The Motte-and-Bailey Framework for Leakage-Resilient Accordion Modes

Featuring Qaitbay and Alicante

Mario Marhuenda Beltrán[1,2]
and Mustafa Khairallah[2]

[1] Radboud University, Nijmegen, Netherlands
mmarhuenda@cs.ru.nl
[2] Umeå University, Umeå, Sweden
mustafa.khairallah@umu.se

Abstract. Accordion modes have experienced a surge in popularity, partially motivated by the recent NIST Accordion modes project. None of the existing practical constructions is leakage-resilient by default. In this work, we design a leakage-resilient Accordion mode. We start by presenting a generic analysis of the Encode-then-Encipher (EtE) framework in the leakage-resilient setting, assuming the enciphering is a leakage-resilient STPRP (STPRPℓ2). We show that the resulting security, while strong, suffers from some limitations. Next, we introduce Motte-and-Bailey, a general framework for building leakage-resilient accordion modes, in the spirit of the PIV construction. Motte-and-Bailey, or MaB for short, is a leveled construction, requiring light assumptions on most of its components to guarantee good STPRPℓ2, CIML2 and CCAMℓ2 security. In particular, we require two fully protected calls to a TBC, a collision-resistant hash function (with unbounded or light leakage), and an ideal leakage-resilient PRG, secure against single-trace attacks. Additionally, we present particular instantiations, Qaitbay and Alicante. In Qaitbay the PRG and the hash function are replaced by the Sponge function, while an independent TBC is used for the leak-free calls. Alicante makes use of an ideal cipher, and uses the MDPH hash function and the 2PRG construction, while the leak-free calls are implemented using independent calls to the ideal cipher. We also give three flavours of how to instantiate the TBC inside Qaitbay.

Keywords: Accordion mode · AEAD · Leakage · Confidentiality · Integrity · EtE

1 Introduction

Over the years, symmetric-key encryption has seen multiple shifts in focus; from stream ciphers and Block Ciphers (BCs), through plain encryption modes, to Authenticated Encryption with Associated Data (AEAD). These shifts have been motivated by attempts to build more robust notions, while keeping the

encryption algorithms practical. Over the past two decades, intensive research on AEAD schemes [7] has been performed, culminating in the CAESAR competition[3] and the NIST Lightweight Cryptography project[4]. However, AEAD schemes have significant limitations. They rely on adding redundancy in the ciphertext to provide security. This leads to practical challenges. In disk/memory encryption, it is hard to extend the plaintext size, which makes using AEAD infeasible. Similarly, applications that can extend the plaintext size may have a hard time communicating these additional bits; the communication cost of sending the bits may be much higher than the computational cost. Because of this, proposals for short-tag or variable-length-tag AEAD schemes [32, 45, 60] have been considered, assuming that for many applications an occasional random forgery can be tolerated, as long as confidentiality is maintained. Unfortunately, it was shown recently that, for most efficient/cheap AEAD schemes, the relation between confidentiality and integrity is coupled; after the first carefully constructed forgery, confidentiality is lost [46, 48].

These limitations are among the motivations to study a different type of symmetric-key encryption: Tweakable Wide Block Ciphers (TWBCs). A TWBC is a keyed function that takes two inputs of arbitrary length. The first input is the tweak and the second input is the plaintext. The tweak is used to select a family of permutations over strings of length l bits, where l is a variable between two parameters $l_{\min} \leq l \leq l_{\max}$, such that for every plaintext of length l, the TWBC produces a ciphertext also of length l. The expectation is that each such permutation is indistinguishable from a random permutation. Such schemes ensure that any change in any bit of the tweak or plaintext affects the whole ciphertext, and any change in the tweak or ciphertext during decryption affects the whole plaintext. They provide security against chosen ciphertext adversaries, even if there are no integrity guarantees. Additionally, they can be easily adapted to AEAD using known techniques, *e.g.* the Encode-then-Encipher (EtE) framework [8]. The properties of TWBCs have motivated the National Institute for Standards and Technology (NIST), USA, to start a new project with the goal of proposing a new standard. They renamed TWBCs as Accordion modes. This effort shows the pressing need for efficient and secure TWBCs [17]. NIST has expressed its plans to standardize a design based on HCTR2 [21], which does not inherently address leakage resilience. In fact, the problem of designing a secure Accordion mode in the presence of non-trivial[5] leakage has been understudied and was considered to be challenging.

[3] https://competitions.cr.yp.to/caesar-submissions.html (2016)

[4] https://csrc.nist.gov/projects/lightweight-cryptography

[5] By trivial leakage, we mean leakage corresponding to a fully-protected implementation: the adversary cannot infer any information about the function, the secret or the input, by observing the leakage. Such leakage can be modelled by a function that always outputs $\perp$, and in practice corresponds to implementations that are heavily and uniformly protected, *e.g.* every subfunction is protected using higher-order masking.

Leakage-resilient symmetric cryptography Side-channel analysis is an essential aspect of modern applied cryptography. Any computation executed on a physical device is inherently susceptible to unintentional information leakage through observable channels, such as timing variations, power consumption, or electromagnetic emanations. Power-based side-channel attacks are typically categorized into two main classes: Simple Power Analysis (SPA) and Differential Power Analysis (DPA). SPA attempts to extract information from a small number of traces, possibly 1, whereas DPA exploits statistical relations across a large number of traces collected under varying inputs. Here, the number of traces denotes the number of distinct inputs used in the acquisition process. However, repeated measurements of identical inputs are often employed to suppress noise. Analogous classifications apply to other forms of side channels. Countermeasures against such attacks frequently rely on physical protections, among which masking represents a central approach. The robustness of such schemes is characterized by their security order, with higher-order masking offering increased resistance. Nevertheless, the associated implementation cost grows substantially with the masking order. A different, more recent approach, is to design the modes to be leakage-resilient. Early works in this area in the context of symmetric key cryptography started in 2008 [30], and have seen significant progress over the years both on the theoretical (cf. [28, 31]) and practical (cf. [24, 27, 43, 52, 55, 59, 61, 64, 65]) sides. Most of the efforts have been dedicated to designing leakage-resilient AEAD, with the efforts converging towards what is now known as levelled implementations. This approach is explored in depth by Bellizia *et al.*, summarizing the state of the art at the time [10]. The main approach is that the mode is divided into components. A constant number of fixed-size components come with strong assumptions, the most popular of which is being leak-free. These assumptions typically mean that the implementation of these components is protected against side-channel with expensive countermeasures, such as higher-order masking. The rest of the components come with weaker assumptions, such as having unbounded leakage (they leak all their internal values), or hard-to-invert leakage as long as each secret input is never repeated, which translates to cheaper countermeasures against SPA. This approach has seen significant success in the context of AEAD, due to the observation that masking countermeasures are typically extremely expensive, adding several orders of magnitude compared to unprotected implementations. By ensuring that these countermeasures are used only in a handful of components, the overall implementation is significantly cheaper than if we mask the whole mode. Several AEAD designs have been proposed in this direction from (Tweakable) Block Ciphers (TBCs) [11, 24, 35, 37, 38, 49, 52, 61, 62]. Similar approaches have also been applied to permutation-based cryptography [9, 25].

Unlike the black-box setting, there is no unified AEAD security notion that captures both integrity and confidentiality in the presence of leakage. Bellizia *et al.* [10] showed a classification of leakage-resilient AEAD notions in the presence of leakage. For integrity, the strongest notion is Ciphertext Integrity with nonce Misuse and encryption and decryption Leakage (CIML2). A weaker notion is Ciphertext Integrity with nonce Misuse and encryption Leakage (CIML1). For

confidentiality, Bellizia *et al.* described a framework for defining security notions. A notion is described using the notation:

- C: Chosen.
- PA/CA: Plaintext or Ciphertext Adversaries.
- m/M: nonce misuse resilience or resistance. Resilience means that the adversary is only allowed to use fresh nonces for challenge queries, while resistance means the adversary is allowed to repeat nonces even in challenge queries.
- L/ℓ: with or without challenge leakage.
- 1/2: the leakage is only from the encryption oracle or from both encryption and decryption oracles.

In this framework, CCAML2 is the strongest possible confidentiality notion. However, it is a completely deterministic notion. For instance, the adversary can fix the nonce to a constant in all queries. Thus, this notion is believed to be infeasible with non-trivial leakage and reasonable assumptions. CCAmL2 and CCAMℓ2 are tied as the second strongest notions. The strongest schemes discussed in the previous paragraph achieve either CCAmL2 $\wedge$ CIML2 or CCAMℓ2 $\wedge$ CIML2. This leaves an open problem: *can we design a scheme that achieves CCAmL2 $\wedge$ CCAMℓ2 $\wedge$ CIML2 in the framework of Bellizia et al.?* We refer to this as problem C.

When it comes to TWBCs, the situation is less clear. Two results have been published on the leakage resilience of PRPs. The first is by Dodis and Pietrzak [28], who showed that a Feistel cipher built from leakage-resilient round functions requires a super-logarithmic number of rounds to achieve leakage-resilient PRP security. However, this result is shown under two strong adversarial assumptions:

- The adversary is adaptive, not only in choosing the queries, but also the leakage functions.
- The PRFs used in the round functions are leakage-resilient, but the remainder of the Feistel network has unbounded leakage. In other words, all the *"connections"* and *"additions"* in the network leak their inputs and outputs.

