Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea Triennale in Matematica

On Cryptographic Hash Functions: Definitions, Ideas and Practical Cryptographic Attacks

Relatore:
Prof. Markus Fischer

Laureando:
Filippo Dallatorre (1216909)

Anno Accademico 2022/2023
15 Dicembre

Abstract

Hash functions, and cryptographic hash functions in particular, underpin the technologies on which modern life is built. Efficient databases, the Internet, the World Wide Web, online payments and more, would not be possible without the security and efficiency gains derived from the use of these functions. In this thesis we will introduce the definitions of hash functions and cryptographic hash functions, comment on the ideas that underlie and inspire these definitions and design the probabilistic models in which they become rigorous mathematical concepts. In Chapter 3, we will examine three cryptographic attacks against specific hash functions, calculate the expected cost of execution and estimate the real-world security associated such functions.

Contents

Chapter 1

Introduction

Hash functions are fundamental to the design and implementation of cryptographic systems. Their role and the properties these functions must respect change radically depending on the context in which they are to serve. This explains the loose nature of the following definition.

1.1 Hash Functions

Let $\mathbb{A}$ be a non-empty and finite set of 'letters', called the 'alphabet'.
We call one of its letters the 'null letter', denoting it by $\alpha_0 \in \mathbb{A}$.
Let $\mathbb{M}$ be set of 'messages' of unbounded but finite length, defined as follows:

$$\mathbb{M} \doteq \{ \ (a_i)_{i \in \mathbb{N}} \subset \mathbb{A}^{\mathbb{N}} \mid \exists N \in \mathbb{N}, \ a_i = \alpha_0 \ \forall i \geq N \ \}$$

In this context, any function $h : \mathbb{M} \to \mathbb{A}^n$ where $n \in \mathbb{N}$, is called an **hash function**[1].

The set $\mathbb{A}^n$ is the set of all 'words' of fixed length n that can be written using the alphabet $\mathbb{A}$. It is often called the digest set or hash-digest set, and the image via h of $m \in \mathbb{M}$ is called the digest of m or hash-digest of m or simply, the hash of m.

The constraint on the nature and length is motivated by the context in which these functions are almost exclusively used, i.e. within computer software.

[1]In practice, it is often necessary to ensure that $h(m)$ can be calculated efficiently even on low-powered computers $\forall m \in \mathbb{M}$. However, this property is not required when discussing hash functions purely at a theoretical level.

In fact, a computer only understands data that consists of a finite number of bits (e.g. 16-bit, 32-bit...), which can always be mapped to the corresponding string of 0s and 1s. Therefore, without loss of generality, we shall henceforth consider $\mathbb{A} = \{0, 1\}$ and $\alpha_0 = 0$.

The set $\mathbb{M}$ has infinite countable cardinality. Therefore, in general, hash functions are **not injective**. The definitions of the following section relax the concept of injectivity somewhat, but they must be understood and motivated in an adversarial context, which will be defined and discussed in Chapter 2.

In the future, it will be also useful to consider a family of finite subsets of $\mathbb{M}$, defined as:

$$\begin{aligned}
&\mathbb{M}_N \doteq \{\ (a_i)_{i\in\mathbb{N}} \subset \mathbb{A}^{\mathbb{N}} \mid a_i = 0\ \forall i \geq N\ \},\quad N \in \mathbb{N} \\
&\mathbb{M}_N \equiv \mathbb{A}^N
\end{aligned}$$

1.2 Types of Hash Functions

A hash function $h : \mathbb{M} \rightarrow \{0,1\}^n$ is: [1]

- **pre-image-resistant** if given the hash $k \in \{0,1\}^n$, it is 'computationally infeasible' to determine any message m such that $h(m) = k$.
- **second pre-image-resistant** if given a message m it is 'computationally infeasible' to determine any other message $m^* \neq m$ such that $h(m) = h(m^*)$
- **collision-resistant** if it is 'computationally infeasible' to determine two different messages $m_1 \neq m_2$ such that $h(m_1) = h(m_2)$.
- **cryptographic** or **secure** if it is respects the three proprieties above.

First of all, we note that collision-resistance implies second pre-image-resistance but does not imply pre-image resistance. Indeed, if h is not second pre-image-resistant, then one can find a collision by 'randomly' choosing m_1, and then finding m_2 so that they share the same hash.

Secondly, the impossibility that would result from injectivity of finding $m_1 \neq m_2$ such that $h(m_1) = h(m_2)$ is relaxed by the 'practical impossibility' of finding such pre-images, since it is 'computationally infeasible'. This confirms the notion that the last two properties relax the concept of injectivity. However, the precise mathematical meaning of 'computationally infeasible' in these definitions is not explained, but will be in the following chapter.

Chapter 2

Adversarial Context

In cryptography, an **adversary** (or attacker, rarely opponent, enemy) is a malicious entity whose aim is to prevent the users of a cryptographic systems from achieving their goal (e.g. privacy, integrity and availability of data). These adversaries carry out a series of **attacks**, each of which has a cost. If the attack is a cyber-attack (the only one we will consider), then the cost is computational.

Only in this context, the intuitive notions of 'computationally feasible/infeasible' become rigorous mathematical concepts.

2.1 The Attack

Our discussion must begin by defining the attack that the hypothetical adversary attempts to carry out successfully. It is not possible to make a general discussion that does not a-priori and precisely define the type of attack, since, as we shall see later in examples, some hash functions are resistant to some attacks and vulnerable to others.

The range of attacks that attempt to exploit weaknesses in hash functions is extensive, and discussing each one is outside the scope of this paper. Nonetheless, we outline below the three attacks that correspond to the three properties of hash functions as previously described.

Given the hash function $h : \mathbb{M} \rightarrow \{0,1\}^n$, which, according to **Kerckhoffs' principle** [2], should always be assumed to be known to the adversary, we define three types of attacks:

- **pre-image-attack on** h: The adversary knows a given hash $k \in \{0,1\}^n$ and tries to determine any message m such that $h(m) = k$.
- **second pre-image-attack on** h: The adversary knows a given message m and tries to determine any other message $m^* \neq m$ such that $h(m) = h(m^*)$.
- A **collision-attack on** h: The adversary tries to determine two different messages $m_1 \neq m_2$ such that $h(m_1) = h(m_2)$.

A (particularly inefficient) collision attack on h can be trivially designed from a second pre-image attack on h in the following way: choose 'at random' $m \in \mathbb{M}$, and then perform the second pre-image attack on h, given m.

2.2 Attacker Resources

The second thing we need in our construction is an estimate of the computational resources and time available to a hypothetical adversary.

Let $R \in \mathbb{N}$ be the maximum computational power of the adversary we want to defend against. This is the maximum number of computing cycles per second that the adversary can use in his attack attempt (the unit of measurement can be, for example, GHz/s). Depending on the importance of the cryptographic systems and the type of attack, different values of R can be assumed. The same applies to the maximum attack time, which we will call $T \in \mathbb{N}$. T can be a constant, depending on the cryptographic systems and/or the type of attack, or it can be estimated from economic assessments (as they say, time is money).

The product $R \cdot T$ represents how many computational cycles the attacker can perform to execute a given attack. Intuitively, if the successful execution of the attack requires, 'on average', more resources than $R \cdot T$, then the attack is 'computationally infeasible'. The precise meaning of the last sentence is explained in the next section.

2.3 The Expected Cost of Attack

Given a certain type of attack on h, it is always possible for the attacker to use a trivial algorithm to perform it, i.e. trying 'at random'.
For example, in the case of a pre-image-attack on h, given $k \in \{0,1\}^n$, the attacker can 'randomly' choose a message $m \in \mathbb{M}$. Such an algorithm has a low cost and a non-zero probability of success, meaning a non-zero probability that $h(m) = k$.

Even when considering only deterministic algorithms, the cost associated with such an algorithm can be dependent on the specific value k.

Of course, given the specific attack on h and the attacker's resources, the definition of pre-image resistance must be deterministic and independent of the specific value of k. The hash function h is either pre-image resistant or not, and ideally this should also be independent of the algorithm the attacker chooses to perform.

These considerations inspire the following three models, each dedicated to one of the aforementioned types of attacks.

2.3.1 Probability Model

Let $(\Omega, \mathscr{P}(\Omega), P)$ be a discrete probability space.
Let $X : \Omega \to \mathbb{M}$ be a random variable.

X induces a probability measure on $\mathbb{M}$, which we shall call P_X, defined as:

$$P_X \doteq P \circ X^{-1} \tag{2.1}$$

Let $h : \mathbb{M} \to \{0,1\}^n$, where $n \in \mathbb{N}$, be the hash function under analysis.
The random variable $h(X)$ induces a probability measure on $\{0,1\}^n$, which we shall call P_h, defined as:

$$P_h \doteq P \circ h(X)^{-1} \tag{2.2}$$

2.3.2 Expected Cost of Pre-Image-Attack

Now consider the family $\mathscr{A}$ of all **deterministic algorithms** ϕ that can be used to perform the pre-image-attack on h. Given $k \in \{0,1\}^n$, a computational cost is associated with each algorithm. The following cost function is thus well defined:

$$\begin{aligned} c : \mathscr{A} \times \{0,1\}^n &\longrightarrow \mathbb{N} \cup \{\infty\} \\ (\phi, k) &\longmapsto c(\phi, k) \end{aligned} \tag{2.3}$$

The value $c(\phi, k)$, if finite, represents the number of computations that must be performed using the ϕ algorithm to find any pre-image of k. On the other hand, if this value is ∞, it means that the ϕ algorithm will never be able to find a pre-image of k.

Finally, consider the random variable $c(\phi, h(X)) : \omega \mapsto c(\phi, h(X(\omega)))$.
We can calculate the expected value of $c(\phi, h(X))$, and all we need is to know the probability measure P_h.

$$E_P[\ c(\phi, h(X))\ ] = E_{P_h}[\ c(\phi, \mathbb{I}_{\{0,1\}^n})\ ] \tag{2.4}$$

In which $\mathbb{I}_{\{0,1\}^n}$ represents the identity function on the set $\{0,1\}^n$.

