Collision Spectrum, Entropy Loss, T-Sponges, and Cryptanalysis of GLUON-64

Léo Perrin(✉) and Dmitry Khovratovich

University of Luxembourg, Walferdange, Luxembourg
{leo.perrin,dmitry.khovratovich}@uni.lu

Abstract. In this paper, we investigate the properties of iterative non-injective functions and the security of primitives where they are used. First, we introduce the Collision Probability Spectrum (CPS) parameter to quantify how far from a permutation a function is. In particular, we show that the output size decreases linearly with the number of iterations whereas the collision trees grow quadratically.

Secondly, we investigate the T-sponge construction and show how certain CPS and rate values lead to an improved preimage attack on long messages. As an example, we find collisions for the GLUON-64 internal function, approximate its CPS, and show an attack that violates the security claims. For instance, if a message ends with a sequence of 1 Mb (respectively 1 Gb) of zeros, then our preimage search takes time $2^{115.3}$ (respectively $2^{105.3}$) instead of 2^{128}.

Keywords: Random function · Collision probability spectrum · Collision tree · T-sponge · GLUON · Collision search

1 Introduction

Consider a function $g : \mathcal{S} \to \mathcal{S}$ where $\mathcal{S}$ is some finite space of size 2^N and suppose that it is not a permutation, i.e. that it has collisions. It is well known that for a random g the complexity of a collision search is of $2^{N/2}$ calls to g. However, not only the collision search complexity but also some related problems are not well studied when collisions have a certain structure, which is the case in several designs [1,2]. It might be clear that iterating such a function may lead to an entropy loss, but again, the scale of this loss and its implications on the security of stream ciphers and hash functions is not well known or underestimated. In this paper we introduce a particular parameter called the Collision Probability Spectrum (CPS), which is based on the number of solutions for the following equation

$$g(a + y) = g(a). \tag{1}$$

We study the CPS for several designs and show, as an illustration of our methodology, a preimage attack on the sponge-based lightweight hash function GLUON-64.

L. Perrin—The first author is supported by the core ACRYPT project from the *Fond National de la Recherche* (Luxembourg).

C. Cid and C. Rechberger (Eds.): FSE 2014, LNCS 8540, pp. 82–103, 2015.
DOI: 10.1007/978-3-662-46706-0_5

Related work. Bellare and Kohno [3] studied how the number of preimages to $g(a)$ affects the complexity of the collision search with the notion of *balance* of a function. In [4], Flajolet and Odlyzko studied several characteristics of random mappings, in particular the distribution of preimage sizes, the cycle size and the size of the iterated image. Their result was applied by Hong and Kim [5] to the MICKEY [1] cipher. Indeed, they found experimentally that the size of the iterated images of this function was essentially the size of the space divided by the number of iterations, a behavior which they showed experimentally to correspond to the prediction of Flajolet et al. However, the resulting attacks were found to be less efficient than the simple collision search [6], though they allow a time/memory trade-off.

Overview of our results. We introduce the *Collision Probability Spectrum* parameter which quantifies how many solutions Eq. (1) has on average and investigate its consequences over the iterated images and preimages of $\mathcal{S}$ by g. We assume that the composition of two such functions has certain properties, which is formalized as an independence assumption. For a large class of mappings two important facts are proved in Theorem 2 (a reader may refer to Fig. 1):

- First, the size of the iterated image of g is inversely proportional with the number i of iterations:
$$|g^i(\mathcal{S})| \sim \frac{|\mathcal{S}|}{\frac{\kappa}{2} \cdot i},$$
where κ depends on the CPS and where i has to be smaller than $\sqrt{|\mathcal{S}|}$ — otherwise, the result does not hold because of the cycles in the functionnal graph.
- Second, an element $y \in g^i(\mathcal{S})$ is the root of a *collision tree* consisting in elements x_l such that any of $g(x_l), g^2(x_l), \ldots, g^i(x_l)$ is equal to y. The average size of this tree is ν_i:
$$\nu_i \sim \frac{\kappa}{4} \cdot i^2,$$
with the same restriction on i: $i < \sqrt{|\mathcal{S}|}$.

Then we discuss the security of the T-sponge construction provided the CPS of the update function. We amend the collision search bound in the flat sponge claim [7]:

$$P = \frac{Q^2}{2^{c+1}} \cdot \left(1 + \frac{\kappa - 1}{2^r}\right),$$

where c is the capacity and r is the rate of the sponge.

Next, in Theorem 6, we show for small r an improved preimage attack with complexity

$$2^c \cdot 2^{r+2}/(\kappa z),$$

where z is the number of zero bytes in the end of the hashed message (actually, any constant suffices).

Finally, we construct an attack on GLUON-64. Aided with a SAT-solver, we find collisions for the update function and demonstrate a preimage attack of

complexity 2^{105} for a message ending with 1 GByte of zeros, which violate the claimed preimage resistance level of 128 bits.

Structure. This paper is organized as follows. We introduce our theoretical framework in Sect. 2 and discuss its application to existing primitives. We investigate the security of T-sponge against collision and preimage search in Sect. 3. Finally, in Sect. 4, we obtain inner-collisions of the update function of GLUON-64 with the help of a SAT-solver and show a preimage attack. For the sake of concision, the proofs are moved to Appendix A.

Notations. We denote by $|E|$ and $\#E$ the size of a set E, by $\mathbb{P}[\omega]$ the probability of an event ω and by $a \xleftarrow{\$} E$ the fact that a is drawn uniformly at random from a set E.

2 Theoretical Framework

In this section we introduce a model of random functions and highlight its difference with the usual approach. We then give several properties of the (iterated) images and preimages of an element by such functions.

2.1 Collision Probability Spectrum and Function Model

Definition 1 (Collision Probability Spectrum). *Let $\mathcal{S}$ be a finite space and let $g : \mathcal{S} \to \mathcal{S}$ be a function. We denote $\mathfrak{c}_k$ the probability that the following equation has exactly k solutions for $a \in \mathcal{S}$ picked uniformly at random in $\mathcal{S}$:*

$$g(a + x) = g(a), \tag{2}$$

so that

$$\mathfrak{c}_k = \mathbb{P}[\#\{x \in \mathcal{S}, g(a + x) = g(a)\} = k \mid a \xleftarrow{\$} \mathcal{S}] \tag{3}$$

The solutions x of this equation are called vanishing differences. *The set of all the elements a of $\mathcal{S}$ such that Eq. (2) has exactly k solutions is denoted V_k. Finally, the set $\mathfrak{C} = \{\mathfrak{c}_k\}_{k \geq 1}$ is the* Collision Probability Spectrum *(CPS) of g.*

An equivalent definition of the CPS is that it is the probability distribution of the number of solutions of Eq. 2. We now make some remarks regarding these definitions:

- Since 0 is always a solution of Eq. (2), we have that $\mathfrak{c}_0 = 0$.
- If g is a permutation, then $\mathfrak{C}(g) = \{\mathfrak{c}_1 = 1, \mathfrak{c}_k = 0 \text{ for } k > 1\}$.
- The input space can be partitioned in the following way: $\mathcal{S} = \bigcup_{k=1}^{\infty} V_k$. Furthermore, the output space can be partitioned as $g(\mathcal{S}) = \bigcup_{k=1}^{\infty} g(V_k)$. This is also a disjoint union. Indeed, $y \in g(V_k)$ has exactly k preimages, by definition.
- The size of $g(V_k)$ is $|g(V_k)| = |\mathcal{S}| \cdot \mathfrak{c}_k/k$ because to each element in $g(V_k)$ correspond k elements in V_k (see Fig. 2). As a consequence,

$$|g(\mathcal{S})| = |\mathcal{S}| \cdot \sum_{k=1}^{\infty} \frac{\mathfrak{c}_k}{k}$$

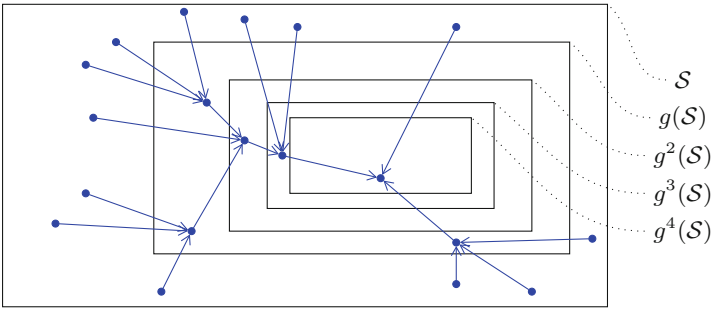

Fig. 1. Collision trees and output shrinkage of iterative non-injective functions. The dots represent elements of $\mathcal{S}$ and there is an edge from x to y if $g(x) = y$. Here, $g(a + x) = g(a)$ always has exactly 3 solutions.