The second result is due to Faust *et al.* [31], where they showed that against non-adaptive adversaries, three rounds are sufficient to reach PRP security (which is weaker than STPRP security). These two results are of theoretical nature, and pre-date a lot of the recent positive results on leakage-resilient AEAD. Defining a leakage-resilient block cipher is not straightforward. A black-box block cipher (without leakage) is expected to behave as a random permutation. However, since a random permutation is a theoretical primitive, it does not experience leakage. Besides, Berti *et al.* [11] discussed how deterministic constructions cannot provide indistinguishability-style notions in the presence of non-trivial leakage, under some reasonable assumptions. Dodis and Pietrzak [28] presented a weakened notion for leakage-resilient PRPs. The queries are split into two categories, challenge and non-challenge queries. Non-challenge queries always use the real construction with non-trivial leakage, while challenge queries use either the

real construction without leakage or a random permutation, depending on which world the challenger is in. This notion captures the best scenario we can hope for without introducing unrealistic assumptions. We will refer to it as STPRPℓ2 and it is the central security notion of the paper. Whether STPRPℓ2 security is achievable using an efficient, levelled Accordion mode is another open problem, which we shall refer to as problem B.

Finally, assuming the existence of such an efficient STPRPℓ2-secure mode, it is not clear if the security properties of the EtE framework carry over to the leakage scenario. Studying the relations (or lack thereof) between the notions of Bellizia *et al.* and STPRPℓ2 is another open problem that we shall refer to as problem A. This paper resolves all three problems.

Non-exhaustive exposition of techniques for building Accordion modes The EtE framework was proposed by Bellare and Rogaway [8], assuming the existence of enciphering schemes. There have been many candidates for TWBCs over the years. The early works on building Pseudo-Random Permutations (PRPs) from Pseudo-Random Functions (PRFs) [54] can be seen as a basis for building TWBCs, and have indeed been used for such purpose [4]. Another early approach is Encrypt-Mix-Encrypt (EME) [41, 42]. A far more popular approach is the HCTR paradigm. This approach was first proposed by Wang *et al.* [66], and later popularized by the designs of Adiantum [20] and HCTR2 [21]. Nevertheless, HCTR was shown to have a security flaw where the hash function used was not almost-XOR-universal as required [50], and the security proof of [16] suffered from a bug [21]. Adiantum and HCTR2 fix these issues. THCTR [29] was an attempt to enhance security, but the security analysis had a flaw that limited its gain [3, 47]. HCTR+ [23] is a more recent candidate that avoids the flaws in THCTR. Cascaded HCTR2 [18] is a very recent candidate that improves the security of HCTR2 by applying the construction twice, with $1.5 - 2\times$ the cost. A different approach based on similar primitives has been proposed by Gunsing *et al.* [33], and its efficiency was demonstrated by Dobraunig *et al.* [26].

Alternatively, the Protected IV (PIV) approach was proposed by Shrimpton and Terashima [63] as a modular approach for building TWBCs from smaller primitives. It has the advantage of being easy to understand and analyze, but early instances have proven more costly compared to HCTR variants. Recently, GEM [15] has been proposed. It is somewhat inspired by PIV, but not in a generic sense.

We note that the NIST has announced its plans to adopt variants of HCTR2 as the new standard. HCTR2 does not provide leakage resilience except if the implementation is heavily protected. This further motivates the necessity of our work. To the best of our knowledge, no scheme in the literature achieves leakage resilience as a TWBC.

1.1 Contributions

Our main contributions of the paper are answering problems A, B, and C:

– We provide both positive and negative results for problem A. We show that STPRPℓ2 implies the following notions with a tight reduction: CIML1 (without decryption leakage) (Theorem 1), CCAMℓ2 (Theorem 4), and r-CIML2 (Theorem 2) (a new notion of security against reforgeability, where the adversary is allowed to observe the decryption leakage up to r successful forgeries, then they cannot observe the leakage any more and have to produce an additional forgery). We also show that STPRPℓ2 implies CIML2, but with a reduction that incurs a linear security drop in the number of forgery attempts (Theorem 3). This drop in security is likely an artefact of the reduction that treats the Accordion mode as a black box, and we show in the solution to problem B that the drop can be improved for specific schemes. On the negative side, we show a separation between STPRPℓ2 and CCAmL2. Our solution to problem B exhibits a concrete construction that achieves STPRPℓ2 but admits a CCAmL2 distinguisher (Theorem 7). These results are given in Section 4. However, one can design an Accordion mode that achieves both STPRPℓ2 and CCAmL2 at an additional cost. We note that CIML2 also requires decryption leakage, but we are able to find a workaround in the security reduction that works for integrity but not for confidentiality. In (Theorem 8) we show that with an additional simulatable leakage assumption (c.f. Definition 1), we can show that STPRPℓ2 implies CCAmL2, but we acknowledge that this additional assumption is strong.
– We construct the `Motte-and-Bailey` (`MaB`) framework, depicted in Figure 1, which resolves problem B. `MaB` achieves STPRPℓ2 security by composing four main components: a leak-free TBC, a collision-resistant hash function, a leakage-resilient PