Information Processing Letters 113 (2013) 60–66

Contents lists available at SciVerse ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Memoryless near-collisions, revisited

Mario Lamberger [a,*], Elmar Teufl [b]

[a] Institute for Applied Information Processing and Communications, Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
[b] Mathematisches Institut, Eberhard Karls Universität Tübingen, Auf der Morgenstelle 10, D-72076 Tübingen, Germany

ARTICLE INFO

Article history:
Received 29 August 2011
Received in revised form 7 October 2012
Accepted 17 November 2012
Available online 20 November 2012
Communicated by D. Pointcheval

Keywords:
Cryptography
Hash functions
Memoryless near-collisions
Covering codes

ABSTRACT

In this paper we discuss the problem of generically finding near-collisions for cryptographic hash functions in a memoryless way. A common approach is to truncate several output bits of the hash function and to look for collisions of this modified function. In two recent papers, an enhancement to this approach was introduced which is based on classical cycle-finding techniques and covering codes. This paper investigates two aspects of the problem of memoryless near-collisions. Firstly, we give a full treatment of the trade-off between the number of truncated bits and the success-probability of the truncation based approach. Secondly, we demonstrate the limits of cycle-finding methods for finding near-collisions by showing that, opposed to the collision case, a memoryless variant cannot match the query-complexity of the "memory-full" birthday-like near-collision finding method.

1. Introduction

The field of hash function research has developed significantly in the light of the attacks on some of the most frequently used hash functions like MD4, MD5 and SHA-1. As a consequence, academia and industry started to evaluate alternative hash functions, *e.g.* in the SHA-3 initiative organized by NIST [1]. During this ongoing evaluation, not only the three classical security requirements *collision resistance*, *preimage resistance* and *second preimage resistance* are considered. Researchers look at (semi-)free-start collisions, near-collisions, distinguishers, etc. A 'behavior different from that expected of a random oracle' for the hash function is undesirable as are weaknesses that are demonstrated only for the compression function and not for the full hash function.

Coding theory and hash function cryptanalysis have gone hand in hand for quite some time now, where a crucial part of the attacks is based on the search for low-weight code words in a linear code (*cf.* [2–4] among others). In this paper, we want to elaborate on a newly proposed application of coding theory to hash function cryptanalysis. In [5,6], it is demonstrated how to use covering codes to find near-collisions for hash functions in a memoryless way. We also want to refer to the recent paper [7] which considers similar concepts from the viewpoint of locality sensitive hashing.

In all of the following, we will work with binary values, where we identify $\{0,1\}^n$ with $\mathbb{Z}_2^n$. Let "+" denote the n-bit exclusive-or operation. The Hamming weight of a vector $v \in \mathbb{Z}_2^n$ is denoted by $\mathrm{w}(v) = |\{i \mid v_i = 1\}|$ and the Hamming distance of two vectors by $\mathrm{d}(u,v) = \mathrm{w}(u+v)$. The Handbook of Applied Cryptography [8, p. 331] defines *near-collision resistance* of a hash function H as follows:

Definition 1 *(Near-collision resistance).* It should be hard to find any two inputs m, m^* with $m \neq m^*$ such that $H(m)$ and $H(m^*)$ differ in only a small number of bits:

$$\mathrm{d}\big(H(m), H\big(m^*\big)\big) \leqslant \epsilon. \qquad (1)$$

For ease of later use we also give the following definition:

Definition 2. A message pair m, m^* with $m \neq m^*$ is called an ϵ-*near-collision* for H if (1) holds.

[*] Corresponding author.
E-mail addresses: mario.lamberger@iaik.tugraz.at (M. Lamberger), elmar.teufl@uni-tuebingen.de (E. Teufl).

0020-0190/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ipl.2012.11.004

Collisions can be considered a special case of near-collisions with the parameter $\epsilon = 0$. The generic method for finding collisions for a given hash function is based on the *birthday paradox* and attributed to Yuval [9]. There are well established cycle-finding techniques (due to Floyd, Brent, Nivasch, *cf.* [10–12]) that remove the memory requirements from an attack based on the birthday paradox (see also [13]). These methods work by repeated iteration of the underlying hash function where in all of these applications the function is considered to behave like a random mapping (*cf.* [14,15]).

In [5,6], the question is raised whether or not the above mentioned cycle-finding techniques are also applicable to the problem of finding near-collisions. We now briefly summarize the ideas of [5,6].

Since Definitions 1 and 2 include collisions as well, the task of finding near-collisions is easier than finding collisions. We now want to have a look at generic methods to construct near-collisions which are more efficient than the generic methods to find collisions.

In the following, let $B_r(x) := \{y \in \mathbb{Z}_2^n \mid \mathrm{d}(x, y) \leqslant r\}$ denote the *Hamming ball* (or *Hamming sphere*) around x of radius r. Furthermore, we denote by $S_n(r) := |B_r(x)| = \sum_{i=0}^{r} \binom{n}{i}$ the cardinality of any n-dimensional Hamming ball of radius r.

A simple adaption of the classical table-based birthday attack for finding ϵ-near-collisions is to start with an empty table, randomly select a message m and compute $H(m)$ and then test whether the table contains an entry $(H(m) + \delta, m^*)$ for some $\delta \in B_\epsilon(0)$ and arbitrary m^*. If so, the pair (m, m^*) is an ϵ-near-collision. If not, $(H(m), m)$ is added to the table and repeat. Then, we know the following:

Lemma 1. *(See [5].) Let H be an n-bit hash function. If we assume that H acts like a random mapping, the average number of messages that we need to hash and store in a table-based birthday-like attack before we find an ϵ-near-collision is $O(2^{n/2} S_n(\epsilon)^{-1/2})$.*

Remark 1. We want to note that in this paper we are measuring the complexity of a problem by counting (hash) function invocations. This constitutes an adequate measure in the case of the memoryless algorithms in this paper, however the real computational complexity of the table-based algorithm above is dominated by the memory access, as the problem of searching for an ϵ-near-collision in the table is much harder than testing for a collision.

The first straight-forward approach to apply the cycle-finding algorithms to the problem of finding near-collisions is a truncation based approach.

Lemma 2. *Let H be an n-bit hash function. Let $\tau_\epsilon : \mathbb{Z}_2^n \to \mathbb{Z}_2^{n-\epsilon}$ be a map that truncates ϵ bits from its input at predefined positions. If we assume that $\tau_\epsilon \circ H$ acts like a random mapping, we can apply a cycle-finding algorithm to the map $\tau_\epsilon \circ H$ to find an ϵ-near-collision in a memoryless way with an expected complexity of about $2^{(n-\epsilon)/2}$.*

Proof. Under the assumptions of the lemma, the results from [14,15] are applied to a random mapping with output length $n - \epsilon$. $\square$

2. A thorough analysis of the truncation approach

As indicated in [5], a simple idea to improve the truncation based approach is to truncate more than ϵ bits. That is, in order to find an ϵ-near-collision we simply truncate μ bits with $\mu > \epsilon$. A cycle-finding method applied to $\tau_\mu \circ H$ has an expected complexity of $2^{(n-\mu)/2}$ and deterministically finds two messages m, m^* such that $\mathrm{d}(H(m), H(m^*)) \leqslant \mu$. However, we can look at the probability that these two messages m, m^* satisfy $\mathrm{d}(H(m), H(m^*)) \leqslant \epsilon$ which is $2^{-\mu} \sum_{i=0}^{\epsilon} \binom{\mu}{i} = 2^{-\mu} S_\mu(\epsilon)$.

For a truly memoryless approach, multiple runs of the cycle-finding algorithm are interpreted as independent events. Therefore, the expected complexity to find an ϵ-near-collision can be obtained as the product of the expected complexity to find a cycle, and the expected number of repetitions of the cycle-finding algorithm, *i.e.* the reciprocal value of the probability that a single run finds an ϵ-near-collision. In other words, we end up with an expected complexity of

$$2^{(n+\mu)/2} S_\mu(\epsilon)^{-1} = 2^{(n+\mu)/2} \left(\sum_{i=0}^{\epsilon} \binom{\mu}{i} \right)^{-1}. \tag{2}$$

Remark 2. In [5], the above approach was already proposed with $\mu = 2\epsilon + 1$. In this case (2) results in a complexity of

$$2^{(n+2\epsilon+1)/2} S_{2\epsilon+1}(\epsilon)^{-1} = 2^{(n+1)/2-\epsilon},$$

which clearly improves upon Lemma 2. Here we have used that $S_{2\epsilon+1}(\epsilon) = \frac{1}{2} S_{2\epsilon+1}(2\epsilon + 1) = 2^{2\epsilon}$.

An interesting question that now arises is to find the number of truncated bits μ that constitutes the best trade-off between a larger μ, *i.e.* a faster cycle-finding part, and a higher number of repetitions for this probabilistic approach. In other words, we would like to determine the value of μ which minimizes (2) for a given ϵ. Analogously, we can search for an integer $\mu > \epsilon$ such that for a given ϵ the expression $2^{-\mu/2} S_\mu(\epsilon)$ is maximized. For small values of ϵ, values for μ were already computed in [5] by an exhaustive search. In this section, we want so solve this problem analytically.

We first show a result that tells us something about the behavior of the sequence of real numbers

$$a_\mu := 2^{-\mu/2} S_\mu(\epsilon) = 2^{-\mu/2} \sum_{i=0}^{\epsilon} \binom{\mu}{i}. \tag{3}$$

We want to note that based on the origin of the problem, we are only interested in values a_μ for $\mu > \epsilon$. Our analysis is still valid starting with $\mu = 1$. We will need the following two properties of sequences: