IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 4, pp. 135–162. DOI:10.46586/tches.v2022.i4.135-162

Triplex: an Efficient and One-Pass Leakage-Resistant Mode of Operation

Yaobin Shen[1], Thomas Peters[1], François-Xavier Standaert[1], Gaëtan Cassiers[1] and Corentin Verhamme[1]

[1] UCLouvain, ICTEAM, Crypto Group, Louvain-la-Neuve, Belgium
firstname.lastname@uclouvain.be

Abstract. This paper introduces and analyzes Triplex, a leakage-resistant mode of operation based on Tweakable Block Ciphers (TBCs) with $2n$-bit tweaks. Triplex enjoys beyond-birthday ciphertext integrity in the presence of encryption and decryption leakage in a liberal model where all intermediate computations are leaked in full and only two TBC calls operating a long-term secret are protected with implementation-level countermeasures. It provides beyond-birthday confidentiality guarantees without leakage, and standard confidentiality guarantees with leakage for a single-pass mode embedding a re-keying process for the bulk of its computations (i.e., birthday confidentiality with encryption leakage under a bounded leakage assumption). Triplex improves leakage-resistant modes of operation relying on TBCs with n-bit tweaks when instantiated with large-tweak TBCs like Deoxys-TBC (a CAESAR competition laureate) or Skinny (used by the Romulus finalist of the NIST lightweight crypto competition). Its security guarantees are maintained in the multi-user setting.

Keywords: Leakage-Resistance · Authenticated Encryption · Single-Pass Modes

1 Introduction

Protecting cryptographic implementations against side-channel attacks is a tedious process and generally leads to significant performance overheads [MOP07]. Research to design modes of operation with good properties against leakage has therefore gained interest over the last decade. One popular approach for this purpose is to leverage so-called "leveled implementations", in which security is obtained by combining the minimum use of a highly protected component while the rest of the computations only requires frugal protections [PSV15]. As surveyed by Bellizia et al., this approach has led to different authenticated encryption schemes, which can be viewed as different tradeoffs between mode-level and implementation-level protection mechanisms [BBC+20].

Starting with the top of the hierarchy established in [GPPS19], ISAP [DEM+17, DEM+20] and TEDT [BGP+20] are two-pass modes of operation that guarantee both Ciphertext Integrity with Misuse-resistance and Leakage in encryption and decryption (CIML2) and CCA security with misuse-resilience and Leakage in encryption and decryption (CCAmL2). Relaxing the confidentiality with decryption leakage requirement, Ascon [DEMS21] and Spook [BBB+20] are one-pass algorithms that guarantee CIML2 and CCAmL1, and follow the blueprint of the TETSponge mode of operation [GPPS20]. Such leakage-resistant designs have also been recognized as relevant for lightweight cryptography: ISAP and Ascon are both finalists in the ongoing NIST competition[1], one of the modes

[1] https://csrc.nist.gov/Projects/lightweight-cryptography

Licensed under Creative Commons License CC-BY 4.0.
Received: 2022-04-15 Accepted: 2022-06-15 Published: 2022-08-31

proposed by the Romulus finalist relies on an adaptation of TEDT (that takes advantage of its $2n$-bit tweaks) [IKMP20] and Spook was a Round-2 candidate.

Looking at their internal components, ISAP, Ascon and Spook are sponge-based designs, building on the good leakage properties of the Duplex construction [DM19]. By contrast, only TEDT is based on a Tweakable Block Cipher (TBC). As for one-pass modes relying on TBCs, the only existing options are the TET scheme given in the appendices of [BGP+20], which aims at CIML2 and CCAmL1 guarantees like Ascon and Spook, and the AET-LR scheme in [GKP], which only targets CIML2 (ignoring confidentiality with leakage).

In this paper, we focus on the challenge of designing a more efficient leakage-resistant mode of operation that provides similar guarantees as Ascon and Spook, but is based on TBCs. For this purpose, we follow the recent trend of leveraging TBCs with large tweaks, which it is the case for Romulus, but also for Deoxys-TBC-384 [JNPS21]. As discussed by List [Lis21], large-tweak TBCs are handy to improve the bounds and efficiency of TEDT (while also simplifying the analysis). We therefore leverage such a primitive to design a new mode of operation, coined Triplex, that additionally improves the rate of the TET construction. Namely, while TET requires two TBC calls (with n-bit tweaks) to encrypt and authenticate n bits of message, Triplex can encrypt and authenticate $2n$ bits with three TBC calls (with $2n$-bit tweaks).[2] Concretely, Triplex enjoys:

- $n - \log_2(n)$ bits of confidentiality without leakage in the nonce misuse-resilient setting [ADL17]. That is, the confidentiality of messages under unique nonces holds as long as the total query complexity of the adversary does not exceed $2^n/n$, even when other messages are compromised due to nonce misuse.

- $n - \log_2(n)$ bits of CIML2 in the unbounded leakage model. That is, the integrity guarantees hold as long as the query complexity of the adversary does not exceed $2^n/n$, even with nonce misuse and full leakage of the unprotected components.

Furthermore, these security guarantees do not vanish in the multi-user setting: Triplex still provides $n - \log_2(n)$ bits of confidentiality and integrity in this context.

Besides its excellent features for leakage-resistance, we believe Triplex is also an interesting candidate to feed the comparison between Sponge-based and TBC-based designs in general [Pey20]. In particular, its improved rate makes it similar to TETSponge, Ascon & Spook regarding this metric. We initiate a comparative discussion of these ciphers by analyzing prototype hardware implementations in the last section of the paper.

Cautionary note. The confidentiality and the integrity of Triplex are analyzed in the ideal TBC model. This is a common trait of most formal analyzes in the multi-user regime. In this setting, we are concerned with how local computation (captured by the number of ideal TBC queries) affects security. The classical assumption on TBC (i.e., TPRP security) is not helpful for this estimation and will induce a security loss due to the black-box replacement. Thus, the bounds exclude generic attacks with local computations that just call the primitive. Besides, in the unbounded leakage model (for CIML2), the internal values (including ephemeral keys) are leaked and the standard model becomes vacuous.

Related works. With the goal to take advantage of efficient AES co-processors that are frequently available in embedded devices, the retrofitting mode of [USS+20] and the LR-BC mode of [BMPS21] only rely on n-bit block cipher calls. These modes are in general less efficient than TBC-based modes due to this additional constraint, but they can lead to excellent performance in practice when these co-processors are indeed available.

[2] TBCs with larger tweaks require slightly more rounds, but current values suggest that the tradeoff is positive. For example, Triplex instantiated with Deoxys-TBC-384 (which has 16 rounds) would require less rounds per message blocks than TET instantiated with Deoxys-TBC-256 (which has 14 rounds).

Structure of the paper. After providing the necessary background in Section 2, we give the high-level ideas behind Triplex and its full specification in Section 3. Sections 4 and 5 provide the confidentiality analysis without leakage and the integrity analysis with leakage of Triplex. In the confidentiality section, we also provide a short discussion of its confidentiality with leakage, which we do not detail due to place constraints and its strong similarity with the analysis of other leakage-resistant modes leveraging re-keying. We end the paper with some hardware implementation results in Section 6.

2 Preliminaries

Notation. Let ε denote the empty string. Let $\{0,1\}^*$ be the set of all finite bit strings including the empty string ε. For a finite set S, let $x \xleftarrow{\$} S$ denote the uniform sampling from S assigning a value to x. Let $|x|$ denote the length of the string x. Let $x[i:j]$ denote the substring from the i-th bit to the j-th bit (inclusive) of x. Concatenation of strings x and y is written as $x \,\|\, y$ or simply xy. If A is an algorithm, let $y \leftarrow A(x_1, \ldots; r)$ denote running A with randomness r on inputs $x_1, \ldots$ and assigning the output to y. Let $y \xleftarrow{\$} A(x_1, \ldots)$ be the result of picking r at random and letting $y \leftarrow A(x_1, \ldots; r)$. Let $\mathrm{Perm}(n)$ denote the set of all permutations over $\{0,1\}^n$, and let finally $\mathrm{Func}(*, n)$ denote the set of all functions from $\{0,1\}^*$ to $\{0,1\}^n$.

Tweakable block cipher [LRW11]. A block cipher $E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ is a family of permutations, where $E_K(\cdot) = E(K, \cdot)$ is a permutation over $\mathcal{M}$. A tweakable block cipher $E : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ (with a slight abuse of notation E) is a family of permutations over $\mathcal{M}$, indexed by two functionally distinct parameters: a key $K \in \mathcal{K}$ that is secret and used to provide the security, and a tweak $t \in \mathcal{T}$ that is public and used to provide variability. We write $E_K(t, \cdot) = E(K, t, \cdot)$, a permutation over $\mathcal{M}$.

Nonce-based authenticated encryption [Rog02]. An AE scheme Π is a triplet of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, where $\mathcal{K}$ is the key-generation algorithm, $\mathcal{E}$ the encryption algorithm and $\mathcal{D}$ the decryption algorithm. The key-generation algorithm $\mathcal{K}$ samples a key K uniformly at random from the key space. The encryption algorithm $\mathcal{E}$ takes as input a key K, a nonce N, an Associated Data (AD) A, a message M, and returns a ciphertext and tag $C \,\|\, \mathsf{tag} \leftarrow \mathcal{E}_K(N, A, M)$. The decryption algorithm $\mathcal{D}$ takes as input a key K, a nonce N, an AD A, a ciphertext and tag $C \,\|\, \mathsf{tag}$, and returns either a message M or a symbol $\perp$ indicating invalidity. For correctness, we assume that if $C \,\|\, \mathsf{tag} \leftarrow \mathcal{E}_K(N, A, M)$ then $M \leftarrow \mathcal{D}_K(N, A, C \,\|\, \mathsf{tag})$. In this paper, the tag is always of fixed length.

Privacy security. We define the privacy security with respect to nonce-misuse resilience as introduced in [ADL17]. The privacy security game $\mathsf{G}^{\mathrm{priv}}_{\Pi}$ is detailed in Figure 1. We consider the security in the multi-user setting. For queries to the same user, the adversary may repeat the nonce in the first encryption oracle Enc_1, but the nonce in the second encryption oracle should be unique and fresh. For queries to different users, the adversary may repeat the nonce in both oracles. With access to oracles Prim, Enc_1 and Enc_2, the goal of the adversary is to distinguish the second encryption oracle of an AE scheme from a random function. Formally, given an adversary $\mathcal{A}$, we define

$$\mathsf{Adv}^{\mathrm{priv}}_{\Pi}(\mathcal{A}) = 2\Pr\left[\mathsf{G}^{\mathrm{priv}}_{\Pi}(\mathcal{A})\right] - 1$$

as the advantage of the adversary against the privacy security of an AE scheme Π in the nonce misuse-resilience setting, with $\mathsf{G}^{\mathrm{priv}}_{\Pi}(\mathcal{A})$ the abbreviation of $\mathsf{G}^{\mathrm{priv}}_{\Pi}(\mathcal{A}) = \mathsf{true}$.

Authenticity security. We consider the authenticity security in the leakage setting, and follow the notion of Ciphertext Integrity with Misuse-resistance and encryption and

Game $\mathsf{G}^{\mathrm{priv}}_{\Pi}(\mathcal{A})$

$K_1, K_2, \ldots, \xleftarrow{\$} \mathcal{K}$; $b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Prim},\mathsf{Enc}_1,\mathsf{Enc}_2}$; **return** $(b' = b)$

procedure $\mathsf{Prim}(J, T, X)$
if $X = (+, x)$ **then return** $E_J(T, x)$
if $X = (-, y)$ **then return** $E_J^{-1}(T, y)$

procedure $\mathsf{Enc}_1(i, N, A, M)$
$C \,\|\, \mathsf{tag} \leftarrow \mathcal{E}(K_i, N, A, M)$
return $C \,\|\, \mathsf{tag}$

procedure $\mathsf{Enc}_2(i, N, A, M)$
$C_1 \,\|\, \mathsf{tag}_1 \leftarrow \mathcal{E}(K_i, N, A, M)$
$C_0 \,\|\, \mathsf{tag}_0 \xleftarrow{\$} \{0,1\}^{|C_1|+|\mathsf{tag}|}$
return $C_b \,\|\, \mathsf{tag}_b$

Figure 1: Game $\mathsf{G}^{\mathrm{priv}}_{\Pi}$: multi-user privacy security of an AE Π with nonce-misuse resilience.

Game $\mathsf{G}^{\mathrm{CIML2}}_{\Pi}(\mathcal{A})$

$K_1, K_2, \ldots, \xleftarrow{\$} \mathcal{K}$; $b \xleftarrow{\$} \{0,1\}$
$\mathcal{Q} \leftarrow \emptyset$; $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Prim},\mathsf{Enc},\mathsf{Dec}}$
return $(b' = b)$

procedure $\mathsf{Enc}(i, N, A, M)$
$L_e \leftarrow \mathcal{L}_E(K_i, N, A, M)$
$C \,\|\, \mathsf{tag} \leftarrow \mathcal{E}(K_i, N, A, M)$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(K_i, N, A, C \,\|\, \mathsf{tag})\}$
return $(C \,\|\, \mathsf{tag}, L_e)$

procedure $\mathsf{Dec}(i, N, A, C \,\|\, \mathsf{tag})$
$L_d \leftarrow \mathcal{L}_D(K_i, N, A, C \,\|\, \mathsf{tag})$
$M \leftarrow \mathcal{D}(K_i, N, A, C \,\|\, \mathsf{tag})$
if $(K_i, N, A, C \,\|\, \mathsf{tag}) \in \mathcal{Q}$ **then**
  return (M, L_d)
if $b = 0$ **then return** $(\perp, L_d)$
else return (M, L_d)

procedure $\mathsf{Prim}(J, T, X)$
if $X = (+, x)$ **then return** $E_J(T, x)$
if $X = (-, y)$ **then return** $E_J^{-1}(T, y)$

Figure 2: Game $\mathsf{G}^{\mathrm{CIML2}}_{\Pi}$: multi-user CIML2 security of an AE Π with nonce misuse-resistance.

decryption Leakage (CIML2) by Berti et al. [BPPS17]. In the leakage environment, the adversary not only has access to encryption oracle $\mathcal{E}$ and decryption oracle $\mathcal{D}$, but also to their corresponding leakage functions $\mathcal{L}_E$ and $\mathcal{L}_D$. Here we consider it in the multi-user setting. Given an adversary $\mathcal{A}$, we define

$$\mathsf{Adv}^{\mathrm{CIML2}}_{\Pi}(\mathcal{A}) = 2\Pr\left[\,\mathsf{G}^{\mathrm{CIML2}}_{\Pi}(\mathcal{A})\,\right] - 1$$

as the advantage of the adversary against the CIML2 security of an AE scheme Π, where game $\mathsf{G}^{\mathrm{CIML2}}_{\Pi}$ is illustrated in Fig. 2 and $\mathsf{G}^{\mathrm{CIML2}}_{\Pi}(\mathcal{A})$ is the abbreviation that $\mathsf{G}^{\mathrm{CIML2}}_{\Pi}(\mathcal{A}) = \mathsf{true}$. The adversary is given encryption and decryption oracles, which both contain the corresponding leakage function. She can repeat nonces in encryption and decryption queries. She may also make a decryption query $(i, N, A, C \,\|\, \mathsf{tag})$ even if $(i, N, A, C \,\|\, \mathsf{tag})$ has appeared in previous encryption queries. This kind of decryption query lets her obtain additional leakage during decryption. The goal of the adversary is to output a valid and new tuple $(i, N, A, C \,\|\, \mathsf{tag})$ that passes the decryption oracle of the real AE scheme, while in the ideal world she will always receive a rejection symbol $\perp$.

