{H^{(1)}}(C) = 1$ (as $H^{(1)}(0^k) = H^{(1)}(K)$) and it runs in time $c\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c determined by the time needed to return $(0^k, K)$ on a particular RAM model. □

2.3.2 Pre, Sec vs. Prf

Theorem 21 (Pre, aPre, ePre $\not\to$ Prf to $1/|\mathcal{Y}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exist a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, L, ε)-Pre (aPre, ePre), then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t + c_1\ Time_{H,L+\lambda}, L, \varepsilon)$-Pre (aPre, ePre), but there exists an adversary C attacking H' in Prf sense, running in time $c_2\ Time_{H,\lambda}$, making two queries to its oracle and with advantage $\mathbf{Adv}^{Prf}_{H'}(C) = 1 - \frac{1}{|\mathcal{Y}|}$, where c_1 and c_2 are absolute constants.*

Proof. Consider the construction $H^{(4)}$ from the Figure 2.1

$$H^{(4)}_K(M) = H_K(M[1 \ldots |M| - 1]||0),$$

where $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ is a (t, L, ε)-Pre (aPre, ePre) hash function family. In the proof of Theorem 5 we proved, that $H^{(4)}$ is $(t + c_1\,\mathrm{Time}_{H,L+\lambda}, L, \varepsilon)$-Pre (aPre, ePre) for some absolute constant c_1.

Now we construct an adversary C performing attack in Prf sense against $H^{(4)}$.

Adversary C^f
  if $f(00) = f(01)$ **then return** 1
  otherwise return 0

Running time of C is $c_2\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c_2 and it is making 2 queries to its oracle. When C's oracle is $H^{(4)}_K$ for some key K, then C returns 1 with probability 1. Number of all functions mapping from $\mathcal{M}$ to $\mathcal{Y}$ is $|\mathcal{Y}|^{|\mathcal{M}|}$, from which $|\mathcal{Y}|^{|\mathcal{M}|-1}$ are those, where the messages 00 and 01 map to the same image. Therefore

$$\mathbf{Adv}^{\mathrm{Prf}}_{H^{(4)}}(C) = 1 - \frac{1}{|\mathcal{Y}|},$$

what completes the proof. □

Theorem 22 (Sec, aSec, eSec $\not\to$ Prf to $1/|\mathcal{Y}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exist a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, L, ε)-Sec (aSec, eSec), then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t + c_1\, Time_{H,L+\lambda}, L, \varepsilon)$-Sec (aSec, eSec), but there exists an adversary C attacking H' in Prf sense, running in time $c_2\, Time_{H,\lambda}$, making one query to its oracle and with advantage $\mathbf{Adv}^{Prf}_{H'}(C) = 1 - \frac{1}{|\mathcal{Y}|}$, where c_1 and c_2 are absolute constants.*

Proof. We use the construction $H^{(2)}$ from the Figure 2.1. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family, which is (t, L, ε)-Sec (aSec, eSec), then consider the hash function family

$$H^{(2)}_K(M) = \begin{cases} 0^y & \text{if } M = 0 \\ H_K(M) & \text{if } M \neq 0 \text{ and } H_K(M) \neq 0^y \\ H_K(0) & \text{otherwise} \end{cases}$$

The first part of the proof is in the proof of Theorem 3, where we proved, that $H^{(2)}$ is $(t + c_1 \operatorname{Time}_{H,L+\lambda}, L, \varepsilon)$-Sec (aSec, eSec) for some absolute constant c_1. On the other hand, in Theorem 19 we showed, that there exists an adversary C attacking $H^{(2)}$ in Prf sense, running in time $c_2 \operatorname{Time}_{H,\lambda}$ for some absolute constant c_2, making one query to its oracle and with advantage $\mathbf{Adv}^{\mathrm{Prf}}_{H^{(2)}}(C) = 1 - \frac{1}{|\mathcal{Y}|}$, what is the second part of the proof. Thus the proof is complete. □

Theorem 23 (Prf $\not\to$ Pre, aPre, ePre, Sec, aSec, eSec to $q/|\mathcal{K}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, q, L, ε)-Prf, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-Prf, but there exists an adversary C_{xxx} attacking H' in xxx sense, running in time $c\, Time_{H,\lambda}$ and with advantage $\mathbf{Adv}^{xxx[\cdot]}_{H'}(C_{xxx}) = 1$, where c is an absolute constant and xxx $\in \{$Pre, aPre, ePre, Sec, aSec, eSec$\}$.*

Proof. Suppose, that $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ is a (t, q, L, ε)-Prf hash function family and consider the following construction $H^{(3)}$ from the Figure 2.1:

$$H^{(3)}_K(M) = \begin{cases} Y & \text{if } \mathrm{Prefix}_{(k+1+y)}(M) = K||b||Y \text{ for some } b \in \{0,1\} \\ H_K(M) & \text{otherwise} \end{cases}$$

in Theorems 6 and 4 we proved, that for xxx $\in$ {Pre, aPre, ePre, Sec, aSec, eSec} there exists an adversary C_{xxx} attacking $H^{(3)}$ in xxx sense, running in time $c \operatorname{Time}_{H,\lambda}$ and with advantage $\mathbf{Adv}^{\mathrm{xxx}[\cdot]}_{H'}(C_{\mathrm{xxx}}) = 1$. Thus we only need to prove, that $H^{(3)}$ is $(t, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-Prf. Let A be an adversary performing attack in Prf sense, running in time at most t, making at most q queries to its oracle each of length at most L. From the assumption we have, that $\mathbf{Adv}^{\mathrm{Prf}}_H(A) \leq \varepsilon$. When A is attacking $H^{(3)}$, it can

notice a difference only when it queries the message with prefix K, where K is some key chosen randomly by the environment. However A has no access to the key, thus it can only guess. in Theorem 6 we had the similar problem and we showed, that

$$\mathbf{Adv}^{\mathrm{Prf}}_{H^{(3)}}(A) \leq \mathbf{Adv}^{\mathrm{Prf}}_{H}(A) + \frac{q}{|\mathcal{K}|} \leq \varepsilon + \frac{q}{|\mathcal{K}|}.$$

Thus $H^{(3)}$ is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-Prf, what we wanted to prove. □

2.3.3 CTFP, aCTFP vs. Prf

Theorem 24 (CTFP, aCTFP $\not\to$ Prf to $1/|\mathcal{Y}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, L, ε)-CTFP (aCTFP), then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, L, \varepsilon + \frac{2}{2^\lambda})$-CTFP (aCTFP), but there exists an adversary C attacking H' in Prf sense, running in time $c\,\mathrm{Time}_{H,\lambda}$, making one query to its oracle and with advantage $\mathbf{Adv}^{Prf}_{H'}(C) = 1 - \frac{1}{|\mathcal{Y}|}$, where c is an absolute constant.*

Proof. Suppose, that $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ is (t, L, ε)-CTFP (aCTFP), and consider the construction $H^{(7)}$ from the Figure 2.1:

$$H^{(7)}_K(M) = \begin{cases} 0^y & \text{if } M = 0 \\ H_K(M) & \text{otherwise} \end{cases}$$

in Theorem 12 (18) we showed, that $H^{(7)}$ is $(t, L, \varepsilon + \frac{2}{2^\lambda})$-CTFP (aCTFP). For the second part of the proof consider the following adversary performing attack in Prf sense and with oracle access to some function f:

Adversary C^f
  if $f(0) = 0^y$ **then return** 1
  otherwise return 0

The running time of the adversary C is $c\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c. If C's oracle is $H^{(7)}_K$ for some key K, then C's chance to win is 1. On the other hand, if C's oracle is a function $f : \mathcal{M} \to \mathcal{Y}$ chosen randomly by the environment, its chance to win is $\frac{1}{|\mathcal{Y}|}$, as the probability that $f(0) = 0^y$ is $\frac{1}{|\mathcal{Y}|}$. Thus C's advantage against $H^{(7)}$ is

$$\mathbf{Adv}^{\mathrm{Prf}}_{H^{(7)}}(C) = 1 - \frac{1}{|\mathcal{Y}|},$$

what we wanted to prove. □

In the theorem above we assume that $\frac{2}{2^\lambda} \leq \frac{1}{|\mathcal{Y}|}$ (actually from the page 35 we assume that $\mathcal{M} = \{0,1\}^*$, $\mathcal{Y} = \{0,1\}^y$, thus there are only $2y$ possible values of λ so that $\frac{2}{2^\lambda} \leq \frac{1}{|\mathcal{Y}|}$, but for infinitely many values of λ holds $\frac{2}{2^\lambda} > \frac{1}{|\mathcal{Y}|}$), thus we can write *CTFP,*

aCTFP nonimplies Prf to $1/|\mathcal{Y}|$. Otherwise we would replace $1/|\mathcal{Y}|$ with $2^{1-\lambda}$ and write *CTFP, aCTFP nonimplies Prf to* $2^{1-\lambda}$.

Theorem 25 (Prf $\not\to$ CTFP, aCTFP to $q/|\mathcal{K}|$)**.** *Let* λ *be an arbitrary number such that* $\{0,1\}^\lambda \subseteq \mathcal{M}$*. If there exists a hash function family* $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$*, which is* (t, q, L, ε)*-Prf, then there exists a hash function family* $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$*, which is* $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$*-Prf, but there exists an adversary* C_{xxx} *attacking* H' *in xxx sense, running in time* $c\,Time_{H,\lambda}$ *and with advantage* $\mathbf{Adv}_{H'}^{xxx[\lambda]}(C_{xxx}) = 1$*, where* c *is an absolute constant and* $xxx \in \{CTFP, aCTFP\}$*.*

Proof. We use the construction $H^{(5)}$ from the Figure 2.1. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a (t, q, L, ε)-Prf hash function family and consider the following construction:

$$H_K^{(5)}(M) = \begin{cases} K[1 \dots \min\{k, y\}] & \text{if } \mathrm{Suffix}_k(M) = K \\ H_K(M) & \text{otherwise} \end{cases}$$

in Theorems 13 (16) we showed, that $H^{(5)}$ is not CTFP (aCTFP) resistant, as there exists an adversary C_{CTFP} (C_{aCTFP}) attacking $H^{(5)}$ in CTFP (aCTFP) sense, running in time $c\,\mathrm{Time}_{H,\lambda}$ with advantage 1, where c is an absolute constant. Thus we only need to show that $H^{(5)}$ is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-Prf. Any adversary A attacking $H^{(5)}$ in Prf sense can notice some difference only when it queries the message with suffix K. Thus if the running time of A is at most t and it makes at most q queries of length at most L, then its advantage against $H^{(5)}$ cannot be greater than the A's advantage against H plus $\frac{q}{|\mathcal{K}|}$ (we have the same situation as in the proof of Theorem 13). Thus the following holds:

$$\mathbf{Adv}_{H^{(5)}}^{\mathrm{Prf}}(A) \le \mathbf{Adv}_{H}^{\mathrm{Prf}}(A) + \frac{q}{|\mathcal{K}|} \le \varepsilon + \frac{q}{|\mathcal{K}|},$$

what completes the proof. □

2.4 Pseudo-random oracle

Pseudo-random oracle seems to be the strongest property. As we will see in this Section, it implies almost all the other security notions. When a hash function family is Pro secure, then it is indifferentiable from a random oracle and it is hard (effectively unfeasible) for non-Pro adversaries (i.e. adversaries attacking in Pre, Sec, Coll,... sense) to win against a random oracle.

Pseudo-random oracle does not imply "always" versions of preimage resistance and second preimage resistance, as it permits insecurity in some small number of keys (what is in contrast to the meaning of "always" notions).

In the following text we will assume, that a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ is build from some ideal compression function $f : \{0,1\}^{y+d} \to \mathcal{Y}; d > 0$ and an algorithm computing H has oracle access to f. For that reason we need to give the oracle access to f also to adversaries attacking in non-Pro sense (i.e. in Pre, aPre, ePre, Sec, aSec, eSec, Coll, CTFP or aCTFP), adversaries attacking in Pro sense already have such access. For example, the advantage in Pre sense of adversary A would look like follows:

$$\mathbf{Adv}^{\mathrm{Pre}}_{H,f}(A) \quad = \quad \Pr\left[K \xleftarrow{\$} \mathcal{K}; M \xleftarrow{\$} \mathcal{M}; Y \leftarrow H_K(M); M' \leftarrow A^f(K, Y) : H_K(M') = Y\right]$$

Advantages in other senses are modified similarly, except MAC and Prf notions. The advantages in MAC and PRF senses stay unchanged, as an adversary attacking in MAC or Prf sense has only oracle access to H. We omit writing H^f even if H has oracle access to f, as all the hash functions used in this section has oracle access to f (in other words they are build from the primitive f).

2.4.1 Pre, Sec, Coll vs. Pro

Theorem 26 (Pre, aPre, ePre $\not\to$ Pro to $\frac{1}{|\mathcal{Y}|}$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$ and let $f = RF_{y+d,y}$ for some $d > 0$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, L, ε)-Pre (aPre, ePre), then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t + c_1\, Time_{H,L+\lambda}, L, \varepsilon)$-Pre (aPre, ePre), but there exists an adversary C attacking H' in Pro sense, running in time $c_2\, Time_{H,\lambda}$, making two queries to its first oracle and with advantage $\mathbf{Adv}^{Pro}_{H',f,\mathcal{S}}(C) = 1 - \frac{1}{|\mathcal{Y}|}$ for any simulator $\mathcal{S}$, where c_1 and c_2 are absolute constants.*

Proof. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family, which is (t, L, ε)-Pre (aPre, ePre). We construct the hash function family $H^{(4)}$ (from the Figure 2.1) defined as follows:

$$H^{(4)}_K(M) = H_K(M[1 \ldots |M| - 1]||0)$$

In the proof of Theorem 5 we proved that $H^{(4)}$ is $(t + c_1\,\mathrm{Time}_{H,L+\lambda}, L, \varepsilon)$-Pre (aPre, ePre). We only need to find an adversary breaking $H^{(4)}$ in Pro sense. For that reason consider the following adversary C performing attack in Pro sense with oracles $f_1 : \mathcal{M} \to \mathcal{Y}$ and $f_2 : \{0,1\}^{y+d} \to \mathcal{Y}$ for some integer $d > 0$:

Adversary C^{f_1,f_2}
  if $f_1(00) = f_1(01)$ **then return** 1
  otherwise return 0

If the first oracle of the adversary C is H_K for some key K, then it returns 1 with probability 1. If its first oracle is $\mathcal{F}$ (a random function), then for any simulator $\mathcal{S}$ the probability that it returns 1 is $\frac{1}{|\mathcal{Y}|}$ (see the proof of Theorem 21). Thus C's advantage

against $H^{(4)}$ is $\mathbf{Adv}^{\mathrm{Pro}}_{H^{(4)},f,\mathcal{S}}(C) = 1 - \frac{1}{|\mathcal{Y}|}$ for any simulator $\mathcal{S}$. The running time of C is $c_2\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c_2 and it makes two queries to its first oracle. □

Theorem 27 (Sec, aSec, eSec, Coll $\not\to$ Pro to $\frac{1}{|\mathcal{Y}|}$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$ and let $f = RF_{y+d,y}$ for some $d > 0$. If there exists a hash function family $H : \mathcal{K}\times\mathcal{M} \to \mathcal{Y}$, which is (t, L, ε)-xxx (xxx $\in$ {Sec, eSec, aSec, Coll}, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t + c_1\ Time_{H,L+\lambda}, L, \varepsilon)$-xxx, but there exists an adversary C attacking H' in Pro sense, running in time $c_2\ Time_{H,\lambda}$, making one query to its first oracle and with advantage $\mathbf{Adv}^{Pro}_{H',f,\mathcal{S}}(C) = 1 - \frac{1}{|\mathcal{Y}|}$ for any simulator $\mathcal{S}$, where c_1 and c_2 are absolute constants.*

Proof. Let xxx denote a member from the set {Sec, eSec, aSec, Coll}. Assume that $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ is (t, L, ε)-xxx and consider construction $H^{(2)}$ from the figure 2.1:

$$H^{(2)}_K(M) = \begin{cases} 0^y & \text{if } M = 0 \\ H_K(M) & \text{if } M \neq 0 \text{ and } H_K(M) \neq 0^y \\ H_K(0) & \text{otherwise} \end{cases}$$

in Theorem 2 we proved, that $H^{(2)}$ is $(t + c_1\,\mathrm{Time}_{H,L+\lambda}, L, \varepsilon)$-Coll and in Theorem 3 we proved that $H^{(2)}$ is $(t + c_1\,\mathrm{Time}_{H,L+\lambda}, L, \varepsilon)$-Sec (aSec, eSec). To prove the second part of Theorem consider the adversary C performing attack in Pro sense with oracles f_1 and f_2, which just verifies whether $f_1(0) = 0^y$. If so, it returns 1, otherwise it returns 0 (the adversary is similar to one in the proof of Theorem 22). Running time of such adversary is $c_2\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c_2, it makes one query to its first oracle and its advantage for any simulator $\mathcal{S}$ is $1 - \frac{1}{|\mathcal{Y}|}$. □

Theorem 28 (Pro $\to$ Pre, ePre, Sec, eSec, Coll to $1/|\mathcal{Y}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$, $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family, $f = RF_{y+d,y}$ for some $d > 0$ and let xxx $\in$ {Pre, ePre, Sec, eSec, Coll}. If there exists an adversary A_{xxx} running in time t, outputting messages of length at most L and with advantage in xxx sense $\mathbf{Adv}^{xxx[\cdot]}_{H,f}(A_{xxx}) = \varepsilon$, then there exists an adversary B_{xxx} attacking H in Pro sense, running in time $t+c\ Time_{H,L+\lambda}$, making at most two queries to its first oracle and with advantage $\mathbf{Adv}^{Pro}_{H,f,\mathcal{S}}(B_{xxx}) \geq \varepsilon - \frac{1}{|\mathcal{Y}|}$ for some simulator $\mathcal{S}$, where c is an absolute constant.*

Proof. Let H be a hash function family and A_{xxx} be an adversary attacking H in xxx (xxx $\in$ {Pre, ePre, Sec, eSec, Coll}) sense, running in time t, outputting messages of length at most L and with advantage ε. Consider the following adversaries B_{xxx} attacking H in Pro sense:

Adversary $B^{f_1,f_2}_{\mathrm{Pre}}(K)$ **let** $Y \xleftarrow{\$} \mathcal{Y}$ $M \leftarrow A^{f_2}_{\mathrm{Pre}}(K, Y)$ **if** $f_1(M) = Y$ **then return** 1 **otherwise return** 0	**Adversary** $B^{f_1,f_2}_{\mathrm{ePre}}(K)$ $(Y, S) \leftarrow A^{f_2}_{\mathrm{ePre}}$ $M \leftarrow A^{f_2}_{\mathrm{ePre}}(K, S)$ **if** $f_1(M) = Y$ **then return** 1 **otherwise return** 0
Adversary $B^{f_1,f_2}_{\mathrm{Sec}}(K)$ **let** $M \xleftarrow{\$} \{0,1\}^\lambda$ $M' \leftarrow A^{f_2}_{\mathrm{Sec}}(K, M)$ **if** $f_1(M) = f_1(M')$ **then return** 1 **otherwise return** 0	**Adversary** $B^{f_1,f_2}_{\mathrm{eSec}}(K)$ $(M, S) \leftarrow A^{f_2}_{\mathrm{eSec}}$ $M' \leftarrow A^{f_2}_{\mathrm{eSec}}(K, S)$ **if** $f_1(M) = f_1(M')$ **then return** 1 **otherwise return** 0
Adversary $B^{f_1,f_2}_{\mathrm{Coll}}(K)$ $(M, M') \leftarrow A^{f_2}_{\mathrm{Coll}}(K)$ **if** $f_1(M) = f_1(M')$ **then return** 1 **otherwise return** 0	

The adversaries above firstly simulate the adversary A_{xxx} attacking in xxx sense, then they verify, whether A_{xxx} returned correct output. If so, they return 1, otherwise they return 0. The probability, that they return 1 if their oracles are H_K and f for some key K is ε (what is equal to the advantage of A_{xxx} against H). If their oracles are a random function $\mathcal{F}$ and the simulator $\mathcal{S}^{\mathcal{F}}$ with oracle $\mathcal{F}$, which always returns 0^y (whatever is its oracle), then they return 1 with the probability $\frac{1}{|\mathcal{Y}|}$. The adversary $A^{S^{\mathcal{F}}}_{\mathrm{xxx}}$ can not win against $\mathcal{F}$ with non-negligible probability, as $\mathcal{F}$ is a random oracle (and any adversary attacking random oracle is doomed to fail) and $\mathcal{S}^{\mathcal{F}}$ always returns 0^y (i.e. $\mathcal{S}^{\mathcal{F}}$ does not return output that is "consistent" with $\mathcal{F}$). In fact if $A^{\mathcal{S}^{\mathcal{F}}}_{\mathrm{Pre}}(Y)$ returns message M, the probability that $\mathcal{F}(M) = Y$ (i.e. that $A^{\mathcal{S}^{\mathcal{F}}}_{\mathrm{Pre}}(Y)$ wins) is $\frac{1}{|\mathcal{Y}|}$. Similar situation is for $A^{\mathcal{S}^{\mathcal{F}}}_{\mathrm{ePre}}$, $A^{\mathcal{S}^{\mathcal{F}}}_{\mathrm{Sec}}$ and $A^{\mathcal{S}^{\mathcal{F}}}_{\mathrm{eSec}}$.

Thus the advantage of B_{xxx} against H is:

$$\mathbf{Adv}^{\mathrm{Pro}}_{H,f,\mathcal{S}}(B_{\mathrm{xxx}}) = \varepsilon - \frac{1}{|\mathcal{Y}|},$$

for the simulator $\mathcal{S}$ given above, therefore for the advantage of adversaries B_{xxx} and the simulator $\mathcal{S}$ the following holds:

$$\mathbf{Adv}^{\mathrm{Pro}}_{H,f,\mathcal{S}}(B_{\mathrm{xxx}}) + \frac{1}{|\mathcal{Y}|} \geq \mathbf{Adv}^{\mathrm{xxx}[\cdot]}_{H}(A_{\mathrm{xxx}}).$$

Note that running time of the adversaries B_{xxx} is $t + c\,\mathrm{Time}_{H,L+\lambda}$ for some absolute constant c. We expect that there exists a sampler from the set $\{0,1\}^\lambda$ ($\mathcal{Y}$), which can sample messages (images) with uniform distribution in time $a\,\mathrm{Time}_{H,\lambda}$ for some absolute constant $a \leq c$. □

Theorem 29 (Pro $\not\to$ aPre, aSec to $1/|\mathcal{K}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$ and let $f = RF_{y+d,y}$ for some $d > 0$. If there exists a hash function*

family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t_A, t_{\mathcal{S}}, q_1, q_2, L, \varepsilon)$-Pro, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t_A, t_{\mathcal{S}}, q_1, q_2, L, \varepsilon + \frac{1}{|\mathcal{K}|})$-Pro, but there exists an adversary C_{xxx} attacking H' in xxx sense, running in time $c\,Time_{H,\lambda}$ and with advantage $\mathbf{Adv}^{xxx[\lambda]}_{H'}(C_{xxx}) = 1$, where c is an absolute constants and $xxx \in \{aPre, aSec\}$.