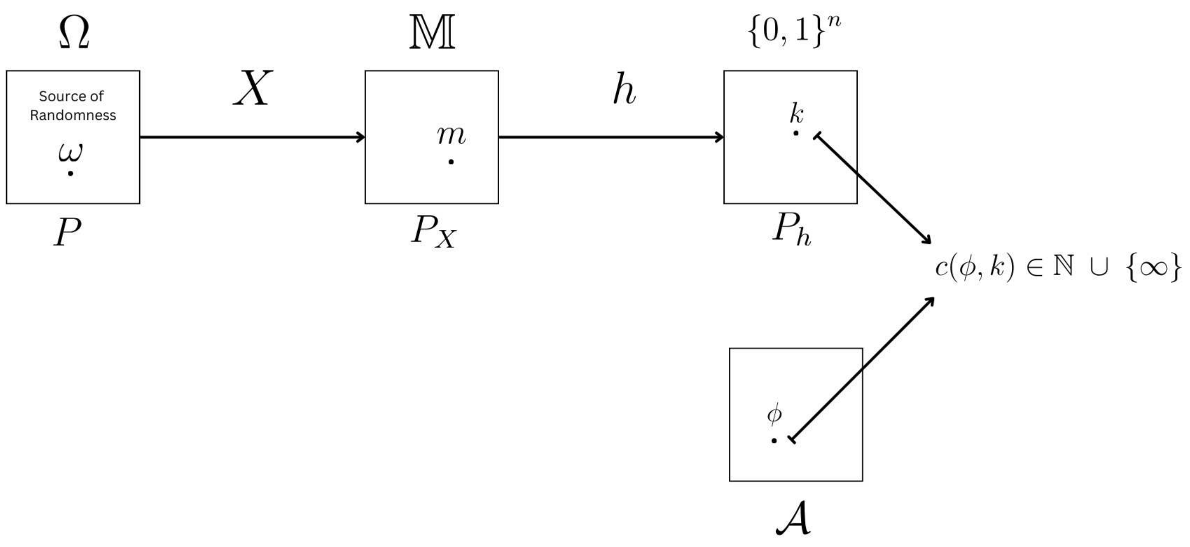

Figure 2.1: Representation of the probability model

2.3.3 Extension to Non-Deterministic Algorithms

So far, we have only examined algorithms of a deterministic nature. This section demonstrates the feasibility of expanding our framework to non-deterministic algorithms. Given the same previous construction, we now define the **source of randomness** of nondeterministic algorithms as follows.

Let $(\tilde{\Omega}, \mathscr{F}, \tilde{P})$ be a probability space.
Let $Y : \tilde{\Omega} \to [0,1] \subset \mathbb{R}$ be a random variable.

Now consider the family $\tilde{\mathscr{A}}$ of all **non-deterministic algorithm** $\tilde{\phi}$ that can be used to perform the pre-image attack on h.
When $\tilde{\phi}$ is evaluated on $y \in [0,1]$, the result is an element $\tilde{\phi}(y) \in \mathscr{A}$ that is, a deterministic algorithm, whose cost can be calculated as shown above.

Now we define the product space:

$$(\Lambda, \mathscr{G}, Q) \doteq (\Omega \times \tilde{\Omega},\ \mathscr{P}(\Omega) \otimes \mathscr{F},\ P \otimes \tilde{P}) \tag{2.5}$$

Consider now the random variable on Λ

$$c(\ \tilde{\phi}(Y), h(X)\ ) : \lambda = (\omega, \tilde{\omega}) \longmapsto c(\ \tilde{\phi}(Y(\tilde{\omega})), h(X(\omega))\ ) \tag{2.6}$$

The expected value of this random variable can be calculated as follows.

$$E_Q[\ c(\ \tilde{\phi}(Y), h(X)\ )\ ] = \int_{\tilde{\Omega}} \left(\sum_{\Omega} c(\tilde{\phi}(Y(\tilde{\omega})), h(X(\omega)) \cdot P(\{\omega\}) \right) \cdot \tilde{P}(d\tilde{\omega}) \tag{2.7}$$

Due to **Tonelli's theorem**, it is possible to swap the order of the integral and the sum, obtaining

$$= \sum_{\Omega} \left(\int_{\tilde{\Omega}} c(\tilde{\phi}(Y(\tilde{\omega})), h(X(\omega))) \cdot \tilde{P}(d\tilde{\omega}) \right) \cdot P(\{\omega\})$$

Thanks to this rewriting, it is easy to see that the following random variable on Ω alone is well defined.

$$\bar{c}(\ \tilde{\phi}, h(X)\ ) \doteq \int_{\tilde{\Omega}} c(\tilde{\phi}(Y(\tilde{w})), h(X)) \cdot \tilde{P}(d\tilde{w}) \tag{2.8}$$

This random variable applied to $\omega \in \Omega$ represents the cost, **averaged over all** $y \in [0, 1]$, of performing $\tilde{\phi}$ with ω as the input.

Here, as in the previous section, it applies that the expected value of $\bar{c}(\tilde{\phi}, h(X))$ can be calculated knowing only the probability measure P_h.

$$E_P[\ \bar{c}(\tilde{\phi}, h(X))\ ] = E_{P_h}[\ \bar{c}(\tilde{\phi}, \mathbb{I}_{\{0,1\}^n})\ ] \tag{2.9}$$

Consequently, in this way we can extend the previous construction to non deterministic algorithms.

Note: From now on, in order not to make the notation too cumbersome, we will use $\mathscr{A}$ to denote the set of all algorithms, **both deterministic and non-deterministic**, and we will use $c(\phi, k)$ to denote the cost of performing the algorithm ϕ with k as input. If the ϕ algorithm is non-deterministic, then $c(\phi, k)$ must be understood as the average cost of ϕ with input k, as discussed in detail in this section.

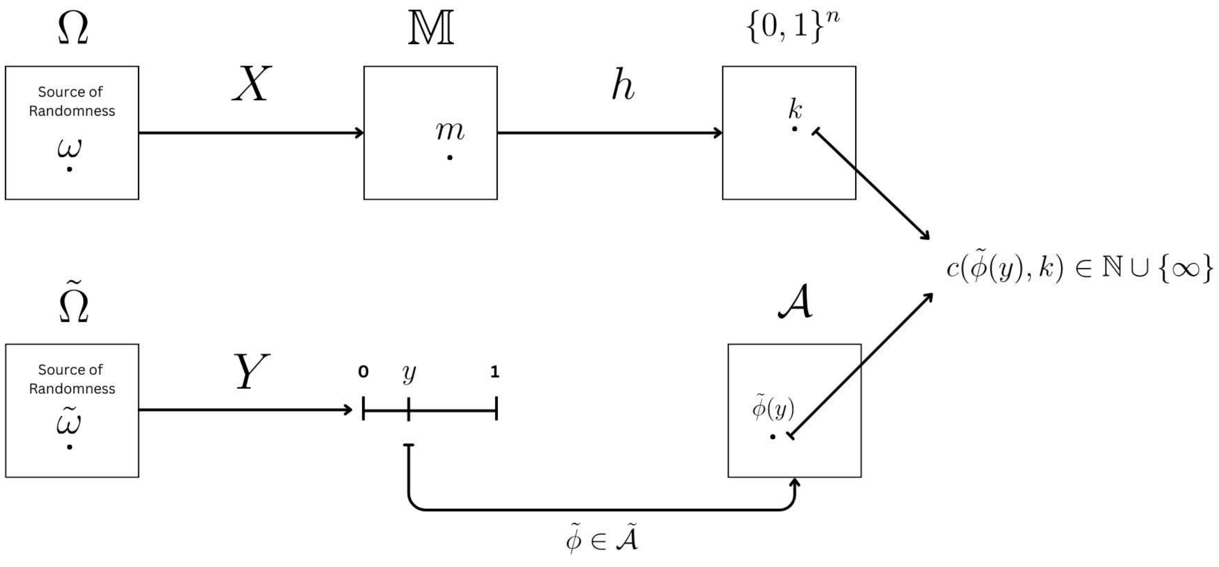

Figure 2.2: Representation of the probability model extended to non deterministic algorithms

Definition: ϕ-infeasible Pre-Image-Attack on h

Given the hash function $h : \mathbb{M} \to \{0,1\}^n$
Let $R \cdot T$ be the resources of the adversary. Let $\phi \in \mathscr{A}$ be the algorithm choose by the adversary for performing the pre-image-attack on h.

We say that the pre-image-attack on h is **ϕ-infeasible** or **ϕ-computationally-infeasible** if

$$E_{P_h}[\ c(\phi, \mathbb{I}_{\{0,1\}^n})\ ] \geq R \cdot T \tag{2.10}$$

To render the definition independent by the choice of ϕ, it suffices for the above inequality to be valid $\forall \phi \in \mathscr{A}$. This is also equivalent to requesting the following.

Definition: infeasible Pre-Image-Attack on h

Given the hash function $h : \mathbb{M} \to \{0,1\}^n$
Let $R \cdot T$ be the resources of the adversary.

We say that the pre-image-attack on h is **infeasible** or **computationally infeasible**, or equivalently that h **is pre-image-resistant** if

$$\inf_{\phi \in \mathscr{A}} E_{P_h}[\ c(\phi, \mathbb{I}_{\{0,1\}^n})\ ] \geq R \cdot T \tag{2.11}$$

2.3.4 Expected Cost of Second Pre-Image-Attack

Consider the family $\mathscr{A}'$ of all algorithms ψ that can be used to perform the second pre-image-attack on h. Given $m \in \mathbb{M}$, a computational cost is associated with each algorithm. The following function is thus well defined:

$$\begin{aligned} c' : \mathscr{A}' \times \mathbb{M} &\longrightarrow \mathbb{N} \cup \{\infty\} \\ (\psi, m) &\longmapsto c'(\psi, m) \end{aligned} \tag{2.12}$$

The value $c'(\psi, m)$, if finite, represents the number of computations that must be performed using the ψ algorithm to find any $m^* \neq m$ such that $h(m^*) = h(m)$. On the other hand, if this value is ∞, it means that the ψ algorithm will never be able to find such m^*.

Note: It is possible to construct an element $\phi' \in \mathscr{A}'$ based on $\phi \in \mathscr{A}$ (an algorithm for performing the pre-image attack on h) in the following way:

Definition of ϕ'
INPUT: $m \in \mathbb{M}$
OUTPUT: $m^* \in \mathbb{M}$
EXECUTION:

1. computing $h(m)$
2. executing ϕ, with $h(m)$ as the input

If $C_{h,m}$ is the cost of computing $h(m)$, then for ϕ and ϕ' considered as above, we have the following inequality.

$$c'(\phi', m) \geq c(\phi, h(m)) + C_{h,m} \tag{2.13}$$

If the OUTPUT m^* of ϕ' is equal to m, then $c'(\phi', m) = \infty$.
The inequality is strict, only when $c'(\phi', m) = \infty$ and $c(\phi, h(m)) < \infty$. In all other cases, it is an equality.

Finally, consider the random variable $c'(\psi, X) : \omega \mapsto c'(\psi, X(\omega))$.
We can calculate the expected value of $c'(\psi, X)$, but this time the knowledge of the probability measure P_X is needed.

$$E_P[\ c'(\psi, X)\ ] = E_{P_X}[\ c'(\psi, \mathbb{I}_{\mathbb{M}})\ ] \tag{2.14}$$

Definition: ψ-infeasible Second Pre-Image-Attack on h

Given the hash function $h : \mathbb{M} \to \{0,1\}^n$
Let $R' \cdot T'$ be the resources of the adversary. Let $\psi \in \mathscr{A}'$ be the algorithm choose by the adversary for performing the second pre-image-attack on h.

