

Permutation-Based Hashing with Stronger (Second) Preimage Resistance

Application to Hash-Based Signature Schemes

Siwei Sun^{1,2}, Shun Li¹, Zhiyu Zhang¹, Charlotte Lefevre³, Bart Mennink³,
Zhen Qin¹, Dengguo Feng^{2,1}

¹ School of Cryptology, University of Chinese Academy of Sciences, China
siweisun.isaac@gmail.com, {lishun,zhangzhiyu}@ucas.ac.cn

² State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

³ Digital Security Group, Radboud University, Nijmegen, The Netherlands
charlotte.lefevre@ru.nl, b.mennink@cs.ru.nl

Abstract. The sponge is a popular construction of hash function design. It operates with a b -bit permutation on a b -bit state, that is split into a c -bit inner part and an r -bit outer part. However, the security bounds of the sponge are most often dominated by the capacity c : If the length of the digest is n bits, the construction achieves $\min\{n/2, c/2\}$ -bit collision resistance and $\min\{n, c/2\}$ -bit second preimage resistance (and a slightly more complex but similar bound for preimage resistance). In certain settings, these bounds are too restrictive. For example, the recently announced Chinese call for a new generation of cryptographic algorithms expects hash functions with 1024-bit digests and 1024-bit preimage and second preimage resistance, rendering the classical sponge design basically unusable, except with an excessively large permutation. We present the SPONGE-DM construction to salvage the sponge in these settings. This construction differs from the sponge by evaluating the permutation during absorption in a Davies-Meyer mode. We also present SPONGE-EDM, that evaluates potentially round-reduced permutations during absorption in Encrypted Davies-Meyer mode, and SPONGE-EDM^c, that optimizes the amount of feed-forward data in this construction. We prove that these constructions generically achieve $\min\{n/2, c/2\}$ -bit collision resistance as the sponge does, but they achieve n -bit preimage resistance and $\min\{n, c - \log_2(\alpha)\}$ -bit second preimage resistance, where α is the maximum size of the first preimage in blocks. With such constructions, one could improve the security (resp., efficiency) without sacrificing the efficiency (resp., security) of hash-based signature schemes whose security relies solely on the (second) preimage resistance of the underlying hash functions. Also, one could use the 1600-bit Keccak permutation with capacity $c = 1088$ and rate $r = 512$ to achieve 512-bit collision resistance and 1024-bit preimage and second preimage resistance, without making extra permutation calls. To encourage further cryptanalysis, we propose two concrete families of instances of SPONGE-EDM (expected to be *weaker* than SPONGE-DM), using SHA3 and Ascon. Moreover, we concretely demonstrate the security and performance advantages of these instances in the context of hashing and hash-based signing.

Keywords: SHA3, Sponge, (Second) preimage resistance, Cryptographic permutations, Hash-based signatures

1 Introduction

The sponge construction of Bertoni et al. [8] is a popular approach for designing cryptographic hash functions and XoFs (eXtendable output Functions). The sponge construction is designed on top of a b -bit permutation f , and operates on a b -bit state, and typically consists of an absorbing phase and a squeezing phase. In the absorbing phase, the state is split into an inner part of c bits (the capacity) and an outer part of r bits (the rate). The input message $\text{msg} \in \mathbb{F}_2^{\text{bitLen}}$ is first injectively padded into an integral number of blocks of r bits. These blocks are absorbed one by one into the outer part of the state, interleaved with an evaluation of f on the entire state. Then, in the squeezing phase, the state is split into an inner part of c' bits and an outer part of r' bits, and a digest is formed by extracting r' bits from the state, again interleaved with an evaluation of f on the entire state, until the required amount of output bits, n , is retrieved. The sponge construction with padding algorithm Pad and n -bit digests denoted by

$$\text{SPONGE}[f : \mathbb{F}_2^b \rightarrow \mathbb{F}_2^b, \text{Pad}, r, r', n]$$

is depicted in Figure 1. Typical sponge functions, in fact, have $c = c'$ and $r = r'$, such as SHA3 [32] and Ascon [16, 38], whereas some take a slightly larger squeezing rate, such as PHOTON [21].

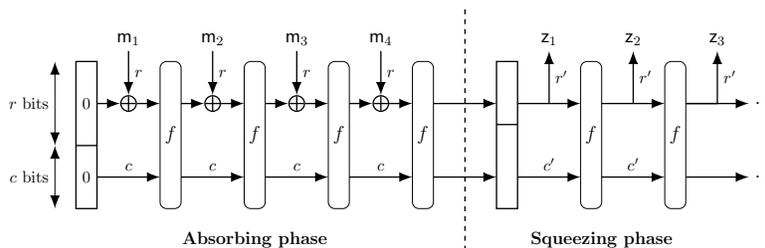


Fig. 1: The sponge construction.

Bertoni et al. [9] proved that, if f is a random permutation, the sponge (in case $c = c'$ and $r = r'$) is indistinguishable from a random oracle [29, 14] as long as the number of evaluations of the permutation does not exceed $2^{c/2}$. Naito and Ohta [30] proved that a similar bound holds in case $r' = r + c/2 - \log_2(c)$ (and with an initial absorption of $r + c/2$ bits). In other words, the sponge behaves like a random oracle up to $2^{c/2}$ queries, and this also implies generic security of the sponge against the classical collision, preimage, and second preimage attacks [1, Appendix A]. Stated differently, the sponge construction truncated to

Table 1: The security bounds of SPONGE and our new constructions SPONGE-DM, SPONGE-EDM, and SPONGE-EDM^c. All parameters are expressed in bits. The bounds assume that $c' \geq c/2 + \log_2(r)$, and α denotes the maximum message length in blocks.

Mode	Output size	Input rate	Output rate	Security			
				Indiff.	Collision	Preimage	2nd preimage
SPONGE	n	r	r'	$\frac{c}{2}$	$\min\{\frac{n}{2}, \frac{c}{2}\}$	$\min\{n, \max\{n - r', \frac{c}{2}\}\}$	$\min\{n, \frac{c}{2}\}$
SPONGE-DM	n	r	r'	$\frac{c}{2}$	$\min\{\frac{n}{2}, \frac{c}{2}\}$	n	$\min\{n, c - \log_2(\alpha)\}$
SPONGE-EDM	n	r	r'	$\frac{c}{2}$	$\min\{\frac{n}{2}, \frac{c}{2}\}$	n	$\min\{n, c - \log_2(\alpha)\}$
SPONGE-EDM ^c	n	r	r'	$\frac{c}{2}$	$\min\{\frac{n}{2}, \frac{c}{2}\}$	n	$\min\{n, c - \log_2(\alpha)\}$

an output of n bits is collision secure up to complexity $\min\{2^{n/2}, 2^{c/2}\}$ and preimage and second preimage secure up to complexity $\min\{2^n, 2^{c/2}\}$. Lefevre and Mennink [26] derived a dedicated security bound for preimage resistance, up to complexity $\min\{2^n, \max\{2^{c/2}, 2^{n-r'}\}\}$. These bounds are also summarized in Table 1, and we refer the readers to Section 3.2 for a more technical treatment of the state-of-the-art security bounds on the sponge.

Above-mentioned bounds on the sponge (as outlined in Table 1) are all tight: they match the generic attacks specified in the original introduction of sponge functions [8, Section 5]. For example, for collision resistance, the attack is quite intuitive: first, there is a generic attack that does not take the internal properties of the sponge into account and that succeeds in complexity approximately $2^{n/2}$. Otherwise, one can vary the first message block, and get around $2^{c/2}$ different states after the first evaluation of f . With high probability, there exist two evaluations with the same inner part, and the difference in the outer part can be annihilated with the next message block. For preimage and second preimage resistance, the attacks are slightly more involved, and they rely on the invertibility of f : we revisit these attacks in more detail in Section 3.3, and in this section we also describe a *new* second preimage attack that particularly works if the length of the message is encoded in the padding.

1.1 Instantiating the Sponge

Existing hash functions based on the sponge have their security parameters based on these bounds. For example, the algorithms in the SHA3 family are sponge constructions instantiated with a 24-round iterative permutation $\text{Keccak-}p[1600, 24] : \mathbb{F}_2^{1600} \rightarrow \mathbb{F}_2^{1600}$ [32]. Denoting by $\text{Keccak}[c, n]$ the SPONGE construction instantiated with $\text{Keccak-}p[1600, 24]$ and capacity $c = c'$ and output n , then the SHA3 family consists of 4 hash functions

$$\begin{aligned}
 \text{SHA3-224}(\text{msg}) &= \text{Keccak}[448, 224](\text{msg} \parallel 01), \\
 \text{SHA3-256}(\text{msg}) &= \text{Keccak}[512, 256](\text{msg} \parallel 01), \\
 \text{SHA3-384}(\text{msg}) &= \text{Keccak}[768, 384](\text{msg} \parallel 01), \\
 \text{SHA3-512}(\text{msg}) &= \text{Keccak}[1024, 512](\text{msg} \parallel 01),
 \end{aligned} \tag{1}$$

and two XoFs

$$\begin{aligned} \text{SHAKE128}(\text{msg}, n) &= \text{Keccak}[256, n](\text{msg} \parallel 1111), \\ \text{SHAKE256}(\text{msg}, n) &= \text{Keccak}[512, n](\text{msg} \parallel 1111), \end{aligned} \tag{2}$$

and in each of these, the capacity is chosen so as to get 112, 128, 192, 256, 128, or 256 bits of collision resistance.

Likewise, the algorithms in the `Ascon` family are sponge constructions instantiated with a 12-round iterative permutation $\text{Ascon-}p[320, 12] : \mathbb{F}_2^{320} \rightarrow \mathbb{F}_2^{320}$ [38]. Let $\text{Ascon}[c, n]$ denote the SPONGE construction instantiated with $\text{Ascon-}p[320, 12]$ and capacity $c = c'$ and output n , then, according to the NIST initial public draft [38], we have

$$\begin{aligned} \text{Ascon-Hash256}(\text{msg}) &= \text{Ascon}[256, 256](\text{msg}), \\ \text{Ascon-XOF128}(\text{msg}, n) &= \text{Ascon}[256, n](\text{msg}), \\ \text{Ascon-CXOF128}(\text{msg}, n) &= \text{Ascon}[256, n](\text{msg}) \end{aligned} \tag{3}$$

(where domain separation is applied using the initial value). Also here, the capacity is selected so as to get 128 bits of security, provided d is large enough. See also the systematization of knowledge of Lefevre and Mennink on the generic security of `Ascon` [27, Section 8].

1.2 The Limits on (Second) Preimage Resistance

Despite the tunability of the sponge, for preimage and second preimage resistance, c is often the limiting factor. Indeed, consider the case of $\text{SHAKE128}(\text{msg}, d)$ of (2). Assuming $n \geq 256$, it generically achieves 128-bit collision resistance but also only 128-bit second preimage resistance (and possibly slightly larger preimage resistance depending on d). Likewise, `Ascon-Hash256` achieves 128-bit collision resistance as well as 128-bit second preimage resistance. (The NIST initial public draft [38] also claims only 128-bit preimage resistance, but according to the tight bounds of Table 1 it achieves 192-bit preimage resistance [27, Section 8].)

In order to achieve higher preimage and second preimage resistance, one typically has to increase the capacity and typically also the permutation size, and this has a negative impact on the performance. We stress that the quest for improved preimage and second preimage resistance is not just a theoretical exercise: to the contrary, in certain cases, one actually *wants* better (second) preimage resistance. For example, hash functions with stronger (second) preimage resistance can be applied to enhance the security of hash-based post-quantum signature schemes whose security relies solely on the security of the underlying hash function against (second) preimage attacks. Moreover, in March 2025, the National Institute of Commercial Cryptography Standards (NICCS) of China released the *Announcement on Launching the Next-generation Commercial Cryptographic Algorithms Program*, where it calls for hash functions with 1024-bit digest and 1024-bit security against preimage and second preimage attacks [31]. In sponge

terms, this restriction implies a capacity of at least $1024 \cdot 2 = 2048$ bits, making the 1600-bit Keccak- $p[1600, 24]$ permutation a priori unsuitable. This leads to the natural question of whether we can achieve λ -bit security against preimage and second preimage attacks with only about λ -bit capacity.

1.3 SPONGE-DM, SPONGE-EDM, and SPONGE-EDM^c

In this work, we propose SPONGE-DM, SPONGE-EDM, and SPONGE-EDM^c: these are sponge functions as described above, but where the permutations during absorption are replaced with a Davies-Meyer [28] evaluation $x \mapsto f(x) \oplus x$ for the permutation f , an Encrypted Davies-Meyer [13] evaluation $x \mapsto h(g(x) \oplus x)$ for two permutations g, h , and a feed-forward-optimized Encrypted Davies-Meyer evaluation, respectively. The three constructions are described in more detail in Section 4. The constructions are loosely inspired by Foekens who, in turn inspired by the fact that (second) preimage attacks require the inverse of the permutation, investigated whether improved (second) preimage security could be achieved with a random function [18]. However, our approach differs in two aspects. Firstly, we do not want to rely on a random function, and secondly, we only manipulate the primitive during absorption.

In Section 5, we prove the security bounds of these three constructions. First, we prove that SPONGE-DM, SPONGE-EDM, and SPONGE-EDM^c achieve the same levels of indifferenciability and collision resistance as the SPONGE. This result should not come as a surprise, as the generic attacks do not rely on the invertibility of the permutation. However, next we prove that SPONGE-DM, SPONGE-EDM, and SPONGE-EDM^c achieve (second) preimage resistance *beyond the bound proven for the sponge*. In detail, we prove that they achieve preimage resistance up to complexity 2^n (i.e., independent of the capacity) and second preimage resistance up to $\min\{2^n, 2^c/\alpha\}$, where α is the length of the first preimage in blocks. See also Table 1. These bounds are actually tight: as we explain in Section 5, there are matching attacks for these bounds.

1.4 Instantiation and Application

These are powerful bounds! For example, with just a 1600-bit permutation, we can use our sponge-like construction SPONGE-EDM to achieve 512-bit collision resistance and 1024-bit preimage and second resistance by taking $c = 1088$ and $r = 512$, where we assume that the maximum size of a message is at most around 2^{64} blocks. Note that this construction does not make extra permutation calls compared to the sponge: it only makes one extra exclusive-or operation and it maintains a double state. Therewith, the Keccak- $p[1600, 24]$ permutation could be easily used in a sponge-like construction to meet the conditions of the Chinese *Announcement on Launching the Next-generation Commercial Cryptographic Algorithms Program* [31].

We remark that SPONGE-EDM is generically more expensive than SPONGE-DM as it makes twice as many permutation calls during absorption. However, this

scheme is mostly introduced to support constructions with round-reduced permutations during absorption. For example, taking the 24-round iterative permutation $\text{Keccak-}p[1600, 24]$ as f , one can take g to be its first half and h its second half. This construction is particularly interesting as stepping stone to cryptanalyze SPONGE-DM when instantiated with a concrete permutation. As a matter of fact, to encourage further cryptanalysis, we present two families of SPONGE-EDM (the weaker mode) instances derived from SHA3 [32] and Ascon [38] in Section 6. In the first family, let

$$\text{Keccak-EDM}[c, n] = \text{SPONGE-EDM}[g, h, \text{Pad}, 1600 - c, 1600 - c, n],$$

where g and h be the first and last 12 rounds of $\text{Keccak-}p[1600, 24]$, respectively. We denote the function mapping $x \in \mathbb{F}_2^b$ to $h(g(x) \oplus x)$ by

$$\text{Keccak-}p[1600, 24]\text{-EDM}(12, 12).$$

In the second family, let

$$\text{Ascon-EDM}[c, n] = \text{SPONGE-EDM}[g, h, \text{Pad}, 320 - c, 320 - c, n], \quad (4)$$

where g and h be the first and last 6 rounds of $\text{Ascon-}p[320, 12]$ respectively. Also, we denote the function mapping $x \in \mathbb{F}_2^b$ to $h(g(x) \oplus x)$ by

$$\text{Ascon-}p[320, 12]\text{-EDM}(6, 6).$$

We expect $\text{Keccak-EDM}[512 + 64, 512]$ to enjoy comparative concrete security levels with $\text{SHA3-512} = \text{Keccak}[1024, 512]$ in regard to collision, preimage, and second preimage attacks when the length of the longest message allowed to be processed is limited to $512 \cdot 2^{64}$ bits. This means that we can achieve the same security level with smaller capacity and larger rate, which increases the efficiency of the hash function (especially for long messages). Moreover, we expect $\text{Keccak-EDM}[1024 + 64, 1024]$ to reach about 1024-bit security against preimage and second preimage attacks when the length of the longest message allowed to be processed is limited to $512 \cdot 2^{64}$ bits. Similarly, we expect that $\text{Ascon-EDM}[256, 256]$ offers a comparative level of concrete security to Ascon-Hash256 in terms of collision resistance. However, it is expected to provide 256-bit and 192-bit security against preimage and second preimage attacks respectively when the maximum message length is restricted to $64 \cdot 2^{64}$ bits. In short, a higher level of security could be offered by the SPONGE-EDM mode while maintaining comparable efficiency as the original hash function. This has some implications on the security and efficiency of hash-based post-quantum signature schemes whose security solely rely on the (second) preimage resistance of the underlying hash functions. By leveraging the Ascon-EDM instances, we can remedy the flaws of the hash-based signature scheme Ascon-Sign [37] submitted to the NIST PQC project in achieving 192-bit security, and improve the efficiency of the 128-bit secure version of the signature algorithm in Ascon-Sign .

1.5 Outline

We start with basic notation and our security model in Section 2. We recall the sponge construction, its state-of-the-art security bounds, and its preimage and second preimage attacks in Section 3. In this section, and more specifically in Section 3.3.3, we present our new long message second preimage attack on the sponge. Then, we present our new constructions SPONGE-DM, SPONGE-EDM, and SPONGE-EDM^c in Section 4, and derive their security bounds in Section 5. Then, we present example instantiations of these constructions using SHA3 and Ascon in Section 6. In this section, we also demonstrate how our results are particularly useful for hash-based signature schemes.

2 Security Model

For $n \in \mathbb{N}$, we denote by \mathbb{F}_2^n the set of n -bit strings and \mathbb{F}_2^* the set of arbitrarily long strings. The set of arbitrarily long strings whose length is a multiple of n is given by $(\mathbb{F}_2^n)^* = \bigcup_{i \in \mathbb{N}} \mathbb{F}_2^{i \cdot n}$. For an $x \in \mathbb{F}_2^*$, we denote by $\|x\|_{\mathbb{F}_2}$ its length in bits. For $a < \|x\|_{\mathbb{F}_2}$, the leftmost a bits of x are denoted $\lceil x \rceil^{(a)}$ and its rightmost a bits $\lfloor x \rfloor^{(a)}$. For $x \in \mathbb{N}$ such that $x < 2^n$, we denote by $\langle x \rangle_n$ the encoding of x as an n -bit string. We denote by ϵ the empty string.

We denote by $x \xleftarrow{\$} \mathcal{X}$ the uniform random selection of an element x from a finite set \mathcal{X} . Let $\text{perm}(b)$ denote the set of all b -bit permutations. For a bijection f , we denote by f^\pm that the user has forward and inverse access to the function.

An adversary \mathcal{A} is an algorithm, that is given query access to a set of oracles \mathcal{O} , denoted as $\mathcal{A}[\mathcal{O}]$. It can make a certain amount of queries to the oracles, and in the end output something. In our security models, this will either be a decision bit, a collision, or a first or second preimage for the hash function.

Throughout, we will consider security of a XoF \mathcal{H} based on a permutation $f \in \text{perm}(n)$. When we consider generic security, we assume f to be a random permutation, $f \xleftarrow{\$} \text{perm}(n)$, and we typically consider the quality at which \mathcal{H} behaves like a random oracle \mathcal{R} , which is a function that outputs a random string of arbitrary length for each new input [3]. We will consider indistinguishability as well as the classical notions of collision resistance, preimage resistance, and second preimage resistance. These notions are described in Sections 2.1 and 2.2, respectively. These descriptions are adopted from those of Lefevre and Mennink [27, Section 8] on Ascon-XoF.

2.1 Indistinguishability

We consider indistinguishability of \mathcal{H} based on random permutation $f \xleftarrow{\$} \text{perm}(n)$. Intuitively, a XoF is indistinguishable [29, 14] from a random oracle \mathcal{R} if there exists a simulator \mathcal{S} with oracle access to \mathcal{R} such that $(\mathcal{H}[f], f^\pm)$ is hard to distinguish from $(\mathcal{R}, \mathcal{S}[\mathcal{R}]^\pm)$.

Definition 1. Consider a XoF \mathcal{H} . Let \mathcal{R} be a random oracle and $f \xleftarrow{\$} \text{perm}(b)$. Let $\mathcal{S}[\mathcal{R}]$ be a two-sided simulator. The indistinguishability of \mathcal{H} with respect to simulator $\mathcal{S}^{\mathcal{R}}$ against an adversary \mathcal{A} is defined as

$$\text{Adv}_{\mathcal{H}, \mathcal{S}}^{\text{indiff}}(\mathcal{A}) = \left| \Pr(\mathcal{A}[\mathcal{H}[f], f^{\pm}] \rightarrow 1) - \Pr(\mathcal{A}[\mathcal{R}, \mathcal{S}[\mathcal{R}]^{\pm}] \rightarrow 1) \right|.$$

In indistinguishability, the adversarial resources are counted in the number of accumulated permutation evaluations q that would be made in the left world.

2.2 Collision, Preimage, and Second Preimage Resistance

We consider collision, preimage, and second preimage resistance of \mathcal{H} based on random permutation $f \xleftarrow{\$} \text{perm}(n)$, where we take collision resistance, (everywhere) preimage resistance, and (everywhere) second preimage of Rogaway and Shrimpton [36]. Here, we require that the *minimal* output size is fixed to some value n , i.e., $\ell \geq n$. In addition, for second preimage resistance, there is an additional parameter κ that specifies the maximal length of the first preimage.

Definition 2. Consider a XoF \mathcal{H} . Let $f \xleftarrow{\$} \text{perm}(b)$. Let $\kappa, n \in \mathbb{N}$.

– The collision resistance of \mathcal{H} against an adversary \mathcal{A} is defined as

$$\text{Adv}_{\mathcal{H}}^{\text{col}[n]}(\mathcal{A}) = \Pr\left(\mathcal{A}[f^{\pm}] \rightarrow (\text{msg}, \text{msg}') : \text{msg} \neq \text{msg}' \text{ and } \mathcal{H}[f](\text{msg}, n) = \mathcal{H}[f](\text{msg}', n)\right);$$

– The (everywhere) preimage resistance of \mathcal{H} against an adversary \mathcal{A} is defined as

$$\text{Adv}_{\mathcal{H}}^{\text{pre}[n]}(\mathcal{A}) = \max_{z \in \mathbb{F}_2^n} \Pr(\mathcal{A}[f^{\pm}](z) \rightarrow \text{msg} : \mathcal{H}[f](\text{msg}, n) = z);$$

– The (everywhere) second preimage resistance of \mathcal{H} against an adversary \mathcal{A} is defined as

$$\text{Adv}_{\mathcal{H}}^{\text{sec}[\kappa, n]}(\mathcal{A}) = \max_{\text{msg} \in \mathbb{F}_2^{\leq \kappa}} \Pr\left(\mathcal{A}[f^{\pm}](\text{msg}) \rightarrow \text{msg}' : \text{msg} \neq \text{msg}' \text{ and } \mathcal{H}[f](\text{msg}, n) = \mathcal{H}[f](\text{msg}', n)\right).$$

In these security notions, the adversarial resources are counted in the number of permutation evaluations q .

We remark that indistinguishability of a XoF from a random oracle implies its collision, preimage, and second preimage resistance up to the bounds that a random oracle would achieve [1, Appendix A] (see also [27, Section 8]). However, dedicated analysis may (and, in the case of our constructions, will) yield tighter bounds.

3 Sponge Construction and Its Security

Before diving into our constructions (in Section 4), we first recall the sponge construction in Section 3.1. We recap its generic security in Section 3.2. In Section 3.3, we recall the existing preimage and second preimage attacks, and derive a new long message second preimage attack.

Algorithm 1: SPONGE[f, Pad, r, r', n]

Input: $\text{msg} \in \mathbb{F}_2^*$
Output: n -bit digest z

- 1 $\mathbf{m} \leftarrow \text{msg} \parallel \text{Pad}(r, \|\text{msg}\|_{\mathbb{F}_2})$
- 2 $\alpha \leftarrow \|\mathbf{m}\|_{\mathbb{F}_2}/r$
- 3 $c \leftarrow b - r$
- 4 Let $\mathbf{m} = (m_1, \dots, m_\alpha) \in (\mathbb{F}_2^r)^k$
- 5 $S \leftarrow 0^b$
- 6 **for** i from 1 to α **do**
- 7 $S \leftarrow f(S \oplus (m_i \parallel 0^c))$
- 8 Let z be the empty string
- 9 **while** $\|z\|_{\mathbb{F}_2} < n$ **do**
- 10 $z \leftarrow z \parallel \text{Trunc}_{r'}(S)$
- 11 $S \leftarrow f(S)$
- 12 **return** $\text{Trunc}_n(z)$

3.1 Construction

Let $b, c, r, c', r' \in \mathbb{N}$ such that $b = r + c = r' + c'$. Let $f : \mathbb{F}_2^b \rightarrow \mathbb{F}_2^b$ be a b -bit permutation. The SPONGE function takes as input a message $\text{msg} \in \mathbb{F}_2^*$ and a requested output length $n \in \mathbb{N}$, and outputs a digest $z \in \mathbb{F}_2^*$ of size n bits. It is defined as follows:

$$\text{SPONGE}[f, \text{Pad}, r, r', n](\text{msg}) = z.$$

Internally, the function always first injectively pads the message msg by concatenating it with a string $\text{Pad}(r, \|\text{msg}\|_{\mathbb{F}_2})$, in such a way that $\mathbf{m} = \text{msg} \parallel \text{Pad}(r, \|\text{msg}\|_{\mathbb{F}_2})$ is of length a multiple of r bits. We typically write $\alpha = \|\mathbf{m}\|_{\mathbb{F}_2}/r$, and we denote $\beta = \lceil n/r' \rceil$ for the number of r' -bit blocks of z . The Sponge construction is depicted in Figure 1 and given in Algorithm 1.

The type of padding string, $\text{Pad}(r, \|\text{msg}\|_{\mathbb{F}_2})$, is dependent on the construction. For example, SHA3 [32] defines

$$\text{SHA3-Pad}(r, \text{bitLen}) = 1 \parallel 0^{(r - \text{bitLen} - 2) \bmod r} \parallel 1,$$

whereas Ascon [38] defines

$$\text{Ascon-Pad}(r, \text{bitLen}) = 1 \parallel 0^{(r - \text{bitLen} - 1) \bmod r}.$$

One may also encode the length of the original message into the padding, by concatenating $\langle \text{bitLen} \rangle_r$.

3.2 Generic Security

The sponge construction is proven indifferentiable up to bound

$$\text{Adv}_{\text{SPONGE}}^{\text{indiff}}(\mathcal{A}) \leq \frac{q(q+1)}{2^c},$$

as proven by Bertoni et al. [9]. Naito and Ohta proved a similar bound if the initial absorption is done at $r + c/2$ bits and squeezing is performed at $r + c/2 - \log_2(c)$ bits [30]. By implication [1, Appendix A], this bound implies collision, preimage, and second preimage resistance:

$$\begin{aligned} \text{Adv}_{\text{SPONGE}}^{\text{col}[n]}(\mathcal{A}) &\leq \frac{q(q+1)}{2^c} + \frac{(q-1)q}{2^{n+1}}, \\ \text{Adv}_{\text{SPONGE}}^{\text{pre}[n]}(\mathcal{A}) &\leq \frac{q(q+1)}{2^c} + \frac{q}{2^n}, \\ \text{Adv}_{\text{SPONGE}}^{\text{sec}[\kappa, n]}(\mathcal{A}) &\leq \frac{q(q+1)}{2^c} + \frac{q}{2^n}. \end{aligned}$$

(Here, strictly seen, the q in the first term refers to permutation calls but the q in the second term to hash function calls. However, for the sake of the storyline, this slight abuse of notation is not important.) An improved bound on preimage resistance was derived by Lefevre and Mennink [26]:

$$\text{Adv}_{\text{SPONGE}}^{\text{pre}[n]}(\mathcal{A}) \leq \min \left\{ \frac{4^{\lceil n/r' \rceil} q}{2^{n-r'}}, \frac{q(q+1)}{2^c} \right\} + \frac{4q}{2^n}.$$

These bounds are, in fact, tight. Bertoni et al. [8] already derived attacks matching exactly these bounds. The collision attack aims to get an inner collision during absorption by making forward evaluations of f , and this attack also means that the indistinguishability bound is tight. For second preimage and preimage resistance, the attacker makes forward and inverse evaluations of f to meet in the middle.

3.3 (Second) Preimage Attacks

Before describing the SPONGE-DM, SPONGE-EDM, and SPONGE-EDM^c constructions, we first recall the generic preimage and second preimage attacks on the sponge construction instantiated with a permutation. These two attacks are taken from the original specification of the sponge of Bertoni et al. [8].

In addition, we present a *new* long message second preimage attack on the sponge construction, which works even if the length of the processed message is encoded in the padding. We remark that this long-message attack is not very relevant for the SHA3 and Ascon families, since only $\frac{c}{2}$ -bit security is claimed with respect to second preimage attacks. However, the long message second preimage attack is very important in our case, since we intend to overcome the $\frac{c}{2}$ -bit bound with the SPONGE-DM, SPONGE-EDM, and SPONGE-EDM^c constructions in Section 4.

3.3.1 Preimage Attack For a given digest $z_1 \parallel \dots \parallel z_{l-1} \parallel z_l \in \mathbb{F}_2^{r'} \times \dots \times \mathbb{F}_2^{r'} \times \mathbb{F}_2^{n-(l-1)r'}$, when $n \leq \max\{n - r', \frac{c}{2}\}$, i.e., $n \leq \frac{c}{2}$, we can apply the generic preimage attack on the ideal hash function with n -bit output to find a preimage in time complexity $\mathcal{O}(2^n)$. When $n > \max\{n - r', \frac{c}{2}\}$, i.e., $n > \frac{c}{2}$, a more efficient attack is shown in Figure 2.

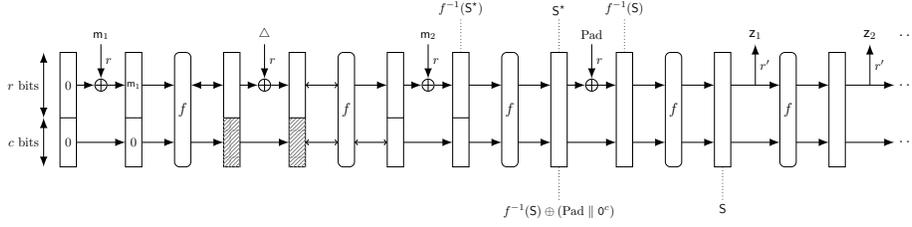


Fig. 2: Generic preimage attack on the sponge construction instantiated with a permutation.

First, we try to find an intermediate $b = (r+c)$ -bit state S , such that $[S]^{(r')} = z_1$ and

$$\begin{cases} [f(S)]^{(r')} = z_2, \\ [f \circ f(S)]^{(r')} = z_3, \\ \dots \dots \\ \underbrace{[f \circ f \circ \dots \circ f(S)]^{(n-(l-1)r')}}_{l-1} = z_l. \end{cases} \quad (5)$$

For a randomly chosen S with $[S]^{(r')} = z_1$, the probability that (5) is fulfilled can be estimated as

$$\left(\frac{1}{2^{r'}}\right)^{l-2} \cdot \frac{1}{2^{n-(l-1)r'}} = \frac{1}{2^{n-r'}}.$$

Therefore, we can identify such a state S with time complexity $2^{n-r'}$. Note that when $n - r' > b - r' = c'$, the time complexity can be reduced to $2^{c'}$, since there are r' bits of $S \in \mathbb{F}_2^b = \mathbb{F}_2^{r'+c'}$ are fixed, and there are only $b - r' = c'$ -bit degrees of freedom. In summary, we can find a satisfied S with a time complexity $\min\{2^{n-r'}, 2^{c'}\}$. After we find S , compute

$$S^* = f^{-1}(S) \oplus (\text{Pad} \parallel 0^c).$$

Then, we try to search for m_1 and m_2 such that

$$[f(m_1 \parallel 0^c)]^{(c)} = [f^{-1}(f^{-1}(S^*) \oplus (m_2 \parallel 0^c))]^{(c)}. \quad (6)$$

The complexity of this procedure is about $\mathcal{O}(2^{\frac{c}{2}})$. At this point, we set

$$\Delta = f(m_1 \parallel 0^c) \oplus f^{-1}(f^{-1}(S^*) \oplus (m_2 \parallel 0^c)).$$

Then, $\text{msg} = m_1 \parallel \Delta \parallel m_2$ is a preimage of $z_1 \parallel \dots \parallel z_{l-1} \parallel z_l$. The overall complexity can be estimated as

$$\min \left\{ \max \left\{ \min \{ 2^{n-r'}, 2^{c'} \}, 2^{\frac{c}{2}} \right\}, 2^n \right\}.$$

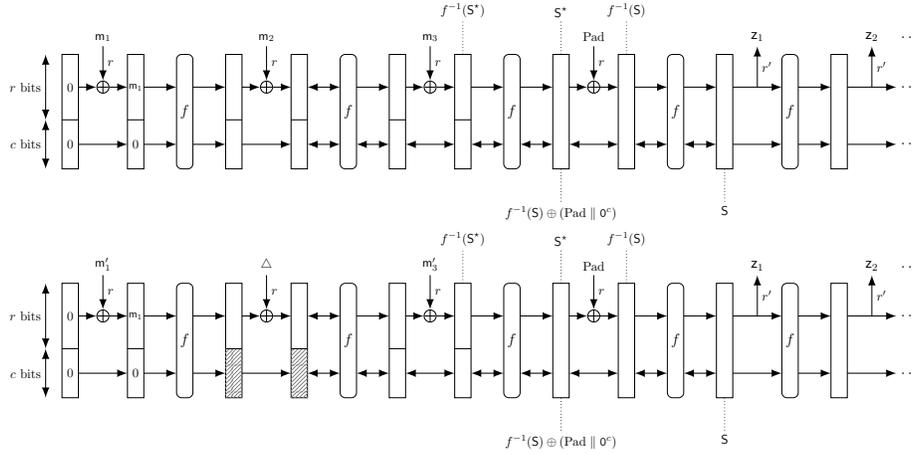


Fig. 3: Generic second preimage attack on the sponge construction instantiated with a permutation.

Remark 1. In Table 1, we listed the security bound proved by Lefevre and Menink in [26], where it assumes $n \leq b$, under which we always have $2^{n-r'} \leq 2^{b-r'} = 2^{c'}$, and thus

$$\min \left\{ \max \{ \min \{ 2^{n-r'}, 2^{c'} \}, 2^{\frac{c}{2}} \}, 2^n \right\} = \min \{ \max \{ 2^{n-r'}, 2^{\frac{c}{2}} \}, 2^n \}.$$

3.3.2 Second Preimage Attack Consider a given message $\text{msg} = m_1 \parallel m_2 \parallel m_3 \in \mathbb{F}_2^{r'} \times \mathbb{F}_2^{r'} \times \mathbb{F}_2^{r'}$ and its digest

$$\mathcal{H}(\text{msg}) = z_1 \parallel \cdots \parallel z_{l-1} \parallel z_l \in \mathbb{F}_2^{r'} \times \cdots \times \mathbb{F}_2^{r'} \times \mathbb{F}_2^{n-(l-1)r'}.$$

When $n \leq \frac{c}{2}$, we can apply the generic second preimage attack on the ideal hash function with n -bit output to find a second preimage in time complexity $\mathcal{O}(2^n)$. When $n > \frac{c}{2}$, there exists a more efficient attack, which is shown in Figure 3.

First, based on the message msg , we can derive the intermediate state S and compute $S^* = f^{-1}(S) \oplus (\text{Pad} \parallel 0^c)$. Then, we search for m'_1 and m'_3 such that

$$\lfloor f(m'_1 \parallel 0^c) \rfloor^{(c)} = \lfloor f^{-1}(f^{-1}(S^*) \oplus (m'_3 \parallel 0^c)) \rfloor^{(c)}.$$

The complexity of this procedure is about $\mathcal{O}(2^{\frac{c}{2}})$. At this point, we set

$$\Delta = f(m_1 \parallel 0^c) \oplus f^{-1}(f^{-1}(S^*) \oplus (m_3 \parallel 0^c)).$$

Then, $\text{msg} = m'_1 \parallel \Delta \parallel m'_3$ is a second preimage of $z_1 \parallel \cdots \parallel z_{l-1} \parallel z_l$.

3.3.3 Long Message Second Preimage Attack The procedure of this attack depends on the padding scheme of the hash function. Unlike Merkle-Damgård hash functions, for many sponge-based hash functions, the last blocks

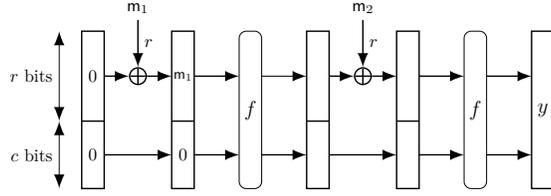


Fig. 4: A visualization of $0^b \xrightarrow{m_1 || m_2} y$.

of two padded messages with different lengths can be the same (e.g., the padding scheme of SHA3). In the following, we describe the long message second preimage attack under the assumption that the last blocks of two padded messages with different lengths are different, which makes the attack more complicated. For example, Wu [40] proposed a sponge-like hash function that employs message length padding. For padding schemes of SHA3, the attack can be carried out more straightforwardly.

The attack starts with the construction of a set of expandable messages [25] based on the multi-collision technique [24]. Let $\mathbf{a} \in \mathbb{F}_2^{r \cdot \ell}$ be a bit string of ℓ r -bit blocks, we use $x \xrightarrow{\mathbf{a}} y$ to denote that after absorbing \mathbf{a} by the sponge with the starting state $x \in \mathbb{F}_2^b$, we get the output state $y \in \mathbb{F}_2^b$. For example, $0^b \xrightarrow{m_1 || m_2} y$ is visualized in Figure 4.

Then, the entire attack operates as follows, and it is visualized in Figure 5. We first construct two sequences $[\mathbf{a}_1^{(0)}, \dots, \mathbf{a}_t^{(0)}]$ and $[\mathbf{a}_1^{(1)}, \dots, \mathbf{a}_t^{(1)}]$ with $\mathbf{a}_i^{(0)} \in \mathbb{F}_2^{2r}$, $\mathbf{a}_i^{(1)} = 0^{r \cdot 2^{i-1}} || \mathbf{s}_i$, $\mathbf{s}_i \in \mathbb{F}_2^{2r}$ ($1 \leq i \leq t$) such that the following condition

$$\begin{cases} 0^b \xrightarrow{\mathbf{a}_1^{(0)}} x_1, x_1 \xrightarrow{\mathbf{a}_2^{(0)}} x_2, \dots, x_{t-1} \xrightarrow{\mathbf{a}_t^{(0)}} x_t, \\ 0^b \xrightarrow{\mathbf{a}_1^{(1)}} x_1, x_1 \xrightarrow{\mathbf{a}_2^{(1)}} x_2, \dots, x_{t-1} \xrightarrow{\mathbf{a}_t^{(1)}} x_t \end{cases} \quad (7)$$

is fulfilled. With these two sequences, we can construct a set

$$\mathbb{M} = \{\mathbf{a}_1^{(b_1)} || \mathbf{a}_2^{(b_2)} || \dots || \mathbf{a}_t^{(b_t)} : (b_t, b_{t-1}, \dots, b_1) \in \mathbb{F}_2^t\}$$

of 2^t messages called an expandable message set. It is easy to check that the length of the message $\mathbf{a}_1^{(b_1)} || \mathbf{a}_2^{(b_2)} || \dots || \mathbf{a}_t^{(b_t)}$ is $((b_1 \cdot 2^0 + 2)r + (b_2 \cdot 2^1 + 2)r + \dots + (b_t \cdot 2^{t-1} + 2)r)$ -bit, and the lengths of the messages in \mathbb{M} range from $2t$ to $2^t + 2t - 1$ r -bit blocks. Moreover, for any $\text{msg} \in \mathbb{M}$, $0^b \xrightarrow{\text{msg}} x_t$. Such an expandable message set can be produced with time $\mathcal{O}(t \cdot 2^{\frac{t}{2}})$. Given x_i , we can find in time $\mathcal{O}(2^{\frac{t}{2}})$ two r -bit blocks $\mathbf{u} \in \mathbb{F}_2^r$ and $\dot{\mathbf{u}} \in \mathbb{F}_2^r$ such that the capacity parts of y and \dot{y} collide or $[y]^{(c)} = [\dot{y}]^{(c)}$, where $x_i \xrightarrow{\mathbf{u}} y$ and $x_i \xrightarrow{0^{r \cdot (2^i - 1)} || \dot{\mathbf{u}}} \dot{y}$. At this point, if we set

$$\begin{cases} \mathbf{a}_{i+1}^{(0)} = \mathbf{u} || \lceil \dot{y} \rceil^{(r)}, \\ \mathbf{a}_{i+1}^{(1)} = 0^{r \cdot (2^i - 1)} || \dot{\mathbf{u}} || \lceil y \rceil^{(r)}, \end{cases} \quad (8)$$

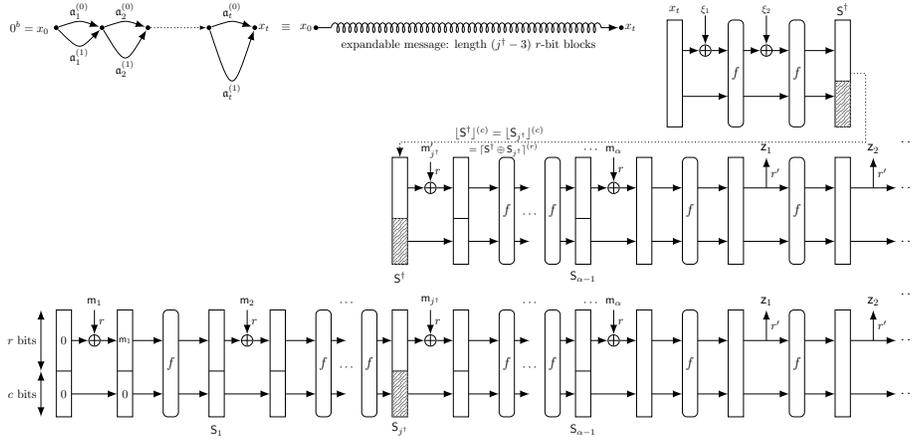


Fig. 5: The long-message second preimage attack

then $x_i \xrightarrow{a_{i+1}^{(0)}} x_{i+1}$ and $x_i \xrightarrow{a_{i+1}^{(1)}} x_{i+1}$ for some $x_{i+1} \in \mathbb{F}_2^b$. We can repeat this process t times to construct \mathbb{M} .

Let $\text{msg} = \mathbf{m}_1 \parallel \cdots \parallel \mathbf{m}_\alpha \in \mathbb{F}_2^{r \cdot \alpha}$ be a message and after padding it becomes $\mathbf{m}_1 \parallel \cdots \parallel \mathbf{m}_\alpha \parallel \mathbf{m}_{\alpha+1} \in \mathbb{F}_2^{r \cdot (\alpha+1)}$. Assume that α is much greater than $2t$ and

$$\begin{cases} 0^b \xrightarrow{\mathbf{m}_1} S_1, \dots, S_{2t+1} \xrightarrow{\mathbf{m}_{2t+2}} S_{2t+2}, \\ S_{2t+2} \xrightarrow{\mathbf{m}_{2t+3}} S_{2t+3}, \dots, S_\alpha \xrightarrow{\mathbf{m}_{\alpha+1}} S_{\alpha+1}. \end{cases}$$

Then, we try to identify $(\xi_1, \xi_2) \in \mathbb{F}_2^r \times \mathbb{F}_2^r$ such that $x_t \xrightarrow{\xi_1 \parallel \xi_2} S^\dagger$ and

$$[S^\dagger]^{(c)} \in \{[S_{2t+2}]^{(c)}, \dots, [S_{\alpha-1}]^{(c)}\}.$$

It takes about $\mathcal{O}(\frac{1}{\alpha} \cdot 2^c)$ to find $(\xi_1, \xi_2) \in \mathbb{F}_2^r \times \mathbb{F}_2^r$ such that $[S^\dagger]^{(c)} = [S_{j^\dagger}]^{(c)}$ for some $j^\dagger \in \{2t+2, \dots, \alpha-1\}$. Then, we select a message \mathfrak{M} from \mathbb{M} whose length is $(j^\dagger - 3) \cdot r$ -bit. The second preimage corresponding to the original message msg is

$$\mathfrak{M} \parallel \xi_1 \parallel \xi_2 \parallel [S^\dagger \oplus S_{j^\dagger}]^{(r)} \parallel \mathbf{m}_{j^\dagger+1} \parallel \cdots \parallel \mathbf{m}_{\alpha-1} \parallel \mathbf{m}_\alpha \in \mathbb{F}_2^{r \cdot \alpha}.$$

Note that, given the long message msg with α r -bit blocks, we need to determine t such that the expendable message set \mathbb{M} contains long enough messages to cover about $\alpha - 4$ blocks. On the other hand, we should not waste time producing an expendable message set containing a message that is too long to be useful. For this reason, we choose $t = \mathcal{O}(\log_2 \alpha)$ fulfilling the following inequality

$$2^{t-1} + 2(t-1) - 1 < \alpha - 3 \leq 2^t + 2t - 1.$$

Consequently, the time complexity of the attack can be estimated as

$$\mathcal{O}(\max\{2^\alpha, 2^{c - \log_2 \alpha}, \log_2 \alpha \cdot 2^{\frac{c}{2}}\}).$$

Typically, $\mathcal{O}(2^{c-\log_2(\alpha)})$ dominates.

Remark 2. Unlike other generic attacks introduced in this section, in the long message second preimage attack, we do not need to compute the inverse of the permutation f employed in the sponge function. Therefore, the attack remains valid even if we replace the permutations in the sponge function with random functions.

4 SPONGE-DM, SPONGE-EDM, and SPONGE-EDM^c

From Section 3.3, we can see that the generic preimage and second preimage attacks require the evaluation of the inverses of the permutations in the absorbing phase. Therefore, we expect that replacing the permutations in the absorbing phase by random transformations may increase the security bound with respect to preimage and second preimage attacks. In fact, Foekens showed that if the permutations in the sponge are replaced by random transformations $\mathcal{F} : \mathbb{F}_2^b \rightarrow \mathbb{F}_2^b$, the security levels of the resulting construction for preimage and second preimage attacks are about n -bit and $\min\{n, c - \log_2(\alpha)\}$ -bit respectively, where n is the size of the digest, c is the capacity and $\alpha \cdot r$ is the bit length of the longest message allowed to be processed [18].

Unfortunately, random transformations are harder to design than random permutations. Yet, we can build functions $\mathcal{F} : \mathbb{F}_2^b \rightarrow \mathbb{F}_2^b$ on top of permutations over \mathbb{F}_2^b that behave sufficiently well. In detail, we present SPONGE-DM and SPONGE-EDM: we leave the permutation $f : \mathbb{F}_2^b \rightarrow \mathbb{F}_2^b$ in the squeezing phase unchanged, and replace it in the absorption phase by a one-way function $\mathcal{F} : \mathbb{F}_2^b \rightarrow \mathbb{F}_2^b$. In SPONGE-DM[f , Pad, r, r', n], we instantiate \mathcal{F} as the Davies-Meyer construction [28], or basically f with a feed-forward:

$$\mathcal{F}(x) = f(x) \oplus x, \quad (9)$$

whereas in SPONGE-EDM[g, h , Pad, r, r', n], we take the Encrypted Davies-Meyer construction [13]:

$$\mathcal{F}(x) = h(g(x) \oplus x). \quad (10)$$

These two constructions are depicted in Figure 6a and Figure 6b. To save some bitwise XORs, we may alternatively instantiate the random function \mathcal{F} with the construction given in Figure 6c, leading to SPONGE-EDM^c. If, in the SPONGE-DM construction we only feedforward the inner part (named as SPONGE-DM^c), then there is a generic preimage attack with complexity $\mathcal{O}(2^{\frac{c}{2}})$, which is detailed in Supplementary Material A. This attack is better than brute-force search when $\frac{c}{2} < n \leq \min\{r', b - \frac{c}{2}\}$.

Remark 3. In SPONGE-DM, we build \mathcal{F} on top of f , and this is the basic construction. The idea of SPONGE-EDM, compared to SPONGE-DM, is to split the permutation f into halves and insert a feedforward halfway. This is the reason why SPONGE-EDM and SPONGE-EDM^c are defined for permutations g, h . For example, in

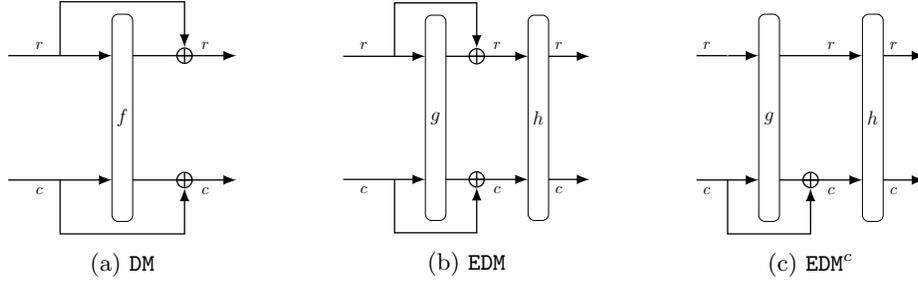


Fig. 6: Candidate one-way functions.

SPONGE-DM, one may opt to take the full 24-round Keccak- p permutation for f , whereas in SPONGE-EDM and SPONGE-EDM^c, permutation g consists of its first 12 rounds and h of its last 12 rounds. Clearly, there is limited efficiency gain in SPONGE-EDM and SPONGE-EDM^c over SPONGE-DM. However, if we consider a particular instantiation (as the above one), cryptanalysis of SPONGE-EDM can serve as stepping stone towards cryptanalysis of SPONGE-DM.

Remark 4. Finally, we note that in the security proofs, we regard f , g , and h as random permutations, and their difference is not important. Therefore, in proofs we simply assume that $f = g = h$. We refer to [27, Remark 1] for a discussion on independence of primitives in security proofs.

5 Security Analysis

We first re-establish the indistinguishability of SPONGE-DM, SPONGE-EDM, and SPONGE-EDM^c up to $2^{c/2}$ queries.

Theorem 1. *Let $b, c, r, c', r' \in \mathbb{N}$ and $f \xleftarrow{\$} \text{perm}(b)$, and consider \mathcal{H} to be SPONGE-DM and \mathcal{H}' to be either SPONGE-EDM and SPONGE-EDM^c of Section 4. Let \mathcal{R} be a random oracle. Let $q \in \mathbb{N}$ be such that $q \leq 2^{b-1}$. There exist simulators \mathcal{S} and \mathcal{S}' such that, for any adversary \mathcal{A} making at most q accumulated queries to f/\mathcal{S} or f/\mathcal{S}' ,*

$$\begin{aligned} \text{Adv}_{\mathcal{H}, \mathcal{S}}^{\text{indiff}}(\mathcal{A}) &\leq \frac{q(3q-1)}{2^c} + \frac{5q^2}{2^b} + \frac{2q(3r+4)}{2^{c'}}, \\ \text{Adv}_{\mathcal{H}', \mathcal{S}'}^{\text{indiff}}(\mathcal{A}) &\leq \frac{2q(6q-1)}{2^c} + \frac{20q^2}{2^b} + \frac{4q(3r+4)}{2^{c'}}. \end{aligned}$$

The proof is given in Section 5.1. By implication, this result implies collision resistance of SPONGE-DM, SPONGE-EDM, and SPONGE-EDM^c up to $\min\{2^{c/2}, 2^{n/2}\}$ queries, as long as $c' \geq c/2 + \log_2(r)$. These bounds are tight as the generic attacks on SPONGE carry over.

Next, we derive an improved preimage resistance bound.

Theorem 2. Let $b, c, r, c', r', n \in \mathbb{N}$ and $f \stackrel{\$}{\leftarrow} \text{perm}(b)$, and consider \mathcal{H} to be SPONGE-DM, SPONGE-EDM, or SPONGE-EDM^c of Section 4. Let $q \in \mathbb{N}$ be such that $q \leq 2^{b-1}$. Then, for any adversary \mathcal{A} making at most q queries to f ,

$$\text{Adv}_{\mathcal{H}}^{\text{pre}[n]}(\mathcal{A}) \leq \frac{4q}{2^n}.$$

The proof is given in Section 5.2. The security proof builds upon the preimage resistance analysis of the sponge by Lefevre and Mennink [26], and its refinement to the sponge based on a random function by Foekens [18].

The proof of second preimage resistance likewise relies on the analysis of Foekens [18].

Theorem 3. Let $b, c, r, c', r', n \in \mathbb{N}$ and $f \stackrel{\$}{\leftarrow} \text{perm}(b)$, and consider \mathcal{H} to be SPONGE-DM and \mathcal{H}' to be either SPONGE-EDM and SPONGE-EDM^c of Section 4. Let $\bar{q} = q + \alpha + \lceil n/r' \rceil - 1$ and $\bar{q}' = q + 2\alpha + \lceil n/r' \rceil - 1$ with $q \in \mathbb{N}$, α the number of r -bit blocks after padding a κ -bit message, and $\bar{q}, \bar{q}' \leq 2^{b-1}$. Then, for any adversary \mathcal{A} making at most q queries to f ,

$$\begin{aligned} \text{Adv}_{\mathcal{H}}^{\text{sec}[\kappa, n]}(\mathcal{A}) &\leq \frac{2\bar{q}}{2^b} + \frac{4\bar{q}}{2^n} + \frac{2\alpha\bar{q}}{2^c}, \\ \text{Adv}_{\mathcal{H}'}^{\text{sec}[\kappa, n]}(\mathcal{A}) &\leq \frac{2\bar{q}'}{2^b} + \frac{4\bar{q}'}{2^n} + \frac{4\alpha\bar{q}'}{2^c}. \end{aligned}$$

The proof is given in Section 5.3.

The bounds on preimage resistance (Theorem 2) and second preimage resistance (Theorem 3) are, in fact, tight. For preimage resistance, this is trivial as the generic attack takes 2^n evaluations. For second preimage resistance, the generic attack takes 2^n work, and hitting any absorption state of the first preimage requires $2^c/\alpha$ work.

For clarity of the presentation, in the proofs we abstract the padding function as $\text{padF} : \mathbb{F}_2^* \rightarrow (\mathbb{F}_2^r)^*$ which outputs the complete sequence of r bit blocks after padding. Its inverse is denoted by upadF and returns \perp if the inputs does not correspond to a valid padding.

5.1 Proof of Indifferentiability (Theorem 1)

We first prove SPONGE-DM, and at the end explain how the proof generalizes to SPONGE-EDM and SPONGE-EDM^c. We have to demonstrate that there exists a simulator \mathcal{S} such that

$$\text{Adv}_{\mathcal{H}, \mathcal{S}}^{\text{indiff}}(\mathcal{A}) = |\Pr(\mathcal{A}[\mathcal{H}[f], f^\pm] \rightarrow 1) - \Pr(\mathcal{A}[\mathcal{R}, \mathcal{S}[\mathcal{R}]^\pm] \rightarrow 1)| \quad (11)$$

is small for any possible adversary \mathcal{A} making q accumulated queries. Here, $(\mathcal{H}[f], f^\pm)$ is called the real world and $(\mathcal{R}, \mathcal{S}[\mathcal{R}]^\pm)$ the ideal world.

The proof closely follows the approach taken by Naito and Ohta [30] for the generalized sponge, and uses the approach of Choi et al. [12] to tame multicollisions. We first describe our simulator in Section 5.1.1. Then, we describe an intermediate world in Section 5.1.2, which allows to decompose the advantage into two distances. Each of these distances is then upper bounded in Sections 5.1.3 and 5.1.4, respectively, leading directly to the conclusion of Theorem 1.

5.1.1 Simulator The simulator \mathcal{S} aims to simulate the function f in both directions in such a way that it is consistent with \mathcal{R} . It maintains an initially empty database \mathcal{T} , storing tuples of the form (x, y, dir) , where $y = \mathcal{S}(x)$ and $\text{dir} \in \{\text{fwd}, \text{inv}\}$ denotes the query direction. For ease of reasoning, we represent these tuples in a graph. This graph consists of all nodes \mathbb{F}_2^b . For each tuple (x, y, dir) that gets added to \mathcal{T} , the following edges are added:

- $x \oplus (\mathbf{m} \| 0^c) \xRightarrow{\mathbf{m}} x \oplus y$ for all $\mathbf{m} \in \mathbb{F}_2^r$;
- $x \rightarrow y$.

Here, the double-lined arrow corresponds to a block absorption (potentially a zero-block) whereas the single-lined arrow does not absorb a block. For simplicity of notation, we denote $S_1 \xRightarrow{\mathbf{m}_1} S_2 \xRightarrow{\mathbf{m}_2} S_3$ as $S_1 \xRightarrow{\mathbf{m}_1 \| \mathbf{m}_2} S_3$, and we write $S \xrightarrow{\epsilon} S$.

Clearly, the simulator should intuitively make sure that, whenever there is a path

$$0^b \xRightarrow{\text{padF}(\text{msg})} y_1,$$

for some $\text{msg} \in \mathbb{F}_2^*$, any subsequent single-lined evaluations starting from y_1 should adhere to the random oracle output on input of msg :

$$[y_1]^{(r')} \| [y_2]^{(r')} \| \dots = \mathcal{R}(\text{msg}).$$

The simulator is described in Algorithm 2.

Next, let \mathcal{Q} be the permutation query history of the adversary. It contains the tuples (x, y, dir) queried by the adversary. In the ideal world, we have $\mathcal{Q} = \mathcal{T}$, but looking ahead, in the intermediate world, \mathcal{T} may contain additional information that is not present in \mathcal{Q} . Given $\mathcal{X} \in \{\mathcal{T}, \mathcal{Q}\}$, let $\text{Rooted}(\mathcal{X})$ denote the set of $y \in \mathbb{F}_2^b$ such that there exists a path that can be reconstructed from \mathcal{X} of the form $0^b \xRightarrow{\text{padF}(\text{msg})} y_1 \rightarrow \dots \rightarrow y_\beta$, for some $\text{msg} \in \mathbb{F}_2^*$ and $\beta \geq 1$ with $y_\beta = y$.

5.1.2 Intermediate World To facilitate the analysis, we introduce an intermediate world $(\mathcal{H}[\mathcal{S}[\mathcal{R}]], \mathcal{S}[\mathcal{R}]^\pm)$. This world implements the real world, except that the random permutation f is replaced by the simulator \mathcal{S} , which internally uses a random oracle \mathcal{R} hidden from the adversary. By the triangle inequality, we have

$$(11) \leq |\Pr(\mathcal{A}[\mathcal{H}[f], f^\pm] \rightarrow 1) - \Pr(\mathcal{A}[\mathcal{H}[\mathcal{S}[\mathcal{R}]], \mathcal{S}[\mathcal{R}]^\pm] \rightarrow 1)| \quad (12)$$

$$+ |\Pr(\mathcal{A}[\mathcal{H}[\mathcal{S}[\mathcal{R}]], \mathcal{S}[\mathcal{R}]^\pm] \rightarrow 1) - \Pr(\mathcal{A}[\mathcal{R}, \mathcal{S}[\mathcal{R}]^\pm] \rightarrow 1)|. \quad (13)$$

Distance (12) is upper bounded in Section 5.1.3, and distance (13) in Section 5.1.4.

Algorithm 2: Simulator

```

1 Function  $\mathcal{S}(x)$ :
2   if  $\exists(x, y, \text{dir}) \in \mathcal{T}$  then
3      $\lfloor$  return  $y$ 
4    $y^{(c')} \xleftarrow{\$} \mathbb{F}_2^{c'}$ 
5   if  $\exists u \in \mathbb{F}_2^r \setminus \{0^r\}, m \in (\mathbb{F}_2^r)^* \cup \{\epsilon\}$  with  $0^b \xrightarrow{m} x \oplus (u \parallel 0^c)$  and
6      $\text{upadF}(m \parallel u) \neq \perp$  then // last absorb call
7      $\lfloor y^{(r')} \leftarrow \mathcal{R}(\text{upadF}(m \parallel u), r') \oplus \lceil x \rceil^{(r')}$ 
8   else if  $\exists \text{msg} \in \mathbb{F}_2^*, \beta \geq 1$  with  $0^b \xrightarrow{\text{padF}(\text{msg})} y_1 \rightarrow \dots \rightarrow y_\beta$  and  $y_\beta = x$ 
9     then // squeezing
10     $\lfloor y^{(r')} \leftarrow \lfloor \mathcal{R}(\text{msg}, \beta \cdot r') \rfloor^{(r')}$ 
11  else
12     $\lfloor y^{(r')} \xleftarrow{\$} \mathbb{F}_2^{r'}$ 
13     $\mathcal{T} \leftarrow \mathcal{T} \cup \{(x, y^{(r')} \parallel y^{(c')})\}$ 
14  return  $y^{(r')} \parallel y^{(c')}$ 
15
16 Function  $\mathcal{S}^{-1}(y)$ :
17   if  $\exists(x, y, \text{dir}) \in \mathcal{T}$  then
18      $\lfloor$  return  $x$ 
19    $x \xleftarrow{\$} \mathbb{F}_2^b$ 
20    $\mathcal{T} \leftarrow \mathcal{T} \cup \{(x, y)\}$ 
21  return  $x$ 

```

5.1.3 Distance Between Real and Intermediate World The only distinction between these worlds is the primitive oracle's source of randomness: in the ideal world this is a random permutation, whereas in the intermediate world the simulator implements a two-sided random function. As long as no full-state collision occurs, the simulator is indistinguishable from a random permutation. Therefore, by the fundamental lemma of game playing [4],

$$(12) \leq \frac{q^2}{2^{b+1}}. \quad (14)$$

5.1.4 Distance Between Intermediate and Ideal World For $i = 1, \dots, q$, let \mathcal{T}_i and \mathcal{Q}_i denote the state of respectively \mathcal{T} and \mathcal{Q} after the first i queries.

Bad Events. We define a family of bad events, indexed by a query index $i = 1, \dots, q$, and defined over \mathcal{T} and \mathcal{Q} .

- COLL_i : $\exists(x, y, \text{dir}) \in \mathcal{T}_i \setminus \mathcal{T}_{i-1}, (x', y', \text{dir}') \in \mathcal{T}_{i-1}$ such that
 - either $\text{dir} = \text{fwd}$ and

$$\left\{ \lfloor y \rfloor^{(c)}, \lfloor y \oplus x \rfloor^{(c)} \right\} \cap \left\{ \lfloor y' \rfloor^{(c)}, \lfloor y' \oplus x' \rfloor^{(c)}, \lfloor x' \rfloor^{(c)}, 0^c \right\} \neq \emptyset,$$

- or $\text{dir} = \text{inv}$ and

$$\lfloor x \rfloor^{(c)} \in \left\{ \lfloor y' \rfloor^{(c)}, \lfloor y' \oplus x' \rfloor^{(c)}, \lfloor x' \rfloor^{(c)}, 0^c \right\};$$

- GUESS_i : $\exists(x, y, \text{dir}) \in \mathcal{Q}_i \setminus \mathcal{Q}_{i-1}$ such that $y \in \text{Rooted}(\mathcal{T}_i) \setminus \text{Rooted}(\mathcal{Q}_i)$;
- $\text{BAD}_i = \text{COLL}_i \vee \text{GUESS}_i$.

Moreover, for $\text{EVT} \in \{\text{COLL}, \text{GUESS}, \text{BAD}\}$, let $\text{EVT} = \bigvee_{i=1}^q \text{EVT}_i$.

The bad event COLL ensures that the simulator responds consistently with respect to the random oracle by forbidding inner collisions and paths connecting. GUESS is an event that can be set only in the intermediate world. It happens when the adversary queries a state in the middle of a path without having queried the earlier states first, causing the two simulators to behave differently.

Distance Between Intermediate and Ideal World as Long as no Bad. Assuming that BAD does not occur, there are no inner collisions between the intermediate states generated by $\mathcal{H}[\mathcal{S}[\mathcal{R}]]$. Consequently, for any $y \in \text{Rooted}(\mathcal{T})$, there exists a unique path of the form $0^b \xrightarrow{\text{padF}(\text{msg})} y_1 \rightarrow \dots \rightarrow y_\beta$, for some $\text{msg} \in \mathbb{F}_2^*$ and $\beta \geq 1$ with $y_\beta = y$, and no intermediate state hits 0^b . This ensures that the simulator remains consistent with the random oracle, expressed formally as:

$$\lfloor y \rfloor^{(r')} = \lfloor \mathcal{R}(\text{msg}, \beta \cdot r') \rfloor^{(r')}.$$

Since GUESS does not occur, the additional \mathcal{H} -originated queries that \mathcal{S} receives in the intermediate world do not influence its responses to the adversary's queries. Therefore,

$$\Pr(\mathcal{A}[\mathcal{H}[\mathcal{S}[\mathcal{R}]], \mathcal{S}[\mathcal{R}]^\pm] \rightarrow 1 \mid \neg \text{BAD}) = \Pr(\mathcal{A}[\mathcal{R}, \mathcal{S}[\mathcal{R}]^\pm] \rightarrow 1 \mid \neg \text{BAD}). \quad (15)$$

Probability of the Bad Events. We bound the probability of BAD in the following lemma.

Lemma 1. *In both the intermediate and ideal worlds, we have*

$$\Pr(\text{BAD}) \leq \frac{q(3q-1)}{2^c} + \frac{4q^2}{2^b} + \frac{q(6r+8)}{2^{c'}}. \quad (16)$$

Proof. By basic probability theory, we have

$$\Pr(\text{BAD}) \leq \sum_{i=1}^q \Pr(\text{COLL}_i \mid \neg \text{BAD}_{i-1}) + \Pr(\text{GUESS}_i \mid \neg \text{BAD}_{i-1}), \quad (17)$$

where BAD_0 denotes an event always set. COLL covers at most 6 inner collision scenarios involving previous queries, as well as two inner collisions with the initial value 0^b . Therefore,

$$\Pr(\text{COLL}_i \mid \neg \text{BAD}_{i-1}) \leq \frac{6(i-1)+2}{2^c}. \quad (18)$$

For the event GUESS, the adversary must be able to guess some $y \in \text{Rooted}(\mathcal{T}_i) \setminus \text{Rooted}(\mathcal{Q}_i)$. Each of these values are uniformly random, and the only potential information available to the adversary consists of the outer r' bits of these nodes. To bound the maximum expected multicollisions on these outer bits, we adopt the approach of Choi et al. [12]. Define the random variable F as follows:

$$F = \max_{v \in \mathbb{F}_2^{r'}} \left| \{(x, y, \text{fwd}) \in \mathcal{T} \mid \lceil y \rceil^{(r')} = v\} \right| \\ + \max_{v \in \mathbb{F}_2^{r'}} \left| \{(x, y, \text{fwd}) \in \mathcal{T} \mid \lceil y \oplus x \rceil^{(r')} = v\} \right| .$$

F counts the size of the largest multicollision among the intermediate states during the squeezing phase. Now, given a permutation query from the adversary, the probability that GUESS $_i$ is set, conditioned on \mathcal{T}_{i-1} is upper bounded by $\frac{F}{2^{c'}}$. Summing over all possible \mathcal{T}_{i-1} , we obtain

$$\Pr(\text{GUESS}_i \mid \neg \text{BAD}_{i-1}) \leq \frac{\mathbf{Ex}(F)}{2^{c'}} .$$

Using [12, 27], we have $\mathbf{Ex}(F) \leq \frac{4q}{2^{r'}} + 6r + 8$. By combining those equations along with (18) into (17), we obtain (16), hence the lemma. \square

Combining (15) with (16), and using the fundamental lemma of game playing [4], we obtain

$$(13) \leq \frac{q(3q-1)}{2^c} + \frac{4q^2}{2^b} + \frac{q(6r+8)}{2^{c'}} . \quad (19)$$

Finally, we plug (19) and (14) into (11), which completes the proof.

Extension to SPONGE-EDM and SPONGE-EDM^c. The proof for SPONGE-EDM and SPONGE-EDM^c would be almost identical. The most notable difference is that the simulator, whenever a forward query with input x is made, if there exists a path $0^b \xrightarrow{m} x \oplus (u \parallel 0^c)$, then the simulator samples a random y , and proactively sets $\mathcal{S}(x \oplus y)$ (for SPONGE-EDM) or $\mathcal{S}(0^r \parallel \lceil x \rceil^{(c)} \oplus y)$ (for SPONGE-EDM^c) to remain consistent with the random oracle. The bad events remain the same, but the query complexity of the simulator is at worst doubled, resulting in a corresponding loss in the bounds.

5.2 Proof of Preimage Resistance (Theorem 2)

Again, we first prove SPONGE-DM, and at the end explain how the proof generalizes to SPONGE-EDM and SPONGE-EDM^c.

Let $z \in \mathbb{F}_2^n$ be any image, and write $z_1 \parallel \dots \parallel z_\beta$, where $\|z_i\|_{\mathbb{F}_2} = r'$ for $i = 1, \dots, \beta - 1$ and $\|z_\beta\|_{\mathbb{F}_2} = n - (\beta - 1)r' \leq r'$. Consider any adversary \mathcal{A} making queries to f^\pm . These queries are summarized in a transcript \mathcal{Q} as tuples of the form (x, y, dir) , where $y = f(x)$ and $\text{dir} \in \{\text{fwd}, \text{inv}\}$ denotes the query direction.

The adversary can make a total amount of q queries, and we denote by \mathcal{Q}_i the query transcript up to (and including) tuple i . We assume that \mathcal{A} never repeats any query.

As in the proof of indifferentiability (Section 5.1), we represent queries that \mathcal{A} makes in a graph. This graph consists of all nodes \mathbb{F}_2^b . For each query (x, y, dir) , the following edges are added:

- $x \oplus (\mathbf{m} \| 0^c) \xrightarrow{\mathbf{m}} x \oplus y$ for all $\mathbf{m} \in \mathbb{F}_2^r$;
- $x \rightarrow y$.

For $i = 1, \dots, \beta$, we define

$$\mathbf{Z}_i := \begin{cases} \{y_i \in \mathbb{F}_2^b \mid \lceil y_i \rceil^{(r')} = \mathbf{z}_i\}, & \text{for } i \in \{1, \dots, \beta - 1\}, \\ \{y_i \in \mathbb{F}_2^b \mid \lceil y_i \rceil^{(n - (\beta - 1)r')} = \mathbf{z}_i\}, & \text{for } i = \beta. \end{cases}$$

The goal of the adversary \mathcal{A} is to find a preimage for \mathbf{z} , which is implied by the following event:

$$\text{PRE}(\mathcal{Q}) : \mathcal{Q} \text{ defines a path } 0^b \xrightarrow{\mathbf{m}_1} y'_1 \cdots \xrightarrow{\mathbf{m}_{\alpha-1}} y'_{\alpha-1} \xrightarrow{\mathbf{m}_\alpha} y_1 \longrightarrow \cdots \longrightarrow y_\beta$$

such that $y_i \in \mathbf{Z}_i$ for $i = 1, \dots, \beta$.

Indeed, once \mathcal{A} sets $\text{PRE}(\mathcal{Q})$ with a sequence of message blocks that correspond to a valid padding, the preimage for \mathbf{z} can be found as $\text{msg} = \text{upadF}(\mathbf{m}_1 \parallel \cdots \parallel \mathbf{m}_\alpha)$.

We say that a state y is a *valid squeezing state* if $\lceil y \rceil^{(r')} = \mathbf{z}_1$, $\lceil f(y) \rceil^{(r')} = \mathbf{z}_2$, and so on, until the last block $\lceil f^{\beta-1}(y) \rceil^{(n - (\beta-1)r')} = \mathbf{z}_\beta$. Formally, we define the set of valid squeezing states:

$$\mathcal{S} = \{y \mid f^{i-1}(y) \in \mathbf{Z}_i \text{ for } i = 1, \dots, \beta\}.$$

Clearly, in order to set $\text{PRE}(\mathcal{Q})$, the adversary must *ever* make a query corresponding to an edge $y'_{\alpha-1} \xrightarrow{\mathbf{m}_\alpha} y_1$, for some α , some $y'_{\alpha-1} \in \mathbb{F}_2^b$, and some $y_1 \in \mathcal{S}$. In terms of how these edges are defined, we can observe that $\text{PRE}(\mathcal{Q}) \Rightarrow \text{BAD}(\mathcal{Q})$, where

$$\text{BAD}(\mathcal{Q}) : (x, y, \text{dir}) \in \mathcal{Q} \text{ such that } x \oplus y \in \mathcal{S},$$

and thus that $\Pr(\text{PRE}(\mathcal{Q})) \leq \Pr(\text{BAD}(\mathcal{Q}))$.

It thus remains to bound $\Pr(\text{BAD}(\mathcal{Q}))$. By basic probability theory

$$\begin{aligned} \Pr(\text{BAD}(\mathcal{Q})) &= \sum_{s=0}^{2^{e'}} \Pr(\text{BAD}(\mathcal{Q}) \mid |\mathcal{S}| = s) \cdot \Pr(|\mathcal{S}| = s) \\ &\leq \sum_{s=0}^{2^{e'}} \sum_{i=1}^q \Pr(\text{BAD}(\mathcal{Q}_i) \mid |\mathcal{S}| = s \wedge \neg \text{BAD}(\mathcal{Q}_{i-1})) \cdot \Pr(|\mathcal{S}| = s) \\ &\leq \sum_{s=0}^{2^{e'}} \sum_{i=1}^q \frac{s}{2^b - (i-1)} \cdot \Pr(|\mathcal{S}| = s) \end{aligned}$$

$$\leq \sum_{s=0}^{2^{c'}} \frac{2qs}{2^b} \cdot \Pr(|\mathcal{S}| = s) = \frac{2q}{2^b} \cdot \mathbf{Ex}(|\mathcal{S}|),$$

assuming that $q \leq 2^{b-1}$.

From Lefevre and Mennink [26, Eq. (17), proof of Lemma 2, p. 198], we know that $\mathbf{Ex}(|\mathcal{S}|) \leq 2\frac{2^b}{2^n}$. Plugging this bound into the above one completes the proof.

Extension to SPONGE-EDM and SPONGE-EDM^c. The proof for SPONGE-EDM would be almost identical. The most notable difference is that $\text{PRE}(\mathcal{Q})$ would now be defined as

$\text{PRE}(\mathcal{Q})$: \mathcal{Q} defines a path

$$0^b \xrightarrow{m_1} y'_1 \longrightarrow y''_1 \cdots \xrightarrow{m_{\alpha-1}} y'_{\alpha-1} \longrightarrow y''_{\alpha-1} \xrightarrow{m_\alpha} y'_\alpha \longrightarrow y_1 \longrightarrow \cdots \longrightarrow y_\beta$$

such that $y_i \in \mathcal{Z}_i$ for $i = 1, \dots, \beta$,

and $\text{BAD}(\mathcal{Q})$ as

$$\text{BAD}(\mathcal{Q}) : (x, y, \text{dir}) \in \mathcal{Q} \text{ such that } x \oplus y \in f^{-1}(\mathcal{S}).$$

However, as f is a permutation, $|f^{-1}(\mathcal{S})| = |\mathcal{S}|$, and the remainder of the proof is identical. For SPONGE-EDM^c, the bad event becomes

$$\text{BAD}(\mathcal{Q}) : (x, y, \text{dir}) \in \mathcal{Q} \text{ such that } \lceil y \rceil^{(r)} \parallel \lfloor x \oplus y \rfloor^{(c)} \in f^{-1}(\mathcal{S}).$$

The expected number of elements in $f^{-1}(\mathcal{S})$ constraint to the outer part being $\lceil y \rceil^{(r)}$ is equal to the expected number of elements in $f^{-1}(\mathcal{S})$ divided by 2^r , as f is a random permutation, and any of them is then hit with probability at most $\frac{2^r}{2^b - (i-1)}$, so the bound will be identical. Note that this adjustment would not work for SPONGE-DM^c: as there is no application of f^{-1} on \mathcal{S} , the adversary can maximize their chances by selecting y so that $\lceil y \rceil^{(r)}$ equals \mathbf{z}_1 (see also Supplementary Material A).

5.3 Proof of Second Preimage Resistance (Theorem 3)

Again, we first prove SPONGE-DM, and at the end explain how the proof generalizes to SPONGE-EDM and SPONGE-EDM^c.

Let $\text{msg} \in \mathbb{F}_2^k$ be any preimage, and write $\mathbf{m}_1 \parallel \cdots \parallel \mathbf{m}_\alpha = \text{padF}(\text{msg})$. Let $\mathbf{z} = \text{SPONGE-DM}(\text{msg})$, and write $\mathbf{z}_1 \parallel \cdots \parallel \mathbf{z}_\beta$, where $\|\mathbf{z}_i\|_{\mathbb{F}_2} = r'$ for $i = 1, \dots, \beta - 1$ and $\|\mathbf{z}_\beta\|_{\mathbb{F}_2} = n - (\beta - 1)r' \leq r'$. Consider any adversary \mathcal{A} making queries to f^\pm . Here, the adversary is aware of the first preimage msg . The queries corresponding to this evaluation of SPONGE-DM are given to \mathcal{A} for free, prior to the experiment. These queries, along with the adversarial ones, are summarized in a transcript

\mathcal{Q} as tuples of the form (x, y, dir) , where $y = f(x)$ and $\text{dir} \in \{\text{fwd}, \text{inv}\}$ denotes the query direction. The adversary can make q queries, thus has access to a total of $\bar{q} = q + \alpha + \lceil n/r' \rceil - 1$ tuples in \mathcal{Q} . We denote by \mathcal{Q}_i the query transcript up to (and including) tuple i . We assume that \mathcal{A} never repeats any query.

As in the proof of indifferentiability (Section 5.1), we represent queries that \mathcal{A} makes in a graph. This graph consists of all nodes \mathbb{F}_2^b . For each query (x, y, dir) , the following edges are added:

- $x \oplus (\mathbf{m} \| 0^c) \xrightarrow{\text{m}} x \oplus y$ for all $\mathbf{m} \in \mathbb{F}_2^r$;
- $x \rightarrow y$.

Likewise, for $i = 1, \dots, \beta$, we define

$$\mathbf{Z}_i := \begin{cases} \{y_i \in \mathbb{F}_2^b \mid \lceil y_i \rceil^{(r')} = \mathbf{z}_i\}, & \text{for } i \in \{1, \dots, \beta - 1\}, \\ \{y_i \in \mathbb{F}_2^b \mid \lceil y_i \rceil^{(n - (\beta - 1)r')} = \mathbf{z}_i\}, & \text{for } i = \beta. \end{cases}$$

The goal of the adversary \mathcal{A} is to find a second preimage msg , which is implied by the following event:

$$\text{SEC}(\mathcal{Q}) : \mathcal{Q} \text{ defines a path } 0^b \xrightarrow{\mathbf{m}'_1} y'_1 \cdots \xrightarrow{\mathbf{m}'_{\alpha-1}} y'_{\alpha-1} \xrightarrow{\mathbf{m}'_\alpha} y_1 \longrightarrow \cdots \longrightarrow y_\beta$$

such that $y_i \in \mathbf{Z}_i$ for $i = 1, \dots, \beta$ and $\text{padF}(\text{msg}) \neq \mathbf{m}'_1 \parallel \cdots \parallel \mathbf{m}'_\alpha$.

Indeed, once \mathcal{A} sets $\text{SEC}(\mathcal{Q})$ with a sequence of message blocks that correspond to a valid padding, the second preimage for msg can be found by unpadding $\mathbf{m}'_1 \parallel \cdots \parallel \mathbf{m}'_\alpha$.

As before, we define the set of *valid squeezing states*:

$$\mathcal{S} = \{y \mid f^{i-1}(y) \in \mathbf{Z}_i \text{ for } i = 1, \dots, \beta\}.$$

The remainder of the proof, however, will be slightly more involved than the preimage proof, most importantly as the adversary *knows* the first preimage msg . For this message, we denote the states on the path with overlines, as in

$$0^b \xrightarrow{\mathbf{m}_1} \bar{y}'_1 \cdots \xrightarrow{\mathbf{m}_{\alpha-1}} \bar{y}'_{\alpha-1} \xrightarrow{\mathbf{m}_\alpha} \bar{y}_1 \longrightarrow \cdots \longrightarrow \bar{y}_\beta.$$

Note that, for this path, we have $\bar{y}_i \in \mathbf{Z}_i$ for $i = 1, \dots, \beta$ basically by definition. In other words, the adversary *knows* $\bar{y}_1 \in \mathcal{S}$. The adversary may succeed in setting $\text{SEC}(\mathcal{Q})$ the way it did for preimage resistance, $\text{BAD}(\mathcal{Q})$ of Section 5.2, but it may also succeed in setting $\text{SEC}(\mathcal{Q})$ if it manages to connect an absorption path from 0^b to any of the states $0^b, \bar{y}'_1, \dots, \bar{y}'_{\alpha-1}$. However, to this end, the adversary must *ever* make a query corresponding to an edge $y'_{i-1} \xrightarrow{\mathbf{m}'_i} \bar{y}$, for some i , some $y'_{i-1} \in \mathbb{F}_2^b$, and some $\bar{y} \in \mathbb{F}_2^b$ such that $\lceil \bar{y} \rceil^{(c)} \in \{0^c, \lceil \bar{y}'_1 \rceil^{(c)}, \dots, \lceil \bar{y}'_{\alpha-1} \rceil^{(c)}\}$. (Indeed, we have to focus on the inner parts, as the adversary can correct the outer part with the next message block.)

More formally, we have the following two bad events:

$$\text{BAD}(\mathcal{Q}) : (x, y, \text{dir}) \in \mathcal{Q} \text{ such that } (x, x \oplus y) \neq (\bar{y}'_{\alpha-1} \oplus (\mathbf{m}_\alpha \| 0^c), \bar{y}_1)$$

and $x \oplus y \in \mathcal{S}$,

$\text{CONNECT}(\mathcal{Q}) : (x, y, \text{dir}) \in \mathcal{Q}$ such that either $[x \oplus y]^{(c)} = 0^c$,

or $\exists a < \alpha$ with $(x, x \oplus y) \neq (\bar{y}'_{a-1} \oplus (\mathbf{m}_a \| 0^c), \bar{y}'_a)$ and $[x \oplus y]^{(c)} = [\bar{y}'_a]^{(c)}$.

Note that the bad events have become a bit more technical than the one on preimage resistance, as we have to include trivial wins coming from the hash evaluation of the first preimage. We have that $\text{SEC}(\mathcal{Q}) \Rightarrow \text{BAD}(\mathcal{Q}) \vee \text{CONNECT}(\mathcal{Q})$, and thus that $\Pr(\text{SEC}(\mathcal{Q})) \leq \Pr(\text{BAD}(\mathcal{Q})) + \Pr(\text{CONNECT}(\mathcal{Q}))$.

It thus remains to bound $\Pr(\text{BAD}(\mathcal{Q}))$ and $\Pr(\text{CONNECT}(\mathcal{Q}))$. For $\text{BAD}(\mathcal{Q})$, the analysis of Section 5.2 carries over, but with $\mathbf{Ex}(|\mathcal{S}|) \leq 2^{\frac{2^b}{2^n}} + 1$.

$$\Pr(\text{BAD}(\mathcal{Q})) \leq \frac{2\bar{q}}{2^b} + \frac{4\bar{q}}{2^n},$$

assuming that $\bar{q} \leq 2^{b-1}$.

For $\text{CONNECT}(\mathcal{Q})$, we obtain:

$$\begin{aligned} \Pr(\text{CONNECT}(\mathcal{Q})) &\leq \sum_{i=1}^{\bar{q}} \Pr(\text{CONNECT}(\mathcal{Q}_i) \mid \neg \text{CONNECT}(\mathcal{Q}_{i-1})) \\ &\leq \sum_{i=1}^{\bar{q}} \frac{\alpha 2^r}{2^b - (i-1)} \leq \frac{2\alpha\bar{q}}{2^c}, \end{aligned}$$

assuming that $\bar{q} \leq 2^{b-1}$.

Extension to SPONGE-EDM and SPONGE-EDM^c. Unlike for preimage resistance, the extension to SPONGE-EDM is a bit more involved. First of all, the first preimage consists of $2\alpha + \lceil n/r' \rceil - 1$ permutation calls that \mathcal{A} gets for free. These result in the following states:

$$0^b \xrightarrow{\mathbf{m}_1} \bar{y}'_1 \longrightarrow \bar{y}''_1 \cdots \xrightarrow{\mathbf{m}_{\alpha-1}} \bar{y}'_{\alpha-1} \longrightarrow \bar{y}''_{\alpha-1} \xrightarrow{\mathbf{m}_\alpha} \bar{y}'_\alpha \longrightarrow \bar{y}_1 \longrightarrow \cdots \longrightarrow \bar{y}_\beta.$$

The event $\text{SEC}(\mathcal{Q})$ would now be defined as:

$\text{SEC}(\mathcal{Q}) : \mathcal{Q}$ defines a path

$$0^b \xrightarrow{\mathbf{m}'_1} y'_1 \longrightarrow y''_1 \cdots \xrightarrow{\mathbf{m}'_{\alpha-1}} y'_{\alpha'-1} \longrightarrow y''_{\alpha'-1} \xrightarrow{\mathbf{m}'_\alpha} y'_{\alpha'} \longrightarrow y_1 \longrightarrow \cdots \longrightarrow y_\beta$$

such that $y_i \in \mathcal{Z}_i$ for $i = 1, \dots, \beta$ and $\text{padF}(\text{msg}) \neq \mathbf{m}'_1 \| \cdots \| \mathbf{m}'_{\alpha'}$,

and bad event $\text{BAD}(\mathcal{Q})$ as

$$\text{BAD}(\mathcal{Q}) : (x, y, \text{dir}) \in \mathcal{Q} \text{ such that } (x, x \oplus y) \neq (\bar{y}''_{\alpha-1} \oplus (\mathbf{m}_\alpha \| 0^c), \bar{y}'_\alpha)$$

$$\text{and } x \oplus y \in f^{-1}(\mathcal{S}).$$

Again, as f is a permutation, $|f^{-1}(\mathcal{S})| = |\mathcal{S}|$ for this bad event, and the analysis carries over.

Bad event $\text{CONNECT}(\mathcal{Q})$, that considers an attack connecting an absorption path from 0^b to any of the states $0^b, \bar{y}_1'', \dots, \bar{y}_{\alpha-1}''$, now gets split in $\text{CONNECT1}(\mathcal{Q})$, that covers preimages where the first permutation call is new and hits an original value $f^{-1}(0^b), \bar{y}_1', \dots, \bar{y}_{\alpha-1}'$ on its inner part, and $\text{CONNECT2}(\mathcal{Q})$, that covers preimages where the second permutation call is new and hits an original value $0^b, \bar{y}_1'', \dots, \bar{y}_{\alpha-1}''$ on its inner part. These two events are formally defined as follows:

$$\begin{aligned} \text{CONNECT1}(\mathcal{Q}) : (x, y, \text{dir}) \in \mathcal{Q} \text{ such that either } x \oplus y = \lfloor f^{-1}(0^b) \rfloor^{(c)}, \\ \text{or } \exists a < \alpha \text{ with } (x, x \oplus y) \neq (\bar{y}_{a-1}'' \oplus (\mathbf{m}_a \| 0^c), \bar{y}_a') \text{ and } \lfloor x \oplus y \rfloor^{(c)} = \lfloor \bar{y}_a' \rfloor^{(c)}, \\ \text{CONNECT2}(\mathcal{Q}) : (x', y', \text{dir}'), (x, y, \text{dir}) \in \mathcal{Q} \text{ such that } x = x' \oplus y' \text{ and} \\ \text{either } \lfloor y \rfloor^{(c)} = 0^c, \\ \text{or } \exists a < \alpha \text{ with } (x, y) \neq (\bar{y}_a', \bar{y}_a'') \text{ and } \lfloor y \rfloor^{(c)} = \lfloor \bar{y}_a'' \rfloor^{(c)}. \end{aligned}$$

Event $\text{CONNECT1}(\mathcal{Q})$ is basically as $\text{CONNECT}(\mathcal{Q})$, and the analysis carries over. For $\text{CONNECT2}(\mathcal{Q})$, we distinguish between forward and inverse queries for (x, y, dir) : if the query is in forward direction (so $\text{dir} = \text{fwd}$), the event is set with $\frac{2\alpha\bar{q}'}{2^c}$ as before; if the query is in inverse direction, we can see that there are \bar{q}' choices for (x', y', dir') , at most $\min\{\alpha \cdot 2^r, \bar{q}'\}$ for (x, y, dir) , and for any combination, the collision is set with probability at most $2/2^b$. Thus, in this case, we also get $\frac{2\alpha\bar{q}'}{2^c}$. As any query (x, y, dir) is either forward or inverse, we have to take the maximum of this bound, instead of the addition.

For SPONGE-EDM^c , the bad event $\text{BAD}(\mathcal{Q})$ generalizes the same way as it did for preimage resistance (Section 5.2) and $\text{CONNECT1}(\mathcal{Q})$ and $\text{CONNECT2}(\mathcal{Q})$ remain unchanged as they only concern the inner part.

6 Concrete Instances for Experimental Cryptanalysis and Potential Applications

In this section, we provide two families of SPONGE-EDM instances derived from SHA3 and Ascon named as Keccak-EDM and Ascon-EDM . The algorithms in Keccak-EDM and Ascon-EDM instantiate the permutations g and h of

$$\text{SPONGE-EDM}[g, h, \text{Pad}, r, r', n]$$

with non-ideal but strong cryptographic permutations withstanding years of extensive cryptanalysis. These instances serve as experimental targets of concrete cryptanalysis for a better understanding of the security of the SPONGE-EDM mode in practice. Moreover, we show that instances in Keccak-EDM and Ascon-EDM outperform their counterparts in SHA3 and Ascon with comparable security levels for long messages. In addition, we discuss the security and performance benefit of using algorithms in Ascon-EDM instead of algorithms in Ascon as the underlying hash functions of the post-quantum signature scheme Ascon-Sign submitted to the NIST post-quantum cryptography competition of additional digital signature schemes [37]. Note that from the perspective of concrete security, the

Table 2: A comparison of the security bounds of the Keccak-EDM instances and the hash functions in the SHA3 family. All parameters are expressed in bits.

Algorithm	Output size	r	c	Security		ρ
				Collision (2nd)	preimage	
SHA3-224	224	1152	448	112	224	1.14
Keccak-EDM-224		1312	288	112	224	
SHA3-256	256	1088	512	128	256	1.18
Keccak-EDM-256		1280	320	128	256	
SHA3-384	384	832	768	192	384	1.38
Keccak-EDM-384		1152	448	192	384	
SHA3-512	512	576	1024	256	512	1.77
Keccak-EDM-512		1024	576	256	512	
Keccak-EDM-768	768	768	832	384	768	-
Keccak-EDM-1024	1024	512	1088	512	1024	-

SPONGE-EDM mode with $\mathcal{F}(x) = h(g(x) \oplus x)$ is expected to be weaker than the SPONGE-DM mode with $\mathcal{F}(x) = h(g(x)) \oplus x$, since in general $g \circ h$ is closer to a random permutation than g or h . In this work we are more willing to propose the weaker algorithms to encourage further cryptanalysis.

6.1 The Keccak-EDM Family of Hash Functions

We abbreviate $\text{Keccak-EDM}[n + 64, n]$ as $\text{Keccak-EDM-}n$, where

$$\text{Keccak-EDM}[c, n] = \text{SPONGE-EDM}[g, h, \text{Pad}, 1600 - c, 1600 - c, n],$$

and g and h are the first and last 12 rounds of $\text{Keccak-}p[1600, 24]$ respectively. We argue that it is reasonable to regard g and h as random permutations in practice despite the non-ideal property of $\text{Keccak-}p[1600, 24]$ [11]. The best known collision and (second-)preimage attacks on variants of the hash functions in SHA3 penetrate at most 6 rounds [20, 34, 15, 22, 19, 35, 41, 42]. The community has strong confidence in the security margin provided by the 12-round $\text{Keccak-}p$ permutation. In fact, the 12-round $\text{Keccak-}p$ permutation is employed in TurboSHAKE and KangarooTwelve [7, 10], which are being standardized by IETF [39]. Some researchers even suggest that 10-round $\text{Keccak-}p$ is sufficient [2]. Therefore, we believe that using 12-round $\text{Keccak-}p$ ensures the security of the Keccak-EDM instances listed in Table 2 under the condition that the longest message allowed to be processed is limited to $r \cdot 2^{64}$ bits.

We denote the function mapping $x \in \mathbb{F}_2^b$ to $h(g(x) \oplus x)$ by

$$\mathcal{F} = \text{Keccak-}p[1600, 24]\text{-EDM}\langle 12, 12 \rangle.$$

Compared to $f = \text{Keccak-}p[1600, 24]$, \mathcal{F} has an extra exclusive-or operation from $\mathbb{F}_2^{1600} \times \mathbb{F}_2^{1600}$ to \mathbb{F}_2^{1600} . This extra operation is negligible compared to the

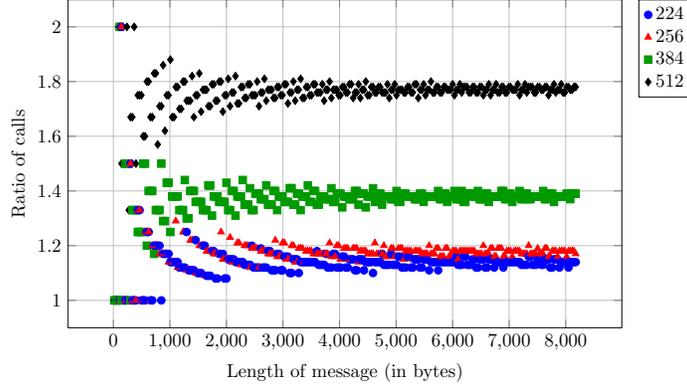


Fig. 7: Ratio of primitive calls between the $\text{SHA3-}n$ and $\text{Keccak-EDM-}n$.

permutation f . Let T_f be the time required for computing $\text{Keccak-}p[1600, 24]$, and let $T_{\mathcal{F}}$ be the time required for computing \mathcal{F} . We implement the function $\mathcal{F} = \text{Keccak-}p[1600, 24]\text{-EDM}\langle 12, 12\rangle$ based on the standalone code from XKCP [6] and run both f and \mathcal{F} for 1,000,000 times. The running times for f and \mathcal{F} are 13.45 seconds and 13.49 seconds, respectively. Hence, we can assume that $\frac{T_{\mathcal{F}}}{T_f} \leq 1.003$. To demonstrate the efficiency of Keccak-EDM , we calculate the ratio of the numbers of internal permutation or function calls made by Keccak and Keccak-EDM when processing messages of the same length, as illustrated in the Figure 7. To process an m -bit message, the number of permutation calls for $\text{SHA3-}n$ is

$$C_{\text{SHA3-}n}(m) = \left\lceil \frac{m + 4}{1600 - 2 \cdot n} \right\rceil.$$

For $\text{Keccak-EDM-}n$, the number of calls is

$$C_{\text{Keccak-EDM-}n}(m) = \left\lceil \frac{m + 4}{1600 - n - 64} \right\rceil.$$

For certain lengths of short messages, the efficiency of Keccak-EDM is almost twice that of SHA3 . For example, for messages with lengths greater than 1084 bits but less than 1276 bits, SHA3-256 requires two calls to the permutation f , whereas Keccak-EDM-256 requires only one. For long messages, the ratio of the times for executing $\text{SHA3-}n$ and $\text{Keccak-EDM-}n$ is about

$$\rho = \frac{C_{\text{SHA3-}n}(m) \cdot T_f}{C_{\text{Keccak-EDM-}n}(m) \cdot T_{\mathcal{F}}} = \frac{1600 - n - 64}{1600 - 2 \cdot n} \cdot \frac{1}{1.003}.$$

Table 2 gives a comparison and clearly shows the efficiency advantages of the Keccak-EDM instances.

Table 3: **Ascon-EDM** and **Ascon-HASH**. All parameters are expressed in bits.

Algorithm	Output size	Parameters		Security		
		Rate	Capacity	Collision	Preimage	2nd preimage
Ascon-HASH	256	64	256	128	192	128
Ascon-EDM	256	64	256	128	192	192
Ascon-EDM-128	128	128	192	64	128	128

6.2 The **Ascon-EDM** Family of Hash Functions

Ascon [16] is a family of lightweight cryptographic algorithms designed to provide efficient and secure encryption, hashing, and extendable-output functions (XoFs) for resource-constrained environments. **Ascon** is currently being standardized by the National Institute of Standards and Technology (NIST) for constrained devices [38]. We denote the 12-round permutation used in **Ascon** by $f = \text{Ascon-}p[12]$. Let g and h be the first and last 6 rounds of $f = \text{Ascon-}p[12]$ respectively. We denote the function mapping $x \in \mathbb{F}_2^{320}$ to $h(g(x) \oplus x) \in \mathbb{F}_2^{320}$ by $\mathcal{F} = \text{Ascon-}p[12]\text{-EDM}\langle 6, 6 \rangle$. Then, we propose the following two **SPONGE-EDM** instances derived from **Ascon**:

$$\begin{cases} \text{Ascon-EDM} = \text{SPONGE-EDM}[g, h, \text{Pad}, 64, 64, 256], \\ \text{Ascon-EDM-128} = \text{SPONGE-EDM}[g, h, \text{Pad}, 128, 128, 128], \end{cases} \quad (20)$$

whose parameters and security bounds are listed in Table 3. Next, we show how to employ these instances to improve the security and efficiency of hash-based post-quantum signature schemes.

*The Failed Attempt of **Ascon-Sign** to Reach 192-bit Security.* We see several attempts to build post-quantum signature schemes by instantiating the **SPHINCS⁺** framework [5] with the **Ascon** family of hash functions. Such signature schemes are expected to be more efficient and cost effective than those algorithms in the FIPS 205 standard [33]. Taking the signature scheme **SLH-DSA-SHAKE-128s** with 128-bit security for example, we will get a more efficient signature scheme with 128-bit security if we instantiate the **SPHINCS⁺** framework with **Ascon-HASH** whose state size is only 320-bit while the state of **SHAKE** is of 1600-bit. In fact, a 16% speedup for signature generation is reported in [17]. From the performance data extracted from [37] and [23] (listed in Table 4), we can observe even more significant improvement.

However, the security of the **SPHINCS⁺** framework is upper bounded by the security of its underlying hash functions against second preimage attacks. **Ascon-HASH** is a sponge construction with 256-bit capacity whose security against second preimage attacks is 128-bit. **Ascon-Sign** [37] is a hash-based signature scheme submitted to the NIST PQC project for additional digital signature schemes. It attempts to achieve 192-bit security by instantiating the **SPHINCS⁺** framework with the **Ascon** family of hash functions. As discussed previously, this

Table 4: A comparison of **Ascon-Sign** and **SPHINCS⁺-SHAKE** for the numbers of cycles required for key generation, signing and verification.

Version	Ascon-Sign			SPHINCS ⁺ -SHAKE		
	Keygen	Sign	Verify	KeyGen	Sign	Verify
128s-simple	315840896	2413174678	2429047	616484336	4682570992	4764084
128s-robust	554679600	4225825170	5516617	1195409786	8995481640	9232084
128f-simple	5939611	115382780	6972950	9649130	239793806	12909924
128f-robust	10156899	198139090	12469524	18726982	460757304	28152828
192s-simple	599392072	5458909051	4696353	898362434	8091419556	6465506
192s-robust	1046162651	9916984141	10281218	1753646932	15306007790	13509022
192f-simple	10939221	243023163	13058030	14215518	386861992	19876926
192f-robust	18827117	419872255	23006148	27463376	734072042	39295686

Table 5: Performance (measured in cycles/byte) comparison of **Ascon-HASH** and **Ascon-EDM-128**.

Algorithm	Size of input (bytes)				
	64	80	96	104	128
Ascon-HASH	22.5	23.72	21.75	20.64	18.7
Ascon-EDM-128	11.8	11.82	11.29	10.47	9.7

is not possible since the resistance to second preimage attacks of **Ascon-HASH** is well below the 192-bit level. In fact, **Ascon-Sign** did not advance to the second round of competition, and in the NIST official comments for the first-round signatures, Saarinen presented a concrete forgery attack on **Ascon-Sign**'s 192-bit security, which is in essence a second preimage attack on **Ascon-HASH** with time $\mathcal{O}(2^{128})$. To solve this issue, we can simply employ **Ascon-EDM** within **SPHINCS⁺**.

Improving the efficiency of Ascon-Sign with 128-bit security. For the 128-bit secure instance of **Ascon-Sign**, the performance can be further improved by using **Ascon-EDM** with $(128 + 64)$ -bit capacity, i.e., a 256-bit capacity as in **Ascon-HASH** is not necessary for **Ascon-EDM** to reach 128-bit security against second preimage attacks. Note that in our design, we require the length of the longest message to be signed is within the 2^{64} -bit limit. In the **SPHINCS⁺** framework, the most computation extensive part comes from the hash function invocations for computing the non-leaf nodes of the underlying **XMSS** trees. For **Ascon-Sign-128**, the function $\mathbf{H}(\text{PK.seed}, \text{ADRS}, \mathbf{n}_{\text{Left}} \parallel \mathbf{n}_{\text{Right}})$ used to compute the non-leaf nodes of the **XMSS** trees is defined as

$$\text{Trunc}_{128}(\text{Ascon-HASH}(\text{PK.seed} \parallel \text{ADRS} \parallel \mathbf{n}_{\text{Left}} \parallel \mathbf{n}_{\text{Right}})),$$

where **PK.seed** is a 16-byte public seed, **ADRS** is a 32-byte structure indicating the relative position of the computed node, and \mathbf{n}_{Left} and $\mathbf{n}_{\text{Right}}$ are two 16-byte nodes of an **XMSS** tree. Therefore the size of the input to **Ascon-HASH** is 640-bit. Taking the padding scheme into account, it takes $\frac{640+64}{64} = 11$ evaluations

of `Ascon-p`[12] to compute one non-leaf XMSS tree node. If we use `Ascon-EDM-128` instead, we can shrink the capacity to $128 + 64 = 192$ -bit, resulting in a $320 - 192 = 128$ -bit rate. Consequently, we only need $\lceil \frac{640+128}{128} \rceil = 6$ evaluations of the `Ascon-p`[12]-EDM[6, 6] permutation. To further demonstrate the advantage of using `Ascon-EDM-128` in this scenario, we perform more experiments measuring the software speed of `Ascon-HASH` and `Ascon-EDM-128` for several input lengths used in `Ascon-Sign` and the results are listed in Table 5.

ACKNOWLEDGEMENTS. This work is supported by the National Key Research and Development Program of China (2022YFB2701900), the National Cryptologic Science Foundation of China (2025NCSF01012), the National Natural Science Foundation of China (62032014), and the Fundamental Research Funds for the Central Universities. Charlotte Lefevre is supported by the Netherlands Organisation for Scientific Research (NWO) under grant OCENW.KLEIN.435. Bart Mennink is supported by the Netherlands Organisation for Scientific Research (NWO) under grant VI.Vidi.203.099.

References

1. Andreeva, E., Mennink, B., Preneel, B.: Security reductions of the second round SHA-3 candidates. In: Burmester, M., Tsudik, G., Magliveras, S.S., Ilic, I. (eds.) Information Security - 13th International Conference, ISC 2010, Boca Raton, FL, USA, October 25-28, 2010, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6531, pp. 39–53. Springer (2010). https://doi.org/10.1007/978-3-642-18178-8_5, https://doi.org/10.1007/978-3-642-18178-8_5
2. Aumasson, J.P.: Too much crypto. Cryptology ePrint Archive, Paper 2019/1492 (2019), <https://eprint.iacr.org/2019/1492>
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993. pp. 62–73. ACM (1993). <https://doi.org/10.1145/168588.168596>, <https://doi.org/10.1145/168588.168596>
4. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4004, pp. 409–426. Springer (2006). https://doi.org/10.1007/11761679_25, https://doi.org/10.1007/11761679_25
5. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The SPHINCS⁺ signature framework. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019. pp. 2129–2146. ACM (2019). <https://doi.org/10.1145/3319535.3363229>, <https://doi.org/10.1145/3319535.3363229>
6. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Assche, G.V., Keer, R.V.: The eXtended Keccak Code Package (XKCP). GitHub Repository, <https://github.com/XKCP/XKCP>

7. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Assche, G.V., Keer, R.V., Viguier, B.: TurboSHAKE. Cryptology ePrint Archive, Paper 2023/342 (2023), <https://eprint.iacr.org/2023/342>
8. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Sponge Functions. Ecrypt Hash Workshop (2007)
9. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: On the indistinguishability of the sponge construction. In: Smart, N.P. (ed.) Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings. Lecture Notes in Computer Science, vol. 4965, pp. 181–197. Springer (2008). https://doi.org/10.1007/978-3-540-78967-3_11, https://doi.org/10.1007/978-3-540-78967-3_11
10. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V., Keer, R.V., Viguier, B.: KangarooTwelve: Fast hashing based on Keccak- p . In: Preneel, B., Vercauteren, F. (eds.) Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10892, pp. 400–418. Springer (2018). https://doi.org/10.1007/978-3-319-93387-0_21, https://doi.org/10.1007/978-3-319-93387-0_21
11. Boura, C., Canteaut, A., Cannière, C.D.: Higher-order differential properties of Keccak and Luffa. In: Joux, A. (ed.) Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6733, pp. 252–269. Springer (2011). https://doi.org/10.1007/978-3-642-21702-9_15, https://doi.org/10.1007/978-3-642-21702-9_15
12. Choi, W., Lee, B., Lee, J.: Indistinguishability of Truncated Random Permutations. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11921, pp. 175–195. Springer (2019). https://doi.org/10.1007/978-3-030-34578-5_7, https://doi.org/10.1007/978-3-030-34578-5_7
13. Cogliati, B., Seurin, Y.: EWCDM: an efficient, beyond-birthday secure, nonce-misuse resistant MAC. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 121–149. Springer (2016). https://doi.org/10.1007/978-3-662-53018-4_5, https://doi.org/10.1007/978-3-662-53018-4_5
14. Coron, J., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3621, pp. 430–448. Springer (2005). https://doi.org/10.1007/11535218_26, https://doi.org/10.1007/11535218_26
15. Dinur, I.: Cryptanalytic applications of the polynomial method for solving multivariate equation systems over $\text{GF}(2)$. In: Canteaut, A., Standaert, F. (eds.) Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I. Lecture Notes in Computer Science,

- vol. 12696, pp. 374–403. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_14, https://doi.org/10.1007/978-3-030-77870-5_14
16. Dobraunig, C., Eichlseder, M., Mendel, F., Schl affer, M.: Ascon v1.2: Lightweight authenticated encryption and hashing. *J. Cryptol.* **34**(3), 33 (2021). <https://doi.org/10.1007/S00145-021-09398-9>, <https://doi.org/10.1007/s00145-021-09398-9>
 17. Faria, H., Valen a, J.M.: Post-quantum authentication with lightweight cryptographic primitives. *Cryptology ePrint Archive*, Paper 2021/1298 (2021), <https://eprint.iacr.org/2021/1298>
 18. Foekens, R.: Security of the Sponge Construction with a Random Transformation. Bachelor’s Thesis (2023)
 19. Guo, J., Liu, G., Song, L., Tu, Y.: Exploring SAT for cryptanalysis: (quantum) collision attacks against 6-round SHA-3. In: Agrawal, S., Lin, D. (eds.) *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security*, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part III. *Lecture Notes in Computer Science*, vol. 13793, pp. 645–674. Springer (2022). https://doi.org/10.1007/978-3-031-22969-5_22, https://doi.org/10.1007/978-3-031-22969-5_22
 20. Guo, J., Liu, M., Song, L.: Linear structures: Applications to cryptanalysis of round-reduced Keccak. In: Cheon, J.H., Takagi, T. (eds.) *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security*, Hanoi, Vietnam, December 4–8, 2016, Proceedings, Part I. *Lecture Notes in Computer Science*, vol. 10031, pp. 249–274 (2016). https://doi.org/10.1007/978-3-662-53887-6_9, https://doi.org/10.1007/978-3-662-53887-6_9
 21. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON family of lightweight hash functions. In: Rogaway, P. (ed.) *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference*, Santa Barbara, CA, USA, August 14–18, 2011. Proceedings. *Lecture Notes in Computer Science*, vol. 6841, pp. 222–239. Springer (2011). https://doi.org/10.1007/978-3-642-22792-9_13, https://doi.org/10.1007/978-3-642-22792-9_13
 22. He, L., Lin, X., Yu, H.: Improved preimage attacks on 4-round Keccak-224/256. *IACR Trans. Symmetric Cryptol.* **2021**(1), 217–238 (2021). <https://doi.org/10.46586/TOSC.V2021.I1.217-238>, <https://doi.org/10.46586/tosc.v2021.i1.217-238>
 23. H ulsing, A., Bernstein, D.J., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.L., Kampanakis, P., K obl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., Aumasson, J.P., Westerbaan, B., Beullens, W.: SPHINCS⁺: Submission to the NIST post-quantum project, v.3 (2020), <https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>
 24. Joux, A.: Multicollisions in iterated hash functions. application to cascaded constructions. In: *Annual International Cryptology Conference*. pp. 306–316. Springer (2004)
 25. Kelsey, J., Schneier, B.: Second preimages on n -bit hash functions for much less than 2^n work. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. pp. 474–490. Springer (2005)
 26. Lefevre, C., Mennink, B.: Tight preimage resistance of the sponge construction. In: Dodis, Y., Shrimpton, T. (eds.) *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference*, CRYPTO 2022, Santa Barbara,

- CA, USA, August 15-18, 2022, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 13510, pp. 185–204. Springer (2022). https://doi.org/10.1007/978-3-031-15985-5_7, https://doi.org/10.1007/978-3-031-15985-5_7
27. Lefevre, C., Mennink, B.: SoK: Security of the Ascon modes. *IACR Trans. Symmetric Cryptol.* **2025**(1), 138–210 (2025). <https://doi.org/10.46586/TOSC.V2025.I1.138-210>, <https://doi.org/10.46586/tosc.v2025.i1.138-210>
 28. Matyas, S.M.: Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin* **27**, 5658–5659 (1985)
 29. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, Cambridge, MA, USA, February 19-21, 2004, Proceedings. *Lecture Notes in Computer Science*, vol. 2951, pp. 21–39. Springer (2004). https://doi.org/10.1007/978-3-540-24638-1_2, https://doi.org/10.1007/978-3-540-24638-1_2
 30. Naito, Y., Ohta, K.: Improved indifferentiable security analysis of PHOTON. In: Abdalla, M., Prisco, R.D. (eds.) *Security and Cryptography for Networks - 9th International Conference, SCN 2014*, Amalfi, Italy, September 3-5, 2014. Proceedings. *Lecture Notes in Computer Science*, vol. 8642, pp. 340–357. Springer (2014). https://doi.org/10.1007/978-3-319-10879-7_20, https://doi.org/10.1007/978-3-319-10879-7_20
 31. NICCS: Announcement on Launching the Next-generation Commercial Cryptographic Algorithms Program (2025), <https://www.niccs.org.cn/tzgg/>
 32. NIST: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions . FIPS PUB 202 (2015), <https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf>
 33. NIST: Stateless Hash-Based Digital Signature Standard. Federal Information Processing Standards Publication, FIPS 205 (2024), <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf>
 34. Qiao, K., Song, L., Liu, M., Guo, J.: New collision attacks on round-reduced Keccak. In: Coron, J., Nielsen, J.B. (eds.) *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Paris, France, April 30 - May 4, 2017, Proceedings, Part III. *Lecture Notes in Computer Science*, vol. 10212, pp. 216–243 (2017). https://doi.org/10.1007/978-3-319-56617-7_8, https://doi.org/10.1007/978-3-319-56617-7_8
 35. Qin, L., Hua, J., Dong, X., Yan, H., Wang, X.: Meet-in-the-middle preimage attacks on sponge-based hashing. In: Hazay, C., Stam, M. (eds.) *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Lyon, France, April 23-27, 2023, Proceedings, Part IV. *Lecture Notes in Computer Science*, vol. 14007, pp. 158–188. Springer (2023). https://doi.org/10.1007/978-3-031-30634-1_6, https://doi.org/10.1007/978-3-031-30634-1_6
 36. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy, B.K., Meier, W. (eds.) *Fast Software Encryption, 11th International Workshop, FSE 2004*, Delhi, India, February 5-7, 2004, Revised Papers. *Lecture Notes in Computer Science*, vol. 3017, pp.

- 371–388. Springer (2004). https://doi.org/10.1007/978-3-540-25937-4_24, https://doi.org/10.1007/978-3-540-25937-4_24
37. Srivastava, V., Gupta, N., Jati, A., Bakshi, A., Breier, J., Chattopadhyay, A., Debnath, S.K., Hou, X.: Ascon-Sign Submission to the NIST Post-quantum Project (2023), <https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures>
38. Turan, M.S., McKay, K.A., Chang, D., Kang, J., Kelsey, J.: Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions. NIST Special Publication 800 – NIST SP 800-232 ipd (2024), <https://csrc.nist.gov/pubs/sp/800/232/ipd>
39. Viguier, B., Wong, D., Assche, G.V., Dang, Q., Daemen, J.: KangarooTwelve and TurboSHAKE. Internet-Draft (2024), <https://www.ietf.org/archive/id/draft-irtf-cfrg-kangarootwelve-12.html>
40. Wu, H.: The hash function JH. Submission to NIST (round 3) **6** (2011)
41. Zhang, Z., Hou, C., Liu, M.: Probabilistic linearization: Internal differential collisions in up to 6 rounds of SHA-3. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14923, pp. 241–272. Springer (2024). https://doi.org/10.1007/978-3-031-68385-5_8, https://doi.org/10.1007/978-3-031-68385-5_8
42. Zhang, Z., Hou, C., Liu, M.: Preimage attacks on up to 5 rounds of SHA-3 using internal differentials. In: Fehr, S., Fouque, P. (eds.) Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part I. Lecture Notes in Computer Science, vol. 15601, pp. 333–363. Springer (2025). https://doi.org/10.1007/978-3-031-91107-1_12, https://doi.org/10.1007/978-3-031-91107-1_12

Supplementary Material

A Generic Preimage Attack on SPONGE-DM^c

Consider the SPONGE-DM^c construction given in Figure 8, where \mathcal{F} is instantiated with the SPONGE-EDM^c compression function, while only feedforward the capacity part.

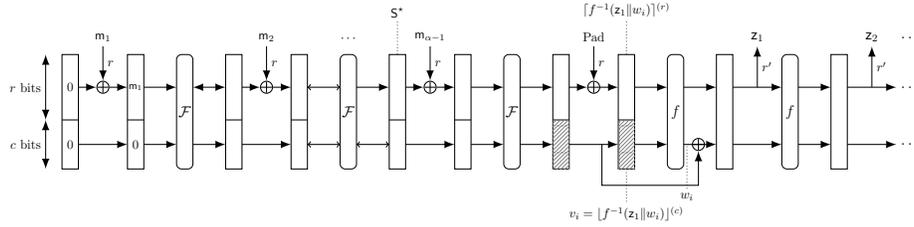


Fig. 8: Generic preimage attack on the SPONGE-DM^c construction.

When $r' \geq n$, the digest consists only z_1 , where $\|z_1\|_{\mathbb{F}_2} = n$. In this case, we randomly select 2^t strings $w_i \in \mathbb{F}_2^{b-n}$ and concatenate each with z_1 to form a set of b -bit candidate states:

$$S_i = z_1 \parallel w_i, \quad i \in \{1, 2, \dots, 2^t\}.$$

For each candidate state, we compute $v_i = \lfloor g^{-1}(S_i) \rfloor^{(c)}$. Next, we search for a message m such that

$$\lfloor \mathcal{F}(S^* \oplus (m_{\alpha-1} \parallel 0^c)) \rfloor^{(c)} = v_j.$$

The expected number of trials required to find such a state m is approximately 2^{c-t} . If m is found, then $m_1 \parallel m_2 \parallel \dots \parallel m_{\alpha-1} \parallel \lfloor \mathcal{F}(S^* \oplus (m_{\alpha-1} \parallel 0^c)) \oplus (g^{-1}(S_j)) \rfloor^{(r)}$ can produce the digest z_1 , regardless of the padding rule. The time complexity is $2^t + 2^{c-t}$, which is minimized to

$$\begin{cases} 2^{c/2} & \text{if } b - n \geq \frac{c}{2}, \\ 2^{b-n} + 2^{n-r} & \text{if } b - n < \frac{c}{2}. \end{cases}$$

When $r' < n$, the process above could be repeated approximately $2^{n-r'}$ times to expect a full n -bit digest matching. Thus, the time complexity scales accordingly:

$$\begin{cases} 2^{n-r'+c/2} & \text{if } b - n \geq \frac{c}{2}, \\ 2^{b-r'} + 2^{2n-r-r'} & \text{if } b - n < \frac{c}{2}. \end{cases}$$