Proof. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a $(t_A, t_{\mathcal{S}}, q_1, q_2, L, \varepsilon)$-Pro hash function family, we use the construction $H^{(9)}$ from the Figure 2.1. Let $K_0 \in \mathcal{K}$ be some particular key, then we define

$$H^{(9)}_K(M) = \begin{cases} H_K(M) & \text{if } K \neq K_0 \\ 0^y & \text{if } K = K_0 \end{cases}$$

The hash function family $H^{(9)}$ is clearly $(t_A, t_{\mathcal{S}}, q_1, q_2, L, \varepsilon + \frac{1}{|\mathcal{K}|})$-Pro. When a key K chosen by the environment is different from K_0, then chance to win of any adversary against $H^{(9)}$ is the same as against H. If the key K is equal to K_0, then in the worst case an adversary wins against $H^{(9)}$ with probability 1. The probability that the key K is equal to K_0 is $\frac{1}{|\mathcal{K}|}$.

On the other hand $H^{(9)}$ is clearly not aPre (aSec) secure. The two stage adversary C_{xxx} attacking in xxx sense (xxx $\in$ {aPre, aSec}), which in the first phase returns the key K_0 and in the second phase returns the message 0 (or any other message different from the one chosen by the environment when attacking in aSec sense) has the advantage 1 and runs in time $c\,\text{Time}_{H,\lambda}$ for some absolute constant c (determined by the time needed to return the key K_0 and message 0 on a particular RAM model). □

2.4.2 MAC vs. Pro

Theorem 30 (MAC $\not\to$ Pro to $q/|\mathcal{K}| + 1/|\mathcal{Y}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$ and let $f = RF_{y+d,y}$ for some $d > 0$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, q, L, ε)-MAC, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-MAC, but there exists an adversary C attacking H' in Pro sense, running in time $c\,Time_{H,\lambda}$, making one query to its first oracle and with advantage $\mathbf{Adv}^{Pro}_{H',f,\mathcal{S}}(C) = 1 - \frac{1}{|\mathcal{Y}|^2}$ for any simulator $\mathcal{S}$, where c is an absolute constant.*

Proof. Consider the construction $H^{(3)}$ from the Figure 2.1:

$$H^{(3)}_K(M) = \begin{cases} Y & \text{if } \text{Prefix}_{(k+1+y)}(M) = K||b||Y \text{ for some } b \in \{0,1\} \\ H_K(M) & \text{otherwise} \end{cases}$$

where $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ is a (t, q, L, ε)-MAC hash function family. in Theorem 4, we

proved, that $H^{(3)}$ is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-MAC. For the second part of the proof consider the following adversary C:

Adversary $C^{f_1,f_2}(K)$
  if $f_1(K||0||0^y) = f_1(K||1||0^y) = 0^y$ **then return** 1
  otherwise return 0

Running time of C is $c\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c and its advantage in Pro sense against $H^{(3)}$ is

$$\mathbf{Adv}^{\mathrm{Pro}}_{H^{(3)},f}(C) = 1 - \frac{1}{|\mathcal{Y}|^2}$$

for any simulator $\mathcal{S}$, where $\frac{1}{|\mathcal{Y}|^2}$ is the probability that a random function $\mathcal{F}$ maps messages $K||0||0^y$ and $K||1||0^y$ to 0^y. □

Theorem 31 (Pro $\to$ MAC to $1/|\mathcal{Y}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$, $f = RF_{y+d,y}$ for some $d > 0$ and $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family. If there exists an adversary A running in time t, outputting or querying messages of length at most L, making q queries to its oracle and with advantage in MAC sense $\mathbf{Adv}^{MAC}_H(A) = \varepsilon$, then there exists an adversary B attacking H in Pro sense, running in time $t + c\,Time_{H,L+\lambda}$, making at most two queries to its first oracle and with advantage $\mathbf{Adv}^{Pro}_{H,f,\mathcal{S}}(A) \geq \varepsilon - \frac{1}{|\mathcal{Y}|}$ for some simulator $\mathcal{S}$, where c is an absolute constant.*

Proof. The proof is very similar to the one of Theorem 28. Let A_{MAC} be an adversary attacking H in MAC sense, running in time t, outputting or querying messages of length at most L, making q queries to its oracle and with advantage ε. We just need to create MAC version of the adversary B from the proof of Theorem 28:

Adversary $B^{f_1,f_2}_{\mathrm{MAC}}(K)$
  $(M, Y) \leftarrow A^{H_K}_{\mathrm{MAC}}$
  if $f_1(M) = Y$ **then return** 1
  otherwise return 0

Running time of such adversary is $t + c\,\mathrm{Time}_{H,L+\lambda}$ and its advantage in Pro sense against H is $\varepsilon - \frac{1}{|\mathcal{Y}|}$ for simulator $\mathcal{S}$ that always returns 0^y no matter what its oracle is (see the proof of Theorem 28 for complete explanation). □

2.4.3 CTFP vs. Pro

Theorem 32 (CTFP $\not\to$ Pro to $1/|\mathcal{Y}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$ and let $f = RF_{y+d,y}$ for some $d > 0$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, L, ε)-CTFP (aCTFP), then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, L, \varepsilon + \frac{2}{2^\lambda})$-CTFP (aCTFP), but there exists an adversary C attacking H' in Pro sense, running in time $c\,Time_{H,\lambda}$,*

making one query to its first oracle and with advantage $\mathbf{Adv}^{Pro}_{H',f,\mathcal{S}}(A) = 1 - \frac{1}{|\mathcal{Y}|}$ *for any simulator* $\mathcal{S}$*, where* c *is an absolute constant.*

Proof. Consider the construction $H^{(7)}$ from the Figure 2.1. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a (t, L, ε)-CTFP (aCTFP) hash function family, then we define:

$$H^{(7)}_K(M) = \begin{cases} 0^y & \text{if } M = 0 \\ H_K(M) & \text{otherwise} \end{cases}$$

in Theorem 12 (18) we showed, that $H^{(7)}$ is $(t, L, \varepsilon + \frac{2}{2^\lambda})$-CTFP (aCTFP). For the second part of the proof consider the following adversary C:

Adversary $C^{f_1,f_2}(K)$
  if $f(0) = 0^y$ **then return** 1
  otherwise return 0

Running time of C is $c\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c. The advantage of such adversary in Pro sense is (see the proof of Theorem 24)

$$\mathbf{Adv}^{\mathrm{Pro}}_{H^{(7)},f,\mathcal{S}}(A) = 1 - \frac{1}{|\mathcal{Y}|}$$

for any simulator $\mathcal{S}$, what completes the proof. □

Theorem 33.

(1) Pro $\to$ *CTFP to* $\frac{1}{|\mathcal{Y}|}$

(2) Pro $\not\to$ *aCTFP to* $\frac{1}{|\mathcal{K}|}$

Proof. The proof of (1) is very similar to the proof of Theorem 28. We just need to construct CTFP version of the adversary B from the proof of Theorem 28 performing attack in Pro sense.

Adversary $B^{f_1,f_2}_{\mathrm{CTFP}}(K)$
  $(Y, S) \leftarrow A^{f_2}_{\mathrm{CTFP}}(K)$
  let $P \xleftarrow{\$} \mathcal{M}$
  $M \leftarrow A^{f_2}_{\mathrm{CTFP}}(P, S)$
  if $f_1(P||M) = Y$ **then return** 1
  otherwise return 0

The proof of (2) is nearly the same as the proof of Theorem 29. Here we just need to find an adversary C attacking the hash function family $H^{(9)}$ in aCTFP sense. The adversary C in the first phase returns triple $(0^y, K_0, K_0)$ and in the second phase it returns empty string. Advantage of such adversary in aCTFP sense against $H^{(9)}$ is 1 and it runs in time $c\,\mathrm{Time}_{H,\lambda}$ for some absolute constant c. □

2.4.4 Prf vs. Pro

Theorem 34 (Prf $\not\to$ Pro to $q/|\mathcal{K}| + 1/|\mathcal{Y}|$)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$ and let $f = RF_{y+d,y}$ for some $d > 0$. If there exists a hash function family $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is (t, q, L, ε)-Prf, then there exists a hash function family $H' : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, which is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-Prf, but there exists an adversary C attacking H' in Pro sense, running in time $c\,Time_{H,\lambda}$, making one query to its first oracle and with advantage $\mathbf{Adv}^{Pro}_{H',f,\mathcal{S}}(C) = 1 - \frac{1}{|\mathcal{Y}|^2}$ for any adversary $\mathcal{S}$, where c is an absolute constant.*

Proof. We use the construction $H^{(3)}$ from the Figure 2.1. Suppose, that $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ is a hash function family, which is (t, q, L, ε)-Prf, then consider a hash function family defined as follows:

$$H^{(3)}_K(M) = \begin{cases} Y & \text{if } \mathrm{Prefix}_{(k+1+y)}(M) = K||b||Y \text{ for some } b \in \{0,1\} \\ H_K(M) & \text{otherwise} \end{cases}$$

in Theorem 23 we showed, that $H^{(3)}$ is $(t, q, L, \varepsilon + \frac{q}{|\mathcal{K}|})$-Prf what completes the first part of the proof. in Theorem 30 we showed the second part of the proof, that there exists an adversary C attacking $H^{(3)}$ in Pro sense running in time $c\,\mathrm{Time}_{H,\lambda}$ and with advantage $1 - \frac{1}{|\mathcal{Y}|^2}$ for any simulator $\mathcal{S}$. Thus the proof is complete. □

Theorem 35 (Pro $\to$ Prf)**.** *Let λ be an arbitrary number such that $\{0,1\}^\lambda \subseteq \mathcal{M}$, $f = RF_{y+d,y}$ for some $d > 0$ and $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family. If there exists an adversary A running in time t, making q queries to its oracle of length at most L and with advantage in Prf sense $\mathbf{Adv}^{Prf}_H(A) = \varepsilon$, then there exists an adversary B attacking H in Pro sense, running in time $t + c\,Time_{H,L+\lambda}$, making at most q queries to its first oracle and with advantage $\mathbf{Adv}^{Pro}_{H,f,\mathcal{S}}(A) = \varepsilon$ for any simulator $\mathcal{S}$, where c is an absolute constant.*

Proof. Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function family and let A be an adversary running in time t, making q queries to its oracle of length at most L and with advantage in Prf sense $\mathbf{Adv}^{\mathrm{Prf}}_H(A) = \varepsilon$. Consider the following adversary B:

Adversary $B^{f_1,f_2}(K)$
  $b \leftarrow A^{f_1}$
  return b

The advantage of B in Pro sense is the same as the advantage of A in Prf sense (as B does not make queries to its second oracle and does not utilize the key). The adversary B runs in time $t + c\,\mathrm{Time}_{H,L+\lambda}$ for some absolute constant c, what completes the proof. □

2.5 Summary

In this Chapter we proved the relationships among all the security notions from the Section 1.2. Note that all provisional implications and separations we proved have their provisional part (i.e. ε value from the part "to ε") negligible (i.e. ε depends on λ, k or y and $\varepsilon(\lambda)$, $\varepsilon(k)$ or $\varepsilon(y)$ descends faster than any polynomial powered to -1), what is necessary condition for meaningfulness of a particular provisional implication or separation. However we have not proven that these provisional parts we provided are the best ones — i.e. the ε value from the statement "to ε" of a provisional implication/separation is the smallest possible. More precisely suppose that between xxx and yyy security notions holds provisional implication (separation) xxx $\rightarrow$ yyy to ε (xxx $\not\rightarrow$ yyy to ε), we say that ε is the smallest possible if for any $\alpha < \varepsilon$ the implication (separation) does not hold. We leave the solution of this problem to our future work.

The provided relationships summarized in the Table 2.1 indicate the strength of the *pseudo-random oracle* security notion (Pro), as Pro implies all the other notions except always preimage resistance, always second-preimage resistance and always chosen target forced prefix preimage resistance. However Pro requires a hash function family H to be build from some ideal compression function f (i.e. a random oracle) or a compression function build from an ideal cipher (e.g. by Davies-Meyer construction), what casts a little shadow over Pro's practical use with "real" hash function families, as it is impossible to practically build a random oracle or an ideal cipher. On the other hand, if we prove that a hash function family H build from an ideal compression function f (or a compression function based on a ideal cipher) is good Pro, then the transformation, which transforms the compression function f to the "big" hash function family H, has no structural flaws and therefore one can believe, that if f is replaced by some well-constructed compression function f', then no such flaws will appear in the resulting hash function family H'. Thus we can say, that pseudo random oracle security notion reduces the building of a "good" hash function family to the building of a "good" compression function.

Such transformation, which transforms a "small" compression function $f : \{0,1\}^{y+d} \rightarrow \mathcal{Y}$ to some "big" hash function family $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$ is called *domain extension*, as it extends the domain of the compression function f to the domain of the hash function family H. A domain extension is called *pseudo-random oracle preserving* (shortly Pro-Pr), if it transforms an ideal compression function (or a compression function based on an ideal cipher) to a hash function family, which is secure in Pro sense. Similarly we can define a *collision resistance preserving* domain extension (shortly Coll-Pr), which transforms a collision resistant compression function to a

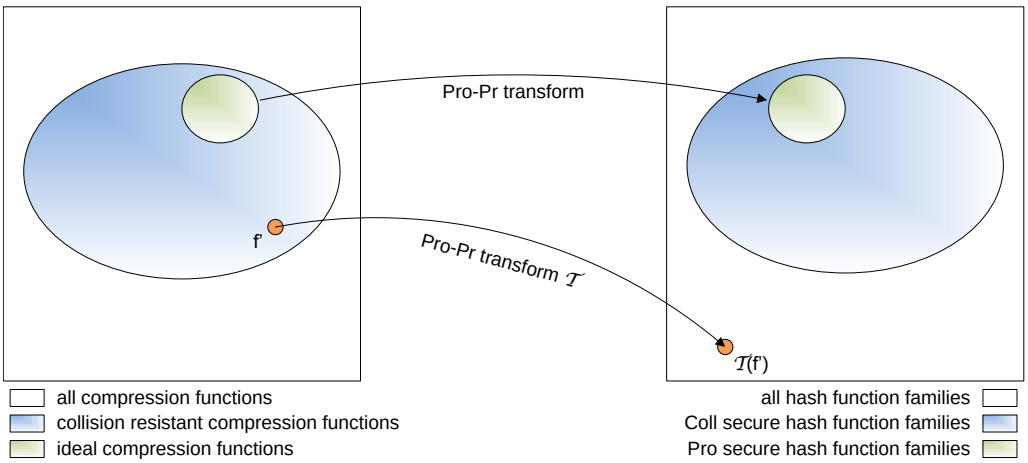

Figure 2.2: Pseudo random random oracle preserving domain extension applied on a non-ideal compression function need not to return Pro secure hash function family. However when applied on an ideal compression function then resulting hash function family is Pro secure, and therefore also Coll secure.

collision resistant hash function family. Bellare and Ristenpart in [3] showed, that Pro-Pr domain extension need not to be Coll-Pr (see Figure 2.2). It means, that they found a domain extension transform $\mathcal{T}(\cdot)$, which is Pro-Pr (i.e. a hash function family $H = \mathcal{T}(f)$ is secure in Pro sense, if f is an ideal compression function), but for some collision resistant compression function f', a hash function family $\mathcal{T}(f')$ is not collision resistant. For that reason they proposed a *multi-property preserving* domain extension, which preserves both pseudo-random oracle and collision resistance security notions, and possibly some others. However our results show, that if a hash function family is secure in Pro sense, then it must be secure in Coll sense, thus a hash function family H produced by some Pro-Pr domain extension transformation $\mathcal{T}(f)$, where f is an ideal compression function, is Coll secure. Therefore the usage of multi-property preserving domain extension transformation is not necessary, instead we can use Pro-Pr domain extension with well chosen compression function.

Summary

In the first part of this Thesis we introduced basic notations and definitions, then we defined twelve notions for hash function security. At the end of the first part we defined the implication and separation between two notions, we also showed some basic properties of such implication/separation and finally we showed the equivalence between one-stage and two-stage versions of some security notions.

In the second part of the Thesis we proved all the relationships among the definitions, except those, which were proven by Rogaway and Shrimpton [13] or by Naor and Reingold [11]. These relationships are summarized in the Table 2.1. Our results indicate that pseudo-random oracle security notion (Pro) is (as expected) the strongest property, as it implies almost all the other security notions (except "always" notions). However Pro has important disadvantage — it requires a hash function family to be build from an ideal compression function (or a compression function based on an ideal cipher). Therefore we rather speak about pseudo-random oracle preserving (Pro-Pr) domain extension transform, which transforms an ideal compression function to a pseudo-random oracle secure hash function family. As Bellare and Ristenpart [3] showed, if a Pro-Pr domain extension transform is applied to a non-ideal compression function f, it can actually "weaken" the resulting hash function family, that is if f is a collision-free compression function, then the resulting hash function family need not to be collision resistant (i.e. Coll secure). Thus we need to choose the compression function for Pro-Pr transforms very carefully. The question is, whether it is possible to build such compression function, which extended by a Pro-Pr domain extension produces Pro secure hash function family. We leave this for our future research.

We note that several Pro-Pr domain extension transforms have been designed already, for example Bellare and Ristenpart in [2] introduced two of them: Strengthened Chain Shift and Enveloped Shoup. They also proved that both of these domain extensions are *multi-property preserving* (MPP), i.e. besides being Pro-Pr, they are also Coll-Pr, MAC-Pr, Prf-Pr and eSec-Pr. Thus MPP transform can guarantee additional security properties (e.g. collision resistance), even if a compression function used with

a particular MPP transform is not "ideal". On the other hand, new hash standard should preserve all mentioned security properties, what can be realized only with a "good" compression function. Therefore we suggest to focus the future research on compression functions and if no suitable compression function will be found, we need to consider the usage of such domain extension transforms.

Bibliography

[1] M. Bellare, R. Canneti, and H. Krawczyk. Keying Hash Functions for Message Authentication. In *Advances in Cryptology – Crypto 96, LNCS vol. 1109*, pages 1–15. Springer, 1996. Available from World Wide Web: `http://www.cs.ucsd.edu/~mihir/papers/kmd5.pdf`.

[2] M. Bellare and T. Ristenpart. Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms. In *International Colloquim on Automata, Languages, and Progamming, LNCS vol. 4596*, pages 399–410. Springer, 2006. Available from World Wide Web: `http://eprint.iacr.org/2007/271`.

[3] M. Bellare and T. Ristenpart. Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In *Advances in Cryptology - ASIACRYPT 2006, LNCS vol. 4284*, pages 299–314. Springer, 2006. Available from World Wide Web: `http://eprint.iacr.org/2006/399`.

[4] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993. Available from World Wide Web: `http://www-cse.ucsd.edu/~mihir/papers/ro.pdf`.

[5] J. Black. The Ideal-Cipher Model, Revisited: An Uninstantiable Blockcipher-Based Hash Function. In *Fast Software Encryption, LNCS vol. 4047*, pages 328–340. Springer, 2006. Available from World Wide Web: `http://eprint.iacr.org/2005/210`.

[6] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *booktitle of the ACM, volume 51, issue 4*, pages 557 – 594. ACM, 2004. Available from World Wide Web: `http://eprint.iacr.org/1998/011.pdf`.

[7] J.S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In *Advances in Cryptology – CRYPTO 2005,*

LNCS vol. 3621, pages 430–448. Springer, 2005. Available from World Wide Web: `http://cs.nyu.edu/~puniya/papers/merkle.pdf`.

[8] J. Kelsey and T. Kohno. Herding Hash Functions and the Nostradamus Attack. In *Advances in Cryptology – EUROCRYPT 2006, LNCS vol. 4004*, pages 183–200. Springer, 2006. Available from World Wide Web: `http://eprint.iacr.org/2005/281.pdf`.

[9] U. Maurer, R. Renner, and C. Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In *Theory of Cryptography, LNCS vol. 2951*, pages 21–39. Springer, 2004. Available from World Wide Web: `ftp://ftp.inf.ethz.ch/pub/crypto/publications/MaReHo04.pdf`.

[10] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[11] M. Naor and O. Reingold. From Unpredictability to Indistinguishability: A Simple Construction of PseudoRandom Functions from MACs. In *Advances in Cryptology – CRYPTO '98, LNCS vol. 1462*, pages 267–281. Springer, 1998. Available from World Wide Web: `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/mac.ps`.

[12] M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Applications. In *21st annual ACM Symposium on Theory of Computing*, pages 33–43. ACM, 1989. Available from World Wide Web: `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/uowhf.ps`.

[13] P. Rogaway and T. Shrimpton. Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In *Fast Software Encryption, LNCS vol. 3017*, pages 371–388. Springer, 2004. Available from World Wide Web: `http://www.inf.unisi.ch/faculty/shrimpton/relates-full.pdf`.

Abstrakt

Kryptografické hešovacie funkcie patria medzi základné kamene súčasnej kryptografie. Inštitút NIST (National Institute for Standards and Technology) nedávno vyhlásil verejnú súťaž, ktorej cieľom je vytvorenie nového hešovacieho štandardu AHS (Advanced Hash Standard). V práci zosumarizujeme základné vlastnosti, ktoré by mal nový hešovací štandard spĺňať, poskytneme ich formálne definície a medzi týmito definíciami dokážeme všetky možné implikácie resp. separácie. Niektoré implikácie/separácie už boli dokázané, niektoré sú nové. Budeme rozlišovat dva typy implikácií resp. separácií - tradičnú a podmienenú. Zatiaľ čo tradičná implikácia (separácia) má význam, aký bežne chápeme pod slovom implikácia (separácia), sila podmienenej implikácie (separácie) závisí na konkrétnej hešovacej funkcii. Ukážeme, že vlastnosť Pseudo-náhodné orákulum, ktorú ako prvý definovali Coron, Dodis, Malinaud a Puniya je (ako sme očakávali) najsilnejšou vlastnosťou, kedže implikuje skoro všetky ostatné vlastnosti. V práci taktiež rozoberáme praktické použitie Pseudo-náhodného orákula a tzv. MPP (Multi Property Transform) transformácií, ktoré poprvý krát navrhli Bellare a Ristenpart.

Kľúčové slová: kryptografická hešovacia funkcia, dokázateľná bezpečnosť, vlastnosti hešovacej funkcie, odolnosť voči kolíziám, pseudo-náhodné orákulum