We say that the second pre-image-attack on h is **ψ-infeasible** or **ψ-computationally-infeasible** if

$$E_{P_X}[\ c'(\psi, \mathbb{I}_{\mathbb{M}})\ ] \geq R' \cdot T' \tag{2.15}$$

To render the definition independent by the choice of ψ, it suffices for the above inequality to be valid $\forall \psi \in \mathscr{A}'$. This is also equivalent to requesting the following.

Definition: infeasible Second Pre-Image-Attack on h

Given the hash function $h : \mathbb{M} \to \{0,1\}^n$
Let $R' \cdot T'$ be the resources of the adversary.

We say that the second pre-image-attack on h is **infeasible** or **computationally infeasible**, or equivalently that h **is second pre-image-resistant** if

$$\inf_{\psi \in \mathscr{A}'} E_{P_X}[\ c'(\psi, \mathbb{I}_{\mathbb{M}})\ ] \geq R' \cdot T' \tag{2.16}$$

It is crucial to highlight that the dependence on the function h here is hidden, but present in the composition of the set $\mathscr{A}'$.

Additionally, in practice it is never possible to know the probability measure P_X, but it is possible to estimate $\inf_{\psi \in \mathscr{A}'} E_{P_X}[\ c'(\psi, \mathbb{I}_{\mathbb{M}})\ ]$ knowing only the distribution P_h in the following way.

$$\inf_{\psi \in \mathscr{A}'} E_{P_X}[\ c'(\psi, \mathbb{I}_{\mathbb{M}})\ ] \ \leq \ \inf_{\phi' | \phi \in \mathscr{A}} E_{P_X}[\ c'(\phi', \mathbb{I}_{\mathbb{M}})\ ] \tag{2.17}$$

This is true because, as shown, $\{\ \phi' \mid \phi \in \mathscr{A}\} \subseteq \mathscr{A}'$. Applying the definition of expected value we get

$$\inf_{\phi' | \phi \in \mathscr{A}} E_{P_X}[\ c'(\phi', \mathbb{I}_{\mathbb{M}})\ ] = \inf_{\phi' | \phi \in \mathscr{A}} \sum_{m \in \mathbb{M}} c'(\phi', m) \cdot P_X(\{m\}) \tag{2.18}$$

Here we add an **additional** but in practice very reasonable assumption about the hash function h and probability measure P_X. Given the given the family of subsets

$$\mathbb{M}_{h,N} \doteq \{m \in \mathbb{M}_N \mid \exists m^* \neq m,\ h(m) = h(m^*)\} \tag{2.19}$$

We request that[1]

$$\exists N \in \mathbb{N} \mid P_X(\mathbb{M}_{h,N}) = 1 \tag{2.20}$$

Thanks to this assumption, it is now possible to apply the inequality (2.13) as an equality. This is legitimate because there exist a ϕ' algorithm whose cost is less than infinity for every message $m \in \mathbb{M}$ (for example, the bruteforce algorithm described in 3.1).

$$\begin{aligned}\inf_{\psi\in\mathscr{A}'} E_{P_X}[\ c'(\psi,\mathbb{I}_\mathbb{M})\ ] &\leq \inf_{\phi'|\phi\in\mathscr{A}} \sum_{\mathbb{M}_{h,N}} c'(\phi',m)\cdot P_X(\{m\}) \\ &= \inf_{\phi\in\mathscr{A}} \sum_{\mathbb{M}_{h,N}} c(\phi,h(m))\cdot P_X(\{m\}) + \sum_{\mathbb{M}_{h,N}} C_{h,m}\cdot P_X(\{m\})\end{aligned}$$

Due to the assumption on the support of P_X, $C_h \doteq \sum_{\mathbb{M}_{h,N}} C_{h,m}\cdot P_X(\{m\})$ is finite. Considering now the partition induced by h on the set $\mathbb{M}_{h,N}$ we have

$$\begin{aligned}\inf_{\psi\in\mathscr{A}'} E_{P_X}[\ c'(\psi,\mathbb{I}_\mathbb{M})\ ] &\leq \inf_{\phi\in\mathscr{A}} \sum_{k\in\{0,1\}^n} \sum_{m\in h^{-1}(k)} c(\phi,h(m))\cdot P_X(\{m\})\ + C_h \\ &= \inf_{\phi\in\mathscr{A}} \sum_{k\in\{0,1\}^n} \sum_{m\in h^{-1}(k)} c(\phi,k)\cdot P_X(\{m\})\ + C_h \\ &= \inf_{\phi\in\mathscr{A}} \sum_{k\in\{0,1\}^n} c(\phi,k)\cdot \sum_{m\in h^{-1}(k)} P_X(\{m\})\ + C_h \\ &= \inf_{\phi\in\mathscr{A}} \sum_{k\in\{0,1\}^n} c(\phi,k)\cdot P_h(\{k\})\ + C_h \\ &= \inf_{\phi\in\mathscr{A}} E_{P_h}[\ c(\phi,\mathbb{I}_{\{0,1\}^n})\ ] + C_h\end{aligned}$$

To recapitulate, we have thus shown that the *infimum* of expected costs for a second pre-image-attack is no greater than the *infimum* of expected costs for a pre-image-attack, plus a constant.

$$\inf_{\psi\in\mathscr{A}'} E_{P_X}[\ c'(\psi,\mathbb{I}_\mathbb{M})\ ]\ \leq\ \inf_{\phi\in\mathscr{A}} E_{P_h}[\ c(\phi,\mathbb{I}_{\{0,1\}^n})\ ] + C_h \tag{2.21}$$

[1]For practical applications, consider $N \gg n$. For instance, $n = 256,\ N = 10^{10}$

2.3.5 Expected Cost of Collision-Attack

Consider the family $\mathscr{A}''$ of all algorithms χ that can be used to perform the collision-attack on h. This time, no additional information (e.g. $m \in \mathbb{M}$ as for the second pre-image-attack) is required.
The following cost function is thus well defined:

$$\begin{aligned} c'' : \mathscr{A}'' &\longrightarrow \mathbb{N} \cup \{\infty\} \\ \chi &\longmapsto c''(\chi) \end{aligned}$$

The value $c''(\chi)$, if finite, represents the number of computations that must be performed using the χ algorithm to find a collision, i.e. to find $m_1 \neq m_2$ such that $h(m_1) = h(m_2)$. On the other hand, if this value is ∞, it means that the χ algorithm will never be able to find a collision.

Note: It is possible to construct an element $\psi' \in \mathscr{A}''$ based on $\psi \in \mathscr{A}'$ (an algorithm for performing the second pre-image attack on h), in the following way:

Definition of ψ'
INPUT: *none*
OUTPUT: $(m_1, m_2) \in \mathbb{M}^2$
EXECUTION:

1. randomly choosing m_1, according to P_X
2. executing ψ, with m_1 as the input

Here we consider the cost of randomly choosing m_1 to be zero. For ψ and ψ' considered as above, we have the following equality.

$$c''(\psi') = E_{P_X}[\ c'(\psi, \mathbb{I}_{\mathbb{M}})\ ] \tag{2.22}$$

Definition: χ-infeasible Collision-Attack on h

Given the hash function $h : \mathbb{M} \to \{0,1\}^n$
Let $R'' \cdot T''$ be the resources of the adversary. Let $\chi \in \mathscr{A}''$ be the algorithm choose by the adversary for performing the collision-attack on h.

We say that the collision-attack on h is **χ-infeasible** or **χ-computationally-infeasible** if

$$c''(\chi) \geq R'' \cdot T'' \tag{2.23}$$

To render the definition independent by the choice of χ, it suffices for the above inequality to be valid $\forall \chi \in \mathscr{A}''$. This is also equivalent to requesting the following.

Definition: infeasible Collision-Attack on h

Given the hash function $h : \mathbb{M} \to \{0,1\}^n$
Let $R'' \cdot T''$ be the resources of the adversary.

We say that the collision-attack on h is **infeasible** or **computationally infeasible**, or equivalently that h **is collision-resistant** if

$$\inf_{\chi \in \mathscr{A}''} c''(\chi) \geq R'' \cdot T'' \tag{2.24}$$

Firstly, we note that the dependence on the function h here is hidden, but present in the composition of the set $\mathscr{A}''$. Secondly, we note that it is possible, using the argument in (2.22), to prove that

$$\inf_{\chi \in \mathscr{A}''} c''(\chi) \leq \inf_{\psi' | \psi \in \mathscr{A}'} c''(\psi) = \inf_{\psi \in \mathscr{A}'} E_{P_X}[\, c'(\psi, \mathbb{I}_{\mathbb{M}}) \,] \tag{2.25}$$

This result is not surprising, since we had already shown that collision-resistance implies second pre-image-resistance in Chapter 1.

We can therefore summarise the relationship between pre-image-resistance, second pre-image-resistance and collision-resistance in the following inequality.

$$\inf_{\chi \in \mathscr{A}''} c''(\chi) \leq \inf_{\psi \in \mathscr{A}'} E_{P_X}[\ c'(\psi, m)\ ] \leq \inf_{\phi \in \mathscr{A}} E_{P_h}[\ c(\phi, k)\ ] + C_h \tag{2.26}$$

Recall that for the second inequality to hold, we must make an assumption on the support of P_X as seen in (2.20)

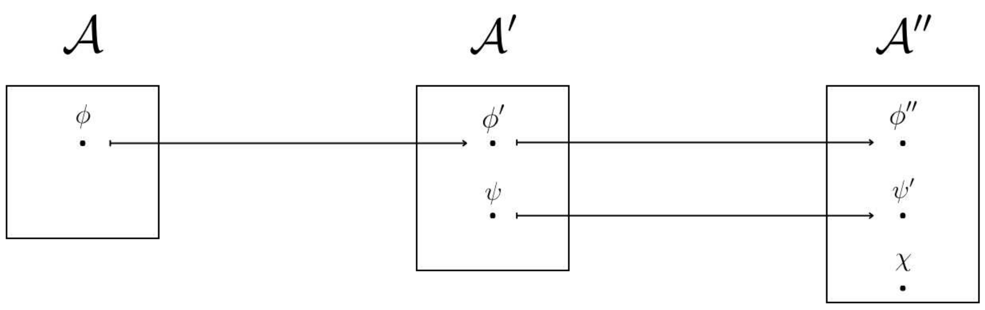

Figure 2.3: Representation of the sets of algorithms and transformations of algorithms

Chapter 3

Examples of Cryptographic Hash Functions and Attacks

Using the theoretical models from the previous chapter, in this chapter we will go on to specify the hash function h and make assumptions about the probability distribution P_h; this will allow us to calculate the expected computational cost for certain algorithms used to perform the three types of attacks discussed earlier.

Finally, we will evaluate what real-world security guarantees the expected cost of the attack.

3.1 Bruteforce Pre-Image-Attack on SHA1

In this example, the hash function under analysis is SHA1, a function designed by the U.S. National Security Agency in 1995, replacing its predecessor SHA0, introduced only two years earlier. [3] [4]

SHA1 is currently a U.S. Federal Information Processing Standard (FIPS). [5]
In the notation used so far, SHA1 is denoted by the function h:

$$h : \mathbb{M} \to \{0,1\}^{160}$$

For those more familiar with coding, we include the pseudocode of SHA1 below. However, knowledge of it is not necessary for our discussion.

SHA-1 pseudocode

Pseudocode for the SHA-1 algorithm follows:

```
Note 1: All variables are unsigned 32-bit quantities and wrap modulo 2^32 when calculating, except for
        ml, the message length, which is a 64-bit quantity, and
        hh, the message digest, which is a 160-bit quantity.
Note 2: All constants in this pseudo code are in big endian.
        Within each word, the most significant byte is stored in the leftmost byte position

Initialize variables:

h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0

ml = message length in bits (always a multiple of the number of bits in a character).

Pre-processing:
append the bit '1' to the message e.g. by adding 0x80 if message length is a multiple of 8 bits.
append 0 ≤ k < 512 bits '0', such that the resulting message length in bits
   is congruent to −64 ≡ 448 (mod 512)
append ml, the original message length in bits, as a 64-bit big-endian integer.
   Thus, the total length is a multiple of 512 bits.

Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w[i], 0 ≤ i ≤ 15

    Message schedule: extend the sixteen 32-bit words into eighty 32-bit words:
    for i from 16 to 79
        Note 3: SHA-0 differs by not having this leftrotate.
        w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1

    Initialize hash value for this chunk:
    a = h0
    b = h1
    c = h2
    d = h3
    e = h4

    Main loop:
    for i from 0 to 79
        if 0 ≤ i ≤ 19 then
            f = (b and c) or ((not b) and d)
            k = 0x5A827999
        else if 20 ≤ i ≤ 39
            f = b xor c xor d
            k = 0x6ED9EBA1
        else if 40 ≤ i ≤ 59
            f = (b and c) or (b and d) or (c and d)
            k = 0x8F1BBCDC
        else if 60 ≤ i ≤ 79
            f = b xor c xor d
            k = 0xCA62C1D6

        temp = (a leftrotate 5) + f + e + k + w[i]
        e = d
        d = c
        c = b leftrotate 30
        b = a
        a = temp

    Add this chunk's hash to result so far:
    h0 = h0 + a
    h1 = h1 + b
    h2 = h2 + c
    h3 = h3 + d
    h4 = h4 + e

Produce the final hash value (big-endian) as a 160-bit number:
hh = (h0 leftshift 128) or (h1 leftshift 96) or (h2 leftshift 64) or (h3 leftshift 32) or h4
```

The algorithm used by the hypothetical adversary in this example is the non-deterministic bruteforce algorithm $\tilde{\phi}_{brute} \in \tilde{\mathscr{A}}$, described below:

Definition of $\tilde{\phi}_{brute}$ (bruteforce)
INPUT: $k \in \{0,1\}^{160}$
OUTPUT: $m \in \mathbb{M}$
EXECUTION:

1. randomly choosing m, according to P_X
2. computing $h(m)$
3. $h(m) = k$?
   (a) if YES, RETURN m
   (b) if NO, go back to point 1.

The cost of running a **single cycle** of the algorithm ϕ is approximated by the quantity $C_{h,m}$, which is the cost of computing $h(m)$. In all generality this cost depends on $m \in \mathbb{M}$.

Assumptions

In order to calculate the expected cost of the ϕ algorithm, we make two assumptions about the distributions P_X and P_h, respectively. The first, concerns the support of the probability measure P_X.

$$\exists N \in \mathbb{N} \mid P_X(\mathbb{M}_N) = 1 \tag{3.1}$$

In the SHA1 definition, the maximum message length is indeed limited to $N = 2^{64} - 1$. This limit is not theoretical but practical in nature, which is why we still decided to present it as an assumption.

In this context, the average cost over all messages of running a single cycle of the algorithm ϕ called $C_h \doteq \sum_{\mathbb{M}_N} C_{h,m} \cdot P_X(\{m\})$ is finite.

Secondly, we assume that the distribution P_h is the Uniform distribution over the set $\{0,1\}^{160}$.

$$P_h \sim \mathscr{U}(\{0,1\}^{160}) \tag{3.2}$$

3.1.1 Expected Cost

Since the $\tilde{\phi}_{brute}$ algorithm is non-deterministic, we must use the framework introduced in 2.3.3. considering the product probability space:

$$(\Lambda, \mathscr{G}, Q) \doteq (\Omega \times \tilde{\Omega},\ \mathscr{P}(\Omega) \otimes \mathscr{F},\ P \otimes \tilde{P})$$

We model the message choices made by the $\tilde{\phi}_{brute}$ algorithm in Step 1. as the the sequence $Y = (Y_1,\ Y_2,\ Y_3, \cdots)$ of random variables on $\tilde{\Omega}$, i.i.d. with common distribution P_X.

We use the random variable $h(X)$ on Ω with distribution P_h to represent the choice of $k \in \{0,1\}^{160}$.

Finally we assume that the family $h(X),\ h(Y_1),\ h(Y_2),\ h(Y_3), \cdots$ to be i.i.d. with common distribution $P_h \sim \mathscr{U}(\{0,1\}^{160})$.

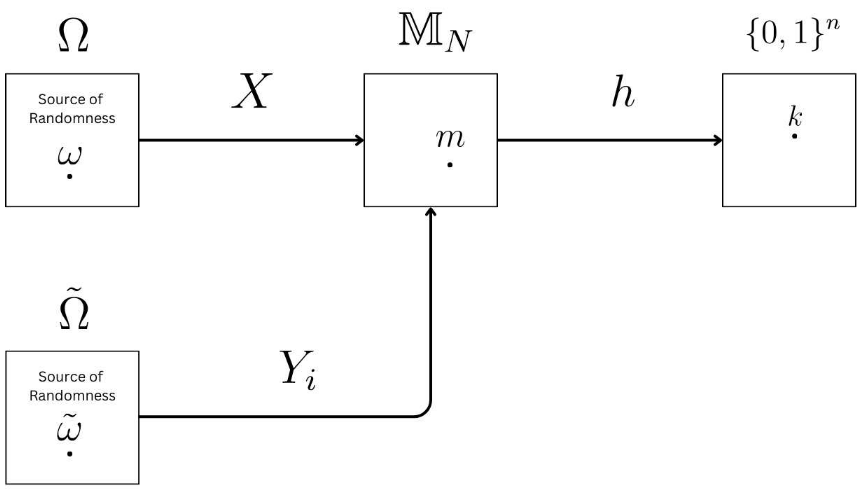

Figure 3.1: Representation of the probability model for the bruteforce pre-image-attack

We now define the random variable S on $\Omega \times \tilde{\Omega}$

$$\begin{aligned} S &\doteq \min\{\ n \in \mathbb{N} \mid h(Y_n) = h(X)\ \} \\ S(\omega, \tilde{\omega}) &= \min\{\ n \in \mathbb{N} \mid h(Y_n(\tilde{w})) = h(X(\omega))\ \} \end{aligned} \tag{3.3}$$

S is the random variable representing the **number of cycles** that the $\tilde{\phi}_{brute}$ algorithm must perform before finding $m \in \mathbb{M}$ such that $h(m) = k$.
We will now prove that S follows a **Geometric distribution** of parameter $p \doteq 2^{-160}$.

$$
\begin{aligned}
&Q(S = n+1) = \\
&\qquad = Q(\min\{\ m \in \mathbb{N} \mid h(Y_m) = h(X)\ \} = n+1) \\
&\qquad = Q(h(Y_i) \neq h(X)\ \forall i \leq n,\ h(Y_{n+1}) = h(X)) \\
&\qquad = \sum_{k \in \{0,1\}^{160}} Q(h(Y_i) \neq k\ \forall i \leq n,\ h(Y_{n+1}) = k \mid h(X) = k) \cdot Q(\{h(X) = k\} \times \tilde{\Omega}) \\
&\qquad = \sum_{k \in \{0,1\}^{160}} Q(\ h(Y_i) \neq k\ \forall i \leq n,\ h(Y_{n+1}) = k \mid h(X) = k) \cdot P(h(X) = k)
\end{aligned}
$$

In the steps above we have we conditioned on the event $\{\omega \in \Omega \mid h(X(\omega)) = k\}$ and used the properties of the product probability $Q = P \otimes \tilde{P}$.

We now use the assumptions of independence and uniform distributions, which implies that $P(h(X) = k) = \tilde{P}(h(Y_i) = k) = p\ \forall i\ \forall k$.

$$
\begin{aligned}
&= \sum_{k \in \{0,1\}^{160}} Q(\ h(Y_i) \neq k\ \forall i \leq n,\ h(Y_{n+1}) = k \mid h(X) = k) \cdot P(h(X) = k) \\
&= \sum_{k \in \{0,1\}^{160}} Q(\ \Omega \times \{h(Y_i) \neq k\ \forall i \leq n,\ h(Y_{n+1}) = k\}) \cdot P(h(X) = k) \\
&= \sum_{k \in \{0,1\}^{160}} \tilde{P}(\ h(Y_i) \neq k\ \forall i \leq n,\ h(Y_{n+1}) = k) \cdot P(h(X) = k) \qquad (3.4) \\
&= \sum_{k \in \{0,1\}^{160}} (1-p)^n \cdot p \cdot p \\
&= (1-p)^n \cdot p
\end{aligned}
$$

Hence, $S \sim \mathscr{G}(p)$. Thanks to the assumptions of independence and uniform distributions, and going over the same calculations, it is easy to see that S and $h(X)$ are **independent**.