2.2 Composition of Functions with Known CPS

The most interesting application of our theory is the properties of iterative constructions where the iterated function has some known CPS. However, to make meaningful and correct statements about composition of such functions, some independency must be assumed.

Assumption 1 (Independence Assumption). *Let g be a function with* CPS *$\mathfrak{C}$. Then there is no correlation between the events $x \in V_j$ and $g(x) \in V_k$ for any j, k.*

This assumption, as we will see, holds for a few (but not for all) real primitives. For the rest of the paper, we implicitly assume that it holds unless stated otherwise.

Definition 2. *Suppose g is a function on $\mathcal{S}$. Then ℓ_i defined as*

$$\ell_i = \frac{|\mathcal{S}|}{|g^i(\mathcal{S})|}$$

is called the shrinking ratio *of g.*

Our first theorem allows to compute the shrinking ratio of the composition of two functions with given CPS.

Theorem 1. *Let g and g' be functions with* CPS *$\mathfrak{C} = \{\mathfrak{c}_k\}_{k\geq 1}$ and $\mathfrak{C}' = \{\mathfrak{c}'_k\}_{k\geq 1}$, respectively. Then the shrinking ratio of the composition $g \circ g'$ is computed as follows:*

$$\ell_1(g \circ g') \;=\; \Big(\frac{1}{\ell_1} - \sum_{k=1}^{\infty} \frac{\mathfrak{c}_k}{k}\big(1 - \frac{1}{\ell'_1}\big)^k\Big)^{-1}.$$

In particular, when $g' = g^i$:

$$\ell_{i+1} = \Big(\frac{1}{\ell_1} - \sum_{k=1}^{\infty} \frac{\mathfrak{c}_k}{k}\big(1 - \frac{1}{\ell_i}\big)^k\Big)^{-1}.$$

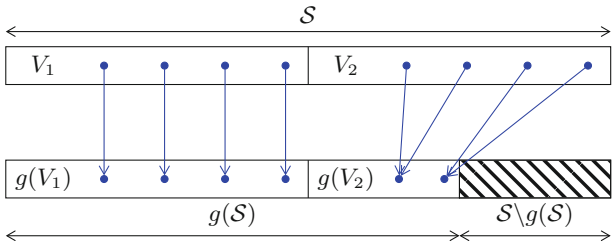

Fig. 2. The effect of g with CPS $\{c_1 = c_2 = 1/2\}$ on $\mathcal{S}$.

A detailed proof is given in Appendix A.1 but we provide a high level view of its structure.

Proof Sketch 1. We consider an element $x_0 \in g'(\mathcal{S})$ such that there exists $\{x_0, ..., x_k\}$ with $g(x_l) = g(x_0)$, i.e. $x_0 \in V_{k+1}$. The number of solutions of $g(x_0 + x) = g(x_0)$ in $g'(\mathcal{S})$ for x_0 drawn at random in $g'(\mathcal{S}) \cap V_{k+1}$ follows a binomial distribution $(m, k, 1/\ell'_1)$ as $x_l \in g'(\mathcal{S})$ with probability $|g'(\mathcal{S})|/|\mathcal{S}| = 1/\ell'_1$.

Using this observation, we can compute the probability that $g(x_0+x) = g(x_0)$ has m solutions in $g'(\mathcal{S})$ for all m: if it has $m+1$ solutions, then it must be that $x_0 \in V_{k+1}$ and that only m of the k non zero solutions "made it" to $g'(\mathcal{S})$. Then, we deduce the size of the image of $g'(\mathcal{S})$ by g, i.e. we give an expression of $\ell_1(g \circ g')$.

Using this theorem, we can give the asymptotic behavior of ℓ_i and of the size of the collision trees as i increases while remaining small enough so that $g(x)$ is not on a cycle. The results stated below have been checked experimentally on the functions for which the independence assumption presumably holds. We need two more definitions.

Definition 3. *Suppose g is a function on $\mathcal{S}$ with* CPS $\mathfrak{C}$. *Then*

- $\kappa(\mathfrak{C})$ *is the* collision average of g — *the average number of non-zero solutions of Eq. (2):* $\kappa = \sum_{k\geq 1} \mathfrak{c}_k \cdot k - 1$.
- $\nu_i(g)$ *is the* average tree size of g — *the average number of elements in a collision tree rooted in $g^i(\mathcal{S})$. Formally, it is the average number of pairs $(x_l, k_l) \in \mathcal{S} \times [1, i]$ such that $g^{k_l}(x_l) = y$ for $y \in g^i(\mathcal{S})$.*

Theorem 2. *Let g be a function with* CPS $\mathfrak{C}$, *then for $i < \sqrt{|\mathcal{S}|}$ the shrinking ratio and the average tree size are approximated as follows for large enough i:*

$$\ell_i \sim \frac{\kappa}{2} \cdot i,\ \nu_i \sim \frac{\kappa}{4} \cdot i^2.$$

Proof Sketch 2. The asymptotic behaviour of ℓ_i can be deduced by using Theorem 1 with $g' = g^i$ and then using the finite expansion of $(1 - 1/\ell_i)^k$ to see that $\ell_{i+1} = \ell_i + \kappa/2$. For ν_i, we simply note that $\nu_i = \sum_{k=1}^{i} \ell_k$. More details are given in Appendix A.2.

Finally, we define the following quantities in the same way as Flajolet et al. [4].

Definition 4. *We call* cycle length *and* tail length*, denoted respectively μ and λ, the average smallest values such that*

$$g^{\lambda}(x) = g^{\lambda+\mu}(x)$$

for x drawn uniformly at random in $\mathcal{S}$.

Experiments (see Appendix A.3) lead us to the following conjecture.

Conjecture 1. Let g be a function of $\mathcal{S}$ with CPS $\mathfrak{C}$. Experimentally, we found the following values for the tail length λ and the cycle length μ:

$$\lambda \sim \sqrt{\frac{\pi}{8 \cdot \kappa}|\mathcal{S}|},\ \mu \sim \sqrt{\frac{\pi}{8 \cdot \kappa}|\mathcal{S}|}.$$

2.3 Independence Assumption in Practice

In this Section, we investigate some results from the literature about particular functions and see how relevant our model is. A summary of this Section is given in Table 1.

Table 1. Characteristics derived from the CPS of some functions.