Hirose's Compression Function [Hir06]. Triplex makes use of Hirose's compression function (based on TBCs) to handle the message and associated data. We next recall the definition of this compression function. It is also represented in Figure 3.

Definition 1. Let $\mathsf{Hir} : \{0,1\}^{2n} \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ be a compression function such that $(h_i, k_i) = \mathsf{Hir}(h_{i-1}, k_{i-1}, m_i)$ where $h_{i-1}, k_{i-1}, h_i, k_i \in \{0,1\}^n$ and $m_i \in \{0,1\}^{2n}$. Hir is built from a tweakable blockcipher $E : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ as follows:

$$\begin{cases} h_i = E_{k_{i-1}}(m_i, h_{i-1}) \oplus h_{i-1} \\ k_i = E_{k_{i-1}}(m_i, h_{i-1} \oplus \theta_1) \oplus h_{i-1} \oplus \theta_1, \end{cases}$$

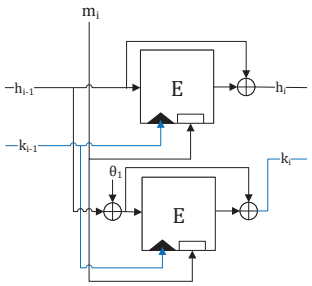


Figure 3: Hirose's compression function that is built on top of a tweakable block cipher E.

where θ_1 is a non-zero constant.

Some useful Lemmas. We introduce some lemmas that will be useful in our analyzes.

Lemma 1. *[HT17, Lemma 15] Suppose that we throw u balls uniformly at random into 2^n bins. Then, with probability at most 2^{-n}, there exists some bin of more than $\max\{4n, 4u/2^n\}$ balls.*

Lemma 2. *[BHT18, Lemma 10] Suppose that we throw u balls into 2^n bins and, conditioning on the result of prior throws, the probability that each ball falls into any particular bin is at most 2^{n-1}. Fix $0 < \epsilon < 1$, and let $u \leq 2^{(1-\epsilon)n-1}$. Then with probabilty at most $2^{-n/2}$, there exists some bin of more than $\lceil 1.5/\epsilon \rceil$ balls.*

The collision probability of Hirose's Double-block-length hash function based on a TBC is formalized by the following lemma.

Lemma 3. *Let H be a hash function composed of the compression function* Hir *specified in Definition 1. Then, for any adversary $\mathcal{A}$ that makes at most p queries to the ideal TBC E, we have*

$$\Pr[\, A \text{ finds a collision on } H \,] \leq \frac{8p^2}{2^{2n}} \ .$$

The proof of this lemma is similar to the one of [Hir06, Theorem 4]. For the sake of completeness, we present it in Appendix A.

3 Specifications of Triplex

In this section, we give the full details of Triplex. Triplex is a one-pass AE mode based on TBCs with large tweaks. To achieve so-called Grade-2 leakage security (i.e., a combination of CCAmL1 for confidentiality and CIML2 for integrity), we combine an ephemeral key evolution process based on a compression function as in [BMPS21] with strengthened Key Derivation Function (KDF) and Tag Generation Function (TGF). On the one hand, the ephemeral key evolution allows iteratively processing each block of message with a fresh key, which is reminiscent of other designs conferring confidentiality guarantees in the presence of encryption leakage [BBC+20]. On the other hand, the compression function is used to progressively absorb the blocks to make the computation more and more dependent on the already processed blocks, leading to a kind of digest that can then be authenticated using a fixed-length Leakage-Resilient Message Authentication Code (LR-MAC).

In the following, we explain the intuition behind the design of Triplex and discuss the difference with sponge-based designs before providing the full specifications. The security analysis is postponed to the next sections.

3.1 Design Blueprint

Triplex follows the general 3-step blueprint suggested in [BBC+20] for Grade-2 designs. In the *initialization* step, we generate a random $2n$-bit state (h_1, k_1) from a key derivation function $\mathsf{KDF}_K(P, N)$, where k_1 is an ephemeral encryption key. P is a random key that can be public. It confers more security in the multi-user setting as in Spook [BBB+20] and TEDT [BGP+20], since in order to implement key-collision attack, the adversary needs to find a collision for both secret key K and public key P among many users. As detailed later, the KDF requires only a single call to the protected TBC. Next, the *bulk* of the computation is instantiated as a one-time encryption of $M_1\| \cdots \|M_\ell$ where, at each successive processing of a message block M_i, the corresponding ciphertext block C_i is created and absorbed, and the state is refreshed. We then also absorb the Associated Data (AD) resulting in a final state (h_f, k_f). Eventually, in the *finalization* step, we use an LR-MAC for the TGF to authenticate (h_f, k_f) and creating a tag using only a second protected TBC call. That means that the linear number of TBC calls in the message (and AD) processing part can remain unprotected, leading to performance benefits.

We now give more details about the intermediate step.

To improve the rate of our AEAD, we start from the Hirose compression function Hir implemented with two calls to the TBC with large tweaks (see Section 2, Figure 3). This allows absorbing $2n$-bit blocks of message per iteration, plugged as the tweak of both TBC calls. By iterating, we get a hash function with $2n$-bit state (h_{i+1}, k_{i+1}), no matter the size of the tweak. Our goal is to turn this hash function into a one-time encryption of the $2n$-bit block of message by making a single additional call to the underlying TBC, thus processing $2n$ bits with only 3 unprotected TBC calls. The reason why a single additional call is enough is because we can already encrypt the first n bits of the block by XORing it with the random half state h_i. We use the other half state k_i as an ephemeral key to encrypt the last n bits of the message block (by XORing it with the output of the additional TBC call). We absorb the resulting $2n$-bit ciphertext block in the compression function from (h_i, k_i). Since the key k_i is used thrice per iteration, we use two constants θ_1, θ_2 to enforce all the plaintext-inputs $h_i, h_i \oplus \theta_1, h_i \oplus \theta_2$ of the TBC to be distinct. This key-input and these 3 plaintext-inputs are thus deterministic in the current state (h_i, k_i), and remain out of the direct control of the adversary.

In Figure 4, we depict the encryption of $M_1\| \dots \|M_4$ with associated data $A_1\|A_2$, where (h_1, k_1) is the initial state. The 3 TBC calls per $2n$-block of message can easily be parsed from the picture. For each processing of message block, the 2 (vertical) TBC calls in black represent Hir. In red, the additional TBC call corresponds to our plug-in that turns Hir into a block encryption. The final state is (h_4, k_4) in this case.

We now picture Triplex's initialization step (which derives the initial state) as well as the finalization step (which generates the tag from the final state) in Figure 5. For the KDF in Figure 5(A), we first use the key K, the public key P and the nonce N to set up an IV (h_0, k_0) for the compression function, where $h_0 = 0^n$ and k_0 is the TBC output. The preimage resistance of the half state h_i-value as the image of a single-length compression function ensures that no internal state can collide with (h_0, k_0) because otherwise $h_i = 0^n$. To avoid initial-state-collisions between encryption and decryption, we simply apply Hir once with the nonce and the public key: we make the unprotected call $\mathsf{Hir}(h_0, k_0, N\|P)$. This very first call to Hir inside KDF forces the initial state (h_1, k_1) to diverge for distinct pairs (N, P) even if some collision on k_0 occurs. For the TGF in Figure 5(B), we borrow the recent LR-MAC due to [BGPS21] which already leverages double-size tweak to get an elegant and simple beyond-birthday authentication mechanism. For checking the validity of the tag in decryption, this LR-MAC relies on the invertibility of the TBC in order to avoid leaking any information on the right tag given any adversarially chosen invalid

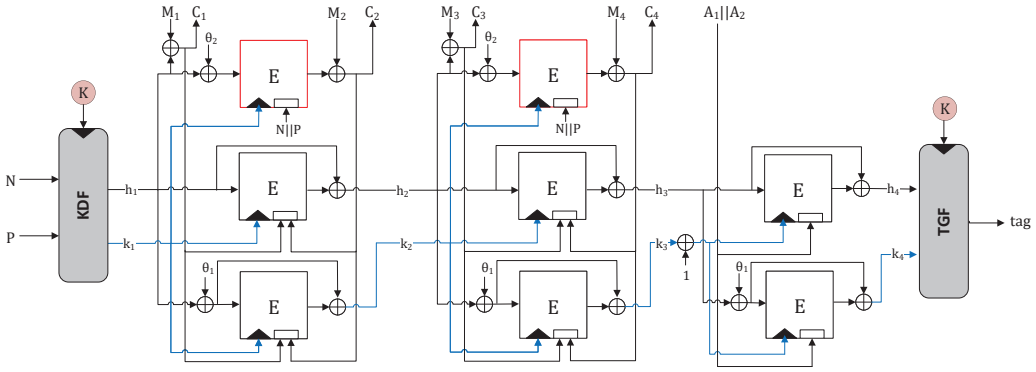
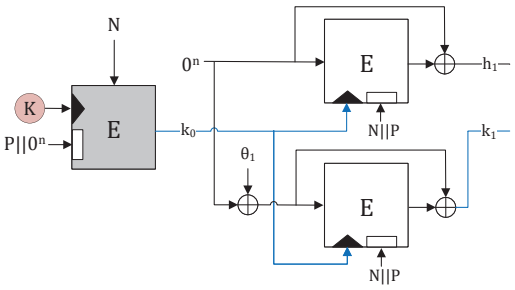
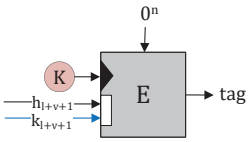


Figure 4: The Triplex mode of operation.



(A) Key Derivation Function (KDF).



(B) Tag Generation Function (TGF) with the final state for $2l$ blocks of message and $2v$ blocks of associated data.

Figure 5: Triplex initialization and finalization (protected TBC in gray).

ciphertexts, as formalized in [BPPS17]. The verification thus simply checks whether the inversion gives back the constant 0^n, and otherwise the computed n-bit string is random and independent of the valid tag. As a result, even this part can leak in full.

We can reuse the same key K in both initialization and finalization. Since the first protected TBC call involves the tweak $P\|0^n$, the n-bit 0-string serves as separation. Indeed, in the second protected TBC call, it is very unlikely that the final state (h_{l+v+1}, k_{l+v+1}) is such that $k_{l+v+1} = 0^n$ for the same preimage reason given above.

Note that it is easy to separate the computation of the unprotected TBCs with the protected ones, which will be handy in our security analysese. While many unprotected TBC calls use $N\|P$ as tweaks with $P \neq 0^n$, with very high probability these computations cannot collide with the first protected TBC. In the same spirit, while the plaintext-input of the protected TBC call in TGF is 0^n, no internal unprotected TBC calls is going to have the same input, except with very low (negligible) probability. The only exception is actually with h_0 in KDF, which will be handled in the proofs.

Regarding the difference between Triplex and sponge-based designs, although security goals may be similar (CCAmL1 + CIML2), the underlying primitives and security proofs are quite different. Triplex is built on top of a TBC and uses Hirose hash to absorb data, has different KDF, TGF and message processing parts as detailed above, and innovatively uses a third TBC to encrypt messages, leading to a better rate than TET. While the state (h_i, k_i) is also used to encrypt $2n$-bit block of message, the Hirose compression function actually absorbs h_i, k_i and the resulting $2n$-bit encrypted block. Somehow, and unlike sponge-based designs, our state can be seen as cleverly playing both the roles of the rate and the capacity in the Duplex mode: while the state is used as one-time encryption key (partially thanks to our third TBC call), and thus as a "rate," it is also absorbed without

any adversarial manipulation, and can also be considered as a "capacity." Therefore, even if h_i can be deduced from the encryption of a known plaintext it still plays a fundamental role in the high collision resistance of the compression function together with k_i. For confidentiality, only the ephemeral key k_i should remain secret to produce the next secret state (h_{i+1}, k_{i+1}). We elaborate further on the comparison with the rate and the capacity in sponge -based designs in Appendix C.

For the security analysis, we cannot generically rely on the collision resistance of Hirose's hash function in the hope to deal independently with our additional third TBC call for confidentiality. This would have made the proof simpler at a first sight but the third TBC call uses the same key k_i of the internal two TBC calls of the Hirose compression function at each iteration. Moreover, exploiting the best of the re-keying mechanism in the proof would become more complex. On the positive side, studying security "from scratch" at the fine-grained TBC level allows us to derive precise and higher security bounds.

Table 1: Parameters for Triplex. For example, the underlying TBC can be instantiated with Skinny-384 for 121-bit security (implementation results are given in Section 6).

Parameters	General n	$n = 128$ (e.g., Skinny-384)
Key size	$2n$ bits	256 bits
	n secret, n public	128 secret, 128 public
Tweak size	$2n$ bits	256 bits
Nonce size	n bits	128 bits
Maximal message	$2^n/n$ blocks	2^{95} GiB
Maximal AD	$2^n/n$	2^{95} GiB
Tag size	n bits	128 bits
Security level	$n - \log_2(n)$ bits	121 bits

3.2 Formal Description

The code description and figure are illustrated in Figure 6 and Figure 7.
The concrete parameters for Triplex are given in Table 1.

Padding Method. The padding function first appends a single 1 and then the smallest number of 0s to the plaintext M such that the length of the padded plaintext is a multiple of $2n$ bits (since in each iteration, it can handle a $2n$-bit string). The resulting padded plaintext is parsed into 2ℓ blocks of n bits where $\ell = \lceil |M|/2n \rceil$, namely $M[1] \| \ldots \| M[2\ell]$ where $|M[i]| = n$. The same padding function is applied to parse the associated data A into $2v$ blocks of n bits where $v = \lceil |A|/2n \rceil$, namely $A[1] \| \ldots \| A[2v]$ where $|A[i]| = n$, except if the associated data A is empty. In this case, no padding is required and no associated data is processed. Formally, for any $M \in \{0,1\}^*$ and $A \in \{0,1\}^*$,

$$
\begin{aligned}
M[1] \| \ldots \| M[2\ell] \leftarrow \mathrm{pad}(M) &= M \| 1 \| 0^{2n-1-(|M| \bmod 2n)}, \\
A[1] \| \ldots \| A[2v] \leftarrow \mathrm{pad}(A) &= \begin{cases} A \| 1 \| 0^{2n-1-(|M| \bmod 2n)} & \text{if } |A| > 0, \\ \emptyset & \text{if} |A| = 0. \end{cases}
\end{aligned}
$$

4 Confidentiality Analysis of Triplex

In this section, we give the privacy analysis for Triplex in the nonce-misuse resilience setting. We also explain how the techniques of [BGP+20, GPPS20] can easily be applied to Triplex to derive its CCAmL1 security at the end of the section.

procedure $\mathcal{E}(K \parallel P, N, A, M)$
Input: key $K \in \{0,1\}^k$,
public key $P \in \{0,1\}^k$
nonce $N \in \{0,1\}^n$
associated data $A \in \{0,1\}^*$
plaintext $M \in \{0,1\}^*$
Output: ciphertext $C \in \{0,1\}^{|M|}$
tag $\mathsf{tag} \in \{0,1\}^n$

Initialize
$M[1] \parallel \ldots \parallel M[2\ell] \leftarrow \mathrm{pad}(M)$
$A[1] \parallel \ldots \parallel A[2v] \leftarrow \mathrm{pad}(A)$
$h_0 \leftarrow 0^n$; $k_0 \leftarrow E_K(P \parallel 0^n, N)$
$(h_1, k_1) \leftarrow \mathsf{Hir}(h_0, k_0, N \parallel P)$

Processing Plaintext
for $i \leftarrow 1$ **to** ℓ **do**
$C[2i-1] = h_i \oplus M[2i-1]$
$C[2i] = E_{k_i}(N \parallel P, h_i \oplus \theta_2) \oplus M[2i]$
$T \leftarrow C[2i-1] \parallel C[2i]$
$(h_{i+1}, k_{i+1}) \leftarrow \mathsf{Hir}(h_i, k_i, T)$
$C \leftarrow \lceil C[1] \parallel \ldots \parallel C[2\ell] \rceil^{|M|}$

Processing Associated Data
$k_{\ell+1} \leftarrow k_{\ell+1} \oplus 1$
for $i \leftarrow 1$ **to** v **do**
$j \leftarrow \ell + i$
$T \leftarrow A[2i-1] \parallel A[2i]$
$(h_{j+1}, k_{j+1}) \leftarrow \mathsf{Hir}(h_j, k_j, T)$

Finalize
$i \leftarrow \ell + v + 1$
$\mathsf{tag} \leftarrow E_K(h_i \parallel k_i, 0^n)$
return $C \parallel \mathsf{tag}$

Inner Function $\mathsf{Hir}(h_{i-1}, k_{i-1}, m_i)$
$h_i \leftarrow E_{k_{i-1}}(m_i, h_{i-1}) \oplus h_{i-1}$
$k_i \leftarrow E_{k_{i-1}}(m_i, h_{i-1} \oplus \theta_1) \oplus h_{i-1} \oplus \theta_1$
return (h_i, k_i)

procedure $\mathcal{D}(K \parallel P, N, A, C \parallel \mathsf{tag})$
Input: key $K \in \{0,1\}^k$,
public key $P \in \{0,1\}^k$
nonce $N \in \{0,1\}^n$
associated data $A \in \{0,1\}^*$
ciphertext $C \in \{0,1\}^*$
tag $\mathsf{tag} \in \{0,1\}^n$
Output: plaintext $M \in \{0,1\}^{|C|}$ or $\perp$

Initialize
$C[1] \parallel \ldots \parallel C[2\ell] \leftarrow \mathrm{pad}(C)$
$A[1] \parallel \ldots \parallel A[2v] \leftarrow \mathrm{pad}(A)$
$h_0 \leftarrow 0^n$; $k_0 \leftarrow E_K(P \parallel 0^n, N)$
$(h_1, k_1) \leftarrow \mathsf{Hir}(h_0, k_0, N \parallel P)$

Processing Ciphertext
for $i \leftarrow 1$ **to** ℓ **do**
$M[2i-1] = h_i \oplus C[2i-1]$
$M[2i] = E_{k_i}(N \parallel P, h_i \oplus \theta_2) \oplus C[2i]$
$T \leftarrow C[2i-1] \parallel C[2i]$
$(h_{i+1}, k_{i+1}) \leftarrow \mathsf{Hir}(h_i, k_i, T)$
$M \leftarrow \lceil M[1] \parallel \ldots \parallel M[2\ell] \rceil^{|C|}$

Processing Associated Data
$k_{\ell+1} \leftarrow k_{\ell+1} \oplus 1$
for $i \leftarrow 1$ **to** v **do**
$j \leftarrow \ell + i$
$T \leftarrow A[2i-1] \parallel A[2i]$
$(h_{j+1}, k_{j+1}) \leftarrow \mathsf{Hir}(h_j, k_j, T)$

Finalize
$i \leftarrow \ell + v + 1$
$x \leftarrow E_K^{-1}(h_i \parallel k_i, \mathsf{tag})$
if $x = 0^n$ **then return** M
else return $\perp$

Figure 6: Authenticated encryption and decryption procedures of Triplex.

NONCE-MISUSE RESILIENCE. We next show that Triplex provides beyond-birthday privacy security in the nonce-misuse resilience setting (see the experiment in Figure 1).

Theorem 1. *For any adversary $\mathcal{A}$ against u users that makes at most q encryptiones queries, p ideal TBC queries, with the total number of primitive calls among these q encryption queries being at most σ, we have*

$$\mathsf{Adv}^{\mathrm{priv}}_{\mathsf{Triplex}}(\mathcal{A}) \leq \frac{u^2 + 16(\sigma+p)^2}{2^{2n+1}} + \frac{3c_1(\sigma+p) + q + 3np + n\sigma + 3\sigma + 3p + 1}{2^n} + \frac{2}{2^{n/2}}$$

where $c_1 = \max\{4n, 4u/2^n\}$ and assuming that $q \leq 2^{n-1}$, $\sigma \leq 2^{n-3}$ and $p + \sigma \leq 2^{n-1}$.

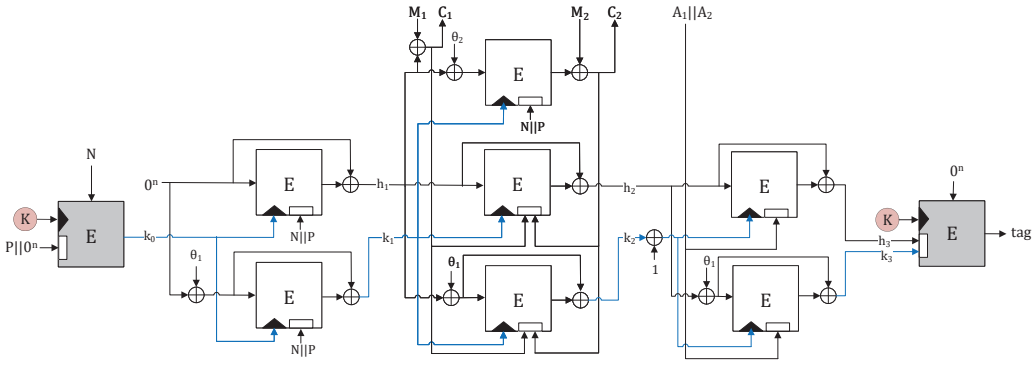


Figure 7: Triplex's full encryption details. The first 3 TBC calls on the left represents KDF producing the initial state (h_1, k_1). The last TBC call on the right represents TGF producing the tag. Only the TBC calls colored in gray at the extremities are protected.

Discussion and overview of the proof. Theorem 1 implies that Triplex provides confidentiality security as long as the total number of primitive calls σ and the total number of ideal TBC queries p (also known as the offline queries) does not exceed $2^n/n$, and the number of users u can be as large as 2^n.

The proof is based on the observation that Triplex is indistinguishable from a random scheme as long as there are no full collisions among the $2n$-bit state value (h_i, k_i). Due to freshness of the nonce in the second encryption oracle, this state value collides with probability approximately $\sigma^2/2^{2n}$. Regarding (key,tweak) collisions between direct calls to TBC and KDF, these happen with probability around $c_1 p/2^n$ for some constant c_1 since multiplicities of the public keys P_i can be bounded by Lemma 1. For collisions between direct calls to the TBC and internal TBC calls among Triplex, the influence on the bound is not significant since the maximum number of state values with the same h_i can be bounded by Lemma 2. Regarding (key, tweak) collisions between direct calls to TBC and TGF, these happen with probability about $p/2^n$ since there is no full collision among the final $2n$-bit state value $(h_{\ell+v+1}, h_{\ell+v+1})$. Regarding key collisions among many users, these happen with probability about $u^2/2^{2n}$ with the help of public key P_i. More details can be found in the formal proof. Note that our security analysis (including the integrity analysis in Section 5) can be modified syntactically to cover the case when Triplex processes associated data (AD) before message M. The proof idea is exactly the same.

Proof. Recall that in the security game as illustrated in Figure 1, the adversary is granted access to three oracles, namely the first encrytion oracle, the second encryption oracle and the ideal TBC oracle. The first encryption oracle and the ideal TBC oracle behave exactly the same in both the real and ideal worlds. Hence our goal is to prove that it is hard for the adversary to distinguish the outputs of the second encryption oracle in the real world from those outputs in the ideal world, except with a negligble probability.

Formally, from the interaction with its oracles, the adversary can obtain:

- **Ideal TBC queries.** For each query $\mathsf{Prim}(J, T, (x, +))$ with answer y, we associate it with an entry $(\mathsf{prim}, J, T, x, y, +)$. Similarly, for each query $\mathsf{Prim}(J, T, (y, -))$ with answer x, we associate it with an entry $(\mathsf{prim}, J, T, x, y, -)$.

- **Queries to the first encryption oracle.** For each query $\mathsf{Enc}_1(i, N, A, M)$ with answer $C \,\|\, \mathsf{tag}$, let $M = M[1] \,\|\, \ldots \,\|\, M[2\ell]$, $C = C[1] \,\|\, \ldots \,\|\, C[2\ell]$, and $A = A[1] \,\|\, \ldots \,\|\, A[2v]$. Let $h_0 = 0^n$, $k_0 = E_{K_i}(P_i \,\|\, 0^n, N)$, $h_1 = E_{k_0}(N \,\|\, P_i, 0^n)$, $k_1 = E_{k_0}(N \,\|\, P_i, \theta_1) \oplus \theta_1$, and for $1 \le j \le \ell$, let $h_{j+1} = E_{k_j}(C[2j-1] \,\|\, C[2j], h_j) \oplus h_j$

and $k_{j+1} = E_{k_j}(C[2j-1] \,\|\, C[2j], h_j \oplus \theta_1) \oplus h_j \oplus \theta_1$. Let $k_{\ell+1} = k_{\ell+1} \oplus 1$, and for $1 \le j \le v$, let $h_{\ell+j+1} = E_{k_{\ell+j}}(A[2j-1] \,\|\, A[2j], h_{\ell+j}) \oplus h_{\ell+j}$ and $k_{\ell+j+1} = E_{k_{\ell+j}}(A[2j-1] \,\|\, A[2j], h_{\ell+j} \oplus \theta_1) \oplus h_{\ell+j} \oplus \theta_1$. We associate the query with an entry $(\mathsf{enc}_1, i, N, A, M, C \,\|\, \mathsf{tag})$. We also use the entry $(\mathsf{inter}, J, T, x, y)$ to record the underlying primitive calls during the computation of this query, which are used for the analysis and hidden from the adversary's view.

- **Queries to the second encryption oracle.** For each query $\mathsf{Enc}_2(i, N, A, M)$ with answer $C \,\|\, \mathsf{tag}$, similarly to the case of the first encryption oracle, we associate it with an entry $(\mathsf{enc}_2, i, N, A, M, C \,\|\, \mathsf{tag})$, and use the entry $(\mathsf{inter}, J, T, x, y)$ to record the underlying primitive calls during the computation of this query. Note that the only difference between queries to the first encryption oracle and second encryption oracle is that the nonce in the former ones may repeat while the nonce in the latter ones should be unique and different from those of the former ones.

We next define some bad events in the real world, and argue that the outputs in the real world are the same as random strings when none of these bad events happen. Denote as $\mathrm{parent}(h^a_{i,b} \,\|\, k^a_{i,b})$ a sequence of state values that lead to $h^a_{i,b} \,\|\, k^a_{i,b}$ at the a-th query to user i, with $\mathrm{parent}(h^a_{i,1} \,\|\, k^a_{i,1}) = h^a_{i,0} \,\|\, k^a_{i,0} = 0^n \,\|\, k^a_{i,0}$ and $\mathrm{parent}(h^a_{i,b} \,\|\, k^a_{i,b}) = (h^a_{i,0} \,\|\, k^a_{i,0}, \ldots, h^a_{i,b-1} \,\|\, k^a_{i,b-1})$. We say the flag bad_1 is set to true if at least one of the following bad conditions is triggered:

(1) There exits two users i and j $(i \neq j)$ such that $K_i = K_j$ and $P_i = P_j$.

(2) The same P_i repeats at least c_1 times among u users.

(3) There exists an ideal TBC query $(\mathsf{prim}, J, T, x, y, *)$ such that $J = K_i$ and $T = P_i \,\|\, 0^n$ for some user i.

(4) There exists an internal primitive call $(\mathsf{inter}, J, T, x, y)$ such that $J = K_i$ and $T = P_i \,\|\, 0^n$ for some user i.

(5) There exists an entry $(\mathsf{enc}_2, i, N^a_i, A^a_i, M^a_i, C^a_i \,\|\, \mathsf{tag}^a_i)$ such that $N^a_i \,\|\, P_i = N^b_j \,\|\, P_j$ and $k^a_{i,0} = k^b_{j,0}$ for some other entry $(\mathsf{enc}_*, j, N^b_j, A^b_j, M^b_j, C^b_j \,\|\, \mathsf{tag}^b_j)$.

(6) There exists an entry $(\mathsf{enc}_*, i, N^a_i, A^a_i, M^a_i, C^a_i \,\|\, \mathsf{tag}^a_i)$ such that $k^a_{i,\ell_a+v_a+1} = 0^n$.

(7) For some entry $(\mathsf{enc}_2, i, N^a_i, A^a_i, M^a_i, C^a_i \,\|\, \mathsf{tag}^a_i)$, there exists an ideal TBC query $(\mathsf{prim}, J, T, x, y, *)$ such that $J = k^a_{i,0}$ and $T = N^a_i \,\|\, P_i$, or there exists some internal primitive call $(\mathsf{inter}, J, T, x, y)$ such that $J = k^a_{i,0}$ and $T = N^a_i \,\|\, P_i$.

(8) For some entry $(\mathsf{enc}_2, i, N^a_i, A^a_i, M^a_i, C^a_i \,\|\, \mathsf{tag}^a_i)$, there exists some $h^{a'}_{i',b'} \,\|\, k^{a'}_{i',b'}$ such that $\mathrm{parent}(h^{a'}_{i',b'} \,\|\, k^{a'}_{i',b'}) \neq \mathrm{parent}(h^a_{i,b} \,\|\, k^a_{i,b})$ and $h^{a'}_{i',b'} \,\|\, k^{a'}_{i',b'} = h^a_{i,b} \,\|\, k^a_{i,b}$.

(9) For any entry $(\mathsf{enc}_2, i, N^a_i, A^a_i, M^a_i, C^a_i \,\|\, \mathsf{tag}^a_i)$, each $k^a_{i,b}$ appears at least c_2 times, or each $h^a_{i,b}$ appears at least c_2 times.

(10) For some entry $(\mathsf{enc}_2, i, N^a_i, A^a_i, M^a_i, C^a_i \,\|\, \mathsf{tag}^a_i)$, there exists some ideal TBC query $(\mathsf{prim}, J, T, x, y, *)$ such that $x \,\|\, J = h^a_{i,b} \,\|\, k^a_{i,b}$.

(11) For some entry $(\mathsf{enc}_2, i, N^a_i, A^a_i, M^a_i, C^a_i \,\|\, \mathsf{tag}^a_i)$, there exits some ideal TBC query $(\mathsf{prim}, J, T, x, y, *)$ such that $J = K_i$ and $T = h^a_{i,\ell_a+v_a+1} \,\|\, k^a_{i,\ell_a+v_a+1}$, or there exists some internal primitive call $(\mathsf{inter}, J, T, x, y)$ such that $J = K_i$ and $T = h^a_{i,\ell_a+v_a+1} \,\|\, k^a_{i,\ell_a+v_a+1}$.

We briefly comment on the intuition behind these bad conditions.

Condition (1) is to avoid key collisions among u users. Condition (2) is to put a threshold on the maximal repeated times of a public P_i among u users, which helps to analyze other bad conditions. Conditions (3) and (4) are to guarantee that the inputs (including the secret key and tweak) of the first TBC remain different from that of the ideal TBC queries or underlying primitive calls, thus preserving the randomness of the output. Condition (5) is to ensure that even when (N, A, M) may repeat between two users, the initial value $(k_0, N \| P)$ of the hash function remains different, thus avoiding trivial collisions for the hash function between two users. Condition (6) is to avoid the input collision between the first TBC and last TBC, that is if $k^a_{i,\ell_a+v_a+1} \neq 0^n$, then the tweaks of these two TBCs are always distinct. Condition (7) is to ensure that for each encryption query $(\mathsf{enc}_2, i, N^a_i, A^a_i, M^a_i, C^a_i \| \mathsf{tag}^a_i)$, the initial input $k^a_{i,0}$ and $N^a_i \| P_i$ are fresh from that of ideal TBC queries and internal primitive calls. Condition (8) is to ensure that for each encryption query, the internal input $h^a_{i,b} \| k^a_{i,b}$ is always fresh from other internal inputs. Condition (9) is to put a threshold on the maximum number of repetitions of the key $k^a_{i,b}$ and the outer part $h^a_{i,b}$ needed in the analysis of following event. Condition (10) is to ensure that for each encryption query, the internal input $h^a_{i,b} \| k^a_{i,b}$ is always fresh from inputs of ideal TBC. Condition (11) is to ensure that the input of last TBC is fresh from that of ideal TBC queries and internal primitive calls.

Denote by $p(K,T)$ the number of ideal TBC queries with key K and tweak T that are issued by the adversary. Then $\sum_{K\in\mathcal{K},T\in\mathcal{T}} p(K,T) = p$. If bad_1 is not set to be true, then for each entry $(\mathsf{enc}_2, i, N^a_i, M^a_i, C^a_i \| \mathsf{tag}^a_i)$, tag^a_i is always a n-bit random string since the (key,tweak) pair of $(K_i, h^a_{i,\ell_a+v_a+1} \| k^a_{i,\ell_a+v_a+1})$ is fresh, and each $C^a_i[b]$ is sampled uniformly at random from a set $\{0,1\}^n \setminus S(k^a_{i,b-1}, T)$ where $S(k^a_{i,b-1}, T)$ is the set of values that have been sampled for the TBC under the pair of key and tweak $(k^a_{i,b-1}, T)$. Here $T = N^a_i \| P_i$ if b is odd and $T = C^a_i[b-2] \| C^a_i[b-1]$ otherwise. Instead of sampling each $C^a_i[b]$ from the corresponding set $\{0,1\}^n \setminus S(k^a_{i,b-1}, T)$ directly, we will first sample a value v uniformly at random from the set $\{0,1\}^n$, and if $v \in S(k^a_{i,b-1}, T)$, then a flag $\mathsf{bad}_2 \leftarrow \mathsf{true}$ and $v \xleftarrow{\$} \{0,1\}^n \setminus S(k^a_{i,b-1}, T)$ is resampled. Finally the value v is assigned to $C^a_i[b]$. So if the flag bad_2 is false, then each $C^a_i[b]$ behaves exactly the same as a random string. Hence, when neither bad_1 nor bad_2 is true, the outputs from the second encryption oracle in the real world are merely random strings, which are independent from queries of the adversary. By applying the fundamental lemma of game playing technique [BR04, BR06],

$$\mathsf{Adv}^{\mathrm{priv}}_{\mathsf{Triplex}}(\mathcal{A}) \leq \Pr[\,\mathsf{bad}_1\,] + \Pr[\,\mathsf{bad}_2 \mid \neg\mathsf{bad}_1\,] \ .$$

In Lemma 4 and Lemma 5 we will bound these two terms by

$$\frac{u^2 + 16(\sigma+p)^2}{2^{2n+1}} + \frac{3c_1(\sigma+p) + q + 3np + n\sigma + 3\sigma + 3p + 1}{2^n} + \frac{2}{2^{n/2}},$$

which completes the proof. □

Lemma 4. *Assume that the adversary makes at most q encryption queries, p ideal TBC queries, with the total number of primitive calls among these q encryption queries being at most σ. Then we have*

$$\Pr[\,\mathit{bad}_1\,] \leq \frac{u^2 + 16(\sigma+p)^2}{2^{2n+1}} + \frac{3c_1(\sigma+p) + 2np + q + 3\sigma + 3p + 1}{2^n} + \frac{2}{2^{n/2}}$$

where $c_1 = \max\{4n, 4u/2^n\}$.

Proof. We now analze the probability that the flag bad_1 is set to be true. Let event_i be the event that the i-th condition is triggered. We consider each event in turn.

For event event_1, since both K_i and P_i $(1 \leq i \leq u)$ are chosen uniformly at random from the set $\{0,1\}^n$, the probability that $K_i = K_j$ and $P_i = P_j$ is exactly $1/2^{2n}$. Summing over at most $u^2/2$ pairs of (i,j),

$$\Pr[\,\mathsf{event}_1\,] \leq \frac{u^2}{2^{2n+1}} \ .$$

Next, we analyze event event_2. In this case, each P_i is public and chosen uniformly at random from the set $\{0,1\}^n$. Let $c_1 = \max\{4n, 4u/2^n\}$. Then by using the balls-into-bins result from Lemma 1,

$$\Pr[\,\mathsf{event}_2\,] \leq \frac{1}{2^n} \ .$$

We then analyze event event_3. Conditioned on $\neg\mathsf{event}_2$, each P_i repeats at most c_1 times among all users. Hence for each ideal TBC query $(\mathsf{prim}, J, T, x, y, *)$, there are at most c_1 users such that $P_i \,\|\, 0^n = T$. The probability that $J = K_i$ for any of these c_1 users is $1/2^n$ since K_i is uniformly and randomly distributed in the set $\{0,1\}^n$. Summing over at most p ideal TBC queries,

$$\Pr[\,\mathsf{event}_3\,] \leq \frac{c_1 p}{2^n} \ .$$

The analysis of event_4 is similar to that of event_3. Summing over at most σ primitive calls,

$$\Pr[\,\mathsf{event}_4\,] \leq \frac{c_1 \sigma}{2^n} \ .$$

We then analyze event event_5. For each $N_i^a \,\|\, P_i$, there is only one corresponding $N_j^b \,\|\, P_j$ such that $N_j^b \,\|\, P_j = N_i^a \,\|\, P_i$. On the other hand, conditioned on $\neg\mathsf{event}_1$, if $P_i = P_j$, then $K_i \neq K_j$ must hold. Thus, the probability that $E_{K_i}(P_i \,\|\, 0^n, N_i^a) = E_{K_j}(P_j \,\|\, 0^n, N_j^b)$ is $1/2^n$ since these two TBCs use different keys. Summing over at most q queries,

$$\Pr[\,\mathsf{event}_5\,] \leq \frac{q}{2^n} \ .$$

Moving to event event_6, if this event happens, it implies that the n-bit output of the Davies-Meyer construction of the Hir compression function equals 0^n. The Davies-Meyer construction cannot be inverted and each of its outputs is uniformly distributed in a set of size at least $2^n - \sigma - p$. Summing over at most $\sigma + p$ TBC calls and internal primitive calls, we get

$$\Pr[\,\mathsf{event}_6\,] \leq \frac{\sigma + p}{2^n - \sigma - p} \leq \frac{2\sigma + 2p}{2^n}$$

by assuming $\sigma + p \leq 2^{n-1}$.

We next consider event event_7. For each ideal TBC entry $(\mathsf{prim}, J, T, x, y, *)$ or internal primitive call $(\mathsf{inter}, J, T, x, y)$, conditioned on $\neg\mathsf{event}_2$, there are at most c_1 encryption queries $(\mathsf{enc}_2, i, N_i^a, A_i^a, M_i^a, C_i^a \,\|\, \mathsf{tag}_i^a)$ such that $N_i^a \,\|\, P_i = T$. Among these queries, the value $k_{i,0}^a$ is chosen uniformly at random from a set of size at least $2^n - q$. Hence the probability that $k_{i,0}^a = J$ is at most $1/(2^n - q)$. Summing over a total of $p + \sigma$ ideal TBC queries and interal primitive calls,

$$\Pr[\,\mathsf{event}_7\,] \leq \frac{c_1(\sigma + p)}{2^n - q} \leq \frac{2c_1(\sigma + p)}{2^n}$$

by assuming $q \leq 2^{n-1}$.

Next, we analyze event event_8. If this event happens, then it implies that the adversary found a collision on the hash function by using at most $\sigma + p$ TBC queries. Hence, from Lemma 3:

$$\Pr[\,\mathsf{event}_8\,] \leq \frac{8(\sigma + p)^2}{2^{2n}} \ .$$

For the event event_9, we rely on the biased balls-into-bins result of Lemma 2. Note that conditioned on $\neg\mathsf{event}_8$, for $b-1 \geq 1$, each $h^a_{i,b-1} \parallel k^a_{i,b-1}$ is fresh, and thus each internal output $k^a_{i,b}$ is uniformaly distributed in a set of size at least $2^n - \sigma - p$. Hence the proability that $k^a_{i,b}$ equals to some particular value is at most $1/(2^n - \sigma - p) \leq 1/2^{n-1}$ by assuming $\sigma + p \leq 2^{n-1}$. The argument for $h^a_{i,b}$ is similar. Let $c_2 = n$. Then from Lemma 2 and assuming $\sigma \leq 2^{n-3}$,

$$\Pr[\,\mathsf{event}_9\,] \leq \frac{2}{2^{n/2}} \ .$$

Next, we analyze event event_{10}. Note that conditioned on $\neg\mathsf{event}_9$, for each ideal TBC query $(\mathsf{prim}, J, T, x, y, *)$, there are at most n values $h^a_{i,b}$ such that $h^a_{i,b} = T$. The probability that $J = k^a_{i,b}$ for any of these $h^a_{i,b} \parallel k^a_{i,b}$ is at most $1/(2^n - \sigma - p)$. Summing over at most p ideal TBC queries,

$$\Pr[\,\mathsf{event}_{10}\,] \leq \frac{np}{2^n - \sigma - p} \leq \frac{2np}{2^n}$$

by assuming $\sigma + p \leq 2^{n-1}$.

We finally analyze event event_{11}. Conditioned on $\neg\mathsf{event}_8$, each $h^a_{i,\ell+v+1} \parallel k^a_{i,\ell+v+1}$ is unique. Hence, for each ideal TBC query $(\mathsf{prim}, J, T, x, y, *)$ or internal primitive call $(\mathsf{inter}, J, T, x, y)$, there exists at most one entry $(\mathsf{enc}_2, i, N^a_i, A^a_i, M^a_i, C^a_i \parallel \mathsf{tag}^a_i)$ such that $h^a_{i,\ell_a+v_a+1} \parallel k^a_{i,\ell_a+v_a+1} = T$. On the other hand, the probability that $K_i = J$ is $1/2^n$ since K_i is a ranom n-bit string. Summing over totally $\sigma + p$ internal primitive calls and ideal TBC queries, we get

$$\Pr[\,\mathsf{event}_{11}\,] \leq \frac{\sigma + p}{2^n} \ .$$

Wrapping up, the probability that the flag bad_1 is set to be true is at most

$$\frac{u^2 + 16(\sigma+p)^2}{2^{2n+1}} + \frac{3c_1(p+\sigma) + q + 2np + 3\sigma + 3p + 1}{2^n} + \frac{2}{2^{n/2}},$$

which concludes the proof. □

Lemma 5. *Assume that the adversary makes at most q encryption queries, p ideal TBC queries, with the total number of primitive calls among these q encryption queries being at most σ. Then*

$$\Pr[\,\mathsf{bad}_2 \mid \neg\mathsf{bad}_1\,] \leq \frac{n\sigma + np}{2^n} \ .$$

Proof. Recall that $S(k^a_{i,b-1}, T)$ is the set of values that have been sampled for the TBC under the pair of key and tweak $(k^a_{i,b-1}, T)$, and $p(K, T)$ is the number of ideal TBC queries with key K and tweak T that are issued by the adversary. Conditioned on $\neg\mathsf{bad}_1$, the size of $S(k^a_{i,b-1}, T)$ is at most $n + p(k^a_{i,b-1}, T)$ since each $k^a_{i,b-1}$ appears at most n times. Hence for each encryption query $(\mathsf{enc}_2, i, N^a_i, A^a_i, M^a_i, C^a_i \parallel \mathsf{tag}^a_i)$, each $C^a_i[b]$ is sampled uniformly at random from the set $\{0,1\}^n \setminus S(k^a_{i,b-1}, T)$ of size at least $2^n - n - p(k^a_{i,b-1}, T)$. Note that the flag bad_2 is set to be true if and only if $v \in S(k^a_{i,b-1}, T)$ where $v \xleftarrow{\$} \{0,1\}^n$, which happens with probability at most $(n + p(k^a_{i,b-1}, T))/2^n$. Recall that $\sum_{K \in \mathcal{K}, T \in \mathcal{T}} p(K, T) = p$ and each $k^a_{i,b}$ appears at most n times among all encryption queries, and summing over at most σ primitive calls,

$$\Pr[\,\mathsf{bad}_2\,] \leq \frac{n\sigma + np}{2^n}$$

which conludes the proof. □

CCAmL1. Here we only present heuristic analysis for the confidentiality with leakage since the proof is standard and follows from previous analyzes for leakage-resistant modes of operations [GPPS20, BMPS21, GPPS19, BGP+20] without technical novelty. The formal definition of CCAmL1 security is given in Appendix B. As a heuristic argument, it is easy to see that ignoring decryption leakages and assuming fresh nonces, every message block is encrypted with a fresh key up to the birthday bound. Note that h_i and k_i are outputs of single-length compression functions. So at high-level, and using the simplified assumptions of [BBC+20], the security of our mode reduces to the SPA security of a single (freshly keyed) iteration encrypting a $2n$-bit block of message.[3]

A bit more formally, to prove the CCAmL1 security of our mode, which stands for CCA security with misuse-resilience and leakage-resistance [GPPS19], we can rely on the hard-to-invert leakage assumption, which follows [YSPY10] and was used in [BGP+20] for TEDT. In the CCAmL1 game, the adversary is not granted access to a leaking decryption oracle but only to a black-box decryption (and the above black-box analysis of misuse-resilience already covers such queries). We briefly sketch the leakage function in encryption and challenge queries. For a fresh nonce in encryption, we can argue that the initial state $(h_1, k_1) \in \{0,1\}^{2n}$ is random (see Figure 5(A)). That is because, up to the birthday bound, all the k_0's computed from the protected TBC are distinct, secret and random.[4] Then, the same holds for all the initial states (h_1, k_1) since there are only two calls to E_{k_0} in Hir, and their leakage thus remains quite limited. For any repeated nonce in encryption, an adversary could easily mount a DPA on the initial state by using many $2n$-bit block of message $M_1 \| M_2$.[5] However, the initial state related to any nonce-respecting query (as required for the computation of challenge ciphertexts) remains independent and secret.

We now argue why any internal state remains sufficiently hidden in any challenge encryption query and why the security follows. Let (h_i, k_i) be the current state and $M_{2i-1} \| M_{2i}$ the $2n$-bit block of message that is being processed in the computation of a challenge ciphertext. The ephemeral random key k_i is only used thrice in this iteration with distinct plaintext-inputs $h_i, h_i \oplus \theta_1, h_i \oplus \theta_2$ (see Figure 4). Since k_i will not be used anywhere else (up to the birthday bound), the 3 TBC calls should not leak enough information about the refreshed state (h_{i+1}, k_{i+1}) which will then be random and will remain secret (at least, until this point). Therefore, the secrecy and the randomness of an internal state propagates to the next one. Moreover, the final state and the TGF computation are independent of the message processing since the last TBC call (see Figure 5(B)) is protected and the long term key then remains secret. The CCAmL1 security thus reduces to the leakage of the one-time XOR computation of $C_{2i-1} = M_{2i-1} \oplus h_i$ and $C_{2i} = M_{2i} \oplus Y_i$, where $Y_i := E_{k_i}(N \| P, h_i \oplus \theta_2)$. Up to the fact that h_i is also involved in $E_{k_i}(C_{2i-1} \| C_{2i}, h_i \oplus \theta_1)$ and $E_{k_i}(C_{2i-1} \| C_{2i}, h_i)$, these XORs are the minimal amount of encrypting manipulations one can hope. Since the involvement of h_i in E_{k_i} is internal and out of the adversary's control (in the challenge ciphertext computations), it is reasonable to assume that very little informative leakage comes out. In the hard-to-invert leakage model, we can iterate the argument until the final state as the leakage between the iterations are independent.

5 Integrity Analysis of Triplex

In this section, we present the authenticity analysis of Triplex in the leakage setting.

CIML2 Security of Triplex. We will analyze the CIML2 security of Triplex in the unbounded leakage model [BPPS17, BKP+18]. In this model, the leakage functions expose

[3] SPAs are attacks that can only exploit the leakage of a constant number of primitive IOs.
[4] State-of-the-art confidentiality with leakage can anyway only be secure up to the birthday bound.
[5] DPAs are attacks that can exploit an the leakage of an adversarially chosen number of primitive IOs.

all the internal states to the adversary during the computation of unprotected building blocks, while the key of strongly protected components remains secret from the adversary and only their inputs and outputs are leaked. Concretely, in Triplex, the first and the last TBCs used for the KDF and TGF have to be strongly protected thanks to implementation-level countermeasures, while the rest of TBC calls (used for the bulk of the computation) do not require any protection to ensure integrity.

Theorem 2. *For any adversary $\mathcal{A}$ against u users that makes at most q encryption and decryption queries, p ideal TBC queries, with the total number of primitive calls of these encryption and decryption queries being at most σ, we have*

$$\mathsf{Adv}^{\mathrm{CIML2}}_{\mathsf{Triplex}}(A) \leq \frac{3q}{2^n} + \frac{u^2 + 16(\sigma+p)^2}{2^{2n+1}} + \frac{1 + (c+3)p + (c+3)\sigma}{2^n} \ ,$$

by assuming $\sigma + p \leq 2^{n-1}$ and $c = \max\{4n, 4u/2^n\}$.

Discussion and overview of the proof. Theorem 2 can be interpreted that Triplex provides integrity security as long as the total number of primitive calls σ does not exceed $2^n/n$ and the total number of ideal TBC queries p does not exceed $2^n/n$, and the number of users can be as large as 2^n. The proof is based on the fact that as long as the final $2n$-bit state value $(h_{\ell+v+1}, k_{\ell+v+1})$ is fresh, then it is hard for the adversary to predict the output of Triplex. For queries to the same user i, since each input tuple (N, A, M) is unique, these collisions can be reduced to the collision probability of Hirose's Double-block-length hash function that is captured by Lemma 3. For queries to different users, although the tuple (N, A, M) may repeat, the key pair (K_i, P_i) is unlikely to collide and thus avoids trivial collisions. Regarding (key, tweak) collisions between direct calls to TBC and KDF, these happen with probability about $cp/2^n$ for some threshold c since multiplicities of P_i can be bounded by Lemma 1. Regarding (key, tweak) collisions between direct calls to TBC and TGF, these happen with probability about $p/2^n$ since each final state value $(h_{\ell+v+1}, k_{\ell+v+1})$ is unique. The formal proof is more detailed.

Proof. From the interaction with its oracles, the adversary obtains responses from the TBC oracle, encryption oracle and decryption oracle, formally leading to:

- **Ideal TBC queries.** For each query $\mathsf{Prim}(J, T, (x, +))$ with answer y, we associate it with an entry $(\mathsf{prim}, J, T, x, y, +)$. Similarly, for each query $\mathsf{Prim}(J, T, (y, -))$ with answer x, we associate it with an entry $(\mathsf{prim}, J, T, x, y, -)$.
- **Encryption queries.** For each query $\mathsf{Enc}(i, N, A, M)$ with answer $(C \,\|\, \mathsf{tag}, L_e)$, let $M = M[1] \| \ldots \| M[2\ell]$, $C = C[1] \| \ldots \| C[2\ell]$, and $A = A[1] \| \ldots \| A[2v]$. Let $h_0 = 0^n$, $k_0 = E_{K_i}(P_i \| 0^n, N)$, $h_1 = E_{k_0}(N \| P_i, 0^n)$, $k_1 = E_{k_0}(N \| P_i, \theta_1) \oplus \theta_1$, and for $1 \leq j \leq \ell$, let $h_{j+1} = E_{k_j}(C[2j-1] \,\|\, C[2j], h_j) \oplus h_j$ and $k_{j+1} = E_{k_j}(C[2j-1] \,\|\, C[2j], h_j \oplus \theta_1) \oplus h_j \oplus \theta_1$. Let $k_{\ell+1} = k_{\ell+1} \oplus 1$, and for $1 \leq j \leq v$, let $h_{\ell+j+1} = E_{k_{\ell+j}}(A[2j-1] \,\|\, A[2j], h_{\ell+j}) \oplus h_{\ell+j}$ and $k_{\ell+j+1} = E_{k_{\ell+j}}(A[2j-1] \,\|\, A[2j], h_{\ell+j} \oplus \theta_1) \oplus h_{\ell+j} \oplus \theta_1$. Define $\mathbf{h} = (h_0, \ldots, h_{\ell+v+1})$ and $\mathbf{k} = (k_0, \ldots, k_{\ell+v+1})$. Since we are working in the unbounded leakage model, except the key K of the first and final TBC call, all the values of $\mathbf{h}$ and $\mathbf{k}$ are leaked to the adversary. Hence, we associate the query with an entry $(\mathsf{enc}, i, N, A, M, T \,\|\, \mathsf{tag}, \mathbf{h}, \mathbf{k})$. Note that the adversary is able to know the underlying primitive calls from this entry. We use $(\mathsf{leak}, J, T, x, y)$ to record each of these underlying primitive calls. That is, the tuple $(\mathsf{leak}, J, T, x, y)$ covers:
  - $(\mathsf{leak}, k_0, N \,\|\, P_i, 0^n, h_1)$ and $(\mathsf{leak}, k_0, N \,\|\, P_i, \theta_1, k_1 \oplus \theta_1)$ during initialization;
  - For $1 \leq j \leq \ell$, $(\mathsf{leak}, k_j, C[2j-1] \,\|\, C[2j], h_j, h_{j+1} \oplus h_j)$, $(\mathsf{leak}, k_j, C[2j-1] \,\|\, C[2j], h_j \oplus \theta_1, k_{j+1} \oplus h_j \oplus \theta_1)$ and $(\mathsf{leak}, k_j, N \,\|\, P_i, h_j \oplus \theta_2, C[2j] \oplus M[2j])$ during the message processing phase;

– For $1 \leq j \leq v$, $(\mathsf{leak}, k_{\ell+j}, A[2j-1] \| A[2j], h_{\ell+j}, h_{\ell+j+1} \oplus h_{\ell+j})$ and $(\mathsf{leak}, k_{\ell+j}, A[2j-1] \,\|\, A[2j], h_{\ell+j} \oplus \theta_1, k_{\ell+j+1} \oplus h_{\ell+j} \oplus \theta_1)$ during the AD processing phase.

REMARK. In the CIML2 game, the adversary is allowed to make a decryption query even this query has appeared in previous encryption phase. The intuition is that the adversary may obtain some additional leakage information via this kind of repeated decryption queries. However, in the unbounded leakage setting, all the internal values are already leaked to the adversary in encryption queries. Therefore, we ignore such trivial decryption queries in the following treatment.

- **Decryption queries.** For each query $\mathsf{Dec}(i, N, A, C \| \mathsf{tag})$ with answer (M, L_d) (here M can be either a message or just a false symbol $\perp$), with $C = C[1] \,\|\, \ldots \,\|\, C[2\ell]$, $A = A[1] \,\|\, \ldots \,\|\, A[2v]$. Similarly to the case of encryption queries, the internal values $\mathbf{h} = (h_0, \ldots, h_{\ell+v+1})$ and $\mathbf{k} = (k_0, \ldots, k_{\ell+v+1})$ are computed as follows: $h_0 = 0^n$, $k_0 = E_{K_i}(P_i \| 0^n, N)$, $h_1 = E_{k_0}(N \| P_i, 0^n)$, $k_1 = E_{k_0}(N \| P_i, \theta_1) \oplus \theta_1$, and for $1 \leq j \leq \ell$, let $h_{j+1} = E_{k_j}(C[2j-1] \,\|\, C[2j], h_j) \oplus h_j$ and $k_{j+1} = E_{k_j}(C[2j-1] \,\|\, C[2j], h_j \oplus \theta_1) \oplus h_j \oplus \theta_1$. Let $k_{\ell+1} = k_{\ell+1} \oplus 1$, and for $1 \leq j \leq v$, let $h_{\ell+j+1} = E_{k_{\ell+j}}(A[2j-1] \,\|\, A[2j], h_{\ell+j}) \oplus h_{\ell+j}$ and $k_{\ell+j+1} = E_{k_{\ell+j}}(A[2j-1] \,\|\, A[2j], h_{\ell+j} \oplus \theta_1) \oplus h_{\ell+j} \oplus \theta_1$. The checking value x is computed as $x \leftarrow E^{-1}_{K_i}(h_{\ell+v+1} \,\|\, k_{\ell+v+1}, \mathsf{tag})$. We associate this query with an entry $(\mathsf{dec}, i, N, A, M, C \,\|\, \mathsf{tag}, \mathbf{h}, \mathbf{k}, x)$. We use $(\mathsf{leak}, J, T, x, y)$ to record the underlying primitive calls that the adversary can learn from this entry.

We say that the adversary forges successfully if any of its decryption queries passes the decryption oracle, namely the returning message M is not a false symbol $\perp$. We now proceed to prove that the probability that the adversary forges successfully is negligible.

To this end, we will first define some bad conditions. A flag bad is set to be true if at least one of the following conditions is triggered:

(1) There exits two users i and j $(i \neq j)$ such that $K_i = K_j$ and $P_i = P_j$.

(2) The same P_i repeats at least c times among u users.

(3) There exists an ideal TBC query $(\mathsf{prim}, J, T, x, y, *)$ such that $J = K_i$ and $T = P_i \| 0^n$ for some user i.

(4) There exists a leaked primitive call $(\mathsf{leak}, J, T, x, y)$ such that $J = K_i$ and $T = P_i \| 0^n$ for some user i.

(5) There exists two queries $(*, i, N^a_i, A^a_i, M^a_i, C^a_i \| \mathsf{tag}^a_i, \mathbf{h}^a_i, \mathbf{k}^a_i)$ and $(*, j, N^b_j, A^b_j, M^b_j, C^b_j \| \mathsf{tag}^b_j, \mathbf{h}^b_j, \mathbf{k}^b_j)$ from two users such that $N^a_i \,\|\, P_i = N^b_j \,\|\, P_j$ and $k^a_{i,0} = k^b_{j,0}$.

(6) There exists an entry $(*, i, N^a_i, A^a_i, M^a_i, C^a_i \,\|\, \mathsf{tag}^a_i, \mathbf{h}^a_i, \mathbf{k}^a_i)$ such that $k^a_{i,\ell_a+v_a+1} = 0^n$.

(7) For the decryption query $(\mathsf{dec}, F, N^a_i, A^a_i, M^a_i, C^a_i \| \mathsf{tag}^a_i, \mathbf{h}^a_i, \mathbf{k}^a_i, x^a_i)$, there exists some previous encryption query $(\mathsf{enc}, i, N^b_i, A^b_i, M^b_i, C^b_i \| \mathsf{tag}^b_i, \mathbf{h}^b_i, \mathbf{k}^b_i)$ such that $(N^a_i, A^a_i, M^a_i) \neq (N^b_i, A^b_i, M^b_i)$ and $h^a_{i,\ell_a+v_a+1} \,\|\, k^a_{i,\ell_a+v_a+1} = h^b_{i,\ell_b+v_b+1} \,\|\, k^b_{i,\ell_b+v_b+1}$.

(8) For the decryption query $(\mathsf{dec}, i, N^a_i, A^a_i, M^a_i, C^a_i \,\|\, \mathsf{tag}^a_i, \mathbf{h}^a_i, \mathbf{k}^a_i, x^a_i)$, there exists some previous decryption query $(\mathsf{dec}, i, N^b_i, A^b_i, M^b_i, C^b_i \,\|\, \mathsf{tag}^b_i, \mathbf{h}^b_i, \mathbf{k}^b_i, x^b_i)$ for $b < a$ such that $(N^b_i, A^b_i, M^b_i) \neq (N^a_i, A^a_i, M^a_i)$ and $h^a_{i,\ell_a+v_a+1} \,\|\, k^a_{i,\ell_a+v_a+1} = h^b_{i,\ell_b+v_b+1} \,\|\, k^b_{i,\ell_b+v_b+1}$.

(9) For the decryption query $(\mathsf{dec}, i, N^a_i, A^a_i, M^a_i, C^a_i \,\|\, \mathsf{tag}^a_i, \mathbf{h}^a_i, \mathbf{k}^a_i, x^a_i)$, there exists some previous encryption query $(\mathsf{enc}, j, N^b_j, A^b_j, M^b_j, C^b_j \,\|\, \mathsf{tag}^b_j, \mathbf{h}^b_j, \mathbf{k}^b_j)$ of different user j such that $P_i \neq P_j$ and $h^a_{i,\ell_a+v_a+1} \,\|\, k^a_{i,\ell_a+v_a+1} = h^b_{j,\ell_b+v_b+1} \,\|\, k^b_{j,\ell_b+v_b+1}$.

(10) For the decryption query $(\mathsf{dec}, i, N^a_i, A^a_i, M^a_i, C^a_i \,\|\, \mathsf{tag}^a_i, \mathbf{h}^a_i, \mathbf{k}^a_i, x^a_i)$, there exists some previous decryption query $(\mathsf{dec}, j, N^b_j, A^b_j, M^b_j, C^b_j \,\|\, \mathsf{tag}^b_j, \mathbf{h}^b_j, \mathbf{k}^b_j, x^b_j)$ of different user j such that $P_i \neq P_j$ and $h^a_{i,\ell_a+v_a+1} \,\|\, k^a_{i,\ell_a+v_a+1} = h^b_{j,\ell_b+v_b+1} \,\|\, k^b_{j,\ell_b+v_b+1}$.

(11) There exists an ideal TBC query $(\mathsf{prim}, J, T, x, y, *)$ such that $J = K_i$ and $T = h^a_{i,\ell_a+v_a+1} \parallel k^a_{\ell_a+v_a+1}$ for some decryption query $(\mathsf{dec}, i, N, A, M, C \parallel \mathsf{tag}, \mathbf{h}, \mathbf{k}, x)$.

(12) There exists a leaked primitive query $(\mathsf{leak}, J, T, x, y)$ such that $J = K_i$ and $T = h_{\ell+v+1} \parallel k_{\ell+v+1}$ for some decryption query $(\mathsf{dec}, i, N, A, M, C \parallel \mathsf{tag}, \mathbf{h}, \mathbf{k}, x)$.

We briefly discuss the intuition behind these bad conditions. Condition (1) is to avoid key collisions among u users. Condition (2) is to put a theshold on the maximal repeated times of a public P_i among u users, which is helpful to anlayze other bad conditions. Conditions (3) and (4) are to guarantee that the inputs (including the secret key and tweak) of the first TBC remain different from that of the ideal TBC queries or underlying primitive calls, thus preserving the randomness of output. Condition (5) is to ensure that even when (N, A, M) may be repeated between two users, the initial values of the hash function remains different, thus avoiding trivial collision for the hash function between two queries. Condition (6) is to avoid the input collision between the first and last TBC calls. That is, if $k^a_{i,\ell_a+v_a+1} \neq 0^n$, then the tweaks of these two TBCs are always distinct. Conditions (7)-(10) are to ensure that for each decryption query $(\mathsf{dec}, i, N^a_i, A^a_i, M^a_i, C^a_i \parallel \mathsf{tag}^a_i, \mathbf{h}^a_i, \mathbf{k}^a_i)$, the tweak $h^a_{i,\ell_a+v_a+1} \parallel k^a_{i,\ell_a+v_a+1}$ is different from that of other encryption or decryption queries. Conditions (11) and (12) are to ensure that for each decryption query $(\mathsf{dec}, i, N^a_i, A^a_i, M^a_i, C^a_i \parallel \mathsf{tag}^a_i, \mathbf{h}^a_i, \mathbf{k}^a_i)$, either the key K_i or the tweak $h^a_{i,\ell_a+v_a+1} \parallel k^a_{i,\ell_a+v_a+1}$ is fresh from that of ideal TBC queries and underlying primitive calls. It can be seen in the following that by excluding these bad conditions, the analysis of the adversary forging successfully apppears to be transparent.

First observe that

$$\Pr[A \text{ forges}] \leq \Pr[A \text{ forges} \mid \neg\mathsf{bad}] + \Pr[\mathsf{bad}] \ . \tag{1}$$

A bound on the probability that bad is set is given in Lemma 6.

We then analyze the chance that A forges given that bad is not set to be true. Such a forgery requires that $x^a_i = E^{-1}_{K_i}(h^a_{i,\ell_a+v_a+1} \parallel k^a_{i,\ell_a+v_a+1}, \mathsf{tag}^a_i) = 0^n$ for some decryption query $(\mathsf{dec}, i, N^a_i, A^a_i, M^a_i, C^a_i \parallel \mathsf{tag}^a_i, \mathbf{h}^a_i, \mathbf{k}^a_i, x^a_i)$. We distinguish several cases according to the type of decryption query $(\mathsf{dec}, i, N^a_i, A^a_i, M^a_i, C^a_i \parallel \mathsf{tag}^a_i, \mathbf{h}^a_i, \mathbf{k}^a_i, x^a_i)$.

- If $(N^a_i, A^a_i, C^a_i) = (N^b_i, A^b_i, C^b_i)$ for some previous encryption query $(\mathsf{enc}, i, N^b_i, A^b_i, M^b_i, C^b_i \parallel \mathsf{tag}^b_i, \mathbf{h}^b_i, \mathbf{k}^b_i, x^b_i)$, then $\mathsf{tag}^a_i \neq \mathsf{tag}^b_i$ and $E^{-1}_{K_i}(h^a_{i,\ell_a+v_a+1} \parallel k^a_{i,\ell_a+v_a+1}, \mathsf{tag}^a_i) \neq x^b_i = 0^n$. Hence the probability that this query is a valid forgery is 0.

- If $(N^a_i, A^a_i, C^a_i) = (N^b_i, A^b_i, C^b_i)$ for some previous decryption query $(\mathsf{dec}, i, N^b_i, A^b_i, M^b_i, C^b_i \parallel \mathsf{tag}^b_i, \mathbf{h}^b_i, \mathbf{k}^b_i)$, then $\mathsf{tag}^a_i \neq \mathsf{tag}^b_i$, and $E^{-1}_{K_i}(h^a_{i,\ell_a+v_a+1} \parallel k^a_{i,\ell_a+v_a+1}, \mathsf{tag}^a_i) = 0^n$ with probability at most $1/(2^n - q)$ since conditioned on $\neg\mathsf{bad}$, the value x^a_i is randomly picked up from a set of size at least $2^n - q$.

- If neither of above two cases happens, then due to bad not happening, either K_i or $h^a_{i,\ell_a+v_a+1} \| k^a_{i,\ell_a+v_a+1}$ is fresh, and thus the probability that $E^{-1}_{K_i}(h^a_{i,\ell_a+v_a+1} \| k^a_{i,\ell_a+v_a+1}, \mathsf{tag}^a_i) = 0^n$ is exactly $1/2^n$.

Therefore, summing over at most q decryption queries, the probability that A forges conditioned on $\neg\mathsf{bad}$ is worth

$$\Pr[A \text{ forges} \mid \neg\mathsf{bad}] \leq \frac{q}{2^n - q} \leq \frac{2q}{2^n}$$

by assuming $q \leq 2^{n-1}$. Equation 1 and the bound of Lemma 6 complete the proof. □

Lemma 6. *Assume that the adversary makes at most q construction queries (including both the encryption queries and decryption queries), p ideal TBC queries, and the total number of primitive calls of these encryption and decryption queries is at most σ. Then*

$$\Pr[\,\mathsf{bad}\,] \leq \frac{u^2 + 16(\sigma+p)^2}{2^{2n+1}} + \frac{q+1+(c+3)p+(c+3)\sigma}{2^n} \ ,$$

where the threshold $c = \max\{4n, 4u/2^n\}$.

Proof. We now bound the chance that the flag bad is set to be true. Denote by event_i the event when the i-th condition is triggered. We analyze each event in turn.