Running one cycle of the algorithm has a message-dependent cost $C_{h,m}$[1]. The algorithm performs a number $S(\omega, \tilde{\omega})$ of cycles in which it calculates the hash of

[1] as we will see later, the SHA1 algorithm is based on the M-D construction 3.3.1, which means that the cost $C_{h,m}$ is basically linear in the length of the message m.

the messages $Y_1(\tilde{\omega}),\ Y_2(\tilde{\omega}),\ \cdots Y_{S(\omega,\tilde{\omega})}(\tilde{\omega})$. Therefore we write:

$$c(\tilde{\phi}_{brute}(Y(\tilde{\omega})), h(X(\omega))) = \sum_{i=1}^{S(\omega,\tilde{\omega})} C_{h,Y_i(\tilde{\omega})} \tag{3.5}$$

Finally, we are ready to calculate the expected cost to perform $\tilde{\phi}_{brute}$.

$$\begin{aligned} E_Q[\ c(\tilde{\phi}_{brute}(Y), h(X))\ ] &= E_Q[\ \sum_{i=1}^{S} C_{h,Y_i}\ ] \\ &= \sum_{\Omega}\Big(\int_{\tilde{\Omega}} \sum_{i=1}^{S(\omega,\tilde{\omega})} C_{h,Y_i(\tilde{\omega})} \cdot \tilde{P}(d\tilde{\omega})\ \Big) \cdot P(\{\omega\}) \\ &= \int_{\tilde{\Omega}}\Big(\sum_{\Omega} \sum_{i=1}^{S(\omega,\tilde{\omega})} C_{h,Y_i(\tilde{\omega})} \cdot P(\{\omega\})\ \Big) \cdot \tilde{P}(d\tilde{\omega}) \\ &= \int_{\tilde{\Omega}}\Big(\sum_{\Omega} \sum_{i=1}^{S(\omega,\tilde{\omega})} C_{h,Y_i(\tilde{\omega})} \cdot Q(\{\omega\}\times\tilde{\Omega})\ \Big) \cdot \tilde{P}(d\tilde{\omega}) \\ &= \int_{\tilde{\Omega}}\Big(\sum_{\Omega} \sum_{n=1}^{\infty} \sum_{i=1}^{n} C_{h,Y_i(\tilde{\omega})} \cdot Q(\{\omega\}\times\tilde{\Omega} \mid S=n) \cdot Q(S=n)\ \Big) \cdot \tilde{P}(d\tilde{\omega}) \end{aligned} \tag{3.6}$$

In the last step we conditioned with respect to the event $\{S = n\}$. Before proceeding, let us now simplify the following expression

$$\begin{aligned} Q(\{\omega\}\times\tilde{\Omega} \mid S=n) &= \frac{Q(\{\omega\}\times\tilde{\Omega}\ ,\ S=n)}{Q(S=n)} \\ &= \frac{Q(\{\omega\}\times\{\tilde{\omega} \mid S(\omega,\tilde{\omega})=n\})}{(1-p)^{n-1}\cdot p} \\ &= \frac{P(\{\omega\})\cdot\tilde{P}(\{\tilde{\omega} \mid S(\omega,\tilde{\omega})=n\})}{(1-p)^{n-1}\cdot p} \\ &= P(\{w\}) \end{aligned} \tag{3.7}$$

The last step is true because, for every ω, the function $\tilde{\Omega} \ni \tilde{\omega} \mapsto S(\omega,\tilde{\omega})$ follows a Geometric distribution, just like S.

By applying this result to the calculation of the expected value we get.

$$\begin{aligned}
E_Q[\ c(\tilde{\phi}_{brute}(Y), h(X)\ ] &= \int_{\tilde{\Omega}} (\sum_{\Omega} \sum_{n=1}^{\infty} \sum_{i=1}^{n} C_{h,Y_i(\tilde{\omega})} \cdot P(\{\omega\}) \cdot Q(S=n)) \cdot \tilde{P}(d\tilde{\omega}) \\
&= \int_{\tilde{\Omega}} (\sum_{n=1}^{\infty} \sum_{i=1}^{n} C_{h,Y_i(\tilde{\omega})} \cdot Q(S=n)) \cdot \tilde{P}(d\tilde{\omega}) \\
&= \sum_{n=1}^{\infty} \sum_{i=1}^{n} (\int_{\tilde{\Omega}} C_{h,Y_i(\tilde{\omega})} \cdot \tilde{P}(d\tilde{\omega})) \cdot Q(S=n) \\
&= \sum_{n=1}^{\infty} \sum_{i=1}^{n} (\sum_{\mathbb{M}_N} C_{h,m} \cdot P_X(m)) \cdot Q(S=n) \\
&= \sum_{n=1}^{\infty} \sum_{i=1}^{n} C_h \cdot Q(S=n) \\
&= C_h \cdot \sum_{n=1}^{\infty} n \cdot Q(S=n) \\
&= C_h \cdot E_Q[\ S\ ] \\
&= C_h \cdot 2^{160}
\end{aligned} \tag{3.8}$$

3.1.2 Considerations

To get an idea of the size of the number 2^{160}, take the Folding@home project as an example. Folding@home (FAH or F@h) is a distributed computing project that aims to help scientists develop new therapies for a range of diseases by simulating protein dynamics. As interest in the project increased following the COVID-19 pandemic, the system reached a speed of about 2.43 exaflops on April 12, 2020, becoming the world's first exaflop computing system. [6].

The number $2^{160} \simeq 10^{48}$ is 30 orders of magnitude bigger than the record achieved by Folding@home. For an adversary operating the computational power of F@h at its maximum, it would take (on average) more than 10^{22} years to find a pre-image. Therefore with our assumptions, we say that the **pre-image-attack on SHA1 is ϕ-computationally-infeasible**.

To date, there is no known algorithm that significantly improves this attack. Thus, SHA1 is still considered pre-image-resistant.

3.2 Bruteforce Collision-Attack on SHA1

We will now examine a collision-attack on SHA1 using the $\tilde{\chi}_{brute} \in \tilde{\mathscr{A}}''$ algorithm with the same assumptions (3.1) and (3.2).

Definition of $\tilde{\chi}_{brute}$ (bruteforce)
INPUT: $\mathscr{L}_h = \emptyset$
OUTPUT: $(m_1, m_2) \in \mathbb{M}^2$
EXECUTION:

```
1. randomly choosing $m$, according to $P_X$
2. computing $h(m)$
3. $h(m) = h(m^*) \in \mathscr{L}_h$ ?
    (a) if YES, RETURN $(m, m^*)$
    (b) if NO,
        i. APPEND $h(m)$ to $\mathscr{L}_h$
        ii. Go back to point 1.
```

Again, we can approximate the cost of executing a **single cycle** of the $\tilde{\chi}_{brute}$ algorithm by the quantity $C_{h,m}$, which is the cost of computing $h(m)$.
In all generality, this cost depends on $m \in \mathbb{M}$, however, as before thanks to the first assumption at (3.1) we have that the average cost over all messages $C_h = \sum_{\mathbb{M}_N} C_{h,m} \cdot P_X(\{m\})$ is finite.

3.2.1 Expected Cost

We model the message choices made by the $\tilde{\chi}_{brute}$ algorithm in Step 1. as the the sequence $Y = (Y_1,\ Y_2,\ Y_3, \cdots)$ of random variables on $\tilde{\Omega}$, i.i.d. with common distribution P_X.

The random variables $h(Y_1),\ h(Y_2),\ h(Y_3), \cdots$ are therefore i.i.d. with common distribution $P_h \sim \mathscr{U}(\{0,1\}^{160})$.

We now define the random variable T on $\tilde{\Omega}$ as

$$T \doteq \min\{\ n \in \mathbb{N} \mid h(Y_n) \in \{\ h(Y_1),\ h(Y_2), \cdots h(Y_{n-1})\ \}\ \} \tag{3.9}$$

T is the random variable representing the **number of cycles** that the $\tilde{\chi}_{brute}$ algorithm must perform before finding a collision.
Calling $M \doteq 2^{160}$, we note that $2 \leq T \leq M + 1$.

We will now proceed to calculate the distribution of T, which we note is an alternative formulation of the **birthday problem**. For $2 \leq n+1 \leq M+1$ we have:

$$\begin{aligned} \tilde{P}(T = n+1) &= \tilde{P}(\ h(Y_i) \neq h(Y_j)\ \forall i \neq j \leq n,\ \exists l \colon h(Y_{n+1}) = h(Y_l)) \\ &= n \cdot \tilde{P}(\ h(Y_i) \neq h(Y_j)\ \forall i \neq j \leq n,\ h(Y_{n+1}) = h(Y_1)) \end{aligned} \tag{3.10}$$

Exploiting again the fact that the random variables $h(Y_i)$ are i.i.d. we get

$$\begin{aligned} &= n \cdot \sum_k \tilde{P}(\ h(Y_i) \neq h(Y_j)\ \forall i \neq j \leq n,\ h(Y_{n+1}) = k \mid h(Y_1) = k) \cdot \tilde{P}(h(Y_1) = k) \\ &= n \cdot \sum_k \tilde{P}(\ h(Y_i) \neq h(Y_j) \neq k\ \forall\ 2 \leq i \neq j \leq n) \cdot \tilde{P}(h(Y_{n+1}) = k) \cdot \tilde{P}(h(Y_1) = k) \\ &= \frac{n}{M^2} \cdot \sum_k \tilde{P}(\ h(Y_i) \neq h(Y_j) \neq k\ \forall\ 2 \leq i \neq j \leq n) \end{aligned} \tag{3.11}$$

To calculate $\tilde{P}(\ h(Y_i) \neq h(Y_j) \neq k\ \forall\ 2 \leq i \neq j \leq n)$ we simply count the possible cases and divide them by the total cases, as all distributions are uniform.
There is only one possibility for the value $h(Y_1)$ since it's fixed ($= k$), for $h(Y_2)$ there are $M-1$ possibilities, for $h(Y_3)$ there are $M-2$ possibilities and so on.

$$P(\ h(X_i) \neq h(X_j) \neq k\ \forall\ 2 \leq i \neq j \leq n) = \frac{1 \cdot (M-1) \cdot (M-2) \cdots (M-n+1)}{1 \cdot M^{n-1}} \tag{3.12}$$

Finally, we can calculate the distribution of T:

$$\begin{aligned} \tilde{P}(T = n+1) &= \frac{n}{M^2} \sum_k \frac{(M-1) \cdot (M-2) \cdots (M-n+1)}{M^{n-1}} \\ &= n \cdot \frac{(M-1)!}{(M-n)!\ \cdot\ M^n} = n \cdot \frac{M!}{(M-n)!\ \cdot\ M^{n+1}} \end{aligned} \tag{3.13}$$

Running one cycle of the algorithm has a message-dependent cost $C_{h,m}$. The algorithm performs a number $T(\tilde{\omega})$ of cycles in which it calculates the hash of the messages $Y_1(\tilde{\omega}),\ Y_2(\tilde{\omega}),\ \cdots\ Y_{T(\tilde{\omega})}(\tilde{\omega})$. Therefore we write:

$$c''(\tilde{\chi}_{brute}(Y(\tilde{\omega}))) = \sum_{i=1}^{T(\tilde{\omega})} C_{h,Y_i(\tilde{\omega})} \tag{3.14}$$

We now have all the elements to calculate the expected value

$$\begin{aligned} E_{\tilde{P}}[\ c''(\tilde{\chi}_{brute}(Y))\ ] &= E_{\tilde{P}}[\ \sum_{i=1}^{T} C_{h,Y_i}\ ] \\ &= \int_{\tilde{\Omega}} \sum_{i=1}^{T(\tilde{\omega})} C_{h,Y_i(\tilde{\omega})} \cdot \tilde{P}(d\tilde{\omega}) \\ &= \int_{\tilde{\Omega}} \sum_{n=1}^{M} \sum_{i=1}^{n+1} C_{h,Y_i(\tilde{\omega})} \cdot \tilde{P}(d\tilde{\omega} \mid T = n+1) \cdot \tilde{P}(T = n+1) \end{aligned} \tag{3.15}$$

In the last step we conditioned with respect to the event $\{T = n+1\}$.
Since the algorithm χ_{brute} chooses only the numerable sequence of messages Y, we can here assume that the space $(\tilde{\Omega}, \mathscr{F}, \tilde{P})$ is also discrete. Without changing the notation, we will now consider $d\tilde{\omega}$ to be a singleton. With this in mind, we simplify the following expression before proceeding.

$$\tilde{P}(d\tilde{\omega} \mid T = n+1) = \tilde{P}(T = n+1 \mid d\tilde{\omega}) \cdot \frac{\tilde{P}(d\tilde{\omega})}{\tilde{P}(T = n+1)} \tag{3.16}$$

In the equality, **Bayes' formula** was applied. We now reason as follows:

- If $d\tilde{\omega} \in \{T^{-1}(n+1)\} \implies \tilde{P}(T = n+1 \mid d\tilde{\omega}) = 1$
- If $d\tilde{\omega} \notin \{T^{-1}(n+1)\} \implies \tilde{P}(T = n+1 \mid d\tilde{\omega}) = 0$

Therefore we have

$$\begin{aligned} &= \tilde{P}(T = n+1 \mid d\tilde{\omega}) \cdot \frac{\tilde{P}(d\tilde{\omega})}{\tilde{P}(T = n+1)} \\ &= E_{\tilde{P}}[\ \mathbb{1}_{\{T^{-1}(n+1)\}}(d\tilde{\omega})\ ] \cdot \frac{\tilde{P}(d\tilde{\omega})}{\tilde{P}(T = n+1)} \\ &= (1 \cdot \tilde{P}(T = n+1) + 0 \cdot \tilde{P}(T \neq n+1)) \cdot \frac{\tilde{P}(d\tilde{\omega})}{\tilde{P}(T = n+1)} \\ &= \tilde{P}(d\tilde{\omega}) \end{aligned} \tag{3.17}$$

Thanks to this simplification, we finally obtain the result.

$$\begin{aligned}
E_{\tilde{P}}[\ c''(\tilde{\chi}_{brute}(Y))\ ] &= \int_{\tilde{\Omega}} \sum_{n=1}^{M} \sum_{i=1}^{n+1} C_{h,Y_i(\tilde{\omega})} \cdot \tilde{P}(d\tilde{\omega} \mid T = n+1) \cdot \tilde{P}(T = n+1) \\
&= \int_{\tilde{\Omega}} \sum_{n=1}^{M} \sum_{i=1}^{n+1} C_{h,Y_i(\tilde{\omega})} \cdot \tilde{P}(d\tilde{\omega}) \cdot \tilde{P}(T = n+1) \\
&= \sum_{n=1}^{M} (\sum_{i=1}^{n+1} \int_{\tilde{\Omega}} C_{h,Y_i(\tilde{\omega})} \cdot \tilde{P}(d\tilde{\omega})) \cdot \tilde{P}(T = n+1) \\
&= \sum_{n=1}^{M} (\sum_{i=1}^{n+1} \sum_{\mathbb{M}_N} C_{h,m} \cdot \tilde{P_X}(\{m\})) \cdot \tilde{P}(T = n+1) \\
&= C_h \cdot \sum_{n=1}^{M} (n+1) \cdot \tilde{P}(T = n+1) \\
&= C_h \cdot \sum_{n=1}^{M} \frac{(n+1)n \cdot M!}{(M-n)!\ \cdot\ M^{n+1}} \\
&= C_h \cdot E_{\tilde{P}}[\ T\ ]
\end{aligned} \tag{3.18}$$

The value $E_{\tilde{P}}[\ T\ ] = \sum_{n=1}^{M} \frac{(n+1)n \cdot M!}{(M-n)!\ \cdot\ M^{n+1}}$ is in fact related to the well-known **Knuth function** $Q(M)$. [7]

$$Q(M) = \sum_{n=1}^{M} \frac{M!}{(M-n)!\ \cdot\ M^{n}} \tag{3.19}$$

The relationship binding the two is the following:

$$E_{\tilde{P}}[\ T\ ] = \ 1 + Q(M) \tag{3.20}$$

Thanks to the work of Ramanujan, Watson and Knuth [8], it is possible to show that $Q(M)$ has asymptotic expansion

$$Q(M) \sim \sqrt{\frac{\pi M}{2}} - \frac{1}{3} + \frac{1}{12}\sqrt{\frac{\pi}{2M}} - \frac{4}{135M} \cdots \tag{3.21}$$

Which implies that the average cost $E_{\tilde{P}}[\ c''(\tilde{\chi}_{brute}(Y))\ ]$ is

$$E_{\tilde{P}}[\ c''(\tilde{\chi}_{brute}(Y))\ ] \sim C_h \cdot \sqrt{\frac{\pi M}{2}} = \sqrt{\frac{\pi}{2}} C_h \cdot 2^{80} \tag{3.22}$$

Approximate Method for the Expected Cost

It is interesting to consider the following approximate but much simpler method for calculating $E_{\tilde{P}}[\ T\ ]$. [10]

Consider the random variables $H_{i,j} \doteq \mathbb{1}_{\{h(Y_i)\ =\ h(Y_j)\}},\ \forall\ i \neq j$
It's easy to see that

$$E_{\tilde{P}}[\ H_{i,j}\ ] = \tilde{P}(\ h(Y_i) = h(Y_j)\ ) = \frac{1}{M} \tag{3.23}$$

Consider the random variable $H \doteq \sum_{i=1}^{K} \sum_{j=i+1}^{K} H_{i,j}$

K for now is an unknown constant. By imposing conditions on K, we will indirectly obtain an estimate of $E_{\tilde{P}}[\ T\ ]$. Reasoning follows in this way:

If $E_{\tilde{P}}[\ H\ ] \geq 1$ it means that at least one of $H_{i,j}$ is equal to 1, that is, there $\exists\ i \neq j$ such that $h(Y_i) = h(Y_j)$. Applying this condition we obtain that:

$$\begin{aligned} E_{\tilde{P}}[\ H\ ] \geq 1 &\iff \\ E_{\tilde{P}}[\sum_{i=1}^{K} \sum_{j=i+1}^{K} H_{i,j}\ ] \geq 1 &\iff \\ \binom{K}{2} \cdot E_{\tilde{P}}[\ H_{i,j}\ ] \geq 1 &\iff \\ K(K-1) \cdot M^{-1} \geq 2 &\iff \\ K^2 - K - 2M \geq\ 0 &\iff \\ K \sim \sqrt{2} \cdot \sqrt{M} = \sqrt{2} \cdot 2^{80} & \end{aligned} \tag{3.24}$$

This approximation differs by $\frac{\sqrt{2}-\sqrt{\frac{\pi}{2}}}{\sqrt{\frac{\pi}{2}}} \approx 12.84\%$ from the previous one.

3.2.2 Considerations

The cost of the algorithm as seen is in the order of $\sqrt{\frac{\pi}{2}} \cdot 2^{80} \simeq 10^{24}$ hash evaluations. This number is only 6 orders of magnitude greater than the power of the most powerful distributed computers like F@h, previously mentioned in (3.1.2). Although such seems like a remote scenario, criminal organizations but especially governments are or will soon be capable of deploying the resources necessary to carry out attacks of this magnitude.

The situation is actually considerably worse, as there are much more efficient algorithms than bruteforce to find a collision.
In fact, on February 23, 2017, the CWI (Centrum Wiskunde & Informatica) and Google announced the **SHAttered attack**, in which they generated two different PDF files with the same SHA-1 hash in about $2^{63.1}$ SHA-1 evaluations. [9]

The existence of such algorithms is possible because the distribution P_h is not in truth the Uniform distribution over the set $\{0,1\}^{160}$. Such is merely an approximation that does not hold in practice.

3.3 Second Pre-Image-Attack on Merkle-Damgård Hash Functions

From what was proved in Chapter 2, equation (2.26) we already know that a second pre-image-attack can have a lower expected cost compared to that of the best pre-image-attack plus the cost of one hash evaluation (which is negligible in practice).

Second pre-image attacks that significantly improve this upper bound are very sophisticated and beyond the scope of this thesis.

However, it is interesting to analyse a **prototype attack** against one of the most widely used class of hash functions, namely those based on the Merkle-Damgård construction. [11] [12] [13]

3.3.1 Merkle-Damgård Construction

In cryptography, the Merkle-Damgård construction is a method for constructing hash functions from appropriate compression functions. This construction has been used in the design of many popular hash algorithms, such as MD5, SHA-1 and SHA-2.

A **compression function** F is nothing more than a hash function that works on exactly two inputs of fixed length.

$$F : \{0,1\}^k \times \{0,1\}^l \longrightarrow \{0,1\}^n \tag{3.25}$$

Hash function security definitions at (1.2) can be applied to compression functions, so we can speak of pre-image-resistant, second pre-image-resistant and collision-resistant compression functions (considering the domain as $\{0,1\}^{k+l}$).

The M-D construction requires that $k = n$, i.e. it implies that the output of the compression function F can be a valid input for it.

In a sense, the M-D construction aims to **extend the domain** of the compression function F to messages of arbitrary length, while maintaining (and in some cases improving) the security properties of it.

It can be proved that if the compression function F is collision-resistant, then so will the hash function based on F that follows the M-D construction (if appropriate padding schemes are used, as we shall see). [12] [13]

From now on, we will denote the **length of a message** $m \in \mathbb{M}$ with the notation $|m| \doteq min\{N \in \mathbb{N} \mid m \in \mathbb{M}_N\}$. Instead, for $m \in \{0,1\}^l$, we say that the length $|m| \doteq l$.

The **M-D hash function** $h_F : \mathbb{M} \longrightarrow \{0,1\}^n$ is defined as follows:

1. Apply a **padding function** Pad to a given message $m \in \mathbb{M}$.
2. **Divide** $Pad(m)$ into blocks of length l.
   This means $Pad(m) = m_1 \mid\mid m_2 \mid\mid \cdots \mid\mid m_K$, such that $|m_j| = l \; \forall j$.
   Here $||$ is is the binary operation of concatenating strings.
   (e.g. '11' || '00' = '1100')
3. **Calculate** $h_1 \doteq F(IV, m_1)$, where IV is the Initialisation Vector specified by the algorithm.
4. **Calculate** $h_i \doteq F(h_{i-1}, m_i)$ for $i = 2, 3, \cdots K$
5. **Return** $h_F(m) \doteq G(h_K)$, where G is called the Finalization Function.
   (G can be the identity function $G(x) = x$).