Function	κ	ℓ_1	ℓ_i/i	ν_i/i^2	Reference for the CPS
MICKEY's update function	0.625	1.407	$2^{-1.7}$	$2^{-2.7}$	[8]
Random mapping	1	1.582	2^{-1}	2^{-2}	[4]
GLUON-64's update function	6.982	3.578	$2^{1.8}$	$2^{0.8}$	Sect. 4.2

Random Mappings. The authors of [4] study random mappings and give the probability that some $x \in \mathcal{S}$ has r preimages by a random mapping g. From this we deduce that the CPS of a random function is given by the Poisson distribution with $\lambda = 1$:

$$\mathfrak{C} = \{e^{-1}/(k-1)!\}_{k \geq 1}.$$

Our framework implies

$$\kappa = 1 \quad \text{and} \quad \ell_1 = 1/(1-e^{-1}) \quad \text{and} \quad \ell_{i+1} = 1/\big(1 - \exp(-1/\ell_i)\big)$$

which fits the results of [4](see also Appendix A.1). The authors of [5] observed that

$$\log_2(\ell_i) \approx \log_2(i) - 1,$$

which also corresponds to $\kappa = 1$. Finally, the trail and cycle length given in Conjecture 1 match those predicted by [4] if we replace κ by 1.

A5/1. The update function of A5/1 does not satisfy the independence assumption. The author of [2] computed its CPS and established that

$$\ell_1 = 1.6, \quad \kappa = 1.25,$$

If the assumption held, then the probability for an element in $\mathcal{S}$ to be in $g^{100}(\mathcal{S})$ would be about 2^{-6}, which is very different from the $2^{-2.5}$ actually observed by Biryukov et al. [9]. The reason is that the update function A5/1 may keep one of its three LFSR's untouched, which means that $x \in V_j$ and $g(x) \in V_k$ are *not* independent events in its case.

MICKEY. The update function of the MICKEY [1] stream-ciphers (v1 and v2) fits our model. Hong and Kim [5] performed some experiments on the first version of MICKEY and, in particular, estimated the size of $g^{2^k}(\mathcal{S})$ for several values of k. Their results are coherent with our model. For instance, they observed that $\log_2(\ell_i)$ (which they denote by $\overline{EL}(f^i)$) is approximated as

$$log_2(\ell_i) \approx \log_2(i) - 1.8$$

The constant term 1.8 implies

$$\kappa/2 \approx 2^{-1.8}.$$

In turn, from the CPS values computed in [8](actually, the values $\mathfrak{c}_k/k$) we obtain the theoretical value

$$\kappa = 0.625,$$

which corresponds to a difference of about 7 % with the experiments in [5].

3 Improved Collision and Preimage Search

In this section we explore generic collision and preimage search methods in their application to functions with fixed collision spectrum.

3.1 Basic Collision Search

First, we reformulate the result from Bellare and Kohno [3] with our notation.

Theorem 3 *[3]. Let g be a function with CPS $\mathfrak{C}$, and let κ be its collision average. Then the birthday collision attack on g requires about*

$$Q = \sqrt{\frac{|\mathcal{S}|}{\kappa + 1}}. \tag{4}$$

queries to g.

The original paper [3] used the parameter *balance* of g, denoted $\mu(g)$, which is computed as

$$\mu(g) = \log_{|g(\mathcal{S})|}\left(\frac{|\mathcal{S}|^2}{\sum_{y\in\mathcal{S}}|g^{-1}(y)|^2}\right) \tag{5}$$

If we know the CPS of g, the balance can be expressed as follows:

$$\mu(g) \approx 1 - \frac{\log_2\left(\sum_{k=1}^{\infty} k\cdot\mathfrak{c}_k\right) + \log_2\left(\sum_{k=1}^{\infty}\mathfrak{c}_k/k\right)}{\log_2\left(|\mathcal{S}|\right)}. \tag{6}$$

If Conjecture 1 holds, then the memory-less collision search based on Floyd's cycle finding algorithm should be $\sqrt{\kappa}$ as fast as in the case of a random function.

3.2 Collision Attacks on T-sponge

Now we demonstrate that the entropy loss because of collisions in the T-sponge construction, though observable, can be mitigated by a large rate parameter.

Sponge Construction. The sponge construction [7] is characterised by its *rate* r, its *capacity* c and its update function g. It is based on an internal state of size $r + c$ where, at each round, r bits of the message are xor-ed. Then the sponge alternates the application of g function with the message injection until the message has been entirely *absorbed*. The digest is then *squeezed* by extracting r bits of the internal state and applying the update function to the internal state again. This is repeated as many times as necessary to obtain a digest of desired length. A representation of a sponge is given in Fig. 3. The sponge-based hash function is indifferentiable from a random oracle in the random-function model up to $2^{c/2}$ queries to g [10]. If g is not a permutation, the sponge is called *transformative sponge* or T-sponge.

We denote a sponge-based hash function by $H : \mathbb{F}_2^* \to \mathbb{F}_2^{rj}$, the internal state space by $\mathcal{S} = \mathbb{F}_2^{r+c}$, and the update function by $g : \mathcal{S} \to \mathcal{S}$.

Collision Search in T-sponge. The following theorem shows that to get a significant speed-up in the collision search, the collision average κ should be at least of the same magnitude as 2^r.

Theorem 4. *Let g be a random mapping from $\mathbb{F}_2^{r+c}$ with* CPS $\mathfrak{C}$. *Let H be a* T*-sponge of capacity c and rate r updated with g. Then the probability of success of a brute-force collision attack on H is*

$$P = \frac{Q^2}{2^{c+1}}\cdot\left(1 + \frac{\kappa - 1}{2^r}\right)$$

where Q is the number of queries to g.

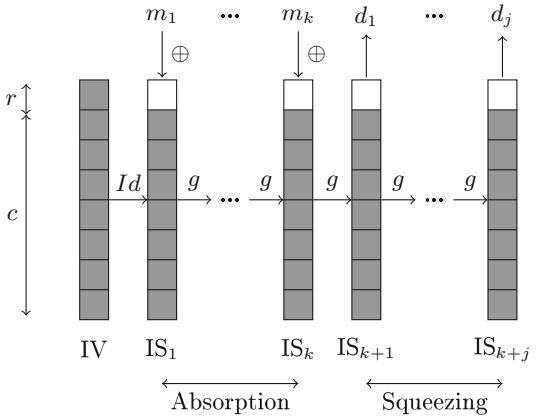

Fig. 3. Principle of a sponge construction. The message m is sliced into k blocks of r bits and "absorbed" during the first phase. Then, the j blocks of size r constituting the digest h are "squeezed".

The proof of Theorem 4 is in Appendix A.4. For a completely random mapping we have $\kappa = 1$, so that the theorem has the same form as in [7].

Nevertheless, since in practice the functions are not drawn at random from the set of all functions, it is of interest to be able to predict the effect of their properties over the security they provide. In particular, we see that a function with $\kappa > 1$ does not exactly provide $c/2$ bits of security against birthday attacks. Such functions can be found in real cryptographic primitives, see Sect. 4. However, we also immediately see that this effect is small since typical value of κ are of order of magnitude 1, 10 being already rather bad, while 2^r is at least in the hundreds. The designers of a T-sponge need not really worry about the number of collisions in the update function *if the rate is high enough.*

3.3 Improved Preimage Attack

Principle of the Iterated Preimage Attacks. Consider a set $\{g_k\}_{k\in\mathcal{K}}$ of random functions of $\mathcal{S}$ with CPS's $\{\mathfrak{C}_k\}_{k\in\mathcal{K}}$ and a fixed starting point $x_0 \in \mathcal{S}$ and let $\{k_1, ..., k_l\}$ be a set of l elements of $\mathcal{K}$. We call *keyed walk* the sequence

$$\big(x_1 = g_{k_1}(x_0),\ x_2 = g_{k_2}(x_1),\ ..., x_l = g_{k_l}(x_{l-1}) = d\big).$$

and it can for instance correspond to the successive values of the internal state of a T-sponge or of a Davies-Meyer based Merkle-Damgård hash function as we discuss in the next sections. Consider a keyed walk directed by a sequence $\{k_1, k_2, ..., \alpha, \alpha, ..., \alpha\}$ ending with z copies of the same symbol α. Then, intuitively, much entropy will have been lost because of the z iterations of g_α so that it should be easier to find a second sequence of keys leading to the same final value. This is formalized by the next theorem and a graphical representation of the phenomena we use is given in Fig. 4.

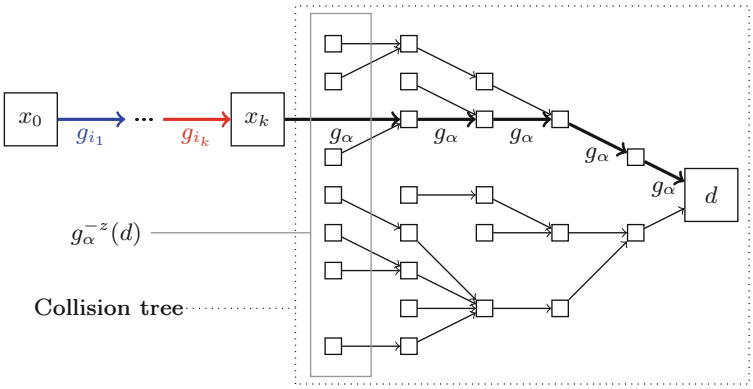

Fig. 4. The two targets of the iterated preimage attacks on d where d is in $g_\alpha^z(\mathcal{S})$ and $z = 5$. Different colors correspond to different function calls (Color figure online).

Theorem 5. *Let $\{g_k\}_{k\in\mathcal{K}}$ be a set of random mappings of $\mathcal{S}$ with* CPS*'s $\{\mathfrak{C}_k\}_{k\in\mathcal{K}}$ and consider a sequence $\{k_1, k_2, ..., \alpha, ..., \alpha\}$ of l keys from $\mathcal{K}$ ending with z identical keys α. Given the final value d of the corresponding keyed walk, the value of α and the number z, it is possible to find, for large enough z:*

1. *a keyed walk ending in d in time $|\mathcal{S}| \cdot 4/(\kappa z)$,*
2. *a keyed walk ending in d after precisely z calls to g_α in time $|\mathcal{S}| \cdot 2/\kappa$.*

where κ is the collision average of $\mathfrak{C}_\alpha$.

Proof. Let d be the final element in the walk. From the structure of the walk, we know that $d \in g_\alpha^z(\mathcal{S})$. Using Theorem 2, we know that there are $(\kappa/2)\cdot z$ elements in $g_\alpha^{-z}(d)$ and that the collision tree rooted at d contains $(\kappa/4) \cdot z^2$ elements. Therefore, such an element of $g_\alpha^{-z}(d)$ is found with probability $\kappa \cdot z/(|\mathcal{S}| \cdot 2)$ and an element in the collision tree with probability $\kappa \cdot z^2/(|\mathcal{S}| \cdot 4)$.

However, in both cases, we need to call g_α z times to know if the element we picked at random is mapped to d after exactly z iterations of g_α (first case) or at most z iterations (second case). Therefore, finding an element in the collision tree (first case) requires $|\mathcal{S}| \cdot z/(\kappa \cdot z^2/4) = |\mathcal{S}| \cdot 4/(\kappa z)$ calls to g_α and finding an element in $g_\alpha^{-z}(d)$ requires $|\mathcal{S}| \cdot z/(\kappa \cdot z/2) = |\mathcal{S}| \cdot 2/\kappa$. □

Note that these attacks can be generalized to the case where the end of the message is periodic instead of constant, i.e. if it ends with z copies of $(\alpha_1, \alpha_2, ..., \alpha_p)$. We simply need to replace g_α by $g' = g_{\alpha_1} \circ ... \circ g_{\alpha_p}$. The κ involved in the complexity computations is then that of g', i.e. $\sum_{i=1}^{p} \kappa_i$ where κ_i is the collision average of g_{α_i} (see Lemma 2 in Sect. 5). The constraint on z being large is only such that we can assume that z has the asymptotical behaviours described in Theorem 2.

Application to a T-sponge. Hashing a message with a T-sponge can be seen as performing a keyed walk where the keys are the message blocks of length r and the initial value x_0 is the all-zero vector. The function g_k is $g_k(x) = g(x \oplus k)$ where k is set to zero after its r first bits and g is the update function of the T-sponge. Clearly, g_k has the same CPS as g.

While the flat sponge claim provides a good description of the security offered by a sponge (be it a T-sponge or a P-sponge) against collision search and, for P-sponge, against second preimage search, there is a gap between the number of queries it allows and the best algorithm known for preimage search. In particular, there is to the best of our knowledge no algorithm allowing a preimage search with complexity below 2^c calls to the sponge function.[1] Theorem 6 bridges the gap between the $2^{c/2}$ bound of the flat sponge claim and the 2^c bound for preimage search by applying Theorem 5 to the T-sponge structure.

Theorem 6. *Let H be a T-sponge with update function g, and let κ be the collision average of g. Let M be a message such that its last z injections to the sponge are identical. Then a preimage to $H(M)$ can be found with complexity*

$$2^c \cdot 2^{r+2}/(\kappa z)$$

Such messages can be quite common. For instance, the last z calls of g can be blank calls for the sole purpose to slow down the hashing as suggested by NIST [12].[2] Such an attack can be prevented by setting an upper-bound of about $2^{r+2}/\kappa$ for the length of the message which in turn means that r has to be high in a T-sponge.

Similarity to the Herding Attack. This attack was introduced in [13] and is also refered to as the Nostradamus attack. In a herding attack, an attacker commits to a digest d and, when given a challenge P, has to find a suffix S such that $H(P||S) = d$. To achieve this she builds, during an offline phase, a so-called *diamond structure* which is essentially a binary collision tree with 2^ℓ nodes and rooted at d. The nodes of the tree contain the value of the internal state as well as the message block which needs to be absorbed to go to its child. During the online phase, she uses the diamond to find efficiently the suffix S: all she has to do is find a way to reach any of the $2^{\ell+1} - 1$ nodes in the diamond from the given starting point.

Application to a Merkle-Damgård Construction. When a block cipher is used in Davies-Meyer mode to build a Merkle-Damgård-based hash function,

[1] This case corresponds to the case where the attacker inverts the squeezing operations in time 2^c to retrieve the last internal state of the sponge before the squeezing and then uses a meet-in-the middle approach to find a valid message leading to this internal state in time $2^{c/2}$ (see [11]). Furthermore, this second step cannot be carried out in the case of a T-sponge since the update function cannot be inverted.

[2] Here, we consider that the message hashed is of a length equal to a multiple of r to begin with, so that the padding consisting in appending a one to the end of the message can be seen as part of the squeezing.

the successive chaining values $h_i \in \mathcal{S}$ are obtained from the previous one and the i-th message block: $h_i = E_{m_i}(h_{i-1}) \oplus h_{i-1} = g_{m_i}(h_{i-1})$. Because of the feedback of h_{i-1}, we do not expect g_k to be a permutation and, therefore, expect such a construction to be vulnerable to iterated preimage attacks. The padding used for Merkle-Damgård constructions usually takes into account the length of the message so that we need a message of the same length. Therefore, it is not enough to aim at an element in the collision tree, we need to find an element which is precisely in $g_\alpha^{-z}(d)$ so that a preimage search requires $2^{N+1}/\kappa$: if the CPS of g_k is such that $\kappa > 2$ then the iterated preimage attack is more efficient than brute-force. Furthermore, if there is an efficient way around the padding (e.g., with expandable messages [14]), the efficiency becomes $2^{N+2}/(\kappa z)$ where N is the size of the internal state of the hash function.

4 Preimage Attack on GLUON-64

4.1 The GLUON Family of Hash Functions

Introduced in [15], the GLUON family of hash functions consists of three members corresponding to different security levels: GLUON-64, GLUON-80 and GLUON-112. They have a T-sponge structure and have characteristics summarized in Table 2.

Table 2. Characteristics of the hash functions of the GLUON family.

name	rate r	capacity c	collision search	preimage search
GLUON-64	8	128	2^{64}	2^{128}
GLUON-80	16	160	2^{80}	2^{160}
GLUON-112	32	224	2^{112}	2^{224}

The function g used to update the internal state has the same structure in the three cases. It can be seen as a stream-cipher based on a Feedback with Carry Shift Register (FCSR). The concept of FCSR has evolved through time as the first stream-cipher [16] based on a component bearing this name got broken [17]. When we talk about FCSR in this paper, we refer to the last version of this structure, i.e. the one used in X-FCSR v2 [18] and, of course, GLUON. For example, the algebraic representation of the FCSR used by GLUON-64 is given in Fig. 5.

A FCSR is made of w cells of r bits. Each cell may be on the receiving end of a feedback. If the cell i receives no feed-backs, then its content at time $t+1$ is the content of the cell of index $i+1$ at time t. Consider now that the cell i receives a feedback. This cell contains an additional memory to store the value of the carry from one clock to the next. The content of the cell at time t is denoted m_i^t and that of the carry register c_i^t. Since it receives a feedback there are a cell of index j and a value of shift s (possibly equal to zero) such that:

$$\begin{aligned} m_i^{t+1} &= m_{i+1}^t + \left(m_j^t \ll s\right) + c_i^t \\ c_i^{t+1} &= m_{i+1}^t \cdot \left(m_j^t \ll s\right) + m_{i+1}^t \cdot c_i^t + \left(m_j^t \ll s\right) \cdot c_i^t \end{aligned}$$

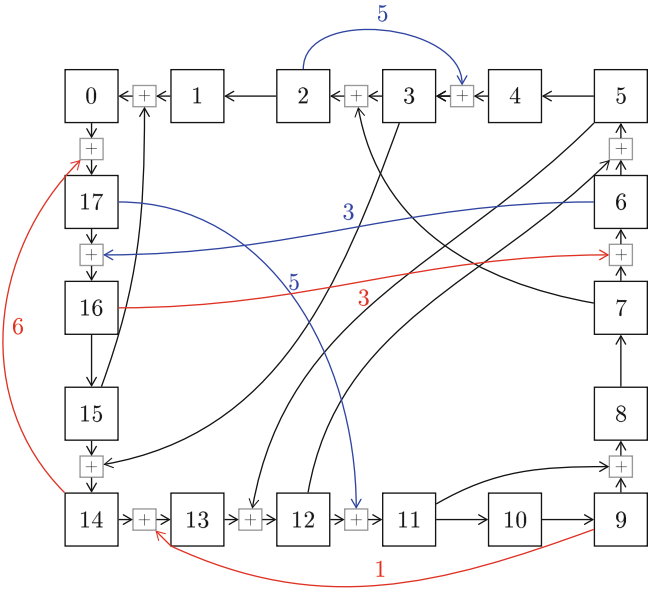

Fig. 5. Algebraic representation of the FCSR used in GLUON-64. Blue arrows correspond to feed-backs shifted to the right and red ones to the left. The value of the shift is given by the label of the arrow (Color figure online).

where "$\ll s$" is a C-style notation for the shift of the content of a cell by s bits to the left (or $|s|$ bits to the right if $s \leq 0$) and $+$ and $\cdot$ are respectively the bitwise addition and multiplication in $\mathbb{F}_2^r$.

The update function of every member of the GLUON family is made of three steps: padding of the input and loading in a FCSR (pad), clocking of the FCSR (ρ) and filtering Φ. We describe these steps separately.

Pad. The internal state of the sponge is of size $r(w-1)$, so that $r(w-1) = r+c$. The padding consists simply in appending a block of r bits all equal to one to the end of the internal state. The rw bits thus obtained are then loaded in a FCSR with an internal state made of w cells of size r. All the carries of the FCSR are set to zero. This operation is denoted $\text{pad} : \mathbb{F}_2^{r+c} \to \mathbb{F}_2^{rw} \times \mathbb{F}_2^{rw}$ as the output is made of the main register *and* the carry register of the FCSR.

ρ^{d+4}. The FCSR is clocked $d+4$ times. One clocking is denoted $\rho : \mathbb{F}_2^{rw} \times \mathbb{F}_2^{rw} \to \mathbb{F}_2^{rw} \times \mathbb{F}_2^{rw}$.

Φ. The output of g is extracted r bits by r bits using the following method: fixed words of the main register are rotated and then xor-ed to obtain r bits and then the FCSR is clocked. This operation is repeated $w-1$ times so as to have $r(w-1) = r+c$ bits of output. The action of clocking the FCSR $w-1$ times while filtering r bits each time is denoted $\Phi : \mathbb{F}_2^{rw} \times \mathbb{F}_2^{rw} \to \mathbb{F}_2^{r+c}$.

Overall, g is equal to $\Phi \circ \rho^{d+4} \circ \text{pad}$. The function pad is a bijection and we shall consider that the restriction of Φ over the set of the pairs main register/carry

register reachable after $d + 4$ calls to ρ starting in the image of pad is collision-free. The designers of GLUON claim:

> After a few iterations from an initial state, the automaton is in a periodic sequence of states of length P. The average number of required iterations to be in such a state is experimentally less than $\log_2(n)$, where n is the size of the main register [...] This leads to consider a function $[g]$ which is really close to a permutation from $\{0,1\}^b$ into itself because the surjective part of the construction is really limited once the function $[g]$ acts on the main cycle.

However, what happens during these first rounds, *before* the main cycle is reached? It is possible to encode the equation

$$(\rho^k \circ \mathrm{pad})(a + x) = (\rho^k \circ \mathrm{pad})(x) \tag{7}$$

for a fixed a into a CNF-formula solvable by a SAT-solver as long as k is not too big, say 10. The encoding is fairly straight-forward and we shall not go into the details for the sake of concision. Note that solving the equation $(\rho^k \circ \mathrm{pad})(x) = y$ using a SAT-solver is fast, meaning that it is possible to run a FCSR backward. However, we tried encoding the filtering so as to solve $(\Phi \circ \rho^k \circ \mathrm{pad})(x) = y$ but no SAT-solver was able to handle the corresponding CNF-formula — we killed the process after 1 week of running time for GLUON-112 (simplest filtering of the three), and for $k = 1$ instead of $k = d + 4 = 46$.

We solved (7) for many values of a and for $k = 10$ for each member of the GLUON family. While no non-zero solutions were found for any a for GLUON-80 and GLUON-112, it turns out that (7) has many solutions for GLUON-64. We used Algorithm 1 to find to which V_k any element $a \in \mathcal{S}$ belongs by enumerating all the values of x such that (7) holds. It works by solving (7) for x, thus (possibly) finding a solution x_1; then solving (7) with the constraint that the solution must be different from x_1, thus (possibly) finding x_2, etc. until no more solutions can be found. If there are k such $x \neq 0$, then a is in V_{k+1}.

4.2 CPS and Preimage Attack on GLUON-64

We ran Algorithm 1 for GLUON-64 on 24,000 different elements chosen uniformly at random in $\mathcal{S} = \mathbb{F}_2^{r+c}$. This allowed us to approximate the CPS of the update function. Our results are Fig. 6.