For event event_1, since both K_i and P_i $(1 \leq i \leq u)$ are chosen uniformly at random from the set $\{0,1\}^n$, the probability that $K_i = K_j$ and $P_i = P_j$ is exactly $1/2^{2n}$. Summing over at most $u^2/2$ pairs of (i,j),

$$\Pr[\,\mathsf{event}_1\,] \leq \frac{u^2}{2^{2n+1}} \ .$$

Next, we analyze the event event_2. In this case, each P_i is public and chosen uniformly at random from the set $\{0,1\}^n$. Let $c = \max\{4n, 4u/2^n\}$. Then by using the balls-into-bins result from Lemma 1,

$$\Pr[\,\mathsf{event}_2\,] \leq \frac{1}{2^n} \ .$$

We then analyze the event event_3. Conditioned on $\neg\mathsf{event}_2$, each P_i repeats at most c times among all users. Hence for each ideal TBC query $(\mathsf{prim}, J, T, x, y, *)$, there are at most c users such that $P_i \,\|\, 0^n = T$. The probability that $J = K_i$ for any of these c users is $1/2^n$ since K_i is uniformly and randomly distributed in the set $\{0,1\}^n$. Summing over at most p ideal TBC queries,

$$\Pr[\,\mathsf{event}_3\,] \leq \frac{cp}{2^n} \ .$$

The analysis of event_4 is similar to that of event_3. Summing over at most σ primitive calls,

$$\Pr[\,\mathsf{event}_4\,] \leq \frac{c\sigma}{2^n} \ .$$

We then analyze event event_5. For each $N_i^a \,\|\, P_i$, there is only one corresponding $N_j^b \,\|\, P_j$ such that $N_j^b \,\|\, P_j = N_i^a \,\|\, P_i$. On the other hand, conditioned on $\neg\mathsf{event}_1$, if $P_i = P_j$, then $K_i \neq K_j$ must hold. Thus the probability that $E_{K_i}(P_i \,\|\, 0^n, N_i^a) = E_{K_j}(P_j \,\|\, 0^n, N_j^b)$ is $1/2^n$ since these two TBCs use different keys. Summing over at most q queries,

$$\Pr[\,\mathsf{event}_5\,] \leq \frac{q}{2^n} \ .$$

Moving to event event_6, if this event happens, it implies that the n-bit output of Davies-Meyer construction of the Hir compression function equals to 0^n. The Davies-Meyer construction cannot be inverted, and each of its outputs is uniformly distributed in a set of size at least $2^n - \sigma - p$. Summing over at most $\sigma + p$ TBC calls and internal primitive calls, we get

$$\Pr[\,\mathsf{event}_6\,] \leq \frac{\sigma+p}{2^n-\sigma-p} \leq \frac{2\sigma+2p}{2^n}$$

by assuming $\sigma + p \leq 2^{n-1}$.

Next, we analyze the events from event_7 to event_{10}. If any of these four events happens, it means that the adversary can find a collision on the hash function H via at most $\sigma + p$ ideal TBC queries. Hence, from Lemma 3 we get

$$\Pr\left[\,\vee_{i=7}^{10}\mathsf{event}_i\,\right] \leq \frac{8(\sigma+p)^2}{2^{2n}} \ .$$

Next, we analyze the event event_{11}. Note that conditioned on that there is not collision on the hash function H, for each ideal TBC query $(\mathsf{prim}, J, T, x, y, *)$, there is at most one decryption query $(\mathsf{dec}, i, N_i^a, A_i^a, M_i^a, C_i^a \| \mathsf{tag}_i^a, \mathbf{h}_i^a, \mathbf{k}_i^a, x_i^a)$ such that $h_{i,\ell_a+v_a+1}^a \| k_{i,\ell_a+v_a+1}^a = T$, and the probability that $J = K_i$ is $1/2^n$ since K_i is a random n-bit string. Summing over at most p ideal TBC queries, we get

$$\Pr[\mathsf{event}_{11}] \leq \frac{p}{2^n} \ .$$

The analysis of event event_{12} is similar to that of event_{11}, and summing over at most σ primitive calls we get

$$\Pr[\mathsf{event}_{12}] \leq \frac{\sigma}{2^n} \ .$$

Wrapping up, the probability that the flag is set to be bad is at most

$$\Pr[\mathsf{bad}] \leq \frac{u^2 + 24(\sigma+p)^2}{2^{2n+1}} + \frac{q+1+(c+3)p+(c+3)\sigma}{2^n} \ ,$$

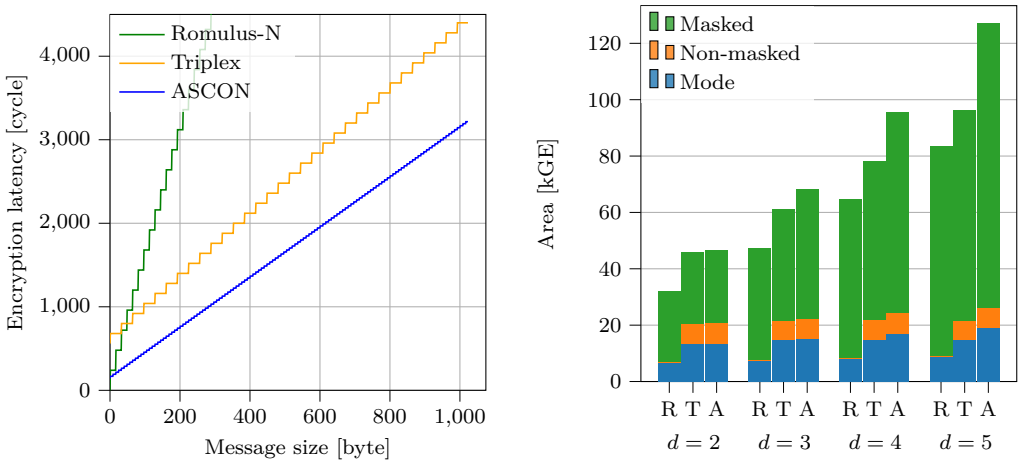
which concludes the proof. □

6 Implementation Results

As mentioned in introduction, the natural competitors of Triplex are TET when it comes to TBC-based modes and Ascon & Spook when it comes to Sponge-based ones. Since the improvement of the rate over TET is clear, we use this last section to initiate a comparative discussion of the implementation features of leakage-resilient modes of operation based on TBCs and permutations. For this purpose, we introduce a leveled implementation of Triplex in hardware with Skinny-384+ as TBC [BJK+16, Kha22]. We compare this implementation with a leveled implementation of Ascon [DEMS21]. For both ciphers, the implementation is made of a masked and an unmasked implementation of the primitive (Skinny-384+ and the Ascon permutation), targeting resistance against DPA and SPA, respectively. The primitive implementations are integrated in top-level mode implementations (containing the datapath of the mode and the control FSM). This architecture has been chosen for its simplicity and, as described below, our implementations of the primitives are also designed for simplicity. As any prototype implementations, they could be further optimized, but we do not expect such optimizations to change our main conclusions.

The implementation of Skinny is round-based (instantiating 32 S-boxes), and takes one cycle per round for the the non-masked implementation, while it takes 6 cycles per round for the masked implementation. Each masked S-box implementation contains two instances of the HPC2 masked AND gadget [CGLS21]. The implementations of the Ascon permutation are more serialized and based on an 80-bit S-box pipeline datapath (instantiating 16 S-boxes). The unmasked implementation therefore takes 4 cycles per round, while the masked implementation requires 6 cycles per round. In order to make the advantages of leakage-resistant modes of operation explicit, we add the results for Romulus-N [IKMP20], which uses the same long-term key in all its call to the Skinny-384+ TBC and therefore requires a uniformly masked implementation.

In Figure 8A, we show the number of cycles required for encrypting messages of various length. For short messages, the encryption time is dominated by the KDF and TGF. This duration is longer for Triplex than for Ascon, mainly due to the larger number of rounds in Skinny (the Hirose part in the KDF only takes less than 15 % of the KDF/TGF cycles). For longer messages, Ascon takes 24 cycles to encrypt a 64 bit message block, while Triplex takes 3×40 cycles to encrypt 256 bits of message. Overall, Ascon has a slightly better throughput despite a more serial architecture (80-bit vs. 128-bit). But



(A) Encryption latency as a fct. of message size.

(B) Area requirements of implementations synthesized for a 65 nm CMOS commercial technology.

Figure 8: Ascon (A), Romulus-N (R) and Triplex (T) hardware implementation figures.

these results are admittedly quite sensitive to the security margins taken by designers.[6] Eventually, the different slope of Romulus-N's performance curve confirms an interest of leveled implementations from the performance (throughput) viewpoint.

The implementations have been synthesized for a 65 nm CMOS commercial technology. In Figure 8B, we can see that the area of these implementations is dominated by the masked primitives (confirming the limited overheads of the leveled approach when high physical security levels are required).[7] So compared to Romulus-N, Triplex has a slightly higher area cost due to the need for both a masked and an unmasked cipher implementations. But this overhead shrinks relatively with a growing number of shares in the masking scheme, and is rewarded with a significantly better performance for messages larger than as little as 48 bytes. Moreover, we observe that the masked Skinny and Ascon have similar areas for small number of shares, while the area of Ascon grows faster than the one of Skinny-384+ as the number shares increases, due to the larger number of AND gates to mask: 80 for the 80-bit Ascon architecture (since it uses 5-bit S-boxes with 5 AND gates), 32 for the 128-bit Skinny-384+ architecture (using 16 serial S-boxes with 2 AND gates). We note that these values could be reduced with a more serialized masked implementation, at the cost of a higher KDF and TGF latency (amortized for long messages).

In both cases the unmasked primitives have a low area. So reducing their serialization would bring a significant throughput increase at the cost of limited area increase.

This study shows that the performances of optimized implementations of modes like Triplex heavily depend on the characteristics of its underlying TBC. Overall, Triplex instantiated with Skinny can achieve performance comparable to a sponge like Ascon, with variations depending on the implementation architectural tradeoff targeted.

Source code is available at https://github.com/uclcrypto/aead_modes_leveled_hw.

[6] The best-known cryptanalysis result against Skinny reaches only 27 rounds out of 40, and is a related-tweakey impossible differential [LGS17, SMB18].

[7] The cost of the mode corresponds to the additional control logic / state machine, registers and multiplexers needed to implement the mode over the masked (and possibly unmasked) primitives.

Table 2: Parameters' comparison between leakage-resistant TBC-based AEAD. Grade 2 means CCAmL1 + CIML2, Grade 3 means CCAmL2 + CIML2 [BBC+20]. The first n key bits correspond to the secret key K and the last ones to the public P (for multi-user security). The rate is the amortized value of the ratio to process n-bit message blocks per number of TBC calls.

	TEDT	TET	Romulus-T	TEDT2	Triplex
Grade	3	2	3	3	2
Key size	$2n-1$	$2n-1$	n	n	$2n$
# pass(es)	2	1	2	2	1
Tweak size	n	n	$2n$	$2n$	$2n$
Rate: $\|M\|$/TBC	1/4	1/2	1/3	1/3	2/3
Rate: $\|AD\|$/TBC	1/4	1/2	1	1/3	1
Nonce size	$3n/4$	n	n	n	n
Message blocks	max $2^{n/4-1}$	max $2^{n/2}$	max $2^{n/2-8}$	max $2^{n/2}$	max $2^n/n$
Multi-user	yes	yes	no	no	yes

7 Conclusion

As a conclusion, we provide a summary of state-of-the-art leakage-resistant constructions in Table 2, which contains the operational parameters of different published modes. We note that the quantitative bounds of their leakage security are similar: CIML2 is beyond birthday for all candidates and CCAmL1 and CCAmL2 can only be birthday secure given current techniques in the model of Guo et al. [GPPS19]. All schemes require two protected TBC calls for this purpose, excepted TEDT2 that needs 3.[8]

We note that the Romulus-T mode is very similar to TEDT implemented with a TBC with $2n$-bit tweaks. The first pass relies on PSV-encryption [PSV15] with a tweak schedule *à la* Romulus. The second pass in an hash-then-MAC based on Hirose's compression function with a large tweak like ours, and with a final TGF as in TEDT. It requires 6 TBC calls to process $2n$ bits of message. It is not designed for multi-user security but a direct adaptation would lead to multi-user security up to the birthday bound.

One additional advantage of Triplex over TET is that does not require the related tweakey security of the TBC (due to the fact that the state (h_i, k_i) is such that $h_i \oplus k_i = M_{i-1} \oplus C_{i-i}$ in TET). Furthermore, Triplex does not need to invert unprotected TBC calls in decryption. As mentioned in introduction, Triplex also makes TBC-based designs more comparable to Sponge-based ones in terms of rate and state size.

Quite naturally, this table casts as main open problem the quest of further improved modes, whether being from the performance (e.g., rate) viewpoint, the simplicity of the underlying primitive or the weakness of the underlying (physical) assumptions.

Acknowledgments. Gaëtan Cassiers, Thomas Peters and François-Xavier Standaert are respectively research fellow, associate researcher and senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in parts by the European Union through the ERC consolidator grant SWORD (num. 724725) and by the Walloon Region through the Win2Wal project PIRATE (num. 1910082).

References

[ADL17] Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting authenticated encryption robustness with minimal modifications. In *CRYPTO (3)*, volume 10403 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2017.

[8] TEDT2 also claims a better CCA security with leakage in a different model.

[BBB+20] Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaëtan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaëtan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters, François-Xavier Standaert, Balazs Udvarhelyi, and Friedrich Wiemer. Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. *IACR Trans. Symmetric Cryptol.*, 2020(S1):295–349, 2020.

[BBC+20] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In *CRYPTO (1)*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400. Springer, 2020.

[BGP+20] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. TEDT, a leakage-resist AEAD mode for high physical security applications. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):256–320, 2020.

[BGPS21] Francesco Berti, Chun Guo, Thomas Peters, and François-Xavier Standaert. Efficient leakage-resilient macs without idealized assumptions. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II*, volume 13091 of *Lecture Notes in Computer Science*, pages 95–123. Springer, 2021.

[BHT18] Priyanka Bose, Viet Tung Hoang, and Stefano Tessaro. Revisiting AES-GCM-SIV: multi-user security, faster key derivation, and better bounds. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, pages 468–499, 2018.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *CRYPTO (2)*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

[BKP+18] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Ciphertext integrity with misuse and leakage: Definition and efficient constructions with symmetric primitives. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, pages 37–50, 2018.

[BMPS21] Olivier Bronchain, Charles Momin, Thomas Peters, and François-Xavier Standaert. Improved leakage-resistant authenticated encryption based on hardware AES coprocessors. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):641–676, 2021.

[BPPS17] Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On leakage-resilient authenticated encryption with decryption leakages. *IACR Trans. Symmetric Cryptol.*, 2017(3):271–293, 2017.

[BR04] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. *IACR Cryptol. ePrint Arch.*, page 331, 2004.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 409–426, 2006.

[CGLS21] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Trans. Computers*, 70(10):1677–1690, 2021.

[DEM+17] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP - towards side-channel secure authenticated encryption. *IACR Trans. Symmetric Cryptol.*, 2017(1):80–105, 2017.

[DEM+20] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. ISAP v2.0. *IACR Trans. Symmetric Cryptol.*, 2020(S1):390–416, 2020.

[DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *J. Cryptol.*, 34(3):33, 2021.

[DM19] Christoph Dobraunig and Bart Mennink. Leakage resilience of the duplex construction. In *ASIACRYPT (3)*, volume 11923 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2019.

[GKP] Chun Guo, Mustafa Khairallah, and Thomas Peyrin. AET-LR: Rate-1 leakage-resilient AEAD based on the Romulus family. NIST LWC Workshop, 2020.

[GPPS19] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated encryption with nonce misuse and physical leakage: Definitions, separation results and first construction - (extended abstract). In *LATINCRYPT*, volume 11774 of *Lecture Notes in Computer Science*, pages 150–172. Springer, 2019.

[GPPS20] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards low-energy leakage-resistant authenticated encryption from the duplex sponge construction. *IACR Trans. Symmetric Cryptol.*, 2020(1):6–42, 2020.

[Hir06] Shoichi Hirose. Some plausible constructions of double-block-length hash functions. In *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, pages 210–225, 2006.

[HT17] Viet Tung Hoang and Stefano Tessaro. The multi-user security of double encryption. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, pages 381–411, 2017.

[IKMP20] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the titans: The Romulus and Remus families of lightweight AEAD algorithms. *IACR Trans. Symmetric Cryptol.*, 2020(1):43–120, 2020.

[JNPS21] Jérémy Jean, Ivica Nikolic, Thomas Peyrin, and Yannick Seurin. The deoxys AEAD family. *J. Cryptol.*, 34(3):31, 2021.

[Kha22] Mustafa Khairallah. Romulus: Lighweight aead from tweakable block ciphers. In *Hardware Oriented Authenticated Encryption Based on Tweakable Block Ciphers*, pages 115–134. Springer, 2022.

[LGS17] Guozhen Liu, Mohona Ghosh, and Ling Song. Security analysis of SKINNY under related-tweakey settings (long paper). *IACR Trans. Symmetric Cryptol.*, 2017(3):37–72, 2017.

[Lis21] Eik List. TEDT2 - highly secure leakage-resilient tbc-based authenticated encryption. In *LATINCRYPT*, volume 12912 of *Lecture Notes in Computer Science*, pages 275–295. Springer, 2021.

[LRW11] Moses D. Liskov, Ronald L. Rivest, and David A. Wagner. Tweakable block ciphers. *J. Cryptol.*, 24(3):588–613, 2011.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards.* Springer, 2007.

[Pey20] Thomas Peyrin. Tweakable block cipher-based cryptography, 2020. FSE, Invited talk.

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In *CCS*, pages 96–108. ACM, 2015.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 98–107, 2002.

[SMB18] Sadegh Sadeghi, Tahereh Mohammadi, and Nasour Bagheri. Cryptanalysis of reduced round SKINNY block cipher. *IACR Trans. Symmetric Cryptol.*, 2018(3):124–162, 2018.

[USS+20] Florian Unterstein, Marc Schink, Thomas Schamberger, Lars Tebelmann, Manuel Ilg, and Johann Heyszl. Retrofitting leakage resilient authenticated encryption to microcontrollers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):365–388, 2020.

[YSPY10] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 141–151. ACM, 2010.

A Proof of Lemma 3

Let $\mathcal{A}$ be a collision-finding algorithm of H with oracle access to the ideal TBC E (including both forward and backward direction). Let (K, T, X, Y) be the entry that records the query and response of E, where K is the key, T the tweak, X the plaintext and Y the ciphtertext. To compute the underlying compression function Hir, it requires a pair of entries $(k_{i-1}, m_i, h_{i-1}, h_i \oplus h_{i-1})$ and $(k_{i-1}, m_i, h_{i-1} \oplus \theta_1, k_i \oplus h_{i-1} \oplus \theta_1)$. We assume that $\mathcal{A}$ makes at most p such pairs of queries to E. For two inputs (h, k, m) and (h', k', m') to Hir, we say they are matching if $(h, k) = (h' \oplus \theta_1, k')$.

We first consider the collision events when the inputs to Hir are non-matching. For $2 \le j \le p$, let C_j be the event that a collision of non-mathcing inputs is found for Hir at the j-th pair of queries. It implies that for some $j' < j$,

$$\mathsf{Hir}(h_{j-1}, k_{j-1}, m_j) = \mathsf{Hir}(h_{j'-1}, k_{j'-1}, m_{j'}) \text{ or } \mathsf{Hir}(h_{j'-1} \oplus \theta_1, k_{j'-1}, m_{j'})$$

or

$$\mathsf{Hir}(h_{j-1} \oplus \theta_1, k_{j-1}, m_j) = \mathsf{Hir}(h_{j'-1}, k_{j'-1}, m_{j'}) \text{ or } \mathsf{Hir}(h_{j'-1} \oplus \theta_1, k_{j'-1}, m_{j'}),$$

which is the same as:

$$(h_j \oplus h_{j-1}, k_j \oplus h_{j-1} \oplus \theta_1) \in \{(h_{j'} \oplus h_{j'-1}, k_{j'} \oplus h_{j'-1} \oplus \theta_1), (k_{j'} \oplus h_{j'-1} \oplus \theta_1, h_{j'} \oplus h_{j'-1})\} \ .$$

Hence,

$$\Pr[C_j] \le \frac{2(j-1)}{(2^n - (2j-2))(2^n - (2j-1))} \le \frac{2(j-1)}{(2^n - (2j-1))^2},$$

since h_j and k_j are chosen without replacement uniformly at random from a set of size at least $2^n - (2j-2)$. Let $C = \vee_{j=1}^{p} C_j$. We then have:

$$\Pr[C] \le \sum_{j=2}^{p} \Pr[C_j] \le \sum_{j=2}^{p} \frac{2(j-1)}{(2^n - (2j-1))^2} \ .$$

We next consider the collison events when the inputs to Hir are matching. Let (h, k, m) and (h', k', m') be the pair of inputs of Hir for the collision. Then $(h, k) = (h' \oplus \theta_1, k')$ since they are matching. (h, k) and (h', k') are both the outputs of Hir, or at most one of them is the initial value (h_0, k_0) of H. Denote as $(\hat{h}, \hat{k}, \hat{m})$ and $(\hat{h'}, \hat{k'}, \hat{m'})$ the inputs to Hir to produce (h, k) and (h', k') respectively. Then these matching inputs require the following equation:

$$(h, k) = (h' \oplus \theta_1, k') \text{ or } (h, k) = (h_0 \oplus \theta_1, k_0).$$

If the former equation holds, then $(\hat{h}, \hat{k}, \hat{m})$ and $(\hat{h'}, \hat{k'}, \hat{m'})$ are non-matching since

$$(h, k) = (h' \oplus \theta_1, k') \neq (k', h'),$$

where θ_1 is a non-zero constant. Denote as C_j^m the event that a collision from matching inputs is found for Hir at the j-th pair of queries. Then from the above analysis,

$$\Pr[C_j^m] \le \frac{2(j-1)+1}{(2^n - (2j-2)(2^n - (2j-1)))} \le \frac{2j-1}{(2^n - (2j-1))^2} \ .$$

Let $C^m = \vee_{j=1}^{p} C_j^m$. Then we finally have:

$$\Pr[C^m] \le \sum_{j=1}^{p} \Pr[C_j^m] \le \sum_{j=1}^{p} \frac{2j-1}{(2^n - (2j-1))^2} \ .$$

Hence by the union bound,

$$\begin{aligned}
\Pr[\mathcal{A} \text{ finds a collision on } H] &\le \Pr[C] + \Pr[C^m] \\
&\le \sum_{j=2}^{p} \frac{2(j-1)}{(2^n - (2j-1))^2} + \sum_{j=1}^{p} \frac{2j-1}{(2^n - (2j-1))^2} \\
&\le \sum_{j=1}^{p} \frac{4j-3}{(2^n - (2j-1))^2} \\
&\le \frac{2p^2 - p}{(2^{n-1})^2} \le \frac{8p^2}{2^{2n}},
\end{aligned}$$

which concludes the proof.

Game $\mathsf{G}_{\Pi}^{\mathsf{CCAmL1}}(\mathcal{A})$

$K_1, K_2, \ldots, \xleftarrow{\$} \mathcal{K}$; $b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Prim},\mathsf{LEnc}_1,\mathsf{LEnc}_2,\mathsf{Dec}}$
return $(b' = b)$

procedure $\mathsf{Prim}(J, T, X)$
if $X = (+, x)$ **then return** $E_J(T, x)$
if $X = (-, y)$ **then return** $E_J^{-1}(T, y)$

procedure $\mathsf{Dec}(i, N, A, C \,\|\, \mathsf{tag})$
$M \leftarrow \mathcal{D}(K_i, N, A, C \,\|\, \mathsf{tag})$
return M

procedure $\mathsf{LEnc}_1(i, N, A, M)$
$C \,\|\, \mathsf{tag} \leftarrow \mathcal{E}(K_i, N, A, M)$
$L_e \leftarrow \mathcal{L}_E(K_i, N, A, M)$
return $(C \,\|\, \mathsf{tag}, L_e)$

procedure $\mathsf{LEnc}_2(i, N, A, M^0, M^1)$
if $|M^0| \neq |M^1|$ **then return** $\perp$
$C^b \,\|\, \mathsf{tag}^b \leftarrow \mathcal{E}(K_i, N, A, M^b)$
$L_e^b \leftarrow \mathcal{L}_E(K_i, N, A, M^b)$
return $(C^b \,\|\, \mathsf{tag}^b, L_e^b)$

Figure 9: Game $\mathsf{G}_{\Pi}^{\mathsf{CCAmL1}}$: multi-user CCAmL1 security of an AE Π.

B Definition of CCAmL1 Security

In this section, we recall the notion of CCAmL1 security (Chosen-Ciphertext Attacks security with misuse-resilience and Leakage) [GPPS19] in the multi-user setting. In this model, the adversary is granted the access to encryption oracle with leakage and decryption oracle, and aims at attacking the confidentiality of several messages encrypted under unique and fresh nonces. The advantage of the adversary is formalized by the left-or-right paradigm, namely for any two different messages M^0 and M^1 of equal length, the probability that the adversary can distinguish the encryption of these two messages is negligible. The selection of this paradigm is due to the conceptual difficulty to define the leakage of idealized objects in the real-or-random game. The CCAmL1 security game is illustrated in Figure 9. In this game, for queries to the same user, the adversary may repeat the nonce in the first leaking encryption oracle LEnc_1, but the nonce in the second leaking encryption oracle LEnc_2 should be unique and fresh. For queries to different users, the adversary may repeat the nonce in both oracles. The adversary also has access to the decryption oracle Dec, but cannot forward queries from the second encryption oracle to Dec since this will result in trivial win. Given an adversary $\mathcal{A}$, define

$$\mathsf{Adv}_{\Pi}^{\mathsf{CCAmL1}}(\mathcal{A}) = 2\Pr\left[\,\mathsf{G}_{\Pi}^{\mathsf{CCAmL1}}(\mathcal{A})\,\right] - 1$$

as the advantage of the adversary against the CCAmL1 security of an AE scheme Π.

C More State Comparison with Sponges

Among the existing one-pass sponge-based designs, [BMPS21] highlights that Ascon and Spook achieve Grade-2 leakage security (i.e., CCAmL1 + CIML2), where the integrity holds in the unbounded leakage model. Like Triplex, that means that as long as the KDF and TGF functions are well-protected it is still infeasible to forge a ciphertext even if the full state leaks. Up to the generation of the initial state, this is not surprising since the design reduces to a hash-then-MAC in the CIML2 experiment, and the bulk of the computation which processes the message is a Duplex-like hash function in both modes.

To formally prove CIML2, the permutations must be modeled as ideal objects, as if the underlying permutation considered in the mode was uniformly picked among all the permutations of the appropriate size. This size is the size of the state which is usually denoted by b and decomposed into the rate part of r bits and the capacity part of c bits, so that $b = r + c$. Let S be a state, so the output of a permutation call. The input of the

next permutation call is thus of the form $S \oplus (M \oplus 0^c)$, where the processed message block has size $|M| = r$. Therefore, even if the adversary knows the full state S in the unbounded leakage model, she has no direct control on the capacity that is being processed through the several permutation calls. She can only explicitly choose the rate of the next permutation input but not the capacity. Consequently, the bit size c plays an important role in the CIML2 security bound since it is easy to see that state collision can be generically reached from $2^{c/2}$ ideal permutation queries. Indeed, once the adversary finds a collision on the c-bit state, she can easily choose the next message blocks to equalize the rate part she fully controls to get the same states. As a result, we must have $c \geq 2n$ if we target n-bit security for CIML2, and the TETSponge mode shows that taking $c = 2n$ gives $n - \log n$ bits of security in the single-user setting, and $n - 2\log n$ bits of security in the multi-user setting with appropriate KDF and TGF functions. With respect to Triplex based on TBC with n-bit keys and outputs, these sponge-based designs must satisfy $c = 2n$ to have a similar CIML2 security. We eventually insist that the state leaks in full and that this security result cannot rely on the fact the the capacity remains secret. Actually, the capacity is *known* in the unbounded leakage setting. For the same reason, we can interpret (h, k) in Triplex as being used also as a capacity.

To prove confidentiality, we can of course not rely on the unbounded leakage setting. However, to generate the next random state it is enough that a sufficient number of bits of the capacity remain secret. For instance, the secrecy of the last n-bit of the capacity might be enough even if $c > n$. With n bit of secret in the state, the output of the next permutation call is kept random and hidden and the ephemeral encryption process with re-keying can be safely iterated. To make a comparison with Triplex, only the n-bit k in the state (h, k) needs to be secret to iterate the process block by block. Obviously, neither the rate of sponge-based modes nor h of Triplex of challenge ciphertexts should leak as otherwise distinghuishing becomes easy. But given the ciphertext returned as an answer to an encryption query for a chosen message, it is easy to deduce these values.