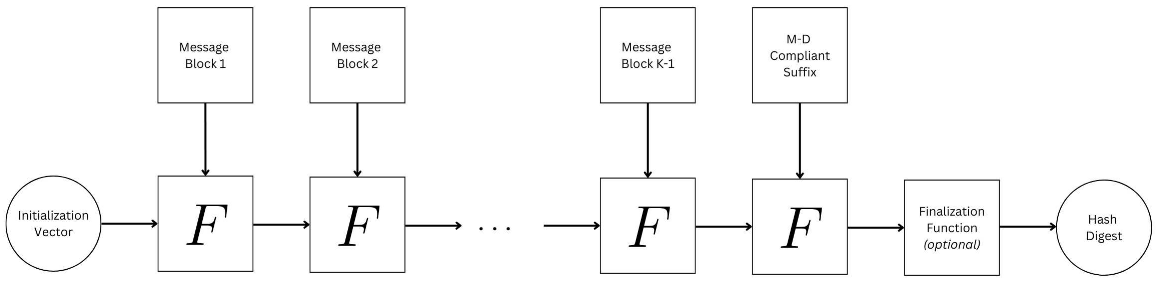

Figure 3.2: Representation of the Merkle-Damgård construction

M-D Compliant Padding

At first glance, the importance of padding may be underestimated, and one might think that any padding that makes the message length a multiple of l would work. Far from it.

Take for example the padding function Pad_0 which adds, at the end of the message m, a number of '0's sufficient to make the length of $Pad_0(m)$ a multiple of l. With this padding scheme, in the event that $|m|$ is not already a multiple of l, it is very easy to find a second pre-image, since m and $m \mid\mid '0'$ will produce the same hash.

It is possible to give **sufficient conditions** for a padding scheme Pad to possess to ensure that the M-D construction is secure: [14]

1. $|Pad(m)|$ is a multiple of $l \ \forall m \in \mathbb{M}$
2. $Pad(m) = m \mid\mid S(m)$, where $S(m)$ is the suffix added to the message m.
3. If $|m| = |m^*| \implies |Pad(m)| = |Pad(m^*)|$
4. If $|m| \neq |m^*| \implies$ the last blocks of $Pad(m)$ and $Pad(m^*)$ are different.

If Pad conforms to point 1. to 4. we say that it is **M-D compliant**.
An example of an M-D compliant padding is the one used in the SHA1 (3.1), called **Length padding**, which involves adding the length of m, represented as a string, to $S(m)$.

Pad_0 is not M-D compliant since it does not respect property number 4. and as we have already seen it leads to hash functions that are not second pre-image-resistant, and therefore not collision-resistant either.

We will now **demonstrate a more general result**:
For every padding function Pad, that respects properties 1. to 3. but not 4., we prove that the Merkle-Damgård hash function h_F is subject to a particular type of second pre-image-attack that greatly improved the upper bound at (2.26).

3.3.2 Long Message Second Pre-Image-Attack

A second pre-image-attack in all generality takes as input any message $m \in \mathbb{M}$. In this case, however, the attack is limited only to very **long messages**, hence the name. We will express the length of these messages as the number of blocks that compose them. For instance, a message $m \in \mathbb{M}$ that is 2^R blocks long means that $|m| = 2^R \cdot l$

Let $h_F : \mathbb{M} \longrightarrow \{0,1\}^n$ be the hash function based on the Merkle-Damgård construction that uses compression function F and the padding scheme Pad. Pad is defined as follows:

- Given $m = m_1 \mid\mid m_2 \mid\mid \cdots \mid\mid m_{K-1} \mid\mid m_K$
  where $|m_i| = l \ \forall i \leq K-1$ and $|m_K| \leq l$
- Let $Pad(m) \doteq m_1 \mid\mid m_2 \mid\mid \cdots \mid\mid m_{K-1} \mid\mid m_K \mid\mid S(m_K)$
  where $|m_K \mid\mid S(m_K)|$ is a multiple of l.

In other words, the suffix $S(m) = S(m_K)$ added to the message only depends on the last $|m| - (K-1) \cdot l$ digits, here called m_K.

The algorithm used by the adversary to perform the Long Message Second Pre-Image-Attack is the non-deterministic algorithm $\tilde{\psi}_{LM} \in \tilde{\mathscr{A}}'$, described below:

Definition of $\tilde{\psi}_{LM}$
INPUT: $m \in \mathbb{M}$ such that $|m| = (2^R + 1) \cdot l$
OUTPUT: $m^* \in \mathbb{M}$
EXECUTION:

1. apply $Pad(m) = m_1 \mid\mid m_2 \mid\mid \cdots \mid\mid m_{2^R} \mid\mid m_{2^R+1} \mid\mid S(m_{2^R+1})$
2. compute $\mathscr{L}_{m,R} \doteq \{\ h_1 = F(IV, m_1),\ h_2 = F(h_1, m_2), \cdots h_{2^R} = F(h_{2^R-1}, m_{2^R})\ \}$
3. randomly choose $m_{link} \in \{0,1\}^l$, according to $\mathscr{U}(\{0,1\}^l)$
4. $F(IV, m_{link}) = h_j \in \mathscr{L}$?
   (a) if YES, RETURN $m_{link} \mid\mid m_{j+1} \mid\mid m_{j+2} \mid\mid \cdots \mid\mid m_{2^R}$
   (b) if NO, go back to point 3.

The cost of performing points 1. and 2. is essentially $C_{h_F,m} = 2^R \cdot C_F$.
The cost of performing **one cycle** of point 4. is C_F which we can consider constant since the length of the message is always l.

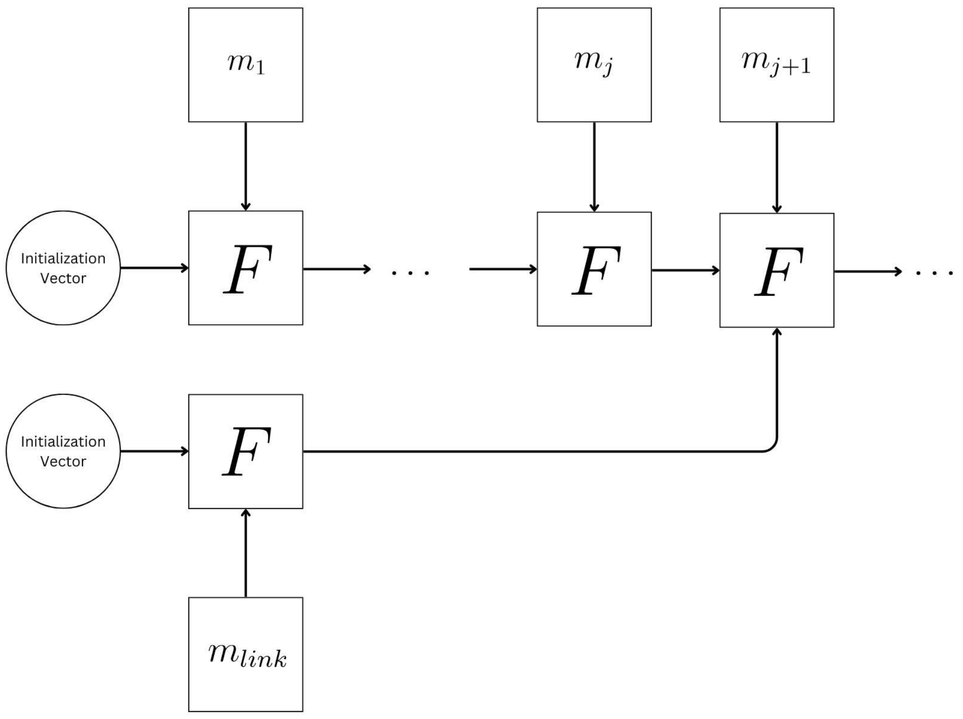

Figure 3.3: Representation of the long message attack performed by the $\tilde{\psi}_{LM}$ algorithm

3.3.3 Expected Cost

We model the choice of $m = m_1 \mid\mid m_2 \mid\mid \cdots \mid\mid m_{2^R} \mid\mid m_{2^R+1}$ with the random variable $X^{(R)} = X_1 \mid\mid X_2 \mid\mid \cdots \mid\mid X_{2^R} \mid\mid X_{2^R+1}$ on Ω, with distribution $P_{X^{(R)}} \doteq P_X(\bullet \mid \mathbb{M}_{2^R+1} \setminus \mathbb{M}_{2^R})$.

We model the choices of m_{link} made by the $\tilde{\psi}_{LM}$ algorithm in Step 3. as the the sequence $Y = (Y_1,\ Y_2,\ Y_3, \cdots)$ of random variables on $\tilde{\Omega}$, i.i.d. with common distribution $P_{Y_1} \sim \mathscr{U}(\{0,1\}^l)$.

The random variables $h_1 \doteq F(IV, X_1),\ h_2 \doteq F(h_1, X_2), \cdots h_{2^R} \doteq F(h_{2^R-1}, X_{2^R})$ are assumed to be i.i.d. with common distribution $P_F \sim \mathscr{U}(\{0,1\}^n)$.
We call $\mathscr{L}_{X,R} \doteq \{\ h_1, h_2, \cdots h_{2^R}\ \}$

The random variables $F_1 \doteq F(IV, Y_1),\ F_2 \doteq F(IV, Y_2), \cdots$ are assumed to be i.i.d. with common distribution $P_F \sim \mathscr{U}(\{0,1\}^n)$.

We now define the random variable U on $\Omega \times \tilde{\Omega}$ as follows:

$$\begin{aligned} U &\doteq \min\{\ m \in \mathbb{N} \mid F_m \in\ \mathscr{L}_{X,R}\} \\ U(\omega, \tilde{\omega}) &= \min\{\ m \in \mathbb{N} \mid F_m(\tilde{w}) \in\ \mathscr{L}_{X(w),R}\ \} \end{aligned} \tag{3.26}$$

As a consequence of the previous cost analysis, it is easy to see that the cost of $\tilde{\psi}_{LM}$ is the following:

$$c'(\tilde{\psi}_{LM}(Y(\tilde{\omega})), X^{(R)}(\omega)) = 2^R \cdot C_F + \sum_{j=1}^{U(\omega,\tilde{\omega})} C_F = (2^R + U(\omega,\tilde{\omega})) \cdot C_F \tag{3.27}$$

Which implies that

$$E_Q[\ c'(\tilde{\psi}_{LM}(Y), X^{(R)})\ ] = (2^R + E_Q[\ U\ ]) \cdot C_F \tag{3.28}$$

If we know the size of the list $\mathscr{L}_{X,R}$ then it is easy to calculate the distribution of U, since the random variables $h_1,\ h_2,\ \cdots,\ F_1,\ F_2,\ \cdots$ are independent. Henceforth, we will consider $L \leq 2^R < 2^n$.

$$\begin{aligned} Q(U = m+1 \mid |\mathscr{L}_{X,R}| = L) &= Q(\min\{m' \in \mathbb{N} \mid F_{m'} \in\ \mathscr{L}_{X,R}\} = m+1 \mid |\mathscr{L}_{X,R}| = L) \\ &= Q(F_i \notin \mathscr{L}_{X,R}\ \forall\ i \leq m,\ F_{m+1} \in \mathscr{L}_{X,R} \mid |\mathscr{L}_{X,R}| = L) \\ &= Q(F_1 \notin \mathscr{L}_{X,R} \mid |\mathscr{L}_{X,R}| = L)^m \cdot Q(F_{m+1} \in \mathscr{L}_{X,R} \mid |\mathscr{L}_{X,R}| = L) \\ &= \left(1 - \frac{L}{2^n}\right)^m \cdot \frac{L}{2^n} \end{aligned} \tag{3.29}$$

From which it follows that under the condition $|\mathscr{L}_{X,R}| = L$, U is a Geometric distribution of parameter $\frac{L}{2^n}$.

To proceed, we would now like to calculate the probability $P(|\mathscr{L}_{X,R}| = L)$, which again is an alternative formulation of the **birthday problem**. However, this seems to be difficult. For this reason, we are going to use an approximation motivated by the next observation on the expected value of $|\mathscr{L}_{X,R}|$.

We first rewrite $|\mathscr{L}_{X,R}|$ using the indicator random variables

$$H_i \doteq \mathbb{1}_{\{h_i \neq h_j \ \forall j<i\}}, \quad i = 2,\ 3, \cdots 2^R \tag{3.30}$$

Now we note that:

$$|\mathscr{L}_{X,R}| = 1 + \sum_{i=2}^{2^R} H_i \ \leq 2^R \tag{3.31}$$

Turning to the expected value, we have that:

$$\begin{aligned}
E_P[\ |\mathscr{L}_{X,R}|\ ] &= 1 + \sum_{i=2}^{2^R} E_P[H_i] \\
&= 1 + \sum_{i=2}^{2^R} P(\{h_i \ \neq \ h_j \ \forall j < i\}) \\
&= 1 + \sum_{i=2}^{2^R} \sum_{k\in\{0,1\}^n} P(\{h_i \ \neq \ h_j \ \forall j < i\} \mid h_i = k) \cdot P(h_i = k) \\
&= 1 + \sum_{i=2}^{2^R} \sum_{k\in\{0,1\}^n} P(\{k \ \neq \ h_j \ \forall j < i\}) \cdot P(h_i = k) \\
&= 1 + \sum_{i=2}^{2^R} 2^n \cdot \left(\frac{2^n-1}{2^n}\right)^{i-1} \cdot 2^{-n} \\
&= \sum_{i=0}^{2^R-1} \left(\frac{2^n-1}{2^n}\right)^{i} \\
&= \frac{1-\left(\frac{2^n-1}{2^n}\right)^{2^R}}{1-\left(\frac{2^n-1}{2^n}\right)} \\
&= 2^n\left(1-\left(1-\frac{1}{2^n}\right)^{2^R}\right) > 2^n\left(1-e^{-2^{R-n}}\right)
\end{aligned} \tag{3.32}$$

We will now make the assumption that $R \leq \frac{n}{2}$. Under this condition, we have

$$\begin{aligned}
2^n\left(1-e^{-2^{R-n}}\right) &= 2^n\left(1-1+2^{R-n}-\frac{1}{2}\cdot 2^{2R-2n}+O(2^{3R-3n})\right) \\
&> 2^n\left(2^{R-n}-\frac{1}{2}\cdot 2^{2R-2n}\right) \\
&= 2^R-\frac{1}{2}\cdot 2^{2R-n} \geq 2^R-\frac{1}{2}
\end{aligned} \tag{3.33}$$

What we have proved, is the following surprising result:

$$2^R \geq E_P[\ |\mathscr{L}_{X,R}|\ ] > 2^R - \frac{1}{2} \quad \forall R \leq \frac{n}{2} \tag{3.34}$$

For sufficiently large R, this implies that the probability measure $P(|\mathscr{L}_{X,R}| = \bullet)$ concentrates nearly **all the mass** close to the value 2^R.

Taking $R > 50$ (which implies $n > 100$), we can with good reason approximate

$$P(|\mathscr{L}_{X,R}| = \bullet) \approx \delta_{2^R} \tag{3.35}$$

Where δ_{x_0} is the Dirac delta function centered in x_0.
Therefore, the expected value of U is:

$$\begin{aligned} E_Q[\ U\ ] &= \sum_{m=0}^{\infty}(m+1)\cdot Q(U = m+1) \\ &= \sum_{m=0}^{\infty}\sum_{k=1}^{2^R}(m+1)\cdot Q(U = m+1 \mid |\mathscr{L}_{X,R}| = k)\cdot P(|\mathscr{L}_{X,R}| = k) \\ &\approx \sum_{m=0}^{\infty}(m+1)\cdot(Q(U = m+1 \mid |\mathscr{L}_{X,R}| = 2^R) \\ &= \sum_{m=0}^{\infty}(m+1)(1-2^{R-n})^m\cdot 2^{R-n} \\ &= 2^{n-R} \end{aligned} \tag{3.36}$$

And finally, the expected cost is

$$\begin{aligned} E_Q[\ c'(\tilde{\psi}_{LM}(Y), X^{(R)})\ ] &= (2^R + 2^{n-R})\cdot C_F \\ \xrightarrow{R=\frac{n}{2}} &= C_F\cdot 2^{\frac{n}{2}+1} \ll C_{h_F}\cdot 2^n \end{aligned} \tag{3.37}$$

3.3.4 Considerations

This result is a great improvement over upper bound in (2.26), however, at the moment it seems to depend on the suffix function $S(m) = S(m_K)$.
It is actually not difficult to generalize to $S(m) = S(m_K, m_{K-1}, \cdots, m_{K-J+1})$, in which case the only thing that changes is the target list $\mathscr{L}_{m,R}$ which has a reduced length of J compared to that of the message. It will suffice to use messages $2^R + J$ blocks long to have the same expected cost of $2^R + 2^{n-R}$ compression function evaluations.

For an M-D compliant padding function such as length padding, the attack as it is described would not work, because $|m^*| < |m|$ and property number 4. implies that the two suffixes will be different.
However, not without difficulty, it is possible to generalize the described attack in a way to bypass length padding, resulting in an expected cost of
$R \cdot 2^{n/2+1} + 2^{n-R+1} \ll 2^n$ compression function evaluations. [15]

Applying these results to the SHA1 function ($n = 160$), and considering $R = 64$, the cost for performing the attack is $64 \cdot 2^{81} + 2^{97} \simeq 10^{29}$ compression function evaluations. This number is 11 orders of magnitude greater than the power of the most powerful distributed computers like F@h, previously mentioned in (3.1).

Because of this, and the fact that the attack applies only on extremely long messages, SHA1 is still considered a second pre-image-resistant function, but this may change in the future thanks to new developments and optimizations.

The work of John Kelsey and Bruce Schneier [15], combined with the work of Joux [16] raise questions about the usefulness of the widely-used Merkle-Damgård construction.

References

[1] *Aiden A. Bruen; Mario A. Forcinito; James M. McQuillan, (2021), "Cryptography, Information Theory and Error-Correction," in Cryptography, Information Theory, and Error-Correction: A Handbook for the 21st Century, Wiley, doi: 10.1002/9781119582397.oth,*
`https://ieeexplore.ieee.org/document/9821713`.

[2] *Auguste Kerckhoffs (January 1883). "La cryptographie militaire". Journal des sciences militaires*
`https://www.petitcolas.net/kerckhoffs/crypto_militaire_1_b.pdf`.

[3] *National Institute of Standards and Technology (11 May 1993), "Secure Hash Standard, Federal Information Processing Standards Publication FIPS PUB 180"*

[4] *Samuel Kramer (11 July 1994), "Proposed Revision of Federal Information Processing Standard (FIPS) 180, Secure Hash Standard", Federal Register*
`https://www.federalregister.gov/documents/1994/07/11/94-16666/`
`proposed-revision-of-federal-information-processing-standard-fips-180-secu`

[5] *National Institute of Standards and Technology (August 2015), "Secure Hash Standard, Federal Information Processing Standards Publication FIPS PUB 180-4"*
`https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf`

[6] *"Folding@home Tops World's Fastest Supercomputer With 2.4 ExaFLOPS For COVID-19 Research", (15 April 2020)*
`https://hothardware.com/news/foldinghome-surpasses-24-exaflops-covid-19-re`

[7] *Donald E. Knuth (1973), "The Art of Computer Programming", Vol. 3, Sorting and Searching, Reading, Massachusetts: Addison-Wesley*

[8] *Philippe Flajolet, Peter J. Grabner, Peter Kirschenhofer, Helmut Prodinger (1995), "On Ramanujan's Q-Function", Journal of Computational and Applied Mathematics, 58: 103–116*
https://www.sciencedirect.com/science/article/pii/0377042793E0258N

[9] *Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, Yarik Markov, Alex Petit Bianco, Clement Baisse (February 23, 2017), "Announcing the first SHA1 collision", Google Security Blog*
https://shattered.io/static/shattered.pdf

[10] *Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, "Introduction To Algorithms", I Foundations, chapter 5.4: page 130*
https://edutechlearners.com/download/Introduction_to_algorithms-3rd%20Edition.pdf

[11] *Ralph C. Merkle (1979), "Secrecy, authentication, and public key systems", Stanford Ph.D. thesis , pages 13-15*
http://www.ralphmerkle.com/papers/Thesis1979.pdf

[12] *Ralph C. Merkle (1989), "A Certified Digital Signature", In Advances in Cryptology – CRYPTO '89 Proceedings, Lecture Notes in Computer Science Vol. 435, G. Brassard, ed, Springer-Verlag, pages 218-238*

[13] *Ivan Damgård (1989), "A Design Principle for Hash Functions", In Advances in Cryptology – CRYPTO '89 Proceedings, Lecture Notes in Computer Science Vol. 435, G. Brassard, ed, Springer-Verlag, , pages 416-427*

[14] *Shafi Goldwasser, Mihir Bellare (July 2008), "Lecture Notes on Cryptography"* https://cseweb.ucsd.edu/~mihir/papers/gb.pdf

[15] *John Kelsey, Bruce Schneier (2005), "Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work"*
https://link.springer.com/content/pdf/10.1007/11426639_28.pdf

[16] *Antoine Joux (2004), "Multicollisions in Iterated Hash Functions. Applications to Cascaded Constructions," Advances in Cryptology–Crypto 2004 Proceedings, Springer-Verlag, 2004.*
https://link.springer.com/content/pdf/10.1007/978-3-540-28628-8_19.pdf