We deduce that $\mathfrak{c}_1 = 0.065$, $\ell_1 = 3.578$ and $\kappa = 6.982$ which are much worse than what one should expect from a random function, namely $\mathfrak{c}_1 = e^{-1} \approx 0.368$, $\ell_1 = 1/(1 - e^{-1}) \approx 1.582$ and $\kappa = 1$. This means that finding a preimage in a scheme equivalent to appending z identical words at the end of the message has a complexity of $2^{136+2}/(6.982 \cdot z) = 2^{128} \cdot (146.7/z)$. For $z > 147$, this is more efficient than classical brute-force. The complexities for some values of $z < 2^{(r+c)/2} = 2^{68}$ are given in Table 3 (Fig. 7).

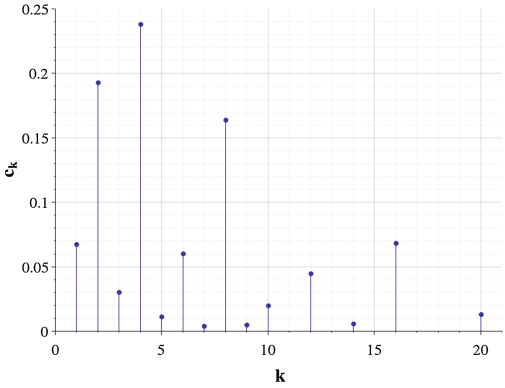

Fig. 6. Approximation of the CPS of the function used by GLUON-64 to update its internal state over 24,000 random elements of $\mathbb{F}_2^{136}$. Note that non-zero $\mathfrak{c}_k$ were observed well after $k = 20$.

Table 3. Complexity $\mathcal{C}$ of a preimage search for $d = H(m)$ where H is GLUON-64 and m is unkown except for the z identical bytes in its end.

z	$\log_2(\mathcal{C})$
147 b	127.99
500 b	126.23
1 kb	125.23
1 Mb	115.27
1 Gb	105.30
10^9 Gb	75.19

5 Other Properties of CPS

The CPS of a function is preserved by some transformation as shown in Lemma 1. The collision average of $g_1 \circ g_2$ has a simple expression given in Lemma 2.

Lemma 1. *Let g be a function with* CPS $\mathfrak{C}$*, $P : \mathcal{S} \to \mathcal{S}$ be a permutation and $J : \mathcal{S} \to \mathcal{S}$ be injective over $g(\mathcal{S})$. Then $g' = J \circ g \circ P$ has* CPS $\mathfrak{C}$ *as well.*

Proof. Since J is injective over $g(\mathcal{S})$, we have $g'(y) = g'(a)$ if and only if $g(P(y)) = g(P(a))$. Since the events "$g'(y) = g'(a)$ has k solutions" and "$g(x) = g(P(a))$ has k solutions" have the same probability, namely $\mathfrak{c}_k$, we see that g and g' have the same CPS. □

Lemma 2. *Let g_1 have collision average κ_1 and g_2 have collision average κ_2. Then $g_1 \circ g_2$ has collision average $\kappa_1 + \kappa_2$.*

Proof. Suppose that $(g_2 \circ g_1)(x) = (g_2 \circ g_1)(y)$ with $x \neq y$. There are two distinct ways this could happen:

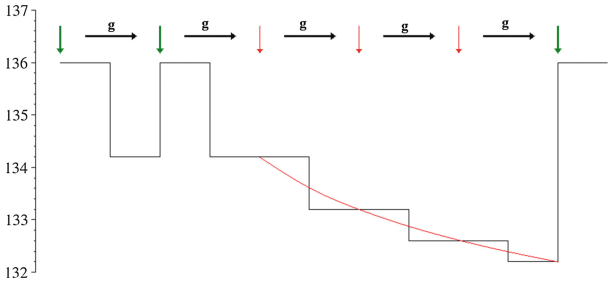

Fig. 7. Evolution of the number of possible values for the internal state of GLUON-64 when a message block is absorbed (thick vertical arrow) or when it absorbs a constant several times (thin vertical arrow) $z = 3$ times.

- if $g_1(x) = g_1(y)$, which happens in κ_1 cases on average,
- or if $g_1(x) \neq g_1(y)$ but $g_2\big(g_1(x)\big) = g_2\big(g_1(y)\big)$. There are on average κ_2/ℓ_1 solutions for $g_2(X) = g_2(Y)$ in $g_1(\mathcal{S})$ where ℓ_1 is the shrinking ratio of g_1. However, each of these solutions is the image of ℓ_1 elements of $\mathcal{S}$ by g_1.

Overall, the equation has $\kappa_1 + \ell_1 \cdot \kappa_2/\ell_1 = \kappa_1 + \kappa_2$ solutions, which proves the lemma. □

Note that Lemma 2 had to hold at least for $g_1 = g_2$ because otherwise we would have had a contradiction with the asymptotic behaviour of ℓ_i described in Theorem 2.

6 Conclusion

We introduced the notion of CPS and of the collision average κ, which is computed from the CPS. The collision average value determines the shrinking ratio and the collision tree size of an iterative construction, and hence directly affects their security, in particular preimage and collision resistance.

We have showed that the T-sponge is a fragile object when the rate parameter is small. For instance, preimages to long messages of specific structure become much easier to find. We gave specific recommendations for the designers of such constructions. Hopefully, our framework might become a useful tool in the design.

Finally, we demonstrated a practical application of our methodology. Aided with a SAT-solver, we found collisions for the GLUON-64 update function and then approximated its CPS and the collision average κ. We showed that for not so long messages of 1 Gigabyte a preimage can be found with complexity 2^{105} compared to the security claim of 2^{128}, and shortcut attacks are possible for messages of only a kilobyte long.

Acknowledgement. The authors thank the designers of the GLUON family of hash functions for providing a reference implementation, Alex Biryukov for very helpful discussions and the anonymous reviewers for their comments.

References

1. Babbage, S., Dodd, M.: The MICKEY stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 191–209. Springer, Heidelberg (2008)
2. Golić, J.D.: Cryptanalysis of alleged A5 stream cipher. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 239–255. Springer, Heidelberg (1997)
3. Bellare, M., Kohno, T.: Hash function balance and its impact on birthday attacks. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 401–418. Springer, Heidelberg (2004)
4. Flajolet, P., Odlyzko, A.M.: Random mapping statistics. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 329–354. Springer, Heidelberg (1990)
5. Hong, J., Kim, W.-H.: TMD-tradeoff and state entropy loss considerations of streamcipher MICKEY. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 169–182. Springer, Heidelberg (2005)
6. Röck, A.: Stream ciphers using a random update function: study of the entropy of the inner state. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 258–275. Springer, Heidelberg (2008)
7. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. In: ECRYPT hash Workshop, vol. 2007, Citeseer (2007)
8. Teo, S.G., Bartlett, H., Alhamdan, A., Simpson, L., Wong, K.K.H., Dawson, E.: State convergence in bit-based stream ciphers (2013)
9. Biryukov, A., Shamir, A., Wagner, D.: real time cryptanalysis of A5/1 on a PC. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 1–18. Springer, Heidelberg (2001)
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008)
11. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON family of lightweight hash functions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 222–239. Springer, Heidelberg (2011)
12. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak and the sha-3 standardization. Slides of a NIST presentation (2013). http://csrc.nist.gov/groups/ST/hash/sha-3/documents/Keccak-slides-at-NIST.pdf
13. Kelsey, J., Kohno, T.: Herding hash functions and the nostradamus attack. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 183–200. Springer, Heidelberg (2006)
14. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
15. Berger, T.P., D'Hayer, J., Marquet, K., Minier, M., Thomas, G.: The GLUON family: a lightweight hash function family based on FCSRs. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 306–323. Springer, Heidelberg (2012)
16. Arnault, F., Berger, T.P.: F-FCSR: design of a new class of stream ciphers. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 83–97. Springer, Heidelberg (2005)
17. Hell, M., Johansson, T.: Breaking the stream ciphers F-FCSR-H and F-FCSR-16 in real time. J. Cryptology **24**(3), 427–445 (2011)

18. Arnault, F., Berger, T., Lauradoux, C., Minier, M., Pousse, B.: A new approach for FCSRs. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 433–448. Springer, Heidelberg (2009)

A Proofs and Experiments

A.1 Proof of the Iterated Values of ℓ_i

Let us prove Theorem 1.

