Improving the Security of the Sphinx Mix Network

Filipe Beato[1], Kimmo Halunen[2], and Bart Mennink[1]

[1] Dept. Electrical Engineering, ESAT/COSIC, KU Leuven, and iMinds, Belgium
{filipe.beato,bart.mennink}@esat.kuleuven.be
[2] VTT Technical Research Center of Finland, Oulu, Finland
kimmo.halunen@vtt.fi

Abstract. The need for secure mix networks to protect the privacy of communications has increased in recent times. Sphinx is one widely used mix network protocol (?). We present some improvements on the security of Sphinx by introducing new authenticated encryption methods with large state, that can be employed in the Sphinx construction. This would protect against the weaknesses found in the Lioness cipher originally used in Sphinx

1 Introduction

With the large growth of internet services, modern users rely more on digital and ubiquitous communications. In this digital domain, privacy needs to be protected against the very nature of the communications, which tend to be easily traceable and produce massive amounts of metadata. Such a task requires some cryptographic methods.

It is not easy to hide the metadata of communications, but there are some systems, called mix networks, that provide such capabilities. One recent example is the Sphinx [20] mix network, that is in wide use in privacy protecting applications. The Sphinx mix network format provides security against powerful adversaries and also good communications possibilities such as replies, which are not easily available in other mix network formats.

There are some parts of the protocol that rely on a block cipher with a large state, to provide security in the system. In the original Sphinx format, this block cipher is LIONESS [1]. This blockcipher is made out of the SEAL stream cipher and a keyed version of the SHA-1 hash function, and evaluates these functions on the message via a Feistel structure.

However, LIONESS dates back to 1996, and has been outpaced by reality. In more detail, attacks on SEAL [25] and SHA-1 [22,32,44,45,47], the most recent result being a free-start collision attack on the full SHA-1, shine a negative light on the security of the LIONESS mode.

Thus, we propose to use more recent advances in sponge-based block ciphers that can accommodate a sufficient state size and provide better security than the original LIONESS. This paper will present our constructions and show the

applicability of these to the Sphinx mix format. We will also show alternate constructions that can be used with Sphinx on reduced parameters. In this vein, there is an incentive to design mix formats with smaller headers. Furthermore, we argue that there are also other use cases, where the large state is needed to avoid the possible reduction on security that seems to come from iteration. In this case, there is need for further research on such primitives like LIONESS that have potentially extremely large state.

The paper is organised in the following way BLAH BLAH BLAH...

1.1 Related Work on Authenticated Encryption

Authenticated encryption (AE) enjoys a long and steady line of research, and the ongoing CAESAR competition [19] for the design of new AE schemes has induced renewed attention to the field. The classical approach to design AE schemes is to build the generic mode of operation on top of a blockcipher (usually AES) in order to process data blocks iteratively [5,10,21,30,33,37,41,42]. A more novel approach is to design AEs based on permutations. The most well-known approach is SpongeWrap by Bertoni et al. [12] which got recently generalized by Jovanovic et al. [31] and Mennink et al. [36], and various CAESAR submissions follow this idea [2,7,15,27,23,38,43]. Different permutation based approaches include APE [3] and PAEQ [17].

Yet, the focus of this work is mostly on *compact* and *highly-secure* authenticated encryption, and particularly its connection to above privacy applications, leaving most of the blockcipher based modes inadequate for our purposes. The reason is that these often entail overhead, e.g. in the form of padding, and the security level is in the end dominated by what the underlying primitive offers. For blockcipher based modes, using AES internally delivers at most 128-bit security, and often there exist already distinguishability attacks in complexity of about 2^{64} (cf. Bellare et al. [8]). In addition, for messages of say 1024 bits, solutions of this type still require at least 8 AES evaluations. Aligned with the fact that generally tweets (or text messages) are of a relatively short (but larger than 128 bits) length, alternative approaches will be more suitable.

Although AEZ [30], or more detailed the latest version v4 in the CAESAR competition [29], is an exception to this, AEZ is also inherently a mode based on 4 and 10 rounds of AES, and has 64-bit security as well. Recent cryptanalysis on AEZ [26,35] has moreover shined a negative light on its security. For permutation based approaches the situation is different, which is in part because cryptographic permutations are often *much* larger than blockciphers. For instance, the Keccak [13] and Keyak [15] permutation are of size 1600 bits, and a smart balance among the internal parameters of the SpongeWrap can allow the security level to reach the 800-bit barrier [31], an observation which we have used in our Sponge based instantiation of AITO.

2 Sphinx message format

Mix networks rely on (cryptography based) mix message formats that provide nice efficiency and security properties. Sphinx [20] is the most compact cryptographic mix message format which is provably secure and efficient. To deliver such properties, Sphinx relies on the *Sphinx blinding logic* technique for generating a session key consisting of nested MAC computations over the public pseudonyms of each predecessor mix. The private key associated to the public key (i.e., pseudonym) is only known by the user, while the session key is used for the encryption of the payload message.

Internally, Sphinx uses a plurality of cryptographic primitives. First, there are five hash functions, which are used to cryptographically hash group elements to key bit strings. Then, it uses a pseudorandom generator and a MAC function for the computation of the nested MAC. Finally, it uses an encryption scheme that encrypts the payload at every mix.

For the encryption, Sphinx relies on the LIONESS blockcipher by Anderson and Biham [1]. This blockcipher is made out of the SEAL stream cipher and a keyed version of the SHA-1 hash function, and evaluates these functions on the message via a Feistel structure. In more detail, denote the stream cipher by S_k and the keyed hash function by H_k. Consider a LIONESS key $k = (k_1, k_2, k_3, k_4)$, where k_1, k_3 will be used to key the stream cipher and k_2, k_4 to key the hash function. To encrypt a message m, LIONESS first splits it into two blocks $m_l \| m_r \leftarrow m$. These two blocks are then transformed using a four-round Feistel structure:

$$
\begin{aligned}
m_r &\leftarrow m_r \oplus S_{k_1}(m_l)\,, \\
m_l &\leftarrow m_l \oplus H_{k_2}(m_r)\,, \\
m_r &\leftarrow m_r \oplus S_{k_3}(m_l)\,, \\
m_l &\leftarrow m_l \oplus H_{k_4}(m_r)\,.
\end{aligned}
$$

The updated $m_l \| m_r$ constitutes the ciphertext c.

LIONESS enjoys a security proof if the underlying hash function SHA-1 and stream cipher SEAL are secure [1], therewith making it particularly useful for Sphinx because of its goal to achieve provable security. However, LIONESS dates back to 1996, and has been outpaced by reality. In more detail, attacks on SEAL [25] and SHA-1 [22,32,44,45,47], the most recent result being a free-start collision attack on the full SHA-1, shine a negative light on the security of the LIONESS mode.

The cipher is used in Section 3.3 of [20] to generate a forward message. Because of the security parameter choices, the Sphinx construction needs a blockcipher with a state of at least 1408 bits plus the message length. Thus, LIONESS offers a good option as it has the potential to have a large state and thus act as the permutation required by the Sphinx system.

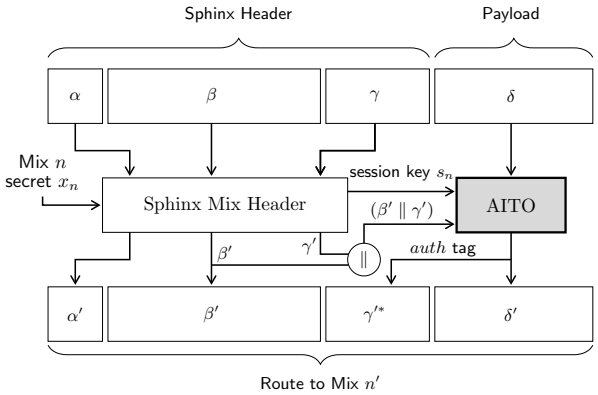

Fig. 1. Using AITO in Sphinx to process a Sphinx message $((\alpha, \beta, \gamma), \delta)$ into $((\alpha', \beta', \gamma'^*), \delta)$ at Mix n with the secret x_n. Note that the key derivation from $s_n = \mathsf{keyderive}(\alpha, x_n)$, blinding $\alpha_n = \mathsf{blind}(\alpha, s_n)$, and $(\beta, \gamma) \to (\beta', \gamma')$ computations occur as in the original Sphinx (see [20, Figure 3]). The session key s_n is used as the key, whereas the header (β', γ') is used as metadata on the input of AITO.

3 Improving the Sphinx

A naive solution to this situation is to replace SEAL by a more modern stream cipher and to replace SHA-1 by SHA-3, but there is little point in doing so: versatility of Sponges in general and SHA-3 in particular enables encryption using SHA-3 on the fly; putting a four-round Feistel construction on top of it is overkill. Instead, it makes more sense to simply *replace* LIONESS by a keyed version of the SHA-3. The AITO Sponge construction of Section 7 is particularly suited for this purpose, as it is an AE scheme based on the SHA-3 permutation. In this way, it seems logical to also replace the other cryptographic functionalities in Sphinx by SHA-3 or SHA-3 based alternatives. Alternatively, one can use the Threefish based version of AITO (Section 5 or Section 6), and use the Skein hash function family [24] to serve for hashing, as it already uses Threefish natively.

Either approach makes the encryption functionality of Sphinx more secure and more efficient. Figure 1 depicts our proposal of using AITO in Sphinx. In addition, AITO supports tweaks as input which could be used for the processing of the header, it allows for authentication, and could potentially be used as MAC function. These advantages could be exploited to integrate part of the nested MAC functionality of Sphinx within AITO. As a bonus, Sphinx using AITO *additionally* authenticates the payload for free, a feature that was missing in the original Sphinx.

3.1 Relaxing the security

The security parameters used in Sphinx are r denoting the maximum number of hops between the sender and the destination and k, which is the traditional

security parameter for the underlying primitives. Because of these values, we get the $1408 (= 2 \times r \times k + k)$ bit minimum requirement for the internal state of the block cipher.

Relaxing the security notions slightly, we can get state sizes that can be accommodated by further constructions based on tweakable blockciphers namely the Threefish and its variations. For example, with $r = 4$ and $k = 100$ would yield 900 bits and $r = 3$ and $k = 128$ 856 bits, that could be processed with the following type of construction.

INSERT THREEFISH CONSTRUCTION HERE

4 Model

This section introduces AITO and the respective threat model. For $n \in \mathbb{N}$, $\{0,1\}^n$ is the set of n-bit strings, and $\{0,1\}^{\leq n} = \bigcup_{i=0}^{n} \{0,1\}^i$. For two bit strings M, N, their concatenation is denoted by $M \| N$ and $M \oplus N$ denotes their bitwise XOR. Furthermore, if $M \in \{0,1\}^{\leq n-1}$, then $\mathsf{pad}_n(M) = M \| 10^{n-1-|M|}$. For a string $N \in \{0,1\}^n$, we define by $\mathsf{unpad}_n(N)$ the unique string $M \in \{0,1\}^{\leq n-1}$ such that $\mathsf{pad}_n(M) = N$. For $m \leq n$ and $N \in \{0,1\}^n$, we denote by $\lceil N \rceil_m$ the leftmost m bits and by $\lfloor N \rfloor_{n-m}$ the rightmost $n-m$ bits of N, in such a way that $N = \lceil N \rceil_m \| \lfloor N \rfloor_{n-m}$.

4.1 AITO

AITO is an AE scheme for short data. Let $\lambda_{\max} \in \mathbb{N}$ be an integer that specifies the maximal size of an authenticated ciphertext (for instance $\lambda_{\max} = 1024$ or 1120). Let $\mu, \nu, \alpha, \tau, \sigma \in \mathbb{N}$ be size values that satisfy $\mu \leq \nu$ and $\nu + \alpha + \tau + \sigma \leq \lambda_{\max}$. Such that, μ denotes the size of the message, ν the size of the ciphertext, τ the size of the meta data, and σ the size of the nonce. The value α determines the size of the authentication tag. If no authentication is needed, we have $\alpha = 0$.

AITO is composed of three algorithms: `KeyGen`, `Enc`, and `Dec`. `KeyGen` is a randomized algorithm that gets as input $\kappa \in \mathbb{N}$ and outputs a random key $key \leftarrow \{0,1\}^\kappa$. The `Enc` and `Dec` algorithms are defined as follows:

$$\begin{aligned}
\texttt{Enc}: \ & \{0,1\}^\kappa \times \{0,1\}^{\leq \mu} \times \{0,1\}^{\leq \tau} \times \{0,1\}^\sigma \to \{0,1\}^{\leq \nu} \times \{0,1\}^\alpha, \\
& (key, msg, meta, nonce) \mapsto (ctxt, auth), \\
\texttt{Dec}: \ & \{0,1\}^\kappa \times \{0,1\}^{\leq \nu} \times \{0,1\}^\alpha \times \{0,1\}^{\leq \tau} \times \{0,1\}^\sigma \to \{0,1\}^{\leq \mu} \cup \{\perp\}, \\
& (key, ctxt, auth, meta, nonce) \mapsto msg/\perp.
\end{aligned}$$

`Dec` outputs the unique msg satisfying $\texttt{Enc}(key, msg, meta, nonce) = (ctxt, auth)$, or it returns $\perp$ if no such message exists. `Enc` also outputs $meta$ and $nonce$. We allow for a small amount of ciphertext expansion (from μ to ν bits), as long as the encrypted ciphertext $(ctxt, auth, meta, nonce)$ is of size at most $\lambda_{\max}$.

As aforementioned various approaches of AE schemes exist; see for instance the CAESAR competition [19] for the design of a new AE scheme. These schemes

are often generic modes of operation that process data blocks iteratively. However, AITO is designed for small data, and orthogonal design approaches turn out to be more suitable. In particular, the key principle of the design of AITO is inspired by tweakable blockciphers, where the meta data and nonce function as the tweak.

4.2 Threat Model

We consider an adversary $\mathcal{A}$ to be any entity attempting to passively access the shared information by monitoring the communication channel, with no incentive to tamper with the content. However, $\mathcal{A}$ is allowed to generate encryptions under a secret and unknown key. In this case, $\mathcal{A}$ should not learn the encrypted content, beyond that revealed in the meta data.

More technically, adversary $\mathcal{A}$ has query access to $\mathtt{Enc}$ under a secret key key, and it tries to find irregularities among the queries, i.e., some relation that is not likely to hold for a random function. Here, we require $\mathcal{A}$ to be nonce respecting, so that every query must be made under a different nonce (see also Section 9). For a function F, let $\mathsf{Func}(F)$ be the set of all functions f with the same interface as F. The advantage of an adversary $\mathcal{A}$ in breaking the secrecy of AITO is defined as follows:

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{AITO}}(\mathcal{A}) = \left| \begin{array}{r} \Pr\left(key \xleftarrow{\$} \mathtt{KeyGen}(\kappa) \ : \ \mathcal{A}^{\mathtt{Enc}_{key}} = 1\right) - \\ \Pr\left(\$ \xleftarrow{\$} \mathsf{Func}(\mathtt{Enc}_{key}) \ : \ \mathcal{A}^{\$} = 1\right) \end{array} \right| .$$

We denote by $\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{AITO}}(Q, T)$ the maximum advantage over all adversaries that make at most Q encryption queries and operate in time T.

For the authenticity of AITO, we consider $\mathcal{A}$ to have access to the encryption functionality $\mathtt{Enc}$ under a secret key key, and say that $\mathcal{A}$ *forges* an authentication tag if it manages to output a tuple $(ctxt, auth, meta, nonce) \in \{0,1\}^{\leq \nu} \times \{0,1\}^{\alpha} \times \{0,1\}^{\leq \tau} \times \{0,1\}^{\sigma}$ such that $\mathtt{Dec}(key, ctxt, auth, meta, nonce) = msg \neq \perp$ and $(msg, meta, nonce)$ was never queried to $\mathtt{Enc}$ before. The forgery attempt may be made under a nonce $nonce$ that has appeared before. The advantage of $\mathcal{A}$ in breaking the authenticity of AITO is defined as follows:

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{AITO}}(\mathcal{A}) = \Pr\left(key \xleftarrow{\$} \mathtt{KeyGen}(\kappa) \ : \ \mathcal{A}^{\mathtt{Enc}_{key}} \text{ forges}\right) .$$

We denote by $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{AITO}}(Q, R, T)$ the maximum advantage over all adversaries that make at most Q encryption queries, R forgery attempts, and operate in time T.

5 AITO: Basic Construction

The first approach is to apply a large tweakable blockcipher. A tweakable blockcipher $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ takes as input a key $k \in \mathcal{K}$, a tweak $t \in \mathcal{T}$, and a

Algorithm 1 $\texttt{Enc}^{\text{3fish}}$

Input: $(key, msg, meta, nonce) \in \{0,1\}^{\kappa} \times \{0,1\}^{\leq \mu} \times \{0,1\}^{\leq \tau} \times \{0,1\}^{\sigma}$
Output: $(ctxt, auth) \in \{0,1\}^{\nu} \times \{0,1\}^{\alpha}$
1: $c \leftarrow \text{3fish}(key, \mathsf{pad}_{128}(nonce \| meta), \mathsf{pad}_{1024}(msg))$
2: **return** $(\lceil c \rceil_{\nu}, \lfloor c \rfloor_{\alpha})$

Algorithm 2 $\texttt{Dec}^{\text{3fish}}$

Input: $(key, ctxt, auth, meta, nonce) \in \{0,1\}^{\kappa} \times \{0,1\}^{\nu} \times \{0,1\}^{\alpha} \times \{0,1\}^{\leq \tau} \times \{0,1\}^{\sigma}$
Output: $msg \in \{0,1\}^{\leq \mu}$ or $\perp$
1: $m \leftarrow \text{3fish}^{-1}(key, \mathsf{pad}_{128}(nonce \| meta), ctxt \| auth)$
2: $msg \leftarrow \mathsf{unpad}_{\mu+1}(\lceil m \rceil_{\mu+1})$
3: **return** $\lfloor m \rfloor_{\alpha} = 0 \ ?\ msg : \perp$

message $m \in \mathcal{M}$, and outputs a ciphertext $c \in \mathcal{M}$. It is a permutation for every choice of (k, t).

For AITO, we suggest using *Threefish*, a tweakable blockcipher by Ferguson et al. used for the Skein hash function family [24]. Threefish supports block sizes of 256, 512, and 1024 bits. The key size equals the block size, and the tweak size is 128 bits. We focus on the largest variant, Threefish-1024, which for readability we simply denote 3fish:

$$\begin{aligned} \text{3fish}: \ & \{0,1\}^{1024} \times \{0,1\}^{128} \times \{0,1\}^{1024} \to \{0,1\}^{1024}, \\ & (k, t, m) \mapsto c\,. \end{aligned}$$

3fish can be used for AE directly, a construction we call $\text{AITO}^{\text{3fish}}$. It operates on keys of size $\kappa = 1024$ bits, messages can be of arbitrary length but of size at most $\mu = 1023 - \alpha$, and the sizes of the meta data and nonce should satisfy $\sigma + \tau \leq 127$. The ciphertexts are of size *exactly* $\nu = 1024 - \alpha$ bits, where α is the size of the authentication tag. The latter is required to make decryption possible. At a high level, the encryption consists of putting $m = \mathsf{pad}_{1024}(msg)$ and $t = \mathsf{pad}_{128}(nonce \| meta)$, and the ciphertext and authentication tag are derived as $ctxt \| auth = c$. Formally, the encryption and decryption of $\text{AITO}^{\text{3fish}}$ are defined as in Algorithms 1 and 2.

Security. In this section we formally derive the security of $\text{AITO}^{\text{3fish}}$ under the assumption that 3fish is a secure tweakable blockcipher. The security of a tweakable blockcipher $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ is captured by an adversary $\mathcal{A}$ that has adaptive two-sided oracle access to either $\widetilde{E}_k$ for some secret key $k \xleftarrow{\$} \mathcal{K}$, or ideal tweakable permutation $\widetilde{\pi}$ with tweak space $\mathcal{T}$ and message space $\mathcal{M}$, and tries to distinguish both worlds. Denote by $\widetilde{\mathsf{Perm}}(\mathcal{T}, \mathcal{M})$ the set of tweakable permutations. We define the strong PRP security of $\widetilde{E}$ as

$$\mathbf{Adv}^{\widetilde{\text{sprp}}}_{\widetilde{E}}(\mathcal{A}) = \left| \Pr\left(k \xleftarrow{\$} \mathcal{K} \ : \ \mathcal{A}^{\widetilde{E}^{\pm}_k} = 1\right) - \Pr\left(\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T}, \mathcal{M}) \ : \ \mathcal{A}^{\widetilde{\pi}^{\pm}} = 1\right) \right| .$$

Algorithm 3 $\mathtt{Enc}^{\mathrm{LRW[3fish]}}$

Input: $(key, msg, meta, nonce) \in \{0,1\}^{\kappa} \times \{0,1\}^{\leq \mu} \times \ bint \times \{0,1\}^{\sigma}$
Output: $(ctxt, auth) \in \{0,1\}^{\nu} \times \{0,1\}^{\alpha}$
1: $c \leftarrow \mathrm{LRW[3fish]}(key, \ \mathsf{pad}_{1024}(meta), \mathsf{pad}_{128}(nonce), \mathsf{pad}_{1024}(msg))$
2: **return** $(\lceil c \rceil_{\nu}, \lfloor c \rfloor_{\alpha})$

Algorithm 4 $\mathtt{Dec}^{\mathrm{LRW[3fish]}}$

Input: $(key, ctxt, auth, meta, nonce) \in \{0,1\}^{\kappa} \times \{0,1\}^{\nu} \times \{0,1\}^{\alpha} \times \{0,1\}^{\leq \tau} \times \{0,1\}^{\sigma}$
Output: $msg \in \{0,1\}^{\leq \mu}$ or $\perp$
1: $m \leftarrow \mathrm{LRW[3fish]}^{-1}(key, \ \mathsf{pad}_{1024}(meta), \mathsf{pad}_{128}(nonce), ctxt \| auth)$
2: $msg \leftarrow \mathsf{unpad}_{\mu+1}(\lceil m \rceil_{\mu+1})$
3: **return** $\lfloor m \rfloor_{\alpha} = 0 \ ? \ msg : \perp$

By $\mathbf{Adv}^{\widetilde{\mathrm{sprp}}}_{\widetilde{E}}(Q,T)$ we denote the maximum security advantage of any adversary $\mathcal{A}$ that makes Q queries and runs in time T.

Theorem 1. *Let $n = 1024$ be the state size of* 3fish. *We have*

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{AITO}^{\mathrm{3fish}}}(Q,T) \leq \mathbf{Adv}^{\widetilde{\mathrm{sprp}}}_{\mathrm{3fish}}(Q,T')\,,$$
$$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{AITO}^{\mathrm{3fish}}}(Q,R,T) \leq \mathbf{Adv}^{\widetilde{\mathrm{sprp}}}_{\mathrm{3fish}}(Q+R,T') + \frac{R2^{n-\alpha}}{2^n-1}\,,$$

where $T' \approx T$.

The proof is given in Appendix A. We briefly remark that the construction is even secure under *release of unverified plaintext*, where *msg* is disclosed before tag verification is done [4].

6 AITO: Expanded Tweak Space

The AITO basic construction presents a rather small tweak space, limiting the sizes of the meta data and nonce. One way to resolve this is to employ a random oracle that maps the (larger) meta data and nonce to a string of size 128 bits, but this would significantly degrade the security of the construction as forgeries can be found in a complexity 2^{64}. Another way to enlarge the tweak space without adjusting the cipher itself is by using it in a tweakable mode of operation.

Liskov et al. [34] introduced two tweakable modes of operation: while these constructions are originally designed to add a tweak input to a blockcipher, they can equally well be applied to tweakable blockciphers themselves to enlarge the tweak space. We will consider one of these constructions, which makes two

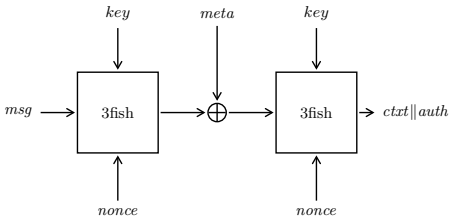

Fig. 2. AITO based on LRW[3fish]. Padding of data is excluded from the figure

evaluations of the underlying cipher:[3]

$$\begin{aligned}\text{LRW[3fish]}: \ &\{0,1\}^{1024} \times \{0,1\}^{1024} \times \{0,1\}^{128} \times \{0,1\}^{1024} \to \{0,1\}^{1024}\,, \\ &(k,t,t',m) \mapsto \text{3fish}(k,t',\text{3fish}(k,t',m) \oplus t)\,.\end{aligned}$$

This construction can be used to realize $\text{AITO}^{\text{LRW[3fish]}}$ as illustrated in Figure 2 and described in Algorithms 3 and 4. The conditions on the sizes of the inputs and outputs carry over from Section 5, with the difference that meta data should now be of size at most $\tau \leq 1023$.

Security. The security of $\text{AITO}^{\text{LRW[3fish]}}$ follows from Theorem 1 and a result from [34]. We refer to Appendix B for the proof.

Theorem 2. *Let $n = 1024$ be the state size of* 3fish*. We have*

$$\mathbf{Adv}^{\text{cpa}}_{\text{AITO}^{\text{LRW[3fish]}}}(Q,T) \leq \Theta\left(\frac{Q^2}{2^n}\right) + \mathbf{Adv}^{\widetilde{\text{sprp}}}_{\text{3fish}}(2Q,T')\,,$$

$$\mathbf{Adv}^{\text{auth}}_{\text{AITO}^{\text{LRW[3fish]}}}(Q,R,T) \leq \Theta\left(\frac{(Q+R)^2}{2^n}\right) + \mathbf{Adv}^{\widetilde{\text{sprp}}}_{\text{3fish}}(2(Q+R),T') + \frac{R2^{n-\alpha}}{2^n-1}\,,$$

where $T' \approx T$.

7 AITO: Sponge Construction

The Sponge functions were originally introduced by Bertoni et al. [14] for cryptographic hashing, but can also be used in a broad spectrum of keyed applications, including message authentication [6,11,39,40] and stream encryption [12,36]. They are also particularly useful for compact AE, and hence allow for an alternative instantiation of AITO. The function realization resembles ideas of the full-state duplex mode [36], transformed to the tweakable setting. Although this instantiation of AITO focuses on compact AE, it can also be used to process longer messages. We remark that, as the keyed Sponges are merely stream based encryption, a unique nonce is required for every encryption.

[3] The other construction is less relevant as it requires an additional key and needs a universal hash function with a 1024-bit range (or smaller, in which case the security of the construction degrades).

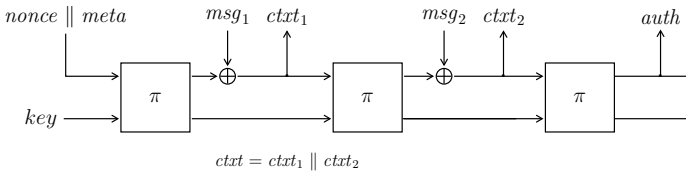

Fig. 3. AITO based on a Sponge. Padding of data is excluded from the figure.

Algorithm 5 $\mathtt{Enc}^{\pi,\ell,n}$

Input: $(key, msg, meta, nonce) \in \{0,1\}^{\kappa} \times \{0,1\}^{\leq \mu} \times \{0,1\}^{\leq \tau} \times \{0,1\}^{\sigma}$
Output: $(ctxt, auth) \in \{0,1\}^{\leq \mu} \times \{0,1\}^{\alpha}$

$\ell' \leftarrow \lceil (|msg| + 1)/n \rceil$
$msg_1 \parallel \cdots \parallel msg_{\ell'} \xleftarrow{n\text{- blocks}} \mathsf{pad}_{\ell',n}(msg)$
$s_0 \leftarrow \mathsf{pad}_n(nonce \| meta) \parallel 0^{b-n-|key|} \parallel key$
for $i = 1, \ldots, \ell'$ **do**
    $s_i \leftarrow \pi(s_{i-1})$
    $s_i \leftarrow s_i \oplus \left(msg_i \parallel 0^*\right)$
    $ctxt_i \leftarrow \lceil s_i \rceil_n$
$s_{\ell'+1} \leftarrow \pi(s_{\ell'})$
return $(\lceil ctxt_1 \parallel \cdots \parallel ctxt_{\ell'} \; rceil_{|msg|}, \lceil s_{\ell'+1} \rceil_{\alpha})$

The realization of $\mathrm{AITO}^{\pi,\ell,n}$ (see Figure 3) is indexed by a permutation π of width b and two parameters ℓ and $n \leq b$ which specify the way it parses the message blocks: it considers at most ℓ message blocks of size n bits. The parameter ℓ can be arbitrarily large, but will be used to show how the length influences the security bound. $\mathrm{AITO}^{\pi,\ell,n}$ operates on keys of size $\kappa \leq b - n$ bits, messages and ciphertexts can be of any length at most $\mu = \ell \cdot n - 1$ (note that the scheme does not use ciphertext expansion, hence $\mu = \nu$), and the sizes of the meta data and nonce should satisfy $\sigma + \tau \leq n - 1$. The size of the authentication tag is $\alpha \leq n$ (this bound is merely for simplicity, the scheme easily generalizes to $\alpha > n$). We require $\mu + \alpha + \tau + \sigma \leq \lambda_{\max}$. The formal encryption and decryption functionalities are given in Algorithms 5 and 6.

Security. $\mathrm{AITO}^{\pi,\ell,n}$ is in fact a full-state duplex construction [36], but it is easier to explain the security of the construction in terms of the Inner-Keyed Sponge (IKS) of Andreeva et al. [6]. This construction gets as input a key k, an arbitrarily sized message m, and a natural number ρ, and it outputs a digest z of size ρ. That is $\mathrm{IKS}^{\pi}(k, m, \rho) = z \in \{0,1\}^{\rho}$. It is defined as the classical Sponge with an outer part of size n and an inner part of size $b - n$, and with the capacity part being initialized using the key. We consider a specific case of IKS where $\rho \leq n$, which means that the squeezing part of the Sponge takes exactly one round.

The security of variable-input-length $\mathrm{IKS} : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^n$ based on a permutation π is slightly different from the CPA security of Section 4.1; it

Algorithm 6 $\mathtt{Dec}^{\pi,\ell,n}$

Input: $(key, ctxt, auth, meta, nonce) \in \{0,1\}^{\kappa} \times \{0,1\}^{\leq \mu} \times \{0,1\}^{\alpha} \times \{0,1\}^{\leq \tau} \times \{0,1\}^{\sigma}$
Output: $msg \in \{0,1\}^{\leq \mu}$ or $\perp$

$\ell' \leftarrow \lceil (|ctxt| + 1)/n \rceil$
$ctxt_1 \parallel \cdots \parallel ctxt_{\ell'} \xleftarrow{n\text{-blocks}} \mathsf{pad}_{\ell' \cdot n}(ctxt)$
$s_0 \leftarrow \mathsf{pad}_n(nonce \| meta) \parallel 0^{b-n-|key|} \parallel key$
for $i = 1, \ldots, \ell' - 1$ **do**
    $s_i \leftarrow \pi(s_{i-1})$
    $msg_i \leftarrow \lceil s_i \rceil_n \oplus ctxt_i$
    if $i < \ell'$ **then**
        $s_i \leftarrow ctxt_i \parallel \lfloor s_i \rfloor_{b-n}$
    else
        $s_i \leftarrow \lceil ctxt_i \rceil_{|ctxt| \bmod n} \parallel \lfloor s_i \rfloor_{b-(|ctxt| \bmod n)}$
$msg \leftarrow \lceil msg_1 \parallel \cdots \parallel msg_{\ell'} \rceil_{|ctxt|}$
$s_{\ell'+1} \leftarrow \pi(s_{\ell'})$
return $\lceil s_{\ell'+1} \rceil_{\alpha} = auth \ ? \ msg : \perp$

differs in two aspects: first, IKS is variable length, so it is compared with a random oracle $\mathcal{RO} : \{0,1\}^* \to \{0,1\}^n$, and second, it is based on an underlying idealized permutation π and the adversary also has two-sided oracle access to π. Denote by $\mathsf{Perm}(\{0,1\}^b)$ the set of b-bit permutations. Abusing notation, we refer to the security of IKS against an adversary that has access to either $(\mathrm{IKS}, \pi^{\pm})$ or $(\mathcal{RO}, \pi^{\pm})$, where $k \xleftarrow{\$} \mathcal{K}$, $\pi \xleftarrow{\$} \mathsf{Perm}(\{0,1\}^b)$, and $\mathcal{RO}$ is a random oracle, by $\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{IKS}}(\mathcal{A})$. We define by $\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{IKS}}(Q, S)$ the maximum advantage over all adversaries with total complexity Q, and that make at most S primitive queries to $\pi^{\pm}$. Here, the total complexity Q counts the number of *fresh calls* to π if $\mathcal{A}$ were conversing with IKS.

Note that if no authentication is needed, then $\mathtt{Enc}^{\pi,\ell,n}$ and $\mathtt{Dec}^{\pi,\ell,n}$ do not require the computation of $s_{\ell'+1} \leftarrow \pi(s_{\ell'})$ at the end, which saves a permutation call. Related to this, we define ℓ_{α} as follows.

$$\ell_{\alpha} = \begin{cases} \ell\,, \text{ if } \alpha = 0\,, \\ \ell + 1\,, \text{ if } \alpha > 0\,. \end{cases}$$

Theorem 3. *Assume $\pi \xleftarrow{\$} \mathsf{Perm}(\{0,1\}^b)$ is an ideal permutation. We have*

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{AITO}^{\pi,\ell,n}}(Q, T) \leq \frac{(\ell_a Q)^2}{2^{b-n}} + \frac{\ell_a Q S}{2^{\kappa}}\,,$$
$$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{AITO}^{\pi,\ell,n}}(Q, R, T) \leq \frac{(\ell_a Q)^2}{2^{b-n}} + \frac{\ell_a Q S}{2^{\kappa}} + \frac{R}{2^{\alpha}}\,.$$

where S is the maximal number of evaluations of π that can be made in time T.

The proof is given in Appendix C.

8 Applications of AITO

In this section we apply the AITO constructions to various scenarios: low-latency messaging, secure mix formatting, and microblogging. Beyond these examples, AITO is in fact suitable for any application requiring a compact output with high efficiency and security benefits.

8.1 Secret Messaging

The popularity of secret message services has been increasing since Edward Snowden's whistleblowing events. Secret message protocols such as OTR [18], TextSecure[4], and Wickr[5] have been deployed into widely used mobile application. Most use common symmetric encryption schemes under an authenticated mode of operation, such as AES-CCM. In addition, for the purpose of forward secrecy, each scheme requires a re-keying protocol for providing a different session key for each communication. The usage of AITO in such protocols allows for higher security and efficiency (for short messages) and provides the possibility to use these protocols through common text messaging channels, which are limited to 140 characters (1120 bits) per message.

One recent proposal that would be especially suitable to be instantiated with AITO is Vuvuzela [46]. Compared with its closest alternatives (e.g., Tor), Vuvuzela achieves better anonymity and performance, but its proposed implementation is geared towards short messages (240 bytes of content). A combination of AITO with Vuvuzela would allow for achieving higher security of the message contents together with the extreme anonymity of messaging. This would mean slightly shorter message content.

8.2 Secure Mix Formatting

8.3 Microblogging

The text input size limitation on microblogging systems, such as Twitter or Instagram, makes AITO a particularly suitable solution for sharing secret messages with a high level of security (significantly higher than ordinary encryption modes). Although the sharing of secret messages on platforms like these sounds illogical at first sense, the incorporation of AITO is useful for sharing private messages among small segregated audiences.

The high level idea of applying AITO to microblogging is as follows:

- The time stamp of a tweet functions as the nonce *nonce*;
- The hashtags (or part of them) function as a public meta data *meta*;
- The tweet itself (and some hashtags, if needed) is authenticated and encrypted.

[4] Cf., `https://whispersystems.org`.
[5] Cf., `https://wickr.com`.

We denote the time stamp by ts and the hashtags by ht. Due to Twitter's limitation of 140 characters, we take $\lambda_{\max} = 1120$ bits, but we use the fact that the time stamp is implicit in the tweet. Thus, we require that $\nu + \alpha + \tau \leq 1120$. We remark that the Unix time stamp is conventionally written in 32 bits. This makes all three AITO constructions suitable for the application.

Note that our specific application of AITO to microblogging, allows the possibility of users to selectively decide on the information to share with providers without affecting their privacy, a term recently defined as the *"gist"* of information shared [16]. Based on this gist, the receiver could choose not to decrypt messages with a for him/her unappealing gist, while providers could enforce data utility and generate revenues.

Without loss of generality, we consider users to be registered, use, and share private information on any Twitter-like system. We also assume that users share a symmetric key using auxiliary out-of-band channels. For example, the users could employ the techniques from [28] to share the necessary keys with the intended recipients. This would also provide some anonymity for the audience as stated in [28]. In the most simple case, the users can get the keys by performing an authenticated Diffie-Hellman or by simply encrypting the secret with the public keys of the intended recipients. However, the exact details for this key exchange are beyond the scope of AITO mechanism and this paper.

For a specific instantiation, we suggest the framework of Figure 4. AITO operates by parsing a 1120-bit tweet into a message/hashtag-tuple $(msg, ht) \in \{0,1\}^{\leq \mu} \times \{0,1\}^{\leq \tau}$. Using the time stamp $ts \in \{0,1\}^{\sigma}$, it is then encrypted to a ciphertext $ctxt \in \{0,1\}^{\leq \nu}$ and authenticated using tag $auth \in \{0,1\}^{\alpha}$ (absent if no authentication is needed, in which case we have $\alpha = 0$). Thus, the encrypted tweet is of the form $(ctxt, auth, ht) \in \{0,1\}^{\leq \nu} \times \{0,1\}^{\alpha} \times \{0,1\}^{\leq \tau}$ (the time stamp ts is implicit from the tweet). In this configuration, nonce respecting behavior of users corresponds to them being "time stamp respecting," and we assume that two tweets under the same secret key are never encrypted with the same time stamp.

9 Conclusion

This paper presented AITO, a novel family of compact and highly secure authenticated encryption schemes that are based on either tweakable blockciphers of Sponge functions. We have shown that AITO can be deployed in various privacy-oriented applications that aim at short messages, including direct messaging protocols such as OTR and Vuvuzela, the Sphinx mix network, and online social networks and microblogging services such as Twitter and Instagram. Due to its specific design goals, AITO performs better than its competitors on applications like these. Applications beyond the ones of Section 8 likely exist. In any case, our work demonstrates that in certain scenarios, an application-oriented cryptographic design may be more suitable than the generic solutions.

We remark that, although the tweakable blockcipher-based versions of AITO offer resistance against nonce reuse (with a slight security degradation), this is

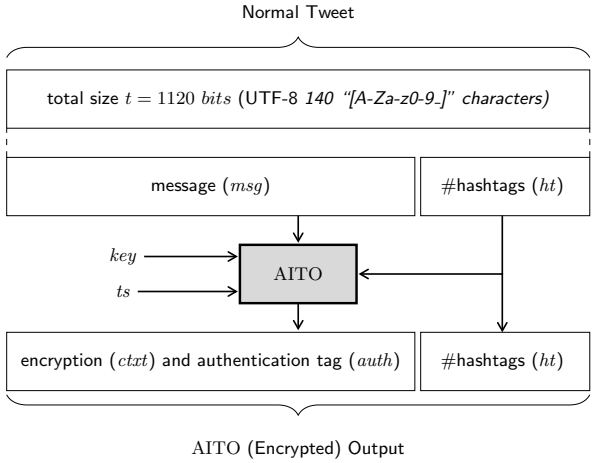

Fig. 4. Using AITO for microblogging to share private messages, while allowing filtering and providing limited data utility through the hashtags.

not the case for the Sponge based version. This may be problematic in certain scenarios. For instance, in our application to microblogging (Section 8.3), this would translate to time stamp respecting adversaries. In the case of protection against time stamp misuse, for example by the OSN itself, the AITO can accommodate a client generated time stamp. This would add some overhead (of 32 bits), but could protect against this type of attack.

References

1. Anderson, R.J., Biham, E.: Two practical and provably secure block ciphers: BEARS and LION. In: FSE '96. LNCS, vol. 1039, pp. 113–120. Springer (1996)
2. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mendel, F., Mennink, B., Mouha, N., Wang, Q., Yasuda, K.: PRIMATEs v1.02 (2015), submission to CAESAR competition
3. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: authenticated permutation-based encryption for lightweight cryptography. In: FSE 2014. LNCS, vol. 8540, pp. 168–186. Springer (2015)
4. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to securely release unverified plaintext in authenticated encryption. In: ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 105–125. Springer (2014)
5. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and authenticated online ciphers. In: ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 424–443. Springer (2015)
6. Andreeva, E., Daemen, J., Mennink, B., Van Assche, G.: Security of keyed Sponge constructions using a modular proof approach. In: FSE 2015. LNCS, vol. 9054, pp. 364–384. Springer (2015)
7. Aumasson, J., Jovanovic, P., Neves, S.: NORX v2.0 (2015), submission to CAESAR competition

8. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS 1997. pp. 394–403. IEEE Computer Society (1997)
9. Bellare, M., Goldreich, O., Mityagin, A.: The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive, Report 2004/309 (2004)
10. Bellare, M., Rogaway, P., Wagner, D.: The EAX mode of operation. In: FSE 2004. LNCS, vol. 3017, pp. 389–407. Springer (2004)
11. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the security of the keyed Sponge construction. Symmetric Key Encryption Workshop (SKEW 2011) (2011)
12. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the Sponge: Single-pass authenticated encryption and other applications. In: SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer (2012)
13. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak implementation overview (May 2012)
14. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions (ECRYPT Hash Function Workshop 2007)
15. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: Keyak v2 (2015), submission to CAESAR competition
16. Bilogrevic, I., Freudiger, J., Cristofaro, E.D., Uzun, E.: What's the gist? privacy-preserving aggregation of user profiles. In: ESORICS 2014. LNCS, vol. 8713, pp. 128–145. Springer (2014)
17. Biryukov, A., Khovratovich, D.: PAEQ: parallelizable permutation-based authenticated encryption. In: ISC 2014. LNCS, vol. 8783, pp. 72–89. Springer (2014)
18. Borisov, N., Goldberg, I., Brewer, E.A.: Off-the-record communication, or, why not to use PGP. In: WPES 2004. pp. 77–84. ACM (2004)
19. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness (March 2014)
20. Danezis, G., Goldberg, I.: Sphinx: A compact and provably secure mix format. In: IEEE S 2009. pp. 269–282. IEEE Computer Society (2009)
21. Datta, N., Nandi, M.: ELmE: A misuse resistant parallel authenticated encryption. In: ACISP 2014. LNCS, vol. 8544, pp. 306–321. Springer (2014)
22. De Cannière, C., Rechberger, C.: Finding SHA-1 characteristics: General results and applications. In: ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer (2006)
23. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.1 (2015), submission to CAESAR competition
24. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein hash function family (2010), submission to NIST's SHA-3 competition
25. Fluhrer, S.R.: Cryptanalysis of the SEAL 3.0 pseudorandom function family. In: FSE 2001. LNCS, vol. 2355, pp. 135–143. Springer (2002)
26. Fuhr, T., Leurent, G., Suder, V.: Collision attacks against CAESAR candidates - forgery and key-recovery against AEZ and marble. In: ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 510–532. Springer (2015)
27. Gligoroski, D., Mihajloska, H., Samardjiska, S., Jacobsen, H., El-Hadedy, M., Jensen, R.: π-Cipher v2.0 (2015), submission to CAESAR competition
28. Günther, F., Manulis, M., Strufe, T.: Cryptographic treatment of private user profiles. In: Financial Cryptography and Data Security, pp. 40–54. Springer (2012)
29. Hoang, V.T., Krovetz, T., Rogaway, P.: AEZ v4: Authenticated Encryption by Enciphering (2015), submission to CAESAR competition

30. Hoang, V., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 15–44. Springer (2015)
31. Jovanovic, P., Luykx, A., Mennink, B.: Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes. In: ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 85–104. Springer (2014)
32. Karpman, P., Peyrin, T., Stevens, M.: Practical free-start collision attacks on 76-step SHA-1. In: CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 623–642. Springer (2015)
33. Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: FSE 2011. LNCS, vol. 6733, pp. 306–327. Springer (2011)
34. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer (2002)
35. Mennink, B.: Observation on weak keys for AEZ (2016), CAESAR mailing list
36. Mennink, B., Reyhanitabar, R., Vizár, D.: Security of full-state keyed and duplex sponge: Applications to authenticated encryption. In: ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 465–489. Springer (2015)
37. Minematsu, K.: Parallelizable rate-1 authenticated encryption from pseudorandom functions. In: EUROCRYPT 2014. LNCS, vol. 8441, pp. 275–292. Springer (2014)
38. Morawiecki, P., Gaj, K., Homsirikamol, E., Matusiewicz, K., Pieprzyk, J., Rogawski, M., Srebrny, M., Wójcik, M.: ICEPOLE v2 (2015), submission to CAESAR competition
39. Naito, Y., Yasuda, K.: New bounds for keyed Sponges with extendable output: Independence between capacity and message length. In: FSE 2016. LNCS, Springer (2016), to appear
40. Peter Gaži, Pietrzak, K., Tessaro, S.: The exact PRF security of truncation: Tight bounds for keyed sponges and truncated CBC. In: CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 368–387. Springer (2015)
41. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer (2004)
42. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: a block-cipher mode of operation for efficient authenticated encryption. In: ACM Conference on Computer and Communications Security. pp. 196–205. ACM (2001)
43. Saarinen, M., Brumley, B.: STRIBOB v2 (2015), submission to CAESAR competition
44. Stevens, M.: New collision attacks on SHA-1 based on optimal joint local-collision analysis. In: EUROCRYPT 2013. LNCS, vol. 7881, pp. 245–261. Springer (2013)
45. Stevens, M., Karpman, P., Peyrin, T.: Freestart collision on full SHA-1. Cryptology ePrint Archive, Report 2015/967 (2015)
46. Van Den Hooff, J., Lazar, D., Zaharia, M., Zeldovich, N.: Vuvuzela: Scalable private messaging resistant to traffic analysis. In: Proceedings of the 25th Symposium on Operating Systems Principles. pp. 137–152. ACM (2015)
47. Wang, X., Yin, Y., Yu, H.: Finding collisions in the full SHA-1. In: CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer (2005)

A Proof of Theorem 1

We start with the secrecy of AITO$^{\text{3fish}}$. Let $\mathcal{A}$ be an adversary that makes Q queries and runs in time T. It has access to either $\texttt{Enc}_{key}$ or \$. Note that Q evaluations of AITO$^{\text{3fish}}$ induce Q evaluations of 3fish. We replace 3fish by an ideal tweakable permutation $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\{0,1\}^{128}, \{0,1\}^{1024})$. Now, any query $\texttt{Enc}^{\widetilde{\pi}}(key, msg, meta, nonce)$ is responded with

$$ctxt \| auth = \widetilde{\pi}(key, \mathsf{pad}_{128}(nonce \| meta), \mathsf{pad}_{1024}(msg)) \,.$$

As $\mathcal{A}$ is required to be nonce respecting, every query is made under a new nonce, which means that every query initiates a new instance of $\widetilde{\pi}$, and $ctxt \| auth$ is a random 1024-bit value. This means that $\texttt{Enc}^{\widetilde{\pi}}_{key}$ is perfectly indistinguishable from \$.

For authenticity, the first part of the proof is identical: we replace 3fish by ideal tweakable permutation $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\{0,1\}^{128}, \{0,1\}^{1024})$, where now the $Q + R$ evaluations of AITO$^{\text{3fish}}$ induce $Q + R$ evaluations of 3fish. It remains to consider the probability to forge an authentication tag for AITO$^{\widetilde{\pi}}$. By [9], it suffices to consider any attempt and sum over all R attempts. Consider any forgery attempt $(ctxt, auth, meta, nonce)$. Note that, as $\mathcal{A}$ is required to be nonce respecting, there has been at most one encryption query under meta data $meta$ and nonce $nonce$. Therefore, the value

$$m = \widetilde{\pi}^{-1}(key, \mathsf{pad}_{128}(nonce \| meta), ctxt \| auth)$$

is randomly drawn from a set of size at least $2^n - 1$, and satisfies $\lfloor m \rfloor_\alpha = 0$ with probability at most $2^{n-\alpha}/(2^n - 1)$.

B Proof of Theorem 2

Note that the derivation in Theorem 1 not only applies to 3fish, but to any tweakable blockcipher. Applied to LRW[3fish] we get

$$\begin{aligned}
\mathbf{Adv}^{\text{cpa}}_{\text{AITO}^{\text{LRW[3fish]}}}(Q,T) &\leq \mathbf{Adv}^{\widetilde{\text{sprp}}}_{\text{LRW[3fish]}}(Q,T'') \,, \\
\mathbf{Adv}^{\text{auth}}_{\text{AITO}^{\text{LRW[3fish]}}}(Q,R,T) &\leq \mathbf{Adv}^{\widetilde{\text{sprp}}}_{\text{LRW[3fish]}}(Q+R,T'') + \frac{R 2^{n-\alpha}}{2^n - 1} \,,
\end{aligned}$$

where $T'' \approx T$. In [34] it is proven that

$$\mathbf{Adv}^{\widetilde{\text{sprp}}}_{\text{LRW[3fish]}}(Q,T'') \leq \Theta\left(\frac{Q^2}{2^n}\right) + \mathbf{Adv}^{\widetilde{\text{sprp}}}_{\text{3fish}}(2Q,T') \,,$$

where $T' \approx T''$.

C Proof of Theorem 3

Let $\mathcal{A}$ be an adversary that makes Q queries and runs in time T. It has access to either $\mathtt{Enc}_{key}$ or \$. Consider any evaluation $\mathtt{Enc}_{key}$ on input of $(msg, meta, nonce)$. If we define $k = 0^{b-n-|key|}\|key$, then its output is as follows,

$$\begin{aligned}
ctxt = {} & \mathrm{IKS}^{\pi}(k, \mathsf{pad}_n(nonce\|meta), n) \oplus msg_1 \,\| \\
& \mathrm{IKS}^{\pi}(k, \mathsf{pad}_n(nonce\|meta)\|msg_1, n) \oplus msg_2 \,\| \\
& \cdots \\
& \mathrm{IKS}^{\pi}(k, \mathsf{pad}_n(nonce\|meta)\|msg_1 \cdots msg_{\ell-2}, n) \oplus msg_{\ell-1} \,\| \\
& \lceil \mathrm{IKS}^{\pi}(k, \mathsf{pad}_n(nonce\|meta)\|msg_1 \cdots msg_{\ell-1}, n) \oplus msg_{\ell} \rceil_{|msg| \bmod n}\,, \\
auth = {} & \mathrm{IKS}^{\pi}(k, \mathsf{pad}_n(nonce\|meta)\|msg_1 \cdots msg_{\ell}, \alpha)
\end{aligned}$$

Where abusing notation, $|msg| \bmod n \in \{1, \ldots, n\}$. In other words, any evaluation of $\mathtt{Enc}_{key}$ entails ℓ_α evaluations of IKS (the computation of *auth* is omitted if $\alpha = 0$). Each of these evaluations adds 1 to the complexity (as it is simply an extension of the previous one). Thus, after Q evaluations of $\mathtt{Enc}_{key}$, IKS is evaluated with a total complexity $\ell_a Q$. We replace IKS by a random oracle $\mathcal{RO}$. This step costs us $\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{IKS}}(\ell_a Q, S)$, where S is as described in the theorem statement.

Now, for the case of secrecy, recall that $\mathcal{A}$ is nonce respecting. Consequently, all evaluations of $\mathcal{RO}$ are made for a different input. This is clear for the $\ell + 1$ queries for a single evaluation; different evaluations of $\mathtt{Enc}_{key}$ are made under a different *nonce* as the adversary is nonce respecting. Consequently, every query to $\mathtt{Enc}_{key}$ is responded with a uniformly randomly generated $|msg|$-bit value, and thus,

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{AITO}^{\pi,\ell,n}}(Q, T) \leq \mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{IKS}}(\ell_a Q, S)\,.$$

Next, for authenticity, it suffices to only focus on the value *auth*. Consider any forgery attempt $(ctxt, auth, meta, nonce)$. Let *msg* be the message that is derived by $\mathtt{Dec}^{\pi,\ell,n}$. As the forgery is required to be non-trivial, $\mathcal{RO}$ has never been queries on

$$\mathsf{pad}_n(nonce\|meta)\|msg_1 \cdots msg_{\ell}$$

before. Its response *auth* is thus a randomly generated value and the forgery is successful with probability $1/2^{\alpha}$. Again using [9],

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{AITO}^{\pi,\ell,n}}(Q, R, T) \leq \mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{IKS}}(\ell_a Q, S) + \frac{R}{2^{\alpha}}\,.$$

Now, in [6] it is proven that[6]

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{IKS}}(Q', S) \leq \frac{(Q')^2}{2^{b-n}} + \frac{Q'S}{2^{\kappa}}\,,$$

which completes the proof of both secrecy and authenticity.

[6] We have slightly re-interpreted the result in order to accommodate the different key length.