Permutation-Based Hash Chains with Application to Password Hashing

Charlotte Lefevre and Bart Mennink

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

Abstract. Hash chain based password systems are a useful way to guarantee authentication with one-time passwords. The core idea is specified in RFC 1760 as S/Key. At CCS 2017, Kogan et al. introduced T/Key, an improved password system where one-time passwords are only valid for a limited time period. They proved security of their construction in the random oracle model under a basic modeling of the adversary. In this work, we make various advances in the analysis and instantiation of hash chain based password systems. Firstly, we describe a slight generalization called U/Key that allows for more flexibility in the instantiation and analysis, and we develop a security model that refines the adversarial strength into offline and online complexity, that can be used beyond the random oracle model, and that allows to argue multi-user security directly. Secondly, we derive a new security proof of U/Key in the random oracle model, as well as dedicated and tighter security proofs of U/Key instantiated with a sponge construction and a truncated permutation. When applied to T/Key, these results improve significantly over the earlier results: whereas the originally suggested instantiation using SHA-256 achieved 128 bits of security using a hash function with a state size of 384 bits, with a truncated permutation construction one can achieve 128 bits of security already with a state size of 256 bits.

Keywords: one-time passwords, hash chain, T/Key, U/Key, security model, sponge, truncated permutation

1 Introduction

The far majority of internet services rely on passwords for authentication. However, despite their broad usage, they introduce significant security weaknesses. For example, major security problems have appeared in the context of static passwords over the last years. These problems have lead the more security-sensitive services to move to two-factor authentication, where an additional device, app, or something else is used for the user to authenticate themselves. The FIDO Alliance has made significant efforts to standardize and promote secure authentication [1].

However, in certain cases, two-factor authentication is not convenient, e.g., it takes more time and one may have to rely on additional physical devices,

and in these cases, one-time password hash chains offer a convenient alternative solution. In a nutshell, the core idea of a hash chain, which is specified in an RFC as S/Key [2], is the following. Let $h : \{0,1\}^n \to \{0,1\}^n$ be a cryptographic hash function. As initialization, the client generates a random password x of n bits, and sends $x_K := h^K(x)$ securely to the server, where h^K is the K-fold evaluation of h. Then, during authentication round k, for $k \in \{1, \ldots, K\}$, the client sends $x_{K-k} := h^{K-k}(x)$ to the server, which verifies that x_{K-k} is indeed a preimage of x_{K-k+1} through h. The server updates its stored value to x_{K-k}. Although this construction is very simple, it suffers from two main weaknesses: passwords remain valid indefinitely (at least, until the next authentication round takes place), and the repeated iteration of the same one-way function makes it weaker than a single iteration [3].

To address these limitations, Kogan et al. introduced a time-based one-time password scheme named T/Key [4]. T/Key solves the validity issue by including timestamps in the iterated evaluations of h and the multi-iteration problem by using domain separation. It is inspired by the TOTP (time-based one-time password) scheme [5], with crucial difference that TOTP relies on a shared secret key whereas T/Key does not. In detail, in T/Key the hash chain transition from password x_{K-k} to x_{K-k+1} for $k \in \{1, \ldots, K\}$ is computed as

$$x_{K-k+1} := h(\langle \mathrm{ctr}_{K-k+1} \rangle_t \parallel id \parallel x_{K-k}) \,, \tag{1}$$

where $\langle \mathrm{ctr}_k \rangle_t$ encodes injectively the timestamp, and id is a random salt used to avoid pre-computation attacks. In other words, the initialization computes x_K from a random x as

$$x_K := h_K^{id} \circ h_{K-1}^{id} \circ \cdots \circ h_1^{id}(x) \,,$$

where $h_k^{id}(\cdot) := h(\langle \mathrm{ctr}_k \rangle_t \parallel id \parallel \cdot)$ for brevity.

Kogan et al. proved that T/Key is a secure hash chain based password system under the assumption that the hash function is a random oracle [6]. In detail, assuming that h is a random oracle, the functions $h_1, \ldots, h_K$ can be considered perfectly random and independent, and breaking the hash chain then corresponds to inverting any point on the chain. An adversary can only succeed in this with probability at most approximately

$$\frac{2q + 2K}{2^n} \,, \tag{2}$$

where q is the number of random oracle queries and n the output size of the random oracle.

However, the security model in which the result (2) was obtained is rather basic, and in particular does not accurately capture the adversarial power. Specifically, two issues arise: (i) the model does not refactor the adversarial complexities into offline and online time, and (ii) it does not consider the multi-user setting. The former issue is particularly relevant in the context of T/Key where time frames are rather short — the developers suggest time frames of 30 seconds.

Short time frames mean that the adversary can only make a limited amount of "live" authenticity attempts after it has learned the value to target, but it has had a much larger pre-computation phase where it could have made a retrospectively lucky hash function evaluation. The latter issue on the multi-user setting is important as T/Key would not be used in isolation but rather by thousands of users at the same time. In addition, dedicated multi-user analysis (as opposed to using a generic single-user to multi-user reduction [7,8]) typically allows for more detailed security bounds, and thus, potentially, a longer lifetime of hash chains.

At first sight, these two issues do not seem very limiting for the current analysis of Kogan et al. [4]. This is due to the fact that their security analysis is in the random oracle model, thus the security loss induced by these simplifications is not significant, and increasing the length of the passwords can compensate for the security loss. In order to use T/Key (or any other hash chain based system) one first has to instantiate the random oracle with an actual hash function, and in this case, the issues become more pronounced. This is caused by the fact that existing hash function security is not tailored towards this particular use case.

In detail, if one would instantiate the random oracle as a chop-Merkle-Damgård construction [9,10,11] (used in SHA-384, SHA-512, SHA-512/224, and SHA-512/256) or a sponge construction [12] (used in SHA3-{224,256,384,512}, SHAKE128, and SHAKE256), it is possible to rely on the indifferentiability security level [13,11] of said constructions [11,14,15], which implies that the constructions "behave like" random oracles,[1] but these analyses do not consider different types of attack phases and different users at the same time. In addition, if one would instantiate the random oracle as a plain (strengthened) Merkle-Damgård construction [9,10] (used in SHA-224 and SHA-256), it is not even possible to rely on indifferentiability composition as this construction is vulnerable to the length extension attack.[2]

1.1 Novel Model

As first main contribution, we introduce in Section 3.2 a new security model for hash chain based password systems. The model applies not only to T/Key but to any hash chain where a transition from one password to the next one is computed using

$$h(\langle \mathrm{ctr}_k \rangle_t \parallel id \parallel x)\,,$$

where $\langle \mathrm{ctr}_k \rangle_t$ encodes injectively a counter or timestamp, and id is a random salt (identical throughout the entire hash chain) used to avoid pre-computation attacks. For brevity, we denote this more universal hash chain based password system by U/Key. A detailed formalism of the construction is given in Section 3.1.

[1] Refer to Ristenpart et al. [16] for limitations of the indifferentiability framework.

[2] Kogan et al. instantiate T/Key with SHA-256. However, they truncate the output of the hash function, de facto making it a chop-Merkle-Damgård construction.

The model does not restrict itself to the random oracle model; to the contrary, h can be any hash function construction based on an idealized underlying primitive. This, in particular, makes the model directly applicable to popular hash function constructions such as chop-Merkle-Damgård [9,10,11] and the sponge [12], but also plain Merkle-Damgård or any other construction which as a hash function is not indifferentiable from a random oracle. (As a matter of fact, the construction that we consider in Section 5.1 is, for some parameter set, differentiable, while still maintaining a negligible adversary success probability.)

The model also allows for a security analysis in a more fine-grained modeling of the adversary than before: the model refactors the adversarial complexity into offline queries and online queries, a strategy previously explored in a general setting by Ghoshal and Tessaro [17]. In addition, it is defined in the multi-user setting. With these refinements, our model is a strict generalization of the model of Kogan et al. [4].

1.2 Refined Analysis of U/Key

Having settled the novel security model, we perform an updated analysis of hash chain based password systems fitting to the U/Key specification in the random oracle model, thus covering more fine-grained adversaries and a multi-user setting. We prove in Section 4.1 that U/Key is secure up to a bound of the order

$$\mathcal{O}\left(\frac{M}{2^s} \times \frac{q_{off}}{2^n} + \max\left\{\frac{M}{2^s}; 1\right\} \times \frac{q_{on}}{2^n}\right), \tag{3}$$

where n is the length of the passwords, M is the number of users, s the size of the salts, and q_{off} and q_{on} denote respectively the number of offline and online queries. Most notably, this bound indicates that in the multi-user setting, security degrades (i) proportionally to the number of users regarding the offline phase and (ii) proportionally to $\max\left\{\frac{M}{2^s}, 1\right\}$ regarding the online phase. The proof is surprisingly subtle, as a special treatment is needed on whether an earlier offline query is "useful" later on; this treatment differs depending on the number of users.

Next, in Section 4.3 we investigate the implication of this result in case the random oracle is instantiated with a hash function construction that is indifferentiable from a random oracle in the framework of Maurer et al. [13] and Coron et al. [11] (see Section 2.3). In these analyses, we focus on a total query complexity $q = q_{off} + q_{on}$. First, in Section 4.3.1 we compose the hash chain based password systems with the chop-Merkle-Damgård [11,14] construction, obtaining a bound of the order

$$\mathcal{O}\left(\frac{q}{2^n} + \frac{q}{2^{b-n}}\right),$$

where b denotes the output size of the compression function. The designers of T/Key in fact instantiated their construction with a truncated version of SHA-256, and above composition result with chop-Merkle-Damgård applies. However,

closer inspection of the state of the art on hash functions suggests that the result based on the sponge is more relevant. After all, SHA-3 is based on the sponge construction, and the sponge has also gained a lot in popularity over the last years in lightweight applications. Most notably, the NIST Lightweight Cryptography winner Ascon [18,19] is based on the sponge. In Section 4.3.2, we analyze the security of U/Key when the construction is instantiated with a sponge [15], and we obtain a bound of the order

$$\mathcal{O}\left(\frac{q}{2^n} + \frac{q^2}{2^c}\right),$$

where c denotes the capacity. In particular, security is not guaranteed beyond half of the permutation size.

However, we remark that the security game underlying the security of U/Key is in fact a complexified variant of preimage resistance. In addition, it is known that the preimage resistance of the sponge can significantly surpass $c/2$ bits of security [20]. These two observations suggest that the above bounding is lossy. Inspired by this, we derive in Section 5 a dedicated security proof. The complexity of this proof arises from the simultaneous presence of multiple chaining values that can be inverted. Additionally, due to the sponge evaluating the digests in multiple rounds, accounting for the release pattern dictated by U/Key becomes markedly difficult. As a matter of fact, the security proof may be considered of independent interest, as internally it solves the problem of understanding the preimage resistance of a cascaded evaluation of the sponge construction. In the single-user setting, restricted to the simpler T/Key, we derive a bound of the order

$$\mathcal{O}\left(\frac{K\ell(q_{off} + q_{on})}{2^n} + \frac{K\ell^2(q_{off} + q_{on})}{2^c} + \min\left\{\frac{K\ell q_{on}}{2^{n-r}}, \frac{(q_{off} + q_{on})q_{on}}{2^c}\right\}\right), \tag{4}$$

where $r = b - c$, and $\ell := \lceil \frac{n}{r} \rceil$. Our analysis targets an optimization of the sponge in the context of U/Key rather than examining U/Key instantiated with the native sponge. In this approach, the state of the sponge is initialized directly with the counter and the salt, which allows to reduce the number of permutation calls required during the absorption phase (see Section 5.1 for more details). Moreover, our proof is modular and can be easily extended to any keyed construction which has been proven to be indistinguishable from a random function.

Looking at these sponge functions in detail, it can be observed that in the context of hash chains, they can be evaluated very efficiently! For example, a typical instantiation of T/Key operates on counters of 32 bits, salts of 80 bits, and a password of 130 bits [4], thus requiring a hash function from 242 to 130 bits. SHA-3 internally uses a 1600-bit permutation, and processes messages and squeezes digests $1600 - 2k$ bits at a time, where k is the security level. This means that if SHA-3 (any of the four instances) is used in the T/Key hash chain, a hash function evaluation simply consists of a truncated permutation evaluation. With

this in mind, we additionally consider the security of U/Key if the random oracle is instantiated with a truncated permutation in Section 4.3.3. This construction is also proven to be indifferentiable from a random oracle [21], and we obtain a bound of the order

$$\mathcal{O}\left(\frac{q}{2^n} + \frac{q^{3/2}}{2^{\frac{2b-n}{2}}} + \frac{q}{2^{b-(n+t)}}\right), \tag{5}$$

where b denotes the permutation size and t the counter size.

However, as before, indifferentiability composition is still a bit lossy, as the indifferentiability analyses of these constructions have not been tailored towards the specific use case of hash chains: in particular, the bound is expressed in the *total* query complexity q only. Therefore, in Section 6, we take a closer look at U/Key using a truncated permutation and derive a *dedicated* security proof. The proof is of a high level of technicality and contains various subtleties, but on the upside, it is much better than the results obtained with the more general sponge in (4), or by combining (3) with the indifferentiability bound of truncation. In detail, we derive a bound of the order

$$\mathcal{O}\left(\frac{Mq_{off}}{2^{n+s}} + \frac{KMq_{off}}{2^b} + \max\left\{\frac{M}{2^s}; 1\right\} \cdot \frac{q_{on}}{2^n} + \max\left\{\frac{KM}{2^n}; 1\right\} \cdot \frac{q_{on}}{2^{b-n}}\right).$$

Concretely, if we consider counters of $t = 32$ bits, and a setting with less than $M = 10^{12}$ users, then one can use permutations *as small as* $b = 200$ *bits*, while still having security guaranteed up to 2^{100} and 2^{128} queries in the online and offline phases, respectively. This improves significantly over the bound in (5), that cannot guarantee more than 84 bits of security (even during the offline phase) in the single-user setting. Note that the obtained security bound is better than that of T/Key instantiated with a (more general) sponge construction, which is essentially caused by the fact that a hash function attacker has fewer freedom in attacking a single truncated permutation than attacking the sponge.

1.3 Application

Using U/Key with a small truncated permutation can offer both security and efficiency advantages over U/Key based on SHA-2. For example, for counters of $t = 32$ bits, salts of $s = 80$ bits, and passwords of 128 bits, U/Key based on SHA-256 (truncated to 128 bits) operates on a state of 384 bits to achieve 128 bits of security. However, there are no guarantees for multi-user scenarios. On the other hand, U/Key based on Keccak-f[400] operates on a state of 400 bits and achieves 128 bits of security as long as the number of users does not exceed 2^{80}. One can even go as low as a 200-bit state using Keccak-f[200], which can be used for $n = 100$ and $s = 68$, providing 100 and 128 bits of security in the online and offline phases, respectively, as long as the number of users stays below 2^{40}.

1.4 Outline

We present preliminaries in Section 2. The generalized hash chain based password system U/Key that we consider in this work is formalized in Section 3.1

and the novel model is presented in Section 3.2. The security of U/Key in the new model is derived in Section 4: first, a new random oracle model result is derived in Section 4.1, and then this result is combined with indifferentiability security results on chop-Merkle-Damgård, the sponge, and a truncated permutation in Section 4.3. Sections 5 and 6 present improved dedicated security results on U/Key with a sponge and a truncated permutation, respectively. Finally, Section 7 concludes.

2 Preliminaries

2.1 Notation

Let $n, y \in \mathbb{N}$. We use $\{0,1\}^n$ to denote the set of n-bit strings, and $\{0,1\}^*$ to denote the set of all binary strings of any length. Assuming $y \leq 2^n - 1$, $\langle y \rangle_n$ denotes the n-bit binary representation of y. For $b \in \mathbb{N}$, $x \in \{0,1\}^b$, $\mathrm{trunc}_n(x)$ denotes the n-bit string obtained by taking the leftmost n bits of x. In some cases, we also use the notation $\mathrm{outer}_n(x)$ to refer to the same string, but in a different context. The notation $\mathrm{inner}_c(x)$ is used to denote the c-bit string obtained by taking the rightmost c bits of x. If $\mathcal{S}$ is a finite set, $x \xleftarrow{\$} \mathcal{S}$ means that x is sampled uniformly at random from $\mathcal{S}$. Given $a, b \in \mathbb{N}$ such that $a \leq b$, the notation $[\![a, b]\!]$ refers to the set $\{a, a+1, \ldots, b\}$. When we mention a construction based on a primitive, *Prim* denotes the set of all possible primitives. Finally, we use $Perm(b)$ to denote the set of permutations over b bits.

2.2 A Balls-and-Bins Result

In the proof, we use a slightly improved version of a balls-and-bins result of Choi et al. [21]. This result is formally stated in Lemma 1. The proof is in Supplementary Material A for completeness.

Lemma 1. *Let $N, R, n \in \mathbb{N}$ be such that R divides N. Consider N balls in a bin, such that for each $r \in [\![1, R]\!]$, $\frac{N}{R}$ balls have label "r". Consider the experiments of sampling n balls in the bin with replacement. For $r \in [\![1, R]\!]$, let $X^{(r)}$ be the number of balls drawn with label "r". Then*

$$\mathsf{E}\left(\max_r X^{(r)}\right) \leq \frac{2n}{R} + 3\ln(R) + 4\,.$$

The result also holds when the sampling is performed without replacement.

2.3 Indifferentiability

Indifferentiability was introduced by Maurer at al. [13], and refined in the context of hash functions by Coron et al. [11]. Indifferentiability is a distinguishing game where the adversary has access to a public primitive $\mathcal{P}$. The adversaries considered are information-theoretic only, meaning that their complexities are

measured solely in terms of the queries made. Moreover, throughout this work, we assume without loss of generality that the adversary never makes a query for which it already knows the answer. The adversary has access to two oracles, one for construction queries, the other for primitive queries. In the so-called "real world" (or W_R for short), primitive queries give access to a random primitive $\mathcal{P}$, and construction queries to the considered construction $\mathcal{H}$ based on $\mathcal{P}$ (denoted by $\mathcal{H}^{\mathcal{P}}$). On the other hand, in the so-called "ideal world" (or W_I for short), the construction queries give access to a random oracle $\mathcal{RO}$, and the primitive queries are implemented by a simulator $\mathcal{S}^{\mathcal{RO}}$. The indifferentiability advantage of an adversary $\mathcal{A}$ is defined as follows:

$$\mathbf{Adv}_{C}^{\text{iff}}(\mathcal{A}) = \left|\mathbf{Pr}\left(\mathcal{A}^{W_R} = 1\right) - \mathbf{Pr}\left(\mathcal{A}^{W_I} = 1\right)\right| .$$

Moreover, $\mathbf{Adv}_{C}^{\text{iff}}(q)$ denotes the maximum of $\mathbf{Adv}_{C}^{\text{iff}}(\mathcal{A})$, over all adversaries allowed to make at most q queries.

2.4 Preimage Resistance

The security game underlying U/Key (of Section 3.1) is a complexified variant of everywhere preimage resistance, defined in Definition 1 [22].

Definition 1. *Let $\mathcal{H}$ be a construction relying on a primitive $\mathcal{P} \in Prim$, producing digests of size n. The everywhere preimage resistance of $\mathcal{H}$ against an adversary $\mathcal{A}$ is defined as follows:*

$$\mathbf{Adv}_{\mathcal{H}}^{\text{ePre}}(\mathcal{A}) = \max_{Z \in \{0,1\}^n} \mathbf{Pr}\left(\mathcal{P} \xleftarrow{\$} Prim, M \leftarrow \mathcal{A}(Z) : \mathcal{H}(M) = Z\right) .$$

Moreover, $\mathbf{Adv}_{\mathcal{H}}^{\text{ePre}}(q)$ denotes the supremum of $\mathbf{Adv}_{\mathcal{H}}^{\text{ePre}}(\mathcal{A})$, over all adversaries allowed to make at most q primitive queries.

3 Hash Chain Based Password System

The generic construction is described in Section 3.1 and the security model in Section 3.2.

3.1 Construction

We will consider a construction that is a bit more general and abstract than T/Key, which in turn was already a generalization of S/Key. For brevity, we will dub our construction U/Key, where U stands for "universal". After the description, we will discuss in more detail how T/Key fits this construction.

Let K, n, s, t be integers, and $h : \{0,1\}^{n+s+t} \to \{0,1\}^n$ a cryptographic hash function. We will describe a hash chain based password system based on h, U/Key[h], with a chain length of K, a password size of n bits, salts of s bits, and a counter over t bits. It is defined through an initialization phase, a chain computation phase, and an authentication phase:

1. Initialization: the client draws a password $x_0 \xleftarrow{\$} \{0,1\}^n$ and a salt $id \xleftarrow{\$} \{0,1\}^s$;
2. Chain computation: for $k = 1, \ldots, K$, the client evaluates $x_k = h_k(x_{k-1})$, where h_k is defined as follows:

$$h_k(x) = h\left(\langle \mathrm{ctr}_k \rangle_t \parallel id \parallel x\right),$$

   where ctr_k is some counter value that may be specific to the actual system. After this computation, the client sends id and x_K to the server;
3. Authentication: if the client wants to authenticate at round k, and the last authentication was at round $k' > k$,[3] it sends the password x_k to the server. The server on its side retrieves the latest password it has stored, $x_{k'}$, and verifies if

$$h_{k'}(h_{k'-1}(\cdots(h_{k+1}(x_k))\cdots)) = x_{k'}$$

   holds. It updates the stored round to k and password to x_k. Note that, depending on the concrete underlying scheme, the counter k may need to be sent along with the password x_k. In the case of T/Key, there is a resynchronization in place based on different successful attempts. This could be used as alternative solution.

Note that, for authentication, the client does not necessarily have to store the whole chain on its side. Instead, it can opt to store certain "checkpoints" [4], or perform an efficient pebbling method [23]. These approaches offer a trade-off between memory and computation time required to evaluate the hashes.

T/Key. The T/Key system [4], specifically, is covered by above description of U/Key with two refinements. First, upon initialization, the client takes the current timestamp t_i and takes as counter values $\mathrm{ctr}_k = t_\mathrm{i} + K - k$, and the value of t_i is sent to the server along with the salt id and the last chaining value x_K. The chain is processed in a pre-described time-based manner, where any transition happens each predetermined period. This means that upon authentication the difference between k' and k depends on the time the user authenticated last.

For reference, the designers of T/Key suggest to use this construction with the following typical parameters:

$$K = 2 \times 10^6\,, \qquad n = 130\,, \qquad s = 80\,, \qquad t = 32\,.$$

Furthermore, each password is valid for 30 seconds (so the chain is reversed each 30 seconds).

3.2 Security Model

We will describe security of U/Key of Section 3.1. It will not be defined for a hash function h but rather for a hash function construction $\mathcal{H}$ based on idealized

[3] Typically, $k' = k + 1$, but for a time-based system such as T/Key, this is not necessarily the case.

primitive $\mathcal{P} \in \mathit{Prim}$: we denote this by U/Key$[\mathcal{H}^{\mathcal{P}}]$. The security game is defined in Algorithm 1. Here, we consider an adversary $\mathcal{A}$ that can make queries to the idealized primitive $\mathcal{P}$ in order to break the scheme. The attack game consists of two phases. In the first one, called the *offline phase*, the adversary $\mathcal{A}$ is allowed to make pre-computation queries, which translates to $\mathcal{A}$ being able to make at most q_{off} primitive queries. In the second phase, called the *online phase*, the adversary $\mathcal{A}$ has access to $M \geq 1$ parallel runs of the construction, where each hash chain has a length K. For simplicity, we assume that the initializations of all sessions are performed at the same time, and so are the traversals through the chains. The online phase itself is subdivided into K rounds, and in each of these rounds $\mathcal{A}$ is allowed to make at most $q_{on,k}$ primitive queries. The total number of online queries is denoted by $q_{on} := \sum_{k=1}^{K} q_{on,k}$. Note that in reality the offline and online phases of the different users are not necessarily synchronized, as online queries of one user could be made during the offline phase of another user. Thus, q_{off} represents the maximum, over all users, of offline queries. Therefore, we believe that enforcing that all users start at the same time in the model does not result in a loss of generality.

For $k \in [\![1, K]\!], m \in [\![1, M]\!]$, let $x_{k,m}$ be the chaining value number k of user number m, and let $\mathcal{H}^{\mathcal{P}}_{k,m}(x) = \mathcal{H}^{\mathcal{P}}(\langle \mathrm{ctr}_k \rangle_t \parallel id_m \parallel x)$. Without loss of generality, we assume that the adversary proposing x as a preimage of $x_{k,m}$ has made all necessary primitive queries to evaluate $\mathcal{H}^{\mathcal{P}}_{k,m}(x)$. We define $\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{H}]}\left(q_{off}, (q_{on,k})_{k\in[\![1,K]\!]}, M\right)$ to be the maximum success probability in SecGame$(\mathcal{A}, \mathcal{H}, q_{off}, (q_{on,k})_{k\in[\![1,K]\!]}, M)$ of Algorithm 1, maximized over all adversaries $\mathcal{A}$. Note, when the security bound is independent on the way the online queries $(q_{on,k})_k$ are distributed (which is the case for our random oracle model analysis and that of the truncated permutation construction), we rather use the notation $\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{H}]}\left(q_{off}, q_{on}, M\right)$.

Adversarial Resources. The adversary is bounded by a certain offline complexity q_{off} and by online complexities $q_{on,k}$ for $k \in [\![1, K]\!]$. Note that the offline complexity may be rather high. The adversary can do pre-computations (possibly on more powerful computers), and we would typically aim at around 128-bit security, which is the level of generic security guaranteed by, e.g., SHA-256 or SHA3-256. This means that q_{off} may get as high as 2^{128}. (Of course, storage may then be a problem for the adversary; refer to Section 7.2 for further discussion regarding this.)

The fine-grained definition of the online complexities allows for analyzing the hash chain based system U/Key of Section 3.1 in full generality, noting that if a user leaves a big gap between two authentication phases, the adversary has more time to attack the scheme *in that round*. That said, for T/Key each period is of the same length, e.g., 30 seconds as suggested by its designers. In this case, all online complexities $q_{on,k}$ are (roughly) identical. A rough bound for the online complexities in U/Key can be derived from the computations for the Bitcoin blockchain. It is estimated [24] that currently all Bitcoin miners in total perform $400 \times 10^{18} \approx 2^{68}$ SHA-256 evaluations per second. In particular, 2^{100} evaluations would require more than 100 years of computation. Although this is a very rough

Algorithm 1: Security model for U/Key based on the construction $\mathcal{H}$, itself relying on a primitive $\mathcal{P} \in \mathit{Prim}$. We write $\mathcal{H}^{\mathcal{P}}_{k,m}(x) := \mathcal{H}^{\mathcal{P}}(\langle \mathrm{ctr}_k \rangle_t \parallel id_m \parallel x)$ for brevity.

Function SecGame$\Big(\mathcal{A}, \mathcal{H}, q_{off}, (q_{on,k})_{k \in [\![1,K]\!]}, M\Big)$
  $\mathcal{P} \xleftarrow{\$} \mathit{Prim}$;
  OfflinePhase$(\mathcal{A}, \mathcal{P})$;
  OnlinePhase$(\mathcal{A}, \mathcal{H}, \mathcal{P})$;
Function OfflinePhase$(\mathcal{A}, \mathcal{P})$
  $\mathcal{A}$ is allowed to make at most q_{off} queries to $\mathcal{P}$;
Function OnlinePhase$(\mathcal{A}, \mathcal{H}, \mathcal{P})$
  for $m = 1, \ldots, M$ **do**
    `/* consider` M `independent initializations */`
    $id_m, (x_{0,m}, x_{1,m}, \ldots, x_{K,m}) \leftarrow \mathrm{U/Key}[\mathcal{H}^{\mathcal{P}}]$;
  $k \leftarrow K$;
  $\mathcal{A}$ is given id_m for all $m \in [\![1, M]\!]$;
  while $k > 0$ **do**
    $\mathcal{A}$ is given $x_{k,m}$ for all $m \in [\![1, M]\!]$;
    $\mathcal{A}$ is allowed to make at most $q_{on,k}$ queries to $\mathcal{P}$;
    PreimageGuess$(\mathcal{A}, \mathcal{H}, \mathcal{P}, x_{k,1}, \ldots, x_{k,M})$;
    $k = k - 1$;
Function PreimageGuess$(\mathcal{A}, \mathcal{H}, \mathcal{P}, x_{k,1}, \ldots, x_{k,M})$
  $\mathcal{A}$ is allowed to submit $y \in \{0,1\}^n$ and wins if there exists $m \in [\![1, M]\!]$ such that $\mathcal{H}^{\mathcal{P}}_{k,m}(y) = x_{k,m}$;

upper bound, in particular if a different hash function is selected, we can use it to upper bound $q_{on,k}$.

4 Random Oracle Results with U/Key

Given the novel security model with refined treatment of adversarial power, it makes sense to have a look at U/Key based on a random oracle and to supersede the original security result on T/Key in the new model. We do so in Section 4.1, with the proof in Section 4.2. In this section, we also particularly elaborate on the impact of the split of adversarial complexity into offline and online complexity.

Then, in Section 4.3 we investigate the meaning of the result if the random oracle is instantiated with some well-established construction. In detail, we consider the instantiation with chop-Merkle-Damgård in Section 4.3.1, with the sponge in Section 4.3.2, and with the truncated permutation in Section 4.3.3.

4.1 U/Key Security in the New Model

In Theorem 1 we state the security of U/Key in the new security model when instantiated with a random oracle.

Theorem 1. *Let $\mathcal{RO} : \{0,1\}^{n+s+t} \to \{0,1\}^n$ be a random oracle, and consider the hash function $\mathcal{H}^{\mathcal{RO}} : \{0,1\}^{n+s+t} \to \{0,1\}^n$ as $\mathcal{H}^{\mathcal{RO}}(x) = \mathcal{RO}(x)$. Then, we have*

$$\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{H}]}\left(q_{off}, q_{on}, M\right) \leq \left(\frac{2M}{2^s} + 3s + 4\right) \times \left(\min\left\{\frac{2M}{2^s}; 1\right\} \times \frac{2q_{off}}{2^n} + \frac{2q_{on}}{2^n}\right).$$

The proof is given in Section 4.2.

Interpretation of the Bound. Remember that the total number of queries is given by $q_{off} + q_{on}$. If $M \leq 2^{s-1}$, the bound is of the order

$$\mathcal{O}\left(\frac{M}{2^s} \times \frac{q_{off}}{2^n} + \frac{q_{on}}{2^n}\right).$$

Thus in the multi-user setting, the offline phase displays a security loss proportional to M, while there is no security loss for the online phase. This is explained by the fact that in the offline phase, the adversary is not provided with the salts and must thus correctly guess them. Increasing the number of users increases the adversary probability to make a query with one correct salt by a factor of M. On the other hand, if $M \geq 2^{s-1}$, then the bound is of the order

$$\mathcal{O}\left(\frac{M}{2^s} \times \left(\frac{q_{off} + q_{on}}{2^n}\right)\right).$$

Indeed, when $M \gg 2^s$, for a well-chosen adversarial strategy, then (almost) all queries of the offline phase are expected to correspond to some existing salt. Moreover in this setting, some salts are expected to collide, thus one query from the adversary can target several chaining values at the same time, hence the factor of $\frac{M}{2^s} > 1$.

The security bound allows for many different interpretations due to its flexibility in the number of users and the number of offline and online queries. For example, for a typical instantiation of our scheme with $q_{off} \ll 2^{128}$, $K = 2^{21}$, and $q_{on} \ll 2^{100}$, with a salt size $s = 80$ and password size $n = 100$, one can guarantee security as long as the number of users is at most $M \leq 2^{52}$.

4.2 Proof of Theorem 1

For $k \in [\![1, K]\!], m \in [\![1, M]\!]$, let $x_{k,m}$ be the chaining value number k of user number m ($x_{0,m}$ denotes the root passwords). Without loss of generality, we assume that the salts are drawn before the offline phase. We define a random variable Coll_s that determines how the salts are colliding together. This variable is represented as a vector with M coordinates, each element taking values in $[\![1, M]\!]$. When two elements at indices i and j share the same value, it means that the salts of users i and j collide. Fixing Coll_s determines the number of distinct salts, that we will denote as $\tilde{M}$. Conditioning on this variable does not provide information about the randomness of the $\tilde{M}$ distinct salt values, except

for the fact that they are sampled without replacement. Moreover, let NC_s be a random variable counting the maximum number of colliding salts, i.e.,

$$\mathrm{NC}_s = \max_{id} \#\left\{m \in [\![1, M]\!] \;\middle|\; id_m = id\right\} .$$

In particular, the value of Coll_s determines the one of NC_s. We can apply Lemma 1 with $N = R = 2^s$, $n = M$, and obtain

$$\mathsf{E}\left(\mathrm{NC}_s\right) \leq \frac{2M}{2^s} + 3s + 4 . \tag{6}$$

Assume that the salts are colliding according to a certain $\mathrm{coll}_s \in ([\![1, M]\!])^M$. Fixing this random variable fixes the number of distinct salts, that we denote by $\tilde{M} \leq M$. Moreover, let $k \in [\![1, K]\!]$, and $\tilde{m} \in [\![1, \tilde{M}]\!]$. Define $h_{k,\tilde{m}}$ to be $h\left(\langle \mathrm{ctr}_k \rangle_t \parallel id_{\tilde{m}} \parallel \cdot\right)$, where $id_{\tilde{m}}$ is the $\tilde{m}^{\text{th}}$ distinct salt.

The adversary's success in winning the security game relies on finding a preimage of one of the chaining values, which can only be obtained through $h_{k,\tilde{m}}$-queries. Since h is a random oracle, having knowledge of other outputs that are not obtained through $h_{k,\tilde{m}}$-queries does not enhance the adversary's probability of success. Therefore, such queries are considered "useless". In particular, during the offline phase the adversary has no access to the salts, thus it has to guess them, and whenever $M \ll 2^s$, this significantly lowers the adversarial success probability. To capture this phenomenon, we use the random variables $Q_{off}^{(k,\tilde{m})}$ which counts the number of useful $h_{k,\tilde{m}}$-queries during the offline phase. Moreover, let $q_{on}^{(k,\tilde{m})}$ be the number of $h_{k,\tilde{m}}$-queries during the online phase. Note that this quantity is not a random variable, since the adversary is given the salts during the online phase. Moreover, we partition the offline (resp., online) queries according to the counter value queried, i.e., $q_{off}^{(k)}$ (resp., $q_{on}^{(k)}$) counts the number of queries of the form $h(\langle \mathrm{ctr}_k \rangle_t \parallel \cdot \parallel \cdot)$. Finally, let

$$Q^{(k,\tilde{m})} = Q_{off}^{(k,\tilde{m})} + q_{on}^{(k,\tilde{m})} . \tag{7}$$

Then, it holds that

$$\begin{aligned}
\sum_{k=1}^{K} \sum_{m=1}^{\tilde{M}} Q_{off}^{(k,\tilde{m})} &\leq \sum_{k=1}^{K} q_{off}^{(k)} \leq q_{off} , \\
\sum_{k=1}^{K} \sum_{m=1}^{\tilde{M}} q_{on}^{(k,\tilde{m})} &\leq \sum_{k=1}^{K} q_{on}^{(k)} \leq q_{on} , \\
\sum_{k=1}^{K} \sum_{m=1}^{\tilde{M}} Q^{(k,\tilde{m})} &\leq q_{off} + q_{on} .
\end{aligned} \tag{8}$$

Furthermore, when $M \leq 2^{s-1}$, we have

$$\mathsf{E}\left(Q_{off}^{(k,\tilde{m})} \;\middle|\; \mathrm{Coll}_s = \mathrm{coll}_s\right) \leq \frac{q_{off}^{(k)}}{2^s - M} \leq \frac{2q_{off}^{(k)}}{2^s} . \tag{9}$$

The adversary $\mathcal{A}$ wins the game whenever there exists $k \in [\![1, K]\!], \tilde{m} \in [\![1, \tilde{M}]\!]$ such that $\mathcal{A}$ found a preimage of one of the x_{k,m_i}, where all m_i's have the same salt equal to the $\tilde{m}^{\text{th}}$ distinct salt (corresponding thus to a $h_{k,\tilde{m}}$-query). Denote this event by $\mathsf{HIT}_{k,\tilde{m}}$. We have

$$\begin{aligned}\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{H}]}(q_{off}, q_{on}, M) &\leq \sum_{\mathrm{coll}_s} \mathbf{Pr}\left(\bigvee_{k,\tilde{m}} \mathsf{HIT}_{k,\tilde{m}} \wedge \mathrm{Coll}_s = \mathrm{coll}_s\right) \\ &\leq \sum_{\mathrm{coll}_s} \sum_{\tilde{m}=1}^{\tilde{M}} \sum_{k=1}^{K} \underbrace{\mathbf{Pr}\left(\mathsf{HIT}_{k,\tilde{m}} \wedge \bigwedge_{k'<k} \neg\mathsf{HIT}_{k',\tilde{m}} \wedge \mathrm{Coll}_s = \mathrm{coll}_s\right)}_{(10)}. \qquad (11)\end{aligned}$$

Here, we introduce the condition $\bigwedge_{k'<k} \neg\mathsf{HIT}_{k',\tilde{m}}$, since if an adversary hits an earlier chaining value, it can set $\mathsf{HIT}_{k,\tilde{m}}$ by making cascaded evaluations. Now, we can split (10) according to the value of $Q^{(k,\tilde{m})}$. Therefore,

$$\begin{aligned}(10) \leq \mathbf{Pr}(\mathrm{Coll}_s = \mathrm{coll}_s) \sum_{q^{(k,\tilde{m})}} \mathbf{Pr}\left(Q^{(k,\tilde{m})} = q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \times \\ \mathbf{Pr}\left(\mathsf{HIT}_{k,\tilde{m}} \mid \bigwedge_{k'<k} \neg\mathsf{HIT}_{k',\tilde{m}} \wedge \mathrm{Coll}_s = \mathrm{coll}_s \wedge Q^{(k,\tilde{m})} = q^{(k,\tilde{m})}\right).\end{aligned}$$

For a given coll_s, let nc_s be the corresponding value of the random variable NC_s. There are two possibilities for the adversary to set the conditioned $\mathsf{HIT}_{k,\tilde{m}}$:

- The adversary guesses one *exact* preimage, or in other words, makes a $h_{k,\tilde{m}}$-query with input x_{k-1,m_i}, where user m_i has the salt number $\tilde{m}$. In that case, since the functions $h_{k,\tilde{m}}$ are independent random oracles, each preimage is sampled uniformly at random, and knowledge of the values $x_{k'',m}$ for $k'' > k$ does not help the adversary. One $h_{k,\tilde{m}}$-query can target simultaneously at most nc_s different chaining values. Therefore, the conditioned probability of this case of $\mathsf{HIT}_{k,\tilde{m}}$ is upper bounded by $\frac{\mathrm{nc}_s q^{(k,\tilde{m})}}{2^n}$;
- The adversary finds a preimage, which is not the *exact* one. Again, one $h_{k,\tilde{m}}$-query can target simultaneously at most nc_s different chaining values, and each new query results in a uniformly random output. Thus, the conditioned probability of this case of $\mathsf{HIT}_{k,\tilde{m}}$ is upper bounded by $\frac{\mathrm{nc}_s q^{(k,\tilde{m})}}{2^n}$.

Therefore,

$$(11) \leq \sum_{\mathrm{coll}_s} \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \times \sum_{\tilde{m}=1}^{\tilde{M}} \sum_{k=1}^{K} \sum_{q^{(k,\tilde{m})}} \mathbf{Pr}\left(Q^{(k,\tilde{m})} = q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \frac{2\mathrm{nc}_s q^{(k,\tilde{m})}}{2^n}$$

$$\leq \sum_{\mathrm{coll}_s} \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \sum_{\tilde{m}=1}^{\tilde{M}} \sum_{k=1}^{K} \frac{2\mathrm{nc}_s}{2^n} \mathsf{E}\left(Q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right)$$

$$\leq \sum_{\mathrm{coll}_s} \frac{2\mathrm{nc}_s}{2^n} \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \mathsf{E}\left(\sum_{\tilde{m}=1}^{\tilde{M}} \sum_{k=1}^{K} Q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right). \qquad (12)$$

We derive two different upper bounds for the expectation. Both always hold, but the best upper bound depends on the value of $\frac{M}{2^s}$.

Case 1. This case will give the best bound when $M \leq 2^{s-1}$. From (7) and (9), we obtain

$$\mathsf{E}\left(Q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \leq q_{on}^{(k,\tilde{m})} + \frac{2q_{off}^{(k)}}{2^s}.$$

Therefore, using (8),

$$\mathsf{E}\left(\sum_{\tilde{m}=1}^{\tilde{M}} \sum_{k=1}^{K} Q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \leq q_{on} + \frac{2M}{2^s} q_{off}. \qquad (13)$$

Case 2. This case will give the best upper bound when $M \geq 2^{s-1}$. In this setting, the salts are expected to cover a large portion of the space $\{0,1\}^s$, therefore trying to count the offline queries that are useful to the adversary does not give an improved probability. We instead use (8) to obtain

$$\mathsf{E}\left(\sum_{\tilde{m}=1}^{\tilde{M}} \sum_{k=1}^{K} Q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \leq q_{on} + q_{off}. \qquad (14)$$

Conclusion. Combining (13) and (14) into (12) gives

$$\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{H}]}\left(q_{off}, q_{on}, M\right) \leq \sum_{\mathrm{coll}_s} \frac{2\mathrm{nc}_s}{2^n} \times \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \left(q_{on} + \min\left\{\frac{2M}{2^s}; 1\right\} \times q_{off}\right)$$

$$\leq \frac{2}{2^n} \times \mathsf{E}\left(\mathrm{NC}_s\right) \times \left(q_{on} + \min\left\{\frac{2M}{2^s}; 1\right\} \times q_{off}\right).$$

Finally, we can use the bound for $\mathsf{E}\left(\mathrm{NC}_s\right)$ found in (6) to obtain

$$\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{H}]}\left(q_{off}, q_{on}, M\right) \leq \left(\frac{2M}{2^s} + 3s + 4\right) \times \left(\min\left\{\frac{2M}{2^s}; 1\right\} \times \frac{2q_{off}}{2^n} + \frac{2q_{on}}{2^n}\right).$$

4.3 Composition With Indifferentiable Hash Functions

In this section we study the security of U/Key instantiated with several indifferentiable hash function constructions in the single-user setting, i.e., if $M = 1$. Bounds in the multi-user setting can be obtained by using generic single-user to multi-user reductions [7,8], which result in multiplying all bounds by a factor of M. However, since our goal is to have a first idea of the tightness of the bounds, for simplicity we only consider the single-user setting.

If a hash function is indifferentiable from a random oracle, this means that it behaves as such and that it can replace a random oracle in almost any practical use case. Andreeva et al. [25] made the composition explicit. Translated to our case, their reduction means that given a hash function construction $\mathcal{H}$, the advantage of the adversary in breaking the security game from Algorithm 1 can be upper bounded as follows:

$$\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{H}]}\left(q_{off}, q_{on}, 1\right) \leq \mathbf{Adv}_{\mathrm{U/Key}[\mathcal{RO}]}\left(q_{off}, q_{on}, 1\right) + \mathbf{Adv}^{\mathrm{iff}}_{\mathcal{H}}\left(q_{off} + q_{on}\right) , \tag{15}$$

where we abuse notation and use $\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{RO}]}\left(q_{off}, q_{on}, 1\right)$ to refer to the advantage of an adversary in breaking Algorithm 1 when the hash function is a random oracle. In other words, this term is bounded in Theorem 1, and all we need to do is to obtain the indifferentiability bound for the hash function construction $\mathcal{H}$.

Admittedly, looking ahead, this generic reduction is not tight. The reason is that the indifferentiability term $\mathbf{Adv}^{\mathrm{iff}}_{\mathcal{H}}\left(q\right)$ takes as security parameter the total number of primitive evaluations $q = q_{off} + q_{on}$ rather than its separation into offline and online queries. This already suggests that a direct analysis will likely give a better bound (and we will do so in Sections 5 and 6). Yet, there is still value in investigating the guaranteed security through composition, which we will do for chop-Merkle-Damgård in Section 4.3.1, the sponge in Section 4.3.2, and truncated permutation in Section 4.3.3. However, as in this reasoning the indifferentiability bounds will be the dominating factors anyway, we will simplify the result of Theorem 1 and simply use that

$$\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{RO}]}\left(q_{off}, q_{on}, 1\right) = \mathcal{O}\left(\frac{q}{2^n}\right) . \tag{16}$$

4.3.1 Chop-Merkle-Damgård Let $b, u, n \in \mathbb{N}$ such that $n < b$. The chop-Merkle-Damgård (or chop-MD for short) construction operates on top of a compression function $\mathcal{F} : \{0,1\}^u \times \{0,1\}^b \to \{0,1\}^b$. The input message $M \in \{0,1\}^*$ is first injectively padded into u-bit blocks as $m_1 \parallel \cdots \parallel m_\ell$. Let $IV \in \{0,1\}^b$ be a fixed initialization vector. Then, chop-MD computes its output as

$$\mathrm{trunc}_n\left(\mathcal{F}\left(m_\ell, \mathcal{F}\left(m_{\ell-1}, \mathcal{F}\left(\cdots \mathcal{F}\left(m_1, IV\right)\right)\right)\right)\right) .$$

Chang and Nandi [14] showed that the chop-MD construction with a state of size b bits and $b - n$ bits truncated is indifferentiable from a random oracle

up to bound

$$\mathbf{Adv}^{\mathrm{iff}}_{\mathrm{ChopMD}}(q) = \mathcal{O}\left(\frac{q}{2^n} + \frac{q}{2^{b-n}}\right).$$

From (15) and (16), we obtain that U/Key instantiated with the chop-MD construction achieves the following level of security:

$$\mathbf{Adv}_{\mathrm{U/Key[ChopMD]}}(q_{off}, q_{on}, 1) = \mathcal{O}\left(\frac{q}{2^n}\right) + \mathcal{O}\left(\frac{q}{2^n} + \frac{q}{2^{b-n}}\right).$$

Note that the designers of T/Key suggested to use SHA-256 truncated to 130 bits of output. This corresponds to the chop-MD construction, giving 126 bits of security.

4.3.2 Sponge Let $b, c, r \in \mathbb{N}$ such that $b = c + r$. The sponge construction operates on top of a permutation $\mathcal{P} : \{0,1\}^b \to \{0,1\}^b$. The input message $M \in \{0,1\}^*$ is first injectively padded into r-bit blocks as $m_1 \parallel \cdots \parallel m_{\ell'}$, under the condition that the last block is non-zero. Let $IV \in \{0,1\}^b$ be a fixed initialization vector. The sponge absorbs its message blocks as

$$S = (m_{\ell'} \parallel 0^c) \oplus \mathcal{P}\Big((m_{\ell'-1} \parallel 0^c) \oplus \mathcal{P}\big(\cdots \oplus \mathcal{P}\left((m_1 \parallel 0^c) \oplus IV\right)\big)\Big),$$

and squeezes its n-bit digest r bits at a time as

$$\mathrm{trunc}_n\left(\mathrm{outer}_r(\mathcal{P}(S)) \parallel \mathrm{outer}_r(\mathcal{P}^2(S)) \parallel \cdots \parallel \mathrm{outer}_r(\mathcal{P}^\ell(S))\right),$$

where $\ell = \lceil \frac{n}{r} \rceil$.

The sponge has been proven to be indifferentiable from a random oracle up to bound [15]

$$\mathbf{Adv}^{\mathrm{iff}}_{\mathrm{Sponge}}(q) = \mathcal{O}\left(\frac{q^2}{2^c}\right).$$

From (15) and (16), we obtain that U/Key instantiated with the sponge construction achieves the following level of security:

$$\mathbf{Adv}_{\mathrm{U/Key[Sponge]}}(q_{off}, q_{on}, 1) = \mathcal{O}\left(\frac{q}{2^n}\right) + \mathcal{O}\left(\frac{q^2}{2^c}\right).$$

In particular, one cannot have a better security than half of the permutation size.

With $t = 32$, and a security goal of 128 bits (thus $n \geq 128$), in the single-user case the minimal permutation size is 257 bits, but this requires at least 288 primitive evaluations to compute one element on the chain as it processes with rate $r = 1$. Taking a slightly larger primitive size, such as $b = 320$ of Ascon-Hash [18,19], allows for a rate of $r = 64$ bits, and one requires 5 permutation calls per hash function evaluation (3 for absorbing, and 2 for squeezing). If we extend the analysis to multiple users and to a salt of size 80 bits, the numbers scale to 368 primitive evaluations, or $4 + 2$ primitive evaluations, respectively.

4.3.3 Truncated Permutation As the hash function is typically used for relatively small data, one might instead consider the truncated permutation construction. As we will see now, this significantly improves the security bound.

Let $b \in \mathbb{N}$ such that $b \geq n + s + t$. The truncated permutation construction operates on top of a permutation $\mathcal{P} : \{0,1\}^b \to \{0,1\}^b$. Due to the condition that $b \geq n + s + t$, we can absorb all data in one permutation call. In detail, for an arbitrary message $M \in \{0,1\}^{n+s+t}$, the truncated permutation computes its output as

$$\mathrm{trunc}_n\left(\mathcal{P}(M \parallel 0^{b-n-s-t})\right) .$$

Choi et al. [21] showed that the truncated permutation construction for fixed salt (i.e., in the single-user setting) is indifferentiable from a random oracle up to bound

$$\mathbf{Adv}^{\mathrm{iff}}_{\mathrm{TruncP}}(q) = \mathcal{O}\left(\frac{q^{3/2}}{2^{\frac{2b-n}{2}}} + \frac{q}{2^{b-(n+t)}}\right) .$$

(In the multi-user setting, when there are multiple random salts, one can rely on the result of Grassi and Mennink [26].) From (15) and (16), we obtain that U/Key instantiated with the truncated permutation construction achieves the following level of security:

$$\mathbf{Adv}_{\mathrm{U/Key[TruncP]}}(q_{off}, q_{on}, 1) = \mathcal{O}\left(\frac{q}{2^n}\right) + \mathcal{O}\left(\frac{q^{3/2}}{2^{\frac{2b-n}{2}}} + \frac{q}{2^{b-(n+t)}}\right) .$$

Observe that, comparing the bounds of Section 4.3.2 and 4.3.3, using a truncated permutation typically gives a better choice than the sponge whenever $n+t \leq \frac{b}{2}$, since this choice allows to minimize the number of primitive calls and maximizes the security at the same time.

With $t = 32$, and a security goal of 128 bits (thus $n \geq 128$), the minimal permutation size is 288 bits and a single permutation evaluation is made. In other words, for the current use case, a truncated permutation is superior over the sponge whenever the used permutation is at least 288 bits. If one uses a smaller permutation, one cannot achieve 128-bit security using the truncated permutation construction, while it may still be possible using the sponge (provided $b \geq 257$) at the cost of extra permutation evaluations.

5 Dedicated Security Proof of U/Key with a Sponge

From the results of Section 4.3 it is clear that indifferentiability is a too strong security property for all of these constructions. For the case of the sponge, on the one hand its indifferentiability bound is undeniably tight since inner collisions within the sponge states can be found in approximately $2^{c/2}$ queries, and those inner collisions can be used among others to find collisions and second preimages. On the other hand, it has been shown that finding a preimage (according to Definition 1) of a value cannot be done in less than 2^{n-r} queries [20]. In

particular, when $n - r > c/2$, this gives a better bound than indifferentiability, thus suggesting that U/Key could as well benefit from this security bound. However, preimage resistance is not the exact security property that we are seeking. Firstly, this is because the target value comes itself from a previous sponge call, which does not correspond to the setting of everywhere preimage (Definition 1). Additionally, we are not aware of results that account for the singular nature of the cascaded hash function evaluations or the release pattern of the chaining values. Therefore, a dedicated security proof seems unavoidable.