Proof. We shall look at the effect multiple iterations of g have over sets $\{x_0, ..., x_k\}$ where $g(x_j) = g(x_{j'})$ for all j, j'.

Let x_0 be in $g'(\mathcal{S})$ and such that there are k other elements $\{x_1, ..., x_k\}$ such that $g(x_0) = g(x_j)$, i.e. $x \in V_{k+1}$.

As every element in $\mathcal{S}$ is in $g'(\mathcal{S})$ with probability only $1/\ell'_1$, the number of elements colliding with x in $g'(\mathcal{S})$ follows a binomial distribution with parameters $(m, k, 1/\ell'_1)$ (we consider that the output of g' are uniformly distributed over $\mathcal{S}$ and that they are independent from one another). Thus, there are m elements colliding with $x \in g'(\mathcal{S})$ with probability $\binom{k}{m}(1/\ell'_1)^m(1-1/\ell'_1)^{k-m}$. Let C_{m+1} be the probability that $g(x_0 + x) = g(x_0)$ has m non-zero solutions in $g'(\mathcal{S})$ knowing that $x_0 \in g'(\mathcal{S})$:

$$C_{m+1} = \sum_{k=m}^{\infty} \mathfrak{c}_{k+1} \binom{k}{m} \Big(\frac{1}{\ell'_1}\Big)^m \Big(1 - \frac{1}{\ell'_1}\Big)^{k-m}. \tag{A.1}$$

Furthermore, we have:

$$\frac{\ell'_1}{\ell_1(g \circ g')} = \frac{|(g \circ g')(\mathcal{S})|}{|g'(\mathcal{S})|} = \sum_{m=1}^{\infty} \frac{C_m}{m},$$

and so:

$$\begin{aligned}\frac{\ell'_1}{\ell_1(g \circ g')} &= \sum_{m=1}^{\infty} \frac{1}{m} \sum_{k=m-1}^{\infty} \mathfrak{c}_{k+1} \binom{k}{m-1} \Big(\frac{1}{\ell'_1}\Big)^{m-1} \Big(1 - \frac{1}{\ell'_1}\Big)^{k-m+1} \\ &= \sum_{k=0}^{\infty} \sum_{m=0}^{k} \frac{\mathfrak{c}_{k+1}}{m+1} \binom{k}{m} \Big(\frac{1}{\ell'_1}\Big)^m \Big(1 - \frac{1}{\ell'_1}\Big)^{k-m}.\end{aligned}$$

This expression can be simplified because $\binom{k}{m}/(m+1) = \binom{k+1}{m+1}/(k+1)$, so that:

$$\begin{aligned}\frac{\ell'_1}{\ell_1(g \circ g')} &= \sum_{k=0}^{\infty} \frac{\mathfrak{c}_{k+1}}{k+1} \sum_{m=0}^{k} \binom{k+1}{m+1} \Big(\frac{1}{\ell'_1}\Big)^m \Big(1 - \frac{1}{\ell'_1}\Big)^{k-m} \\ &= \sum_{k=0}^{\infty} \frac{\mathfrak{c}_{k+1}}{k+1} \sum_{m=1}^{k+1} \binom{k+1}{m} \Big(\frac{1}{\ell'_1}\Big)^{m-1} \Big(1 - \frac{1}{\ell'_1}\Big)^{k-(m-1)} \\ &= \sum_{k=0}^{\infty} \frac{\mathfrak{c}_{k+1} \cdot \ell'_1}{k+1} \Big(\sum_{m=0}^{k+1} \binom{k+1}{m} \Big(\frac{1}{\ell'_1}\Big)^m \Big(1 - \frac{1}{\ell'_1}\Big)^{k+1-m} - \big(1 - \frac{1}{\ell'_1}\big)^{k+1} \Big).\end{aligned}$$

Note that $\sum_{m=0}^{k+1} \binom{k+1}{m} \left(\frac{1}{\ell_1'}\right)^m \left(1-\frac{1}{\ell_1'}\right)^{k+1-m} = 1$ (binomial theorem), so in the end we obtain:

$$\begin{aligned}\frac{1}{\ell_1(g\circ g')} &= \sum_{k=0}^{\infty} \frac{\mathfrak{c}_{k+1}}{k+1}\Big(1-\big(1-\frac{1}{\ell_1'}\big)^{k+1}\Big) \\ &= \sum_{k=1}^{\infty} \frac{\mathfrak{c}_k}{k}\Big(1-\big(1-\frac{1}{\ell_1'}\big)^{k}\Big) = \frac{1}{\ell_1} - \sum_{k=1}^{\infty} \frac{\mathfrak{c}_k}{k}\big(1-\frac{1}{\ell_1'}\big)^k\end{aligned}$$

which proves the Theorem. □

Note that this result is coherent with the one found by [4] in the case of random functions, i.e. when $\{\mathfrak{c}_k\}_{k\geq 1} = \{e^{-1}/\big((k-1)!\big)\}_{k\geq 1}$. Indeed, they prove that

$$\frac{1}{\ell_{i+1}} = 1 - \exp\big(\frac{-1}{\ell_i}\big),$$

which is the same as the one we found:

$$\begin{aligned}\frac{1}{\ell_{i+1}} &= \sum_{k=1}^{\infty} \frac{\mathfrak{c}_k}{k}\Big(1-\big(1-\frac{1}{\ell_i}\big)^{k}\Big) \\ &= e^{-1}\sum_{k=1}^{\infty} \frac{1}{k!}\Big(1-\big(1-\frac{1}{\ell_i}\big)^{k}\Big) \;=\; 1-\exp\big(\frac{-1}{\ell_i}\big).\end{aligned}$$

A.2 Proof of the Asymptotic Behaviors

Theorem 1 gives the recurrence relation ℓ_i satisfies so we can prove its asymptotic behavior.

Proof. Since ℓ_i is obviously increasing (the output space keeps shrinking and we keep $i < 2^{n/2}$ to remain away from the main cycle) we have, for large enough i:

$$\begin{aligned}\frac{1}{\ell_{i+1}} &= \sum_{k=1}^{\infty} \frac{\mathfrak{c}_k}{k}\Big(1-\big(1-\frac{k}{\ell_i}+\frac{k(k-1)}{2\cdot\ell_i^2}+\mathrm{o}(\ell_i^{-2})\big)\Big) \\ &= \frac{1}{\ell_i}\sum_{k=1}^{\infty} \mathfrak{c}_k\,\big(1-\frac{k-1}{2\cdot\ell_i}+\mathrm{o}(\ell_i^{-1})\big),\end{aligned}$$

so that we have:

$$\begin{aligned}\frac{\ell_i}{\ell_{i+1}} &= \sum_{k=1}^{\infty}\mathfrak{c}_k - \sum_{k=1}^{\infty}\frac{\mathfrak{c}_k\cdot(k-1)}{2\cdot\ell_i} + \mathrm{o}(\ell_i^{-1}) \\ &= 1-\frac{\kappa/2}{\ell_i}+\mathrm{o}(\ell_i^{-1})\end{aligned}$$

which in turns implies

$$\begin{aligned}\ell_{i+1} &= \frac{\ell_i}{1 - \frac{\kappa/2}{\ell_i} + \mathrm{o}(\ell_i^{-1})} \\ &= \ell_i + \frac{\kappa}{2} + \mathrm{o}(1),\end{aligned}$$

so that $\ell_i = \frac{\kappa}{2} \cdot i + \mathrm{o}(i)$. This observation concludes the proof of the behavior of ℓ_i.

Let us now look at ν_i. There are on average ℓ_w elements reaching $y \in g^w(\mathcal{S})$ in exactly w iterations. Since $g^i(\mathcal{S}) \subseteq g^w(\mathcal{S})$ for all $w \leq i$, we have that $y \in g^i(\mathcal{S})$ is reached, on average, by:

- ℓ_1 elements in exactly 1 iteration,
- ℓ_2 elements in exactly 2 iterations,

  ...
- ℓ_i elements in exactly i iterations.

Overall, there are on average $\sum_{w=1}^{i} \ell_w \approx \sum_{w=1}^{i} (\kappa/2)w \approx (\kappa/4)i^2$ elements reaching $y \in g^i(\mathcal{S})$ after at most i iterations of g. □

A.3 Experimental Justification of Conjecture 1

For every N between 12 and 17 included, we generated 100 functions with a given CPS and, for each of them, we picked 40 elements at random in $\mathbb{F}_2^N$ and computed $\lambda/2^{N/2}$ and $\mu/2^{N/2}$ for each of them (24,000 data points for each CPS). The average of these values for CPS's corresponding to different values of κ are given in Fig. 8. As we can see, both $\lambda/2^{N/2}$ and $\mu/2^{N/2}$ are almost equal to $\sqrt{\pi/(8\kappa)}$, which is equivalent to Conjecture 1 being correct.

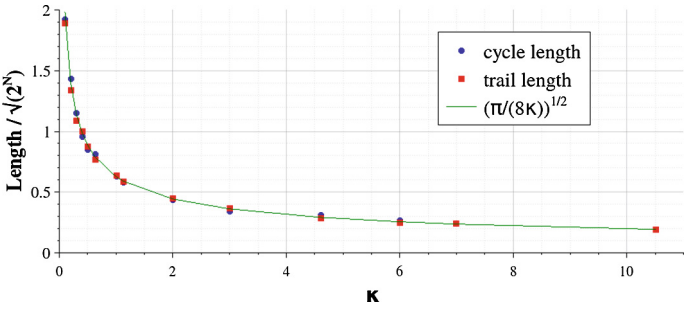

Fig. 8. Average value of $\lambda/2^{N/2}$ and $\mu/2^{N/2}$ for different κ.

A.4 Proof of the Effect of the CPS on a T-sponge

Proof. Our proof is a modified version of the one used by Bertoni et al. in the paper where they introduced the sponge construction [7]. In particular, we use the same terminology: we call the elements of $\mathbb{F}_2^{c+r}$ "nodes" and we partition the space according to the value of the bits in the capacity to obtain 2^c "super-nodes", each containing 2^r nodes. There is an oriented edge from node x to node y if and only if $y = g(x)$. Finding a collision in H boils down to finding two different paths in this graph starting from points in the super-node with capacity zero to an identical super-node.

We shall study the fan-in and the fan-out of these super-nodes, the fan-in of a super-node being the number of edges going to it and the fan-out the number of edges going out of it. In this framework, the fan-out of each super-node is 2^r. However, the number of edges going in each super-node is not constant. Consider some super-node Y made of nodes $y_1, ..., y_{2^r}$. Each y_i has a fan-in $F(y_i)$ so that the $F(y_i)$'s are independent and identically distributed random variables with the distribution

$$\mathbb{P}[F(y_i) = k] = \frac{\mathfrak{c}_k}{k} \text{ if } k \geq 1 \ , \ \mathbb{P}[F(y_i) = 0] = 1 - \frac{1}{\ell_1}$$

which has a mean equal to 1 and a variance equal to κ.

The value of the fan-in of the super-node Y is the sum of the fan-in's of its nodes:

$$F(Y) = \sum_{i=1}^{2^r} F(y_i).$$

We consider that 2^r is large enough to apply the central limit theorem so that $F(Y)$ is normally distributed with mean equal to 2^r and variance equal to $\kappa \cdot 2^r$.

Consider now the set $\mathcal{N}_k$ of all the super-nodes with fan-in equal to k; in other words the set of the super-nodes with exactly k preimages. It has a size equal to $|\mathcal{N}_k| = 2^c G(k)$ where

$$G(k) = \frac{1}{\sqrt{2\pi \cdot \kappa \cdot 2^r}} \cdot \exp\big(-\frac{1}{2} \cdot \frac{(k-2^r)^2}{\kappa \cdot 2^r}\big)$$

and the $\mathcal{N}_k$'s form a partition of the space of the super-nodes. Consider some node x_1: the probability that its image by g is in a super-node of $\mathcal{N}_k$ is

$$\mathbb{P}[g(x_1) \in \mathcal{N}_k] \ = \ \frac{k}{2^{c+r}} \cdot |\mathcal{N}_k| \ = \ \frac{k}{2^r} \cdot G(k)$$

Let V be the super-node $g(x_1)$ is in. The probability that a second element $x_2 \neq x_1$ is such that $g(x_2)$ is in the same super-node as $g(x_1)$ is the probability that x_2 is at the beginning of one of the $k-1$ edges going to V which are not the one starting at x_1. Therefore, the probability that $g(x_1)$ and $g(x_2)$ are in the same super-node V knowing that $V \in \mathcal{N}_k$ is

$$\mathbb{P}[g(x_1), g(x_2) \in V \mid V \in \mathcal{N}_k] = \frac{k-1}{2^{c+r}} \cdot \frac{k}{2^r} \cdot G(k)$$

so that the probability that $g(x_1)$ and $g(x_2)$ have the same capacity bits for x_1 and x_2 chosen uniformly at random is

$$\mathbb{P}[g(x_1), g(x_2) \in V] = \sum_{k=0}^{\infty} \frac{k(k-1)}{2^{c+2r}} \cdot G(k) \approx \frac{(2^r)^2 + \kappa \cdot 2^r - 2^r}{2^{c+2r}}.$$

Therefore, the probability of success of a collision search performed by absorbing Q messages at random until two internal states with the same capacity bits are observed is

$$\mathbb{P}[\text{success of collision search}] \approx \binom{Q}{2} \frac{2^{2r} + 2^r(\kappa - 1)}{2^{c+2r}} \approx \frac{Q^2}{2^{c+1}} \cdot \left(1 + \frac{\kappa - 1}{2^r}\right).$$

□

B Algorithms

Algorithm 1. Enumerating all the solutions of $g(a + \delta) = g(a)$.

$D =$ empty list
$b = 0$
while $b < rw - 1$ **do**
  $F = \mathrm{CNF}(\rho_k^1) + \mathrm{CNF}(\rho_k^2) + \mathrm{CNF}(\rho_k^1(x) = \rho_k^2(y))$
  $F = F + \mathrm{CNF}(x = a) + \mathrm{CNF}(x_b + y_b = 1)$
  for δ in D **do**
    $F = F + \mathrm{CNF}(x + y \neq \delta)$
  end for
  if SAT-solver concludes that F is satisfiable **then**
    Retrieve y from the assignment and append $x + y$ to D
  else
    $b = b + 1$ ▷ We move on only when this bit is exhaustively used
  end if
end while
Return D