The remainder of this section is organized as follows. Section 5.1 describes the exact scheme that we study and the rationale behind it. Section 5.2 recalls the concept of outer-keyed sponges and PRF security, and defined a security variant needed for our proof. Finally, Section 5.3 is dedicated to the main result of this section, and Section 5.4 to the corresponding proof.

5.1 Optimization of Sponge-Based Instantiation

According to the description of U/Key, the chaining value, the counter, as well as the salt are seen as inputs to the hash function. In the context of a sponge (cf., Section 4.3.2), this means that the string $\langle \mathrm{ctr}_{K-k+1} \rangle_t \parallel id \parallel x_{K-k}$ is padded into blocks of r bits, and each of these blocks is added one by one to the outer part of the state of the sponge. For a typical instantiation of U/Key, where the counter and salt can be of total size 112 bits, this induces a significant overhead regarding the number of required permutation evaluations if the rate is small. On the other hand, from a security perspective, this overhead appears to be unnecessary. Indeed, from the perspective of the security proof, there is no distinction between absorbing the counter and the salt into the sponge by blocks of r bits or directly initializing the state with these values. Strictly speaking, in the former case, the initial states are randomized, but this modification does not affect the proof.

Therefore, we consider a family of sponges with initial states of the form $IV_{k,id}$, where

$$IV_{k,id} = IV \parallel \langle \mathrm{ctr}_{K-k+1} \rangle_t \parallel id_m \,,$$

and $IV \in \{0,1\}^{b-t-s}$ can be any value fixed in advance. Note that this is possible only if the capacity is large enough (i.e., $c \geq t + s$). However, looking forward to the main result in this section, namely Theorem 2, for a typical use case of $t = 32, s = 80$, a capacity less than 112 bits does not provide a decent level of security anyway. One round of our refined scheme is depicted in Fig. 1. For simplicity, we consider the simple padding 10^*, which appends to the message a 1 and as much 0's as necessary to obtain a padded message with a length multiple of r. Therefore, the number of rounds necessary to absorb the ℓ message blocks is equal to ℓ or $\ell + 1$.

Note that the construction that is instantiated in U/Key is similar to the sponge variant as used in PHOTON [27], where the initial absorbing rate can also be larger (in our case, $r' = r + t + s$). However, that construction keeps

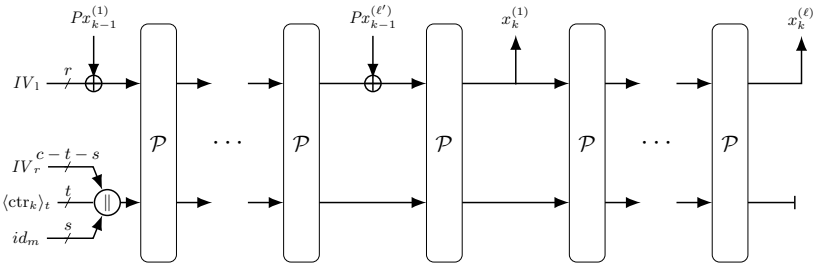

Fig. 1. U/Key with a tweaked version of the sponge. The figure illustrates the computation of $x_k = \mathrm{trunc}_n\left(x_k^{(1)} \parallel \cdots \parallel x_k^{(\ell)}\right)$ from x_{k-1}. Here, $Px_{k-1}^{(1)} \parallel \cdots \parallel Px_{k-1}^{(\ell')} = \mathrm{pad}(x_{k-1})$, where $\ell \leq \ell'$, and $IV := IV_l \parallel IV_r$ is fixed.

an indifferentiability result comparable to the sponge as long as $r' < r + c/2 - \log_2(c)$ [28], but we will show that even beyond this threshold security is still achieved. We stress that the security bound obtained in Theorem 2 holds both for this optimized version of the sponge and the plain sponge construction. In the remainder of this section, we will abuse notation and keep calling this optimized construction the sponge.

5.2 Outer-Keyed Sponges, PRF-Security, and Key-Reveal PRF-Security

Before going to our main result of this section (Theorem 2 in Section 5.3), we will recall the concept of an outer-keyed sponge from Andreeva et al. [29]. Next, we formalize the corresponding notion of PRF security, and define the notion of key-reveal PRF security.

5.2.1 Outer-Keyed Sponge Let $\mathcal{P} \in \mathit{Perm}(b)$, and $m \in \mathbb{N}$. We denote by $\mathrm{Sponge}_m^{\mathcal{P}}$ the construction defined in Section 4.3.2 based on the permutation $\mathcal{P}$ with a digest of size m. The outer-keyed sponge based on the permutation $\mathcal{P}$ with key $\mathcal{K} \in \{0,1\}^\kappa$, is denoted as $\mathrm{OKS}_{\mathcal{K}}^{\mathcal{P}}$ and defined as follows, for any (possibly empty) message M:

$$\mathrm{OKS}_{\mathcal{K}}^{\mathcal{P}}(M, m) = \mathrm{Sponge}_m^{\mathcal{P}}(\mathcal{K} \parallel M)\,.$$

5.2.2 PRF Security Let $\mathcal{RO}$ be a random oracle [6]. It takes as input a (possibly empty) message M, and produces a stream of arbitrary length. From $\mathcal{RO}$, define $\mathcal{RO}^*$ when, given as input a message M and an index m, returns the first m bits of the stream $\mathcal{RO}(M)$. Additionally, let $\mathcal{K} \xleftarrow{\$} \{0,1\}^\kappa$ and $\mathcal{P} \xleftarrow{\$} \mathit{Perm}(b)$. The PRF advantage of an adversary $\mathcal{A}$ against OKS is defined as follows:

$$\mathbf{Adv}_{\mathrm{OKS}}^{\mathrm{PRF}}(\mathcal{A}) = \left| \mathbf{Pr}\left(\mathcal{A}^{\mathrm{OKS}_{\mathcal{K}}^{\mathcal{P}}, \mathcal{P}} = 1\right) - \mathbf{Pr}\left(\mathcal{A}^{\mathcal{RO}^*, \mathcal{P}} = 1\right) \right|.$$

$\mathbf{Adv}^{\mathrm{PRF}}_{\mathrm{OKS}}(M,N)$ denotes the maximum of $\mathbf{Adv}^{\mathrm{PRF}}_{\mathrm{OKS}}(\mathcal{A})$, among all adversaries allowed to make at most N permutation queries and construction queries with a cost of M. (The cost of construction queries is determined in terms of the total number of primitive calls induced by construction calls with OKS.)

5.2.3 Key-Reveal PRF Security We will use a slight variant of PRF security, namely key-reveal PRF security. The key-reveal PRF security game is exactly as the above PRF security game, except that the key is revealed at the end of the interaction, right before the distinguisher outputs its decision bit (this will be a dummy key in the ideal world). Consequently, key-reveal PRF security is strictly stronger than PRF security, as the adversary can opt to ignore the obtained information. While introducing (yet) another security definition might initially appear perplexing, a closer examination of the actual PRF security proof of the outer-keyed sponge [29,30,31] reveals an interesting aspect: *the authors actually established key-reveal PRF security of the sponge.* This phenomenon arises because, in proofs utilizing techniques such as the H-coefficient technique [32], the keys are often assumed to be revealed at the end of the interaction for the sake of simplicity.

That said, we will require a formal definition of the notion of key-reveal PRF security. Let $\mathcal{RO}$ be a random oracle, $\mathcal{K} \xleftarrow{\$} \{0,1\}^{\kappa}$, and $\mathcal{P} \xleftarrow{\$} \mathit{Perm}(b)$ as defined in Section 5.2.2. The adversary is given access to three oracles:

- $\overline{\mathcal{K}}_{\mathcal{K}}$: when queried, it returns $\mathcal{K}$;
- $\overline{\mathcal{C}}^{\mathcal{C}}$ for $\mathcal{C} \in \{\mathrm{OKS}^{\mathcal{P}}_{\mathcal{K}}, \mathcal{RO}^*\}$: if $\overline{\mathcal{K}}_{\mathcal{K}}$ has not been queried yet, this oracle relays the query to $\mathcal{C}$, otherwise, it returns $\perp$;
- $\overline{\mathcal{P}}^{\mathcal{P}}$: if $\overline{\mathcal{K}}_{\mathcal{K}}$ has not been queried yet, this oracle relays the query to $\mathcal{P}$, otherwise, it returns $\perp$.

The key-reveal PRF advantage of an adversary $\mathcal{A}$ against OKS is defined as follows:

$$\mathbf{Adv}^{\text{PRF-krev}}_{\mathrm{OKS}}(\mathcal{A}) = \left| \mathbf{Pr}\left(\mathcal{A}^{\overline{\mathcal{K}}_{\mathcal{K}},\overline{\mathcal{C}}^{\mathrm{OKS}^{\mathcal{P}}_{\mathcal{K}}},\overline{\mathcal{P}}^{\mathcal{P}}} = 1\right) - \mathbf{Pr}\left(\mathcal{A}^{\overline{\mathcal{K}}_{\mathcal{K}},\overline{\mathcal{C}}^{\mathcal{RO}^*},\overline{\mathcal{P}}^{\mathcal{P}}} = 1\right) \right|.$$

$\mathbf{Adv}^{\text{PRF-krev}}_{\mathrm{OKS}}(M,N)$ denotes the maximum of $\mathbf{Adv}^{\text{PRF-krev}}_{\mathrm{OKS}}(\mathcal{A})$, among all adversaries allowed to make at most N permutation queries and construction queries with a cost of M. (The cost of construction queries is determined in terms of the total number of primitive calls induced by construction calls with OKS.)

5.3 Security Result

Now, we state the security of U/Key instantiated with the sponge construction in Theorem 2. At a high level, the security proof relies on several key observations. First, the random secret password x_0 remains undisclosed to the adversary throughout the process. This enables us to treat the evaluation of the sponge on x_0 as an outer-keyed sponge. Consequently, we can replace x_1 with a random

value by leveraging the PRF advantage of the outer-keyed sponge. By repeating this process K times, we find ourselves with K instances of the PRF security of the outer-keyed sponge and K instances of a variant of the everywhere preimage of the sponge. These variants involve a randomized target, and access to chaining values following the release pattern of hash chain protocols. Moreover, in each of these variants, we can use the key-reveal PRF advantage, which allows to conclude that, during most of the attack phase, the adversary does not learn any information about the target to invert. We consider the single-user setting. Given the proof technique that we currently employ, it does not seem that considering a multi-user setting would improve over a generic single-user to multi-user reduction [7,8].

Theorem 2. *Let* Sponge *denote the construction from Section 5.1. Assuming that* $(\ell-1)^2 < 2^b$*, and* $q_{off} + q_{on} + (\ell+\ell')K \leq 2^c/6$*, we have*

$$\mathbf{Adv}_{\mathrm{U/Key[Sponge]}}\left(q_{off}, (q_{on,k})_{k\in[\![1,K]\!]}, 1\right) \leq \sum_{k=1}^{K}\left(\mathbf{Adv}^{\mathrm{PRF}}_{\mathrm{OKS}}\left(\ell'+\ell, q_k\right) + \frac{8\ell q_k}{2^n} + \min\left\{\frac{4\ell q_{on,k}}{2^{n-r}}, \frac{2q_{on,k}\cdot q_k}{2^c}\right\} + \mathbf{Adv}^{\text{PRF-krev}}_{\mathrm{OKS}}(\ell+\ell', 2\ell(q_k - q_{on,k}))\right),$$

where $q_k = (\ell+\ell')(K-k) + q_{off} + \sum_{k'\geq k} q_{on,k'}$*. The result also holds when considering the plain sponge construction from Section 4.3.2.*

The proof is given in Section 5.4.

Interpretation of the Bound. We focus on the setting where all online phases have the same cost (i.e., $\forall k, k', q_{on,k} = q_{on,k'}$). In this case, the bound simplifies to

$$\mathbf{Adv}_{\mathrm{U/Key[Sponge]}}\left(q_{off}, q_{on}, 1\right) \leq \frac{8\ell K q_1}{2^n} + \min\left\{\frac{4\ell q_{on}}{2^{n-r}}, \frac{2q_{on}q_1}{2^c}\right\} + 2K\cdot\mathbf{Adv}^{\text{PRF-krev}}_{\mathrm{OKS}}(\ell+\ell', 2\ell q_1)\,.$$

Moreover, from [29,30,31], we can derive

$$\mathbf{Adv}^{\text{PRF-krev}}_{\mathrm{OKS}}(M,N) = \mathcal{O}\left(\frac{M^2}{2^c} + \frac{NM}{2^c} + \frac{N}{2^n} + \frac{N}{2^c} + \left(2^{\ell' r - b}\right)^c\right).$$

If we additionally assume that $\ell' r < b$ and $\ell K \ll q_{off} + q_{on}$, we obtain

$$\mathbf{Adv}_{\mathrm{U/Key[Sponge]}}\left(q_{off}, q_{on}, 1\right) = \mathcal{O}\left(\frac{K\ell(q_{off}+q_{on})}{2^n} + \frac{K\ell^2(q_{off}+q_{on})}{2^c} + \min\left\{\frac{K\ell q_{on}}{2^{n-r}}, \frac{(q_{off}+q_{on})q_{on}}{2^c}\right\}\right). \quad (17)$$

Consider the typical parameters for T/Key, so that $K \leq 2^{21}$, and suppose we aim for 100 bits of security in the online phase, and 128 bits in the offline phase.

Because of the first term in (17), this implies $n \geq 149$. Consider the Spongent [33] permutation of size $b = 176$. With $n = 150, c = 150$, we have $r = 26, \ell = \ell' = 6$, and the security requirements are met in the single-user setting.

We note that if the permutation is larger than 200 bits, any sponge evaluation in the context of U/Key consists of only one permutation call. In this case, it makes more sense to look at U/Key instantiated with a truncated permutation, and we do so in Section 6.

5.4 Proof of Theorem 2

Denote by $x_0, \ldots, x_K \in \{0,1\}^n$ the chaining values, and id the salt of the user. Let $\mathcal{A}$ be an adversary for the security game of Algorithm 1, allowed to make at most q_{off} offline queries and a total of q_{on} queries during the online phase, where $q_{on,k}$ queries are allowed after the release of x_k and before the release of x_{k+1}. For simplicity, we assume that after each release of x_k, the primitive queries that transition from x_k to x_{k+1} are given for free to the adversary. This gives a total number of queries equal to $q_{on,1}$.

We will use a hybrid argument with K distinct security games. For each $k \in [\![0, K]\!]$, we define S_k to be the following sampling procedure:

- $\forall k' \leq k, x_{k'} \xleftarrow{\$} \{0,1\}^n$,
- $\forall k' > k, x_{k'} \leftarrow \mathrm{Sponge}_m^{\mathcal{P}}(IV_{\mathrm{ctr}_{k'},id}, x_{k'-1})$,

where by abuse of notation $\mathrm{Sponge}_m^{\mathcal{P}}(IV, \cdot)$ denotes the sponge with its state initialized with IV. Note that S_0 corresponds to the original sampling procedure used in the security game. For $k \in [\![1, K]\!]$, we define $G(k)$ to be the security game described in Algorithm 1, but it is aborted when the counter equals k in line 15. In other words, $G(k)$ stops right before x_{k-1} is given to the adversary. In particular, in $G(k)$, the adversary has no opportunities to provide a preimage for $x_{k'}$ for any $k' < k$. Note that, according to this formalism, $G(1)$ corresponds to the full security game.

In the proof, we will transition through these different security games, but the chaining values may not necessarily follow the sampling procedure S_0. The specific sampling procedure will be indicated as a subscript in the probabilities.

Initial Step. We start with the first step of our reasoning:

$$\begin{aligned}&\mathbf{Adv}_{\mathrm{U/Key[Sponge]}}(\mathcal{A}) = \mathbf{Pr}_{S_0}(\mathcal{A} \text{ wins } G(1)) \\ &\leq |\mathbf{Pr}_{S_0}(\mathcal{A} \text{ wins } G(1)) - \mathbf{Pr}_{S_1}(\mathcal{A} \text{ wins } G(1))| + \mathbf{Pr}_{S_1}(\mathcal{A} \text{ wins } G(1)) \ . \quad (18)\end{aligned}$$

The only difference between S_0 and S_1 is in the way in which x_1 is sampled: either randomly or via an evaluation of the sponge. From $\mathcal{A}$, we build a distinguisher $\mathcal{A}'_1$ which returns 1 whenever $\mathcal{A}$ wins the game $G(1)$. Since x_0 is never revealed to the adversary, it can be seen as a secret key. Therefore, $\mathcal{A}'_1$ is a distinguisher in the PRF security game of the outer-keyed sponge initialized

with the state $IV_{1,id}$. Its resources are upper bounded by $\ell' + \ell$ construction queries and q_1 primitive queries. Therefore, we obtain from (18):

$$\mathbf{Adv}_{\mathrm{U/Key[Sponge]}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{PRF}}_{\mathrm{OKS}}(\ell' + \ell, q_1) + \mathbf{Pr}_{S_1}(\mathcal{A} \text{ wins } G(1)) \,. \tag{19}$$

Inductive Reasoning. We proceed with the inductive step to upper bound the term $\mathbf{Pr}_{S_k}(\mathcal{A} \text{ wins } G(k))$ for any $k \in [\![1, K-1]\!]$. Recall that in S_k, the values $x_1, \ldots, x_k$ are sampled uniformly at random, and the remaining chaining values are derived from the hash chain protocol with the root x_k. In particular, in $G(k)$, the adversary is not able to see any inconsistency in the fact that x_k is not the image of x_{k-1} by a sponge call, and finding a preimage of any of these earlier $x_{k'}$'s does not trigger a win.

This means that there are two possibilities for the adversary to win $G(k)$: either $\mathcal{A}$ finds a preimage for x_k by the sponge with the initial state fixed to $IV_{k,id}$, which we abbreviate as "$\mathcal{A}$ solves pre(x_k)", or $\mathcal{A}$ wins $G(k+1)$. Indeed, if the adversary wins $G(k)$ without discovering a preimage for x_k, it indicates that it was already successful in the game before x_k was revealed, thus during $G(k+1)$. We thus have:

$$\begin{aligned}
\mathbf{Pr}_{S_k}(\mathcal{A} \text{ wins } G(k)) &\leq \mathbf{Pr}_{S_k}(\mathcal{A} \text{ solves pre}(x_k) \text{ in } G(k)) + \mathbf{Pr}_{S_k}(\mathcal{A} \text{ wins } G(k+1)) \\
&\leq \mathbf{Pr}_{S_k}(\mathcal{A} \text{ solves pre}(x_k) \text{ in } G(k)) \\
&\quad + \left|\mathbf{Pr}_{S_k}(\mathcal{A} \text{ wins } G(k+1)) - \mathbf{Pr}_{S_{k+1}}(\mathcal{A} \text{ wins } G(k+1))\right| \\
&\quad + \mathbf{Pr}_{S_{k+1}}(\mathcal{A} \text{ wins } G(k+1)) \,.
\end{aligned}$$

Then, from $\mathcal{A}$ we can build a distinguisher $\mathcal{A}'_{k+1}$ returning 1 if and only if $\mathcal{A}$ wins $G(k+1)$. Because S_k and S_{k+1} only differ in the sampling of x_{k+1}, and x_k is never given to the adversary in $G(k+1)$, we can again use the PRF advantage of the sponge. The resources of $\mathcal{A}'_{k+1}$ are upper bounded by $\ell' + \ell$ construction queries and q_{k+1} primitive queries. Thus

$$\begin{aligned}
\mathbf{Pr}_{S_k}(\mathcal{A} \text{ wins } G(k)) &\leq \mathbf{Pr}_{S_k}(\mathcal{A} \text{ solves pre}(x_k) \text{ in } G(k)) \\
&\quad + \mathbf{Adv}^{\mathrm{PRF}}_{\mathrm{OKS}}\left(\mathcal{A}'_{k+1}\right) + \mathbf{Pr}_{S_{k+1}}(\mathcal{A} \text{ wins } G(k+1)) \,.
\end{aligned} \tag{20}$$

We upper bound the quantity $\mathbf{Pr}_{S_k}(\mathcal{A} \text{ solves pre}(x_k) \text{ in } G(k))$ for any $k \in [\![1, K]\!]$ in Lemma 2.

Lemma 2. *If $(\ell - 1)^2 < 2^b$, and $q_k \leq 2^c/6$, we have*

$$\begin{aligned}
\mathbf{Pr}_{S_k}(\mathcal{A} \textit{ solves pre}(x_k) \textit{ in } G(k)) &\leq \frac{8\ell q_k}{2^n} + \min\left\{\frac{4\ell q_{on,k}}{2^{n-r}}, \frac{2q_{on,k} \cdot q_k}{2^c}\right\} \\
&\quad + \mathbf{Adv}^{\text{PRF-krev}}_{\mathrm{OKS}}(\ell + \ell', 2\ell(q_k - q_{on,k})) \,.
\end{aligned}$$

Proof (Sketch). The full proof follows the approach presented in [20]. We therefore only provide the intuition, and give the full proof in Supplementary Material B.

In the context of everywhere preimage resistance (Definition 1), where there are no online or offline phases, we assume that the adversary always knows

x_k. The proof of [20] closely follows the best-known attack for the preimage resistance of the sponge [12], which can be summarized as follows:

1. If $n \leq c/2$: the adversary selects an arbitrary $x \in \{0,1\}^n$, evaluates the sponge with input x. With probability around $\frac{1}{2^n}$, one iteration of this step will successfully output a digest equal to x_k;
2. If $n > c/2$, the attack consists of two phases:
   2.1. The adversary samples $y \xleftarrow{\$} \{0,1\}^c$, obtains $S = x_k^{(1)} \parallel y$, and computes $\mathcal{P}^l(S)$ for $l = 1, \ldots, \ell - 1$. To succeed this step, the condition is that $\text{outer}_r(\mathcal{P}^l(S)) = x_k^{(l+1)}$ for any $\in [\![1, \ell - 2]\!]$, and $\text{outer}_{n-\ell r}(\mathcal{P}^{\ell-1}(S)) = x_k^{(\ell)}$. One iteration of this step succeeds with probability around $\frac{1}{2^{n-r}}$;[4]
   2.2. Once the adversary has found a good state S, it needs to connect this state to the IV of the sponge through finding inner collisions. The i^{th} query of the adversary succeeds with probability around $\frac{i}{2^c}$.

For the steps 1., 2.1., and 2.2., [20] define three bad events, named BADFWD, BADINV, and INNER, respectively. Their analysis can be applied to our setting too, yielding

$$\begin{aligned}\mathbf{Pr}_{S_k}\left(\mathcal{A} \text{ solves pre}(x_k) \text{ in } G(k)\right) &\leq \mathbf{Pr}\left(\mathsf{BADFWD}\right) + \min\left\{\mathbf{Pr}\left(\mathsf{BADINV}\right); \mathbf{Pr}\left(\mathsf{INNER}\right)\right\} \\ &\leq \frac{4q_k}{2^n} + \min\left\{\frac{4\ell q_k}{2^{n-r}}; \frac{q_k(q_k+1)}{2^c}\right\}.\end{aligned}$$

However, this upper bound is too coarse for our setting. Indeed, during the offline phase, no information about the uniformly sampled value x_k is available to the adversary. Therefore, one iteration of step 2.1. does not succeed with probability $\frac{1}{2^{n-r}}$, but $\frac{1}{2^n}$ during the offline phase. During the online stage before x_k is revealed, the situation is slightly different, since the chaining values computed from x_k might leak information. Our situation is comparable to the case studied by Chen et al. [34]: the subsequent chaining values x_{k+X} being revealed correspond to the leakage of intermediate states. However, the leakage of these states should not compromise earlier computed intermediate states. To show this, we use the key-reveal PRF advantage of the construction, which allows to replace x_{k+1} by a random value completely independent from x_k, and this boils down to the same situation as during the offline phase.

Moreover, observing that step 2.2. must be done *after* completing step 2.1., we note that, if BADFWD is triggered during the online phase, this gives the adversary less power to set INNER, as only inner collisions found during the online phase matter. Taking these two considerations into account, we need to split the events BADFWD, BADINV, INNER in a more fine-grained way, which is done in the full version of the proof in Supplementary Material B. □

Now, we can plug the bound derived from Lemma 2 into (20) and obtain

$$\begin{aligned}\mathbf{Pr}_{S_k}\left(\mathcal{A} \text{ wins } G(k)\right) \leq \frac{8\ell q_k}{2^n} + \min\left\{\frac{4\ell q_{on,k}}{2^{n-r}}, \frac{2q_{on,k} \cdot q_k}{2^c}\right\} + \mathbf{Adv}^{\text{PRF}}_{\text{OKS}}(\mathcal{A}'_{k+1}) \\ + \mathbf{Adv}^{\text{PRF-krev}}_{\text{OKS}}(\ell + \ell', 2\ell(q_k - q_{on,k})) + \mathbf{Pr}_{S_{k+1}}\left(\mathcal{A} \text{ wins } G(k+1)\right). \quad (21)\end{aligned}$$

[4] Here, it is assumed that $\ell > 1$; if $\ell = 1$, this step is trivial.

Conclusion. We remark that, eventually, $\mathbf{Pr}_{S_K}(\mathcal{A} \text{ wins } G(K))$ is equal to $\mathbf{Pr}_{S_K}(\mathcal{A} \text{ solves } \mathrm{pre}(x_K) \text{ in } G(K))$. Therefore, by applying the steps taken in (20) and (21) inductively, we obtain from (19):

$$\begin{aligned}
&\mathbf{Adv}_{\mathrm{U/Key[Sponge]}}(\mathcal{A}) \\
&\leq \sum_{k=1}^{K} \mathbf{Pr}_{S_k}(\mathcal{A} \text{ solves } \mathrm{pre}(x_k) \text{ in } G(k)) + \sum_{k=1}^{K} \mathbf{Adv}^{\mathrm{PRF}}_{\mathrm{OKS}}(\ell' + \ell, q_k) \\
&\leq \sum_{k=1}^{K} \left(\frac{8\ell q_k}{2^n} \min\left\{ \frac{4\ell q_{on,k}}{2^{n-r}}, \frac{2q_{on,k} \cdot q_k}{2^c} \right\} + \mathbf{Adv}^{\text{PRF-krev}}_{\mathrm{OKS}}(\ell + \ell', 2\ell(q_k - q_{on,k})) \right. \\
&\qquad \left. + \mathbf{Adv}^{\mathrm{PRF}}_{\mathrm{OKS}}(\ell' + \ell, q_k) \right).
\end{aligned}$$

6 Improved Multi-User Security Proof of U/Key with a Truncated Permutation

For most use cases, U/Key instantiated with a sponge can be evaluated in one permutation call per sponge, making the underlying construction a truncated permutation. Also in this case, the indifferentiability result studied in Section 4.3.3 remains a too strong security property. Indeed, the hash function is required to take $n + s + t$ bits as input, but not all of these inputs can be manipulated freely by the adversary. In the case of a truncated permutation, fixing the salt and the counter fixes the user and thus the elements on the chains that are targeted. Therefore, fixing x_k determines the counters and the users that are targeted. Moreover, the separation between the offline and online phase is not taken into account when using a generic result.

Relying on the simplicity of the construction and its one-pass feature, we obtain in Theorem 3 below a tight bound using a dedicated multi-user proof that additionally accurately captures offline and online time. Section 6.1 describes the construction in more detail, as well as some additional notation used. Section 6.2 is dedicated to the security bound and Appendix C to the proof.

6.1 Description of Scheme

We describe the construction that we consider in this section in more detail. The scheme is used simultaneously by M users, and the user number m runs the hash chain with a random salt id_m. The chaining value number k of user number m is denoted by $x_{k,m}$. In particular, $x_{0,m}$ denote the root passwords. Moreover, let $b > c \geq s + t$, and let $IV \in \{0,1\}^{c-s-t}$ be any fixed initialization vector. (The initialization vector does not play a role in the security.) The state of the permutation is split as $b = n + c$, where n bits are used for the password, and c plays a role similar to the capacity of the sponge. In particular, c includes the counter and the salt. The construction then traverses through the hash chain as

$$x_k = \mathrm{trunc}_n\left(\mathcal{P}(x_{k-1} \parallel \langle \mathrm{ctr}_k \rangle_t \parallel id_m \parallel IV)\right),$$

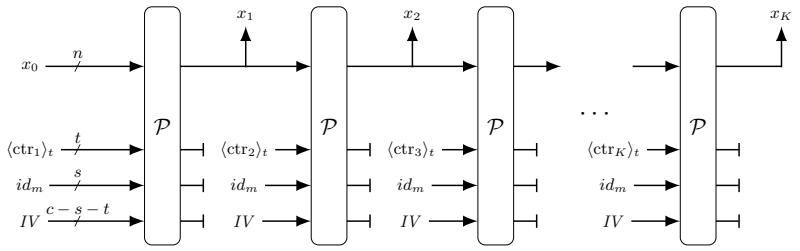

Fig. 2. U/Key with a truncated permutation of size $b = n + c$ for user number m.

for $k \in [\![1, K]\!]$ and $m \in [\![1, M]\!]$. The sequence $(\langle \mathrm{ctr}_k \rangle_t \parallel id_m \parallel IV)_{k,m}$ can be seen as a family of fixed prefixes. The scheme is illustrated in Fig. 2.

6.2 Security Result

In Theorem 3 we state the multi-user security of U/Key on top of the truncated permutation construction.

Theorem 3. *Let* TruncP *denote the construction from Section 6.1. Assuming that* $KM + q_{off} + q_{on} \leq 2^{b-1}$*, we have*

$$\mathbf{Adv}_{\mathrm{U/Key[TruncP]}}(q_{off}, q_{on}, M) \leq \left(\frac{2M}{2^s} + 3s + 4\right)\left(\min\left\{\frac{2M}{2^s}; 1\right\}\frac{4q_{off}}{2^n} + \frac{4q_{on}}{2^n}\right) + \left(\frac{2KM}{2^n} + 3n + 4\right)\left(\min\left\{\frac{2KM}{2^n}; 1\right\}\frac{4q_{off}}{2^c} + \frac{4q_{on}}{2^c}\right).$$

The proof shares similarities with the proof of Theorem 1, particularly in the way that forward queries to $\mathcal{P}$ are treated. A difference is in the fact that, in current setting, the adversary can also make inverse evaluations to $\mathcal{P}^{-1}$. The complete proof is given in Supplementary Material C.

Interpretation of the Bound. The bound includes multiple terms, and we will provide key points that can help to interpret it. The bound is a sum of two main terms. The first (resp., second) one corresponds to a strategy where the adversary only makes forward (resp., inverse) queries. Now, regarding offline queries, whenever $M < 2^s$ (resp., $KM < 2^n$), the number of forward (resp., inverse) offline queries that are "useful" is multiplied by $\frac{M}{2^s}$ (resp., $\frac{KM}{2^n}$). If we then denote by q the (expected) number of useful queries, the bound of Theorem 3 simplifies to

$$\mathcal{O}\left(\frac{q}{2^n} + \frac{q}{2^c} + \frac{qM}{2^{s+n}} + \frac{qKM}{2^b}\right).$$

Moreover, as long as $M = \mathcal{O}(2^s)$, this bound is of the order

$$\mathcal{O}\left(\frac{q}{2^n} + \frac{q}{2^c} + \frac{qKM}{2^b}\right).$$

Just like for the interpretation of Theorem 1, the security bound allows for many different interpretations. For example, if we take $s = 80$ and $t = 32$, and target a security goal of 128 bits, we must take $n \geq 128$ and $c \geq 128$, and thus require a permutation of size at least 256 bits. For more specific settings, we can make use of the separation between offline and online queries. For example, if we assume $M \leq 10^{12}$, $s = 68$, keep t the same, and assume $q_{on} \ll 2^{100}$, then we can take $n = c = 100$, thus having a permutation of size 200 bits, while still achieving 128-bit offline security.

7 Concluding Remarks

One-time passwords using hash chains are a viable option if two-factor authentication is not suitable. With their introduction of T/Key, Kogan et al. [4] made a great effort to improve its state of the art. However, their analysis was in a relatively basic model, and with our novel model we have shown that it is possible to argue security in a more fine-grained treatment of the adversarial resources. In particular, we split the adversarial capacity into offline and online computation, and allow for analysis in the multi-user setting. We demonstrated the relevance of our model on our slightly more general construction U/Key.

7.1 Impact of Online Query Complexity

The separation of offline and online queries allows to more accurately determine the adversarial success probability in terms of the parameters of the scheme. For example, taking shorter timeframes implies that $q_{on,k}$ will be lower, though on the other hand K may need to be higher. For a T/Key construction where the timeframes are all of the same size, this does not make a difference as the bound is determined by $q_{on} = Kq_{on,k}$, but a difference may be present for more general constructions as covered by our new U/Key. More noticeable is the role of the number of users, as we also already showed in the interpretation of the bounds of Theorem 1 and Theorem 3. For example, depending on whether M exceeds 2^s, the bound differs and one can even achieve higher security than the password size n provided $M \ll 2^s$.

7.2 Memory Storage

In our current security model, we assume that the adversary can store all queries made during the offline phase. However, in real life, memory is more expensive than computational power, which can result in suboptimal bounds. To address this, we can adopt an approach similar to the one of Kogan et al. [4], i.e., split the adversary $\mathcal{A}$ of Algorithm 1 into two: $\mathcal{A}_1$ runs the offline phase and passes an advice string S to $\mathcal{A}_2$, which runs the online phase. The adversary $\mathcal{A}_1$ is typically unrestricted, but the advice string S has an upper bound on its size. This approach, however, has a downside: indifferentiability composition does not apply since the adversary is two-stage [16]. Therefore, a dedicated proof is needed

for any security bound at the construction level, and memory-bounded proofs in this context are particularly challenging [35,36,37].

Acknowledgements

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 rant VI.Vidi.203.099

References

1. "FIDO Alliance – Open Authentication Standards More Secure than Passwords," https://fidoalliance.org, 2023, accessed: 2023-04-22.
2. N. Haller, "The S/KEY One-Time Password System," Request for Comments (RFC) 1760, February 1995. [Online]. Available: http://tools.ietf.org/html/rfc1760
3. R. Bhaumik, N. Datta, A. Dutta, N. Mouha, and M. Nandi, "The Iterated Random Function Problem," in *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, ser. Lecture Notes in Computer Science, T. Takagi and T. Peyrin, Eds., vol. 10625. Springer, 2017, pp. 667–697. [Online]. Available: https://doi.org/10.1007/978-3-319-70697-9_23
4. D. Kogan, N. Manohar, and D. Boneh, "T/Key: Second-Factor Authentication From Secure Hash Chains," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM, 2017, pp. 983–999. [Online]. Available: https://doi.org/10.1145/3133956.3133989
5. D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm," Request for Comments (RFC) 6238, May 2011. [Online]. Available: http://tools.ietf.org/html/rfc6238
6. M. Bellare and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols," in *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, Eds. ACM, 1993, pp. 62–73. [Online]. Available: https://doi.org/10.1145/168588.168596
7. E. Biham, "How to decrypt or even substitute des-encrypted messages in 2^{28} steps," *Inf. Process. Lett.*, vol. 84, no. 3, pp. 117–124, 2002. [Online]. Available: https://doi.org/10.1016/S0020-0190(02)00269-7
8. M. Bellare, A. Boldyreva, and S. Micali, "Public-key encryption in a multi-user setting: Security proofs and improvements," in *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, ser. Lecture Notes in Computer Science, B. Preneel, Ed., vol. 1807. Springer, 2000, pp. 259–274. [Online]. Available: https://doi.org/10.1007/3-540-45539-6_18

9. R. C. Merkle, "One Way Hash Functions and DES," in *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, ser. Lecture Notes in Computer Science, G. Brassard, Ed., vol. 435. Springer, 1989, pp. 428–446. [Online]. Available: https://doi.org/10.1007/0-387-34805-0_40
10. I. Damgård, "A Design Principle for Hash Functions," in *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, ser. Lecture Notes in Computer Science, G. Brassard, Ed., vol. 435. Springer, 1989, pp. 416–427. [Online]. Available: https://doi.org/10.1007/0-387-34805-0_39
11. J. Coron, Y. Dodis, C. Malinaud, and P. Puniya, "Merkle-Damgård Revisited: How to Construct a Hash Function," in *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, ser. Lecture Notes in Computer Science, V. Shoup, Ed., vol. 3621. Springer, 2005, pp. 430–448. [Online]. Available: https://doi.org/10.1007/11535218_26
12. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Sponge functions," Ecrypt Hash Workshop 2007, May 2007.
13. U. M. Maurer, R. Renner, and C. Holenstein, "Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology," in *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, ser. Lecture Notes in Computer Science, M. Naor, Ed., vol. 2951. Springer, 2004, pp. 21–39. [Online]. Available: https://doi.org/10.1007/978-3-540-24638-1_2
14. D. Chang and M. Nandi, "Improved Indifferentiability Security Analysis of chopMD Hash Function," in *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, ser. Lecture Notes in Computer Science, K. Nyberg, Ed., vol. 5086. Springer, 2008, pp. 429–443. [Online]. Available: https://doi.org/10.1007/978-3-540-71039-4_27
15. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "On the Indifferentiability of the Sponge Construction," in *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, ser. Lecture Notes in Computer Science, N. P. Smart, Ed., vol. 4965. Springer, 2008, pp. 181–197. [Online]. Available: https://doi.org/10.1007/978-3-540-78967-3_11
16. T. Ristenpart, H. Shacham, and T. Shrimpton, "Careful with Composition: Limitations of the Indifferentiability Framework," in *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, ser. Lecture Notes in Computer Science, K. G. Paterson, Ed., vol. 6632. Springer, 2011, pp. 487–506. [Online]. Available: https://doi.org/10.1007/978-3-642-20465-4_27
17. A. Ghoshal and S. Tessaro, "The query-complexity of preprocessing attacks," in *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II*, ser. Lecture Notes in Computer Science, H. Handschuh and A. Lysyanskaya, Eds., vol. 14082. Springer, 2023, pp. 482–513. [Online]. Available: https://doi.org/10.1007/978-3-031-38545-2_16

18. C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1.2," Winning Submission to NIST Lightweight Cryptography, 2021. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/ascon-spec-final.pdf
19. ——, "Ascon v1.2: Lightweight Authenticated Encryption and Hashing," *J. Cryptol.*, vol. 34, no. 3, p. 33, 2021. [Online]. Available: https://doi.org/10.1007/s00145-021-09398-9
20. C. Lefevre and B. Mennink, "Tight Preimage Resistance of the Sponge Construction," in *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*, ser. Lecture Notes in Computer Science, Y. Dodis and T. Shrimpton, Eds., vol. 13510. Springer, 2022, pp. 185–204. [Online]. Available: https://doi.org/10.1007/978-3-031-15985-5_7
21. W. Choi, B. Lee, and J. Lee, "Indifferentiability of Truncated Random Permutations," in *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*, ser. Lecture Notes in Computer Science, S. D. Galbraith and S. Moriai, Eds., vol. 11921. Springer, 2019, pp. 175–195. [Online]. Available: https://doi.org/10.1007/978-3-030-34578-5_7
22. P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, ser. Lecture Notes in Computer Science, B. K. Roy and W. Meier, Eds., vol. 3017. Springer, 2004, pp. 371–388. [Online]. Available: https://doi.org/10.1007/978-3-540-25937-4_24
23. B. Schoenmakers, "Explicit optimal binary pebbling for one-way hash chain reversal," in *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, ser. Lecture Notes in Computer Science, J. Grossklags and B. Preneel, Eds., vol. 9603. Springer, 2016, pp. 299–320. [Online]. Available: https://doi.org/10.1007/978-3-662-54970-4_18
24. "Blockchain website, total hash rate," https://www.blockchain.com/explorer/charts/hash-rate, 2023, accessed: 2023-07-11.
25. E. Andreeva, B. Mennink, and B. Preneel, "Security Reductions of the Second Round SHA-3 Candidates," in *Information Security - 13th International Conference, ISC 2010, Boca Raton, FL, USA, October 25-28, 2010, Revised Selected Papers*, ser. Lecture Notes in Computer Science, M. Burmester, G. Tsudik, S. S. Magliveras, and I. Ilic, Eds., vol. 6531. Springer, 2010, pp. 39–53. [Online]. Available: https://doi.org/10.1007/978-3-642-18178-8_5
26. L. Grassi and B. Mennink, "Security of truncated permutation without initial value," in *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 II*, ser. Lecture Notes in Computer Science, S. Agrawal and D. Lin, Eds., vol. 13792. Springer, 2022, pp. 620–650. [Online]. Available: https://doi.org/10.1007/978-3-031-22966-4_21
27. J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON Family of Lightweight Hash Functions," in *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, ser.

Lecture Notes in Computer Science, P. Rogaway, Ed., vol. 6841. Springer, 2011, pp. 222–239. [Online]. Available: https://doi.org/10.1007/978-3-642-22792-9_13

28. Y. Naito and K. Ohta, "Improved indifferentiable security analysis of PHOTON," in *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, ser. Lecture Notes in Computer Science, M. Abdalla and R. D. Prisco, Eds., vol. 8642. Springer, 2014, pp. 340–357. [Online]. Available: https://doi.org/10.1007/978-3-319-10879-7_20
29. E. Andreeva, J. Daemen, B. Mennink, and G. V. Assche, "Security of keyed sponge constructions using a modular proof approach," in *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, ser. Lecture Notes in Computer Science, G. Leander, Ed., vol. 9054. Springer, 2015, pp. 364–384. [Online]. Available: https://doi.org/10.1007/978-3-662-48116-5_18
30. Y. Naito and K. Yasuda, "New bounds for keyed sponges with extendable output: Independence between capacity and message length," in *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, ser. Lecture Notes in Computer Science, T. Peyrin, Ed., vol. 9783. Springer, 2016, pp. 3–22. [Online]. Available: https://doi.org/10.1007/978-3-662-52993-5_1
31. B. Mennink, "Key Prediction Security of Keyed Sponges," *IACR Trans. Symmetric Cryptol.*, vol. 2018, no. 4, pp. 128–149, 2018. [Online]. Available: https://doi.org/10.13154/tosc.v2018.i4.128-149
32. J. Patarin, "The "coefficients h" technique," in *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, ser. Lecture Notes in Computer Science, R. M. Avanzi, L. Keliher, and F. Sica, Eds., vol. 5381. Springer, 2008, pp. 328–345. [Online]. Available: https://doi.org/10.1007/978-3-642-04159-4_21
33. A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and I. Verbauwhede, "spongent: A Lightweight Hash Function," in *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, ser. Lecture Notes in Computer Science, B. Preneel and T. Takagi, Eds., vol. 6917. Springer, 2011, pp. 312–325. [Online]. Available: https://doi.org/10.1007/978-3-642-23951-9_21
34. Y. L. Chen, A. Luykx, B. Mennink, and B. Preneel, "Systematic security analysis of stream encryption with key erasure," *IEEE Trans. Inf. Theory*, vol. 67, no. 11, pp. 7518–7534, 2021. [Online]. Available: https://doi.org/10.1109/TIT.2021.3109302
35. S. Tessaro and A. Thiruvengadam, "Provable Time-Memory Trade-Offs: Symmetric Cryptography Against Memory-Bounded Adversaries," in *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part I*, ser. Lecture Notes in Computer Science, A. Beimel and S. Dziembowski, Eds., vol. 11239. Springer, 2018, pp. 3–32. [Online]. Available: https://doi.org/10.1007/978-3-030-03807-6_1
36. J. Jaeger and S. Tessaro, "Tight Time-Memory Trade-Offs for Symmetric Encryption," in *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, ser. Lecture Notes in Computer Science, Y. Ishai and V. Rijmen, Eds., vol. 11476. Springer, 2019, pp. 467–497. [Online]. Available: https://doi.org/10.1007/978-3-030-17653-2_16

37. I. Dinur, "Tight Time-Space Lower Bounds for Finding Multiple Collision Pairs and Their Applications," in *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, ser. Lecture Notes in Computer Science, A. Canteaut and Y. Ishai, Eds., vol. 12105. Springer, 2020, pp. 405–434. [Online]. Available: https://doi.org/10.1007/978-3-030-45721-1_15
38. W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The collected works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

Supplementary Material

A Proof of Lemma 1

Let $p = \frac{1}{R}$ and $r \in [\![1, R]\!]$. It is clear that $X^{(r)}$ follows a binomial law with parameters p and n. Therefore, we can use the Chernoff bound, so that for any $j \geq 2pn$,

$$\mathbf{Pr}\left(X^{(r)} \geq j\right) \leq e^{-\frac{j-pn}{3}} .$$

Finally,

$$\begin{aligned}
&\mathsf{E}\left(\max_r X^{(r)}\right) \\
&= \sum_{j\geq 1} \mathbf{Pr}\left(\max_r X_r \geq j\right) \\
&\leq 2pn + 3\ln(R) + \sum_{j=2pn+3\ln(R)}^{n} \mathbf{Pr}\left(\bigvee_r X_r \geq j\right) \\
&\leq 2pn + 3\ln(R) + \sum_{r=1}^{R} \sum_{j=2pn+3\ln(R)}^{n} e^{-\frac{j-pn}{3}} \\
&\leq 2pn + 3\ln(R) + R \cdot e^{\frac{pn}{3}} \cdot \frac{e^{-\frac{2pn+3\ln(R)}{3}} - e^{\frac{-n-1}{3}}}{1 - e^{-\frac{1}{3}}} \\
&\leq 2pn + 3\ln(R) + 4R \cdot e^{-\frac{pn}{3}} e^{-\ln(R)} \\
&\leq \frac{2n}{R} + 3\ln(R) + 4 .
\end{aligned}$$

Now, when the sampling is performed without replacement, we can use [38, Theorem 4], which states that for any continuous and convex function,

$$\mathsf{E}\left(f\left(X^{(r)}\right)\right) \leq \mathsf{E}\left(f\left(Y^{(r)}\right)\right) ,$$

where $Y^{(r)} \sim$Binomial(p, n). In particular, this holds when $f(x) = e^{t\cdot x}$ for any $t > 0$. Because the Chernoff bound is obtained by upper bounding $\mathsf{E}\left(e^{t\cdot X^{(r)}}\right)$, the proof also carries over to this case.

B Proof of Lemma 2

In this section, we provide a self-contained proof of Lemma 2. The proof closely follows [20].

Lemma 2 aims to bound the probability that an adversary finds a preimage for x_k in $G(k)$. Remember that, for $k' < K$, when $x_{k'}$ is given to the adversary, the permutation evaluations that allow to compute $x_{k'+1}$ from $x_{k'}$ are given for

free to the adversary. Therefore, in the game $G(k)$, the adversary was granted a total of $q^{(k)} = (\ell + \ell')(K - k) + q_{off} + \sum_{k' \geq k} q_{on}^{(k')}$ queries. We remark that the phase *before* the release of x_k is basically the offline phase for the particular security game $G(k)$. We can therefore distinguish between three different stages for the game $G(k)$:

- The offline phase, during which the adversary can make q_{off} queries;
- The online phase up to the release of x_k (free queries included). During that phase, the adversary can make $(\ell + \ell')(K - k) + \sum_{k' > k} q_{on}^{(k')}$ queries;
- The online phase after the release of x_k and before the release of x_{k+1} during which the adversary can make $q_{on,k}$ queries.

We refer to the two first stages as the k-offline phase, and the last as the k-online phase.

Setup. We start by establishing the notation used in this proof. In the probabilities, when no subscript is mentioned, the distribution S_k is implicitly used. Let $0 < s \leq r$ be such that $n = (\ell - 1) \times r + s$, or in words, s is the length of x_k^ℓ. The adversarial query history is denoted as $\mathcal{Q}$. It contains tuples of the form (X, Y, d), indicating that $\mathcal{P}(X) = Y$ and that the query was made in the direction $d \in \{fwd, inv\}$. Given two indices $a \leq b$, $\mathcal{Q}[a : b]$ refers to the subquery history encompassing only the tuples in $\mathcal{Q}$ that were added between query number a and query number b (both indices included). Moreover, $\mathcal{Q}[a]$ denotes $\mathcal{Q}[a : a]$. Given a bad event BAD, for any $i \in [\![1, q_k]\!]$, $\mathsf{BAD}[i]$ denotes that BAD is triggered after the first i queries. If E is an event, $\mathbf{1}_E$ is the Bernoulli variable equal to 1 if and only if E occurs. Finally, for $x \in \mathbb{N}$, $n \leq x$, we use $[x]_n$ to denote the falling factorial of degree n of x, i.e.,

$$[x]_n = \prod_{i=0}^{n-1} (x - i) \,.$$

In order to find a preimage, the adversary must complete two steps: i) Find a well-formed state such that its cascade of $\ell - 1$ consecutive permutation evaluations produces the outer parts that match exactly the components $x_k^{(l)}$; ii) Connect this state to $IV_{k,id}$ with message blocks. Let $\mathcal{S}$ be the following set:

$$\mathcal{S} = \left\{ y \in \{0,1\}^b \;\middle|\; \forall l \in [\![1, \ell]\!],\, \mathrm{outer}_r(\mathcal{P}^l(y)) = x_k^{(l)} \right\} .$$

Moreover, define $\mathcal{S}[l]$ to be $\mathcal{P}^l(\mathcal{S})$. In words, $\mathcal{S}$ captures all the states right before the squeezing phase which cascade successively to produce the desired ℓ outer parts, and $\mathcal{S}[l]$ specifically captures the states associated with squeezing $x_k^{(l)}$. Since $\mathcal{P}$ is a permutation, these sets all have the same size. Moreover, given $0 \leq a \leq b \leq \ell$, $\mathcal{S}[a : b]$ is the multi-set equal to $\bigcup_{l=a}^{b} \mathcal{S}[l]$.

Event Splitting. If the adversary manages to find a preimage for x_k, this implies that there exist $a_1, \ldots, a_{\ell'} \in \{0,1\}^r$, $N_0, \ldots N_{\ell'}, \in \{0,1\}^b$, and $d_1, \ldots, d_{\ell'} \in \{fwd, inv\}$ such that

- $N_0 = IV_{k,id}$;
- $\forall i \in [\![1, \ell']\!], (N_{i-1} \oplus (a_i \parallel 0^c), N_i, d_i) \in Q$;
- $N_{\ell'} \in \mathcal{S}[1]$.

We denote this bad event by PRE. At this stage, we dropped the requirement on the message blocks to correspond to a valid padding. In order to avoid the complexity of reasoning about the entire graph, we can focus on specific trigger points. The pivot taken here is the direction of the query $(N_{\ell'-1} \oplus (a_{\ell'} \parallel 0^c), N_{\ell'}, d_{\ell'})$.

Let BADFWD and, for $l \in [\![1, \ell]\!]$, BADINV^l be defined as follows:

$$\mathsf{BADFWD} : \exists (X, Y, fwd) \in \mathcal{Q} \text{ such that } X \in \mathcal{S}[0]\,,$$
$$\mathsf{BADINV}^l : \exists (S_{fwd}, S_{inv}, d) \in \mathcal{Q} \text{ such that } S_d \in \mathcal{S}[l]\,.$$

Moreover, let $\mathsf{BADINV} := \bigvee_{l=1}^{\ell} \mathsf{BADINV}^l$. BADFWD (resp., BADINV) corresponds to the pivot query made in the forward (resp., inverse) direction. Intuitively, with BADFWD, the adversary cannot freely choose the outer parts of the states it queries. On the other hand, if BADINV is set during the k-online phase, the adversary can freely choose the outer parts of the query it makes, allowing it to set this event with higher probability. However, in the k-offline phase, the adversary additionally needs to guess the outer parts.

Now, if the adversary finds a well-formed state through BADINV, it must further connect this state to the initial state of the sponge $IV_{k,id}$ *after having triggered* BADINV. To capture that, we will consider a slightly more complicated version of bad event INNER from [20]. More precisely, we parametrize INNER by a query index i. This enforces that a fresh inner collision should have been found starting from the query i, and previously found inner collisions do not play a role. This parameterization proves to be valuable, as triggering BADINV during the k-online phase has a lower success probability than during the k-offline phase. $\mathsf{INNER}(i)$ is defined as follows:

$$\mathsf{INNER}(i) : \exists (X, Y, fwd) \in \mathcal{Q}, (X', Y', inv) \in \mathcal{Q}[i:q] \\ \text{such that } \mathrm{inner}_c(Y) = \mathrm{inner}_c(X')\,.$$

In [20], Lefevre and Mennink argued the following splitting:

$$\mathsf{PRE} \implies \mathsf{BADFWD} \vee (\mathsf{BADINV} \wedge \mathsf{INNER}(1))\,,$$

which led to the following bound:

$$\mathbf{Pr}\left(\mathsf{PRE}\right) \leq \mathbf{Pr}\left(\mathsf{BADFWD}\right) + \min\left\{\mathbf{Pr}\left(\mathsf{INNER}(1)\right), \mathbf{Pr}\left(\mathsf{BADINV}\right)\right\}\,.$$

However, this approach is too coarse for our setting, and we need to further split the events to take into account the different phases. More precisely, in order to trigger PRE, there are three possibilities:

– The adversary triggers BADFWD at some point;
– The adversary triggers BADINV during the k-offline phase, and INNER(1) at some point;
– The adversary triggers BADINV and finds inner collisions, both during the k-online phase.

Therefore,

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{PRE}\right) \leq\ & \mathbf{Pr}\left(\mathsf{BADFWD}\right) + \min\left\{\mathbf{Pr}\left(\mathsf{BADINV}[q_k - q_{on,k}]\right), \mathbf{Pr}\left(\mathsf{INNER}(1)\right)\right\} \\ & + \min\left\{\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADINV}[q_k - q_{on,k}]\right), \mathbf{Pr}\left(\mathsf{INNER}(q_k - q_{on,k})\right)\right\}\ ,\end{aligned}$$

where we remind that $\mathsf{BAD}[i]$ denotes that BAD is triggered after the first i queries; while $\mathsf{INNER}(i)$ denotes that a new inner collision has been found starting from the i^{th} query. Looking ahead, $\mathbf{Pr}\left(\mathsf{BADINV}[q_k - q_{on,k}]\right)$ will have a bound similar to $\mathbf{Pr}\left(\mathsf{BADFWD}\right)$. This allows us to eliminate the term involving $\mathsf{INNER}(1)$, resulting in the following expression:

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{PRE}\right) \leq\ & \mathbf{Pr}\left(\mathsf{BADFWD}\right) + \mathbf{Pr}\left(\mathsf{BADINV}[q_k - q_{on,k}]\right) \\ & + \min\left\{\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADINV}[q_k - q_{on,k}]\right), \mathbf{Pr}\left(\mathsf{INNER}(q_k - q_{on,k})\right)\right\}\ .\end{aligned} \tag{22}$$

Now, we can evaluate the probabilities individually. Intuitively, for the events BADFWD and BADINV, we adopt the same approach as [20], which involves conditioning on $|\mathcal{S}|$. The main novelty in our approach consists of showing that setting BADINV during the k-offline phase is (almost) as hard as setting BADFWD.

Probability of BADFWD. We have

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BADFWD}\right) &\leq \sum_{i=1}^{q_k}\sum_{y=1}^{2^c} \mathbf{Pr}\left(\mathsf{BADFWD}[i] \wedge \neg\mathsf{BADFWD}[i-1] \wedge |S[0]| = y\right) \\ &\leq \sum_{i=1}^{q_k}\sum_{y=1}^{2^c} \mathbf{Pr}\left(\mathsf{BADFWD}[i] \;\middle|\; \neg\mathsf{BADFWD}[i-1] \wedge |S[0]| = y\right) \times \\ &\quad\ \mathbf{Pr}\left(|S[0]| = y\right)\ .\end{aligned}$$

Triggering BADFWD is similar to a guessing game. The adversary must make a query such that the answer lies in the set $\mathcal{S}[1]$. As explained in more detail in [20], the values in $\mathcal{S}[0]$ are defined via inverse $\mathcal{P}$-calls, thus random. Moreover, one failed attempt with the query (X, Y, fwd) from the adversary only removes X from the set of possibilities. Therefore,

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BADFWD}\right) &\leq \sum_{i=1}^{q_k}\sum_{y=1}^{2^c} \frac{y}{2^b - (i-1)}\mathbf{Pr}\left(|S[0]| = y\right) \\ &\leq \frac{2q_k}{2^b}\mathsf{E}\left(|S[0]|\right)\ ,\end{aligned} \tag{23}$$

where we used that $q_k \leq 2^{b-1}$.

Probability of $\mathsf{BADINV} \wedge \neg\mathsf{BADINV}[q_k - q_{on,k}]$. Note that we can assume that $\ell > 1$, otherwise this event can be set with probability 1 and the probability will not dominate the "min". Again, by basic probability, we have

$$\begin{aligned}&\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADINV}[q_k - q_{on,k}]\right)\\ &\leq \sum_{i=q_k-q_{on,k}+1}^{q_k}\sum_{y=1}^{2^c}\sum_{l=1}^{\ell}\mathbf{Pr}\left(\mathsf{BADINV}^l[i] \;\middle|\; \neg\mathsf{BADINV}[i-1] \wedge |S[1:\ell]| = \ell y\right) \times \\ &\qquad\qquad \mathbf{Pr}\left(|S[1:\ell]| = \ell y\right) .\end{aligned}$$

The idea used in [20] for the conditioned BADINV involves a close examination of the paths induced by the elements in $\mathcal{S}[1:\ell]$. Here, we only focus on one fixed outer part at a time for the individual bad events BADINV^l. For simplicity, when the adversary makes a query with input Z, both $\mathcal{P}(Z)$ and $\mathcal{P}^{-1}(Z)$ are assumed to be given to the adversary. Therefore, the adversary wins if $\mathcal{P}^{-1}(Z) \to Z \to \mathcal{P}(Z)$ coincides with a path $Y_{l-1} \to Y_l \to Y_{l+1}$ with $Y_l \in \mathcal{S}[l]$ and $Y_{l\pm1} \in \mathcal{P}^{\pm1}(\mathcal{S}[l])$. $\mathcal{S}[l]$ is a set of size y, and is a subset of a set of size at least 2^c (note that this statement is also valid when $l = \ell$, since $2^{b-s} \geq 2^c$). At a high level, either a query sets BADINV, or it fails, and in the latter case, the only information that the adversary learns is that neither Z, $\mathcal{P}(Z)$, nor $\mathcal{P}^{-1}(Z)$ are in $\mathcal{S}[1:\ell]$. Therefore, for any i, y, and l,

$$\mathbf{Pr}\left(\mathsf{BADINV}^l[i] \;\middle|\; \neg\mathsf{BADINV}[i-1] \wedge |S[1:\ell]| = \ell y\right) \leq \frac{y}{2^c - 3(i-1)} \leq \frac{2y}{2^c} ,$$

where we used that $q_k \leq 2^c/6$. Therefore,

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADINV}[q_k - q_{on,k}]\right) &\leq \sum_{i=q_k-q_{on,k}+1}^{q_k}\sum_{y=1}^{2^c}\frac{2\ell y}{2^c}\mathbf{Pr}\left(|S[1:\ell]| = \ell y\right)\\ &\leq \frac{2q_{on,k}}{2^c}\mathsf{E}\left(|S[1:\ell]|\right) . \qquad (24)\end{aligned}$$

Probability of $\mathsf{BADINV}[q_k - q_{on,k}]$. Setting this event implies among others that the adversary wins *before* x_k is released. As a first step, from $\mathcal{A}$, we want to build a distinguisher $\mathcal{D}$ that returns 1 if and only if $\mathcal{A}$ sets BADINV at the end of the k-offline phase. For $\mathcal{D}$ to verify the aforementioned event, we need to provide it additional information at the end of the k-offline phase. As the first piece of additional information, we provide x_k, which can be seen as a key for the outer-keyed sponge during the k-offline phase. Moreover, for every existing $(X, Y, d) \in \mathcal{Q}[1 : q_k - q_{on,k}]$, we give to $\mathcal{D}$ the queries $(\mathcal{P}^l(Y), \mathcal{P}^{l+1}(Y))$, and $(\mathcal{P}^{-l-1}(X), \mathcal{P}^{-l}(X))$ for all $l \in [\![0, \ell - 2]\!]$. Now, $\mathcal{D}$ is a distinguisher in the key-reveal PRF security game of the sponge. The resources of $\mathcal{D}$ can be upper bounded by $\ell + \ell'$ construction queries, and $2\ell(q_k - q_{on,k})$ primitive queries.

Therefore,

$$\begin{aligned}
&\mathbf{Pr}_{S_k}\left(\mathsf{BADINV}[q_k - q_{on,k}]\right) \\
&\leq \left|\mathbf{Pr}_{S_k}\left(\mathsf{BADINV}[q_k - q_{on,k}]\right) - \mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}[q_k - q_{on,k}]\right)\right| \\
&+ \mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}[q_k - q_{on,k}]\right) \\
&\leq \mathbf{Adv}^{\text{PRF-krev}}_{\text{OKS}}(\ell + \ell', 2\ell(q_k - q_{on,k})) + \mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}[q_k - q_{on,k}]\right) . \quad (25)
\end{aligned}$$

Now, we can upper bound the term $\mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}[q_k - q_{on,k}]\right)$. We have

$$\begin{aligned}
&\mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}[q_k - q_{on,k}]\right) \\
&\leq \sum_{y=1}^{2^c} \sum_{i=1}^{q_k - q_{on,k}} \sum_{l=1}^{\ell} \mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}^l[i] \mid \neg\mathsf{BADINV}[i-1] \wedge \mathcal{S}[1:\ell] = \ell y\right) \times \\
&\qquad \mathbf{Pr}\left(\mathcal{S}[1:\ell] = \ell y\right) .
\end{aligned}$$

Remember that the values $x_k^{(l)}$ are sampled at random, but are not given to the adversary. Therefore, the difficulty of guessing an element in $\mathcal{S}[l]$ is augmented by the fact that the adversary must guess $x_k^{(l)}$. Given $i \in [\![1, q_k]\!]$ and $l \in [\![1, \ell]\!]$, we define the bad events GUESS_i^l as follows:

$$\mathsf{GUESS}_i^l : \exists (S_{fwd}, S_{inv}, d) \in \mathcal{Q}[i] \text{ such that } x_k^{(l)} = \begin{cases} \text{outer}_r(S_d) & \text{if } l < \ell\,, \\ \text{outer}_s(S_d) & \text{if } l = \ell\,. \end{cases}$$

Moreover, define the random variables $\mathcal{G}^l$ as follows:

$$\mathcal{G}^l = \left|\left\{ i \in [\![1, q_k - q_{on,k}]\!] \mid \mathsf{GUESS}_i^l \text{ holds} \right\}\right| .$$

In other words, for each l, $\mathcal{G}^l$ counts the number of queries during the k-offline phase that have their outer part equal to $x_k^{(l)}$. Note that contrary to the case of BADINV during the k-online phase, we do have to consider the last chaining value separately. Indeed, when $s < r$, guessing $x_k^{(\ell)}$ becomes easier due to the smaller size of the last message block, but given this correct guess, guessing an element in $\mathcal{S}[\ell]$ becomes harder, since $\mathcal{S}[\ell]$ is a set of size y, and is a subset of a set of size at least 2^{b-s}. We have

$$\mathsf{E}_{S_{k+1}}\left(\mathcal{G}^l\right) \leq \begin{cases} \frac{(q_k - q_{on,k})}{2^r} & \text{if } l < \ell\,, \\ \frac{(q_k - q_{on,k})}{2^s} & \text{if } l = \ell\,. \end{cases}$$

Note that this bounding is independent of the events $\neg\mathsf{BADINV}[i-1]$ and $|\mathcal{S}[1:\ell]| = y$. Now, for the probability that the i^{th} query lies in $\mathcal{S}[l]$, conditioned on GUESS_i^l, the reasoning used for the analysis of BADINV during the k-online phase still applies. However, we need to use a more accurate bounding

for the size of the superset of $\mathcal{S}[\ell]$. We have

$$\begin{aligned}
&\mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}[q_k - q_{on,k}]\right) \\
&\leq \sum_{y=1}^{2^c} \sum_{i=1}^{q_k - q_{on,k}} \mathbf{Pr}\left(\mathcal{S}[1:\ell] = \ell y\right) \times \\
&\quad \left[\sum_{l=1}^{\ell-1} \mathbf{Pr}\left(\mathsf{GUESS}_i^l\right) \frac{y}{2^c - 3(i-1)} + \mathbf{Pr}\left(\mathsf{GUESS}_i^\ell\right) \frac{y}{2^{b-s} - 3(i-1)}\right] \\
&\leq \sum_{y=1}^{2^c} \mathbf{Pr}\left(\mathcal{S}[1:\ell] = \ell y\right) \left[\sum_{l=1}^{\ell-1} \left(\mathsf{E}\left(\mathcal{G}^l\right) \frac{2y}{2^c}\right) + \mathsf{E}\left(\mathcal{G}^\ell\right) \frac{2y}{2^{b-s}}\right] \\
&\leq \frac{2(q_k - q_{on,k})}{2^b} \mathsf{E}\left(|\mathcal{S}[1:\ell]|\right) , \qquad (26)
\end{aligned}$$

where we used that $q_k \leq 2^c/6 \leq 2^{b-s}/6$. Therefore, combining (25) and (26), we obtain

$$\begin{aligned}
\mathbf{Pr}\left(\mathsf{BADINV}[q_k - q_{on,k}]\right) \leq\ & \mathbf{Adv}^{\text{PRF-krev}}_{\text{OKS}}(\ell + \ell', 2\ell(q_k - q_{on,k})) \\
& + \frac{2(q_k - q_{on,k})}{2^b} \mathsf{E}\left(|\mathcal{S}[1:\ell]|\right) . \qquad (27)
\end{aligned}$$

Probability of $\mathsf{INNER}(q_k - q_{on,k})$. For $i \in [\![q_k - q_{on,k}, q_k]\!]$, conditioned on the fact that $\mathsf{INNER}(q_k - q_{on,k})$ has not been set beforehand, the probability that the i^{th} query triggers $\mathsf{INNER}(q_k - q_{on,k})$ is upper bounded by $\frac{i2^r}{2^b - i}$. Therefore, using that $q_k \leq 2^{b-1}$,

$$\mathbf{Pr}\left(\mathsf{INNER}(q_k - q_{on,k})\right) \leq \sum_{i=q_k - q_{on,k}+1}^{q_k} \frac{2i}{2^c} \leq \frac{2q_k q_{on,k}}{2^c} . \qquad (28)$$

Conclusion. Plugging (23), (24), (27), and (28) into (22), we obtain

$$\begin{aligned}
\mathbf{Pr}\left(\mathsf{PRE}\right) \leq\ & \frac{2q_k}{2^b} \mathsf{E}\left(|S[0]|\right) + \frac{2(q_k - q_{on,k})}{2^b} \mathsf{E}\left(|\mathcal{S}[1:\ell]|\right) \\
& + \mathbf{Adv}^{\text{PRF-krev}}_{\text{OKS}}(\ell + \ell', 2\ell(q_k - q_{on,k})) \\
& + \min\left\{\frac{2q_{on,k}}{2^c} \mathsf{E}\left(|S[1:\ell]|\right), \frac{2q_{on,k} \cdot q_k}{2^c}\right\} . \qquad (29)
\end{aligned}$$

Now, it remains to compute the expectation of the sets $\mathcal{S}[0]$ ans $\mathcal{S}[1:\ell]$. Again, we use the same approach as [20], and use the linearity of the expectation. We

have

$$\begin{aligned}
\mathsf{E}\left(\mathcal{S}[1]\right) &= \sum_{y\in\{0,1\}^b} \mathsf{E}\left(\mathbf{1}_{Y\in\mathcal{S}[1]}\right) \\
&\leq \sum_{\substack{y\in\{0,1\}^b \text{ s.t.,} \\ \mathrm{outer}_r(y)=x_k^{(1)}}} \mathbf{Pr}\left(Y\in\mathcal{S}[1]\right) \\
&\leq \sum_{\substack{y\in\{0,1\}^b \text{ s.t.,} \\ \mathrm{outer}_r(y)=x_k^{(1)}}} \frac{2^c}{2^b}\frac{2^c}{2^b-1}\cdots\frac{2^{b-s}}{2^b-(\ell-2)} \\
&= \frac{2^c(2^c)^{\ell-2}\cdot 2^{c-s}}{[2^b]_{\ell-1}} \\
&\leq 2\frac{2^b}{2^n}\,,
\end{aligned}$$

where we used that $2[2^b]_{\ell-1} \geq (2^b)^{\ell-1}$ if $(\ell-1)^2 \leq 2^b$ (see [20, Section 2.1]). Because $\mathcal{P}$ is a permutation, we have for any $l \in [\![0,\ell]\!]$, $\mathcal{S}[l] = \mathcal{S}[1]$, therefore $\mathcal{S}[1:\ell] = \ell \times \mathcal{S}[1]$. Thus,

$$\begin{aligned}
\mathsf{E}\left(\mathcal{S}[0]\right) &\leq 2\frac{2^b}{2^n}\,, \\
\mathsf{E}\left(\mathcal{S}[1:\ell]\right) &\leq 2\ell\frac{2^b}{2^n}\,.
\end{aligned}$$

Plugging these bounds into (29) gives

$$\begin{aligned}
\mathbf{Pr}_{S_k}\left(\mathcal{A} \text{ solves } \mathrm{pre}(x_k) \text{ in } G(k)\right) \leq \frac{8\ell q_k}{2^n} + \min\left\{\frac{4\ell q_{on,k}}{2^{n-r}}, \frac{2q_{on,k}\cdot q_k}{2^c}\right\} \\
+ \mathbf{Adv}^{\text{PRF-krev}}_{\text{OKS}}(\ell+\ell', 2\ell(q_k - q_{on,k}))\,.
\end{aligned}$$

C Proof of Theorem 3

The proof shares similarities with the proof of Theorem 1, particularly in the way that forward queries to $\mathcal{P}$ are treated. A difference is in the fact that, in current setting, the adversary can also make inverse evaluations of $\mathcal{P}^{-1}$.

Setup. We split q_{on} as $q_{fwd,on}+q_{inv,on}$, depending on the direction of the queries, and similarly split $q_{off} := q_{fwd,off} + q_{inv,off}$. We first define useful notation for the treatment of forward queries. As in the proof of Theorem 1, let Coll_s be a random variable with sample space $([\![1,M]\!])^M$ that determines how are the salts colliding together. More precisely, when two elements at position i and j share the same value, it means that the salts of users i and j collide. Fixing Coll_s determines the number of distinct salts, that we will denote as $\tilde{M}$.

Moreover, let NC_s be a random variable defined as follows:

$$\mathrm{NC}_s = \max_{id} \#\left\{m \;\middle|\; id_m = id\right\} .$$

NC_s counts the maximal number of jointly colliding salts. The value of Coll_s determines the one of NC_s. We saw in (6) that

$$\mathsf{E}\left(\mathrm{NC}_s\right) \leq \frac{2M}{2^s} + 3s + 4 . \tag{30}$$

Assume that Coll_s is fixed to a certain coll_s, which gives $\tilde{M} \leq M$ distinct salts, and fixes NC_s to nc_s. Let $k \in [\![1, K]\!]$, and $\tilde{m} \in [\![1, \tilde{M}]\!]$. Note that we can partition the forward queries according to the associated values of $(\tilde{m}, k)$. In particular, a forward permutation query with the unique salt number $\tilde{m}$ and counter k is called a $\mathcal{P}_{k,\tilde{m}}$-query. During the offline phase, whenever $M \ll 2^s$, then with high probability a large portion of the queries do not correspond to $\mathcal{P}_{k,\tilde{m}}$-queries. Let $Q_{fwd,off}^{(k,\tilde{m})}$ be a random variable counting the number of useful $\mathcal{P}_{k,\tilde{m}}$-queries during the offline phase, and let $q_{fwd,on}^{(k,\tilde{m})}$ be the number of useful $\mathcal{P}_{k,\tilde{m}}$ queries during the online phase. Let

$$Q_{fwd} := \sum_{k=1}^{K} \sum_{\tilde{m}=1}^{\tilde{M}} Q_{fwd,off}^{(k,\tilde{m})} + q_{fwd,on}^{(k,\tilde{m})} ,$$

so that Q_{fwd} counts the total number of useful forward queries. Similarly to (13) and (14), we have

$$\mathsf{E}\left(Q_{fwd} \;\middle|\; \mathrm{Coll}_s = \mathrm{coll}_s\right) \leq \min\left\{\frac{2M}{2^s}; 1\right\} \times q_{fwd,off} + q_{fwd,on} . \tag{31}$$

Now, we introduce terminology for the treatment of inverse queries. We define $\mathrm{Coll}_{\mathrm{CV}}$ and $\mathrm{NC}_{\mathrm{CV}}$ respectively, that play a role similar to Coll_s and NC_s, respectively. $\mathrm{Coll}_{\mathrm{CV}}$ determines how the chaining values $x_{k,m}$ from distinct permutation calls are colliding, and is a pair of random variables with sampling space $([\![1, M]\!])^M \times ([\![1, KM]\!])^{KM}$. The first component of $\mathrm{Coll}_{\mathrm{CV}}$ corresponds to the random variable Coll_s, and the second one can be represented as a matrix with K rows and M columns, each element taking values in $[\![1, KM]\!]$. If the element at row k and column m equals the element at row k' and column m', it means that $x_{k,m}$ collides with $x_{k',m'}$. This random variable needs to keep track of Coll_s, as there might be cases where two different salts id_m and $id_{m'}$ collide, and $x_{k,m} = x_{k,m'}$ due to a lucky collision. In that case (and only in that case), the permutation calls to compute $x_{k+1,m}$ and $x_{k+1,m'}$ will be the same. However, we want to avoid counting these cases as collisions between chaining values. We define the random variable $\mathrm{NC}_{\mathrm{CV}}$ as the maximum number of jointly chaining values that come from *distinct* permutation evaluations. Note that fixing the value of $\mathrm{Coll}_{\mathrm{CV}}$ also fixes the value of $\mathrm{NC}_{\mathrm{CV}}$. Then, each chaining value

(without counting repeated permutation evaluations) is the result of a sampling in $\{0,1\}^b$ without replacement. Therefore, from Lemma 1, we know that

$$\mathsf{E}\left(\mathrm{NC}_{\mathrm{CV}}\right) \leq \frac{2KM}{2^n} + 3n + 4\,. \tag{32}$$

Now, assume that $\mathrm{Coll}_{\mathrm{CV}}$ is set to a certain value $\mathrm{coll}_{\mathrm{CV}}$. Given this fixed instance, we can infer the number of distinct chaining values that we denote by $\mathrm{dist}_{\mathrm{CV}}$. Given $u \in [\![1, \mathrm{dist}_{\mathrm{CV}}]\!]$, let $Q^{(u)}_{inv,off}$ (resp., $Q^{(u)}_{inv,on}$) be a random variable which counts the number of inverse queries during the offline (resp., online) phase that have their n leftmost bits fixed to the u^{th} distinct chaining value. Note that this quantity is also a random variable for the online phase, since the adversary might make lucky inverse queries with $x_{k,m}$ *before* the value is revealed. Moreover, let

$$Q_{inv} = \sum_u Q^{(u)}_{inv,off} + Q^{(u)}_{inv,on}\,.$$

We now derive an upperbound for $\mathsf{E}\left(Q_{inv} \mid \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right)$. We first consider the case where $KM \leq 2^{n-1}$. Let $z \in \{0,1\}^n$. Conditionning on the random variable $\mathrm{Coll}_{\mathrm{CV}}$ introduces a supplementary bias in the sampling of the values $x_{k,m}$. Indeed, if the first u distinct chaining values have already been sampled, the $(u+1)^{\mathrm{th}}$ distinct chaining value cannot have its n leftmost bits equal to any of the u previous chaining values. Consequently, the u^{th} distinct chaining value is the result of a sampling from a set of size at least $2^b - u2^c \geq 2^b - KM \times 2^c$, among which at most 2^c values have their n leftmost bits equal to z. Hence, the probability that one of the $\mathrm{dist}_{\mathrm{CV}}$ chaining values equals to z can be upper bounded by

$$\frac{\mathrm{dist}_{\mathrm{CV}} \times 2^c}{2^b - KM \times 2^c} \leq \frac{2KM}{2^n}\,,$$

where we used $KM \leq 2^{n-1}$.

In the case where $KM \geq 2^{n-1}$, we upper bound $\mathsf{E}\left(Q_{inv} \mid \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right)$ simply by $q_{inv,off} + q_{inv,on}$. Therefore,

$$\begin{aligned}\mathsf{E}\left(Q_{inv} \mid \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) &= \mathsf{E}\left(\sum_u Q^{(u)}_{inv,off} + Q^{(u)}_{inv,on} \;\middle|\; \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) \\ &\leq \min\left\{\frac{2KM}{2^n}; 1\right\} \times q_{inv,off} + q_{inv,on}\,. \end{aligned} \tag{33}$$

Bad Event and Probability Splitting. Let $\mathcal{Q}_k$ be the query history of the adversary *before* the $x_{k,m}$'s are revealed. The adversary winning the security game implies that the following bad event happened:

$$\begin{aligned}\mathsf{BAD} : \exists m \in [\![1,M]\!], k \in [\![1,K]\!], d \in \{fwd, inv\}, y \in \{0,1\}^n, z \in \{0,1\}^c \\ \text{such that } \Big((y \parallel \mathrm{ctr}_k \parallel id_m \parallel IV), (x_{k,m} \parallel z), d \Big) \in \mathcal{Q}_{k-1}\,.\end{aligned}$$

In other words, the adversary must make a query which collides with one of the M chains. Now, we split BAD as $\mathsf{BADFWD} \vee \mathsf{BADINV}$, depending on the direction of the query triggering BAD. Without loss of generality, we can stop the security game whenever the adversary has set BAD. Therefore, the bad events BADFWD and BADINV are disjoint, so that

$$\mathbf{Pr}\left(\mathsf{BAD}\right) \leq \mathbf{Pr}\left(\mathsf{BADFWD} \wedge \neg\mathsf{BADINV}\right) + \mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADFWD}\right) . \quad (34)$$

Probability of BADFWD*.* This probability can be upper bounded in a similar way to what has been done in Theorem 1. We have

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADFWD} \wedge \neg\mathsf{BADINV}\right) \\
&\leq \sum_{\mathrm{coll}_s} \sum_{q_{fwd}} \mathbf{Pr}\left(\mathsf{BADFWD} \;\middle|\; \mathrm{Coll}_s = \mathrm{coll}_s \wedge Q_{fwd} = q_{fwd} \wedge \neg\mathsf{BADINV}\right) \times \\
&\qquad \mathbf{Pr}\left(Q_{fwd} = q_{fwd} \;\middle|\; \mathrm{Coll}_s = \mathrm{coll}_s\right) \times \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) . \qquad (35)
\end{aligned}$$

The conditioned event BADFWD can be decomposed in a query-wise fashion. Indeed, due to the condition $Q_{fwd} = q_{fwd}$, the total number of queries likely to trigger BADFWD with a non-zero probability is equal to q_{fwd}. Now, assuming that BADFWD was not set before the useful query number i, then BADFWD can be set in two different ways during the query number i:

- The adversary guesses one *exact* preimage. In that case each of the chaining values are random, with a small bias due to the permutation.[5] More precisely, for each $k \in [\![0, K]\!]$, $m \in [\![1, M]\!]$, and $z \in \{0,1\}^n$,

$$\mathbf{Pr}\left(x_{k,m} = z\right) \leq \frac{2^c}{2^b - KM} \leq \frac{2}{2^n} ,$$

  where we used $KM \leq 2^{b-1}$. Moreover, one attempt of the adversary targets at most nc_s preimages at the same time, thus its success probability is upper bounded by $\frac{2\mathrm{nc}_s}{2^n}$;
- The adversary guesses a preimage which is not the *exact* one. In that case, we can use directly the randomness of the permutation at the time of the query. For a given forward query, the answer is drawn from a set of size at least $2^b - KM - q$, and among them at most $\mathrm{nc}_s \times 2^c$ values hit the targeted chaining values. Therefore, one attempt of the adversary has a success probability of at most $\frac{2\mathrm{nc}_s}{2^n}$, where we used $KM + q \leq 2^{b-1}$.

Therefore,

$$\mathbf{Pr}\left(\mathsf{BADFWD} \;\middle|\; \mathrm{Coll}_s = \mathrm{coll}_s \wedge Q_{fwd} = q_{fwd} \wedge \neg\mathsf{BADINV}\right) \leq \frac{4\mathrm{nc}_s \times q_{fwd}}{2^n} .$$

[5] Actually, the root passwords $x_{0,m}$ are uniformly random, but this does not change the upper bounding.

Now plugging this upper bound into (35) gives

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADFWD} \wedge \neg\mathsf{BADINV}\right) \\
&\leq \sum_{\mathrm{coll}_s} \sum_{q_{fwd}} \frac{4\mathrm{nc}_s \times q_{fwd}}{2^n} \mathbf{Pr}\left(Q_{fwd} = q_{fwd} \;\middle|\; \mathrm{Coll}_s = \mathrm{coll}_s\right) \times \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \\
&= \frac{4}{2^n} \sum_{\mathrm{coll}_s} \mathsf{E}\left(Q_{fwd} \;\middle|\; \mathrm{Coll}_s = \mathrm{coll}_s\right) \times \mathrm{nc}_s \times \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) .
\end{aligned}$$

Plugging (31) into the equation above gives

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADFWD} \wedge \neg\mathsf{BADINV}\right) \\
&\leq \frac{4}{2^n} \left(\min\left\{\frac{2M}{2^s}; 1\right\} \times q_{fwd,off} + q_{fwd,on}\right) \times \sum_{\mathrm{coll}_s} \mathrm{nc}_s \times \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \\
&= \frac{4}{2^n} \left(\min\left\{\frac{2M}{2^s}; 1\right\} \times q_{fwd,off} + q_{fwd,on}\right) \times \mathsf{E}\left(\mathrm{NC}_s\right) .
\end{aligned}$$

Finally, plugging (30) into the equation above allows us to conclude that

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADFWD} \wedge \neg\mathsf{BADINV}\right) \\
&\qquad \leq \left(\frac{2M}{2^s} + 3s + 4\right) \times \left(\min\left\{\frac{2M}{2^s}; 1\right\} \times \frac{4q_{fwd,off}}{2^n} + \frac{4q_{fwd,on}}{2^n}\right) . \quad (36)
\end{aligned}$$

Probability of BADINV. It remains to upper bound the second term of (34). We have

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADFWD}\right) \\
&\leq \sum_{\mathrm{coll}_{\mathrm{CV}}} \sum_{q_{inv}} \mathbf{Pr}\left(\mathsf{BADINV} \mid \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}} \wedge Q_{inv} = q_{inv} \wedge \neg\mathsf{BADFWD}\right) \times \\
&\qquad \mathbf{Pr}\left(Q_{inv} = q_{inv} \mid \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) \times \mathbf{Pr}\left(\mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) . \quad (37)
\end{aligned}$$

Now, regarding the conditioned BADINV, we can reason in a query-wise fashion. The only queries that can trigger BADINV with a non-zero probability are inverse queries with their leftmost bits fixed to one of the $\mathrm{dist}_{\mathrm{CV}}$ chaining values. For a useful query to succeed, there are two possibilities:

– The adversary guesses the *exact* state. In particular, it should guess the full inner part of c bits. These c bits are random, so that for any $y \in \{0,1\}^c$, $m \in [\![1, M]\!]$, $k \in [\![1, K]\!]$,

$$\mathbf{Pr}\left(\mathrm{inner}_c(\mathcal{P}(x_{k-1,m} \parallel \mathrm{ctr}_k \parallel id_m \parallel IV)) = y\right) \leq \frac{2^n}{2^b - KM} \leq \frac{2}{2^c} .$$

Moreover, among the chaining values, there are at most $\mathrm{nc}_{\mathrm{CV}}$ different c-bit states which are attached to the same chaining value $x_{k,m}$. Therefore, one attempt succeeds with probability at most $\frac{2\mathrm{nc}_{\mathrm{CV}}}{2^c}$;

– The adversary guesses a preimage which does not correspond to the *exact* state. In that case we can use directly the randomness of the permutation. The inverse query answer is drawn from a set of size at least $2^b - KM - q_{on} - q_{off} \geq 2^{b-1}$, and among them at most $\mathrm{nc}_{\mathrm{CV}} \times 2^n$ elements set BADINV. Therefore, one attempt to set this event happens with probability at most $\frac{2\mathrm{nc}_{\mathrm{CV}}}{2^c}$.

Plugging these bounds into (37) gives

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADFWD}\right) \\
&\leq \sum_{\mathrm{coll}_{\mathrm{CV}}} \sum_{q_{inv}} \frac{4q_{inv} \times \mathrm{nc}_{\mathrm{CV}}}{2^c} \times \mathbf{Pr}\left(Q_{inv} = q_{inv} \mid \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) \times \mathbf{Pr}\left(\mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) \\
&= \frac{4}{2^c} \sum_{\mathrm{coll}_{\mathrm{CV}}} \mathsf{E}\left(Q_{inv} \;\middle|\; \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) \times \mathrm{nc}_{\mathrm{CV}} \times \mathbf{Pr}\left(\mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right).
\end{aligned}$$

We can use the inequality from (33) to obtain

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADFWD}\right) \\
&\leq \frac{4}{2^c} \sum_{\mathrm{coll}_{\mathrm{CV}}} \left(\min\left\{\frac{2KM}{2^n}; 1\right\} q_{inv,off} + q_{inv,on}\right) \times \mathrm{nc}_{\mathrm{CV}} \times \mathbf{Pr}\left(\mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) \\
&= \mathsf{E}\left(\mathrm{NC}_{\mathrm{CV}}\right) \times \left(\min\left\{\frac{2KM}{2^n}; 1\right\} \times \frac{4q_{inv,off}}{2^c} + \frac{4q_{inv,on}}{2^c}\right).
\end{aligned}$$

Finally, we use (32), and conclude that

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADFWD}\right) \\
&\quad \leq \left(\frac{2KM}{2^n} + 3n + 4\right) \times \left(\min\left\{\frac{2KM}{2^n}; 1\right\} \times \frac{4q_{inv,off}}{2^c} + \frac{4q_{inv,on}}{2^c}\right). \quad (38)
\end{aligned}$$

Conclusion. We conclude by plugging (36) and (38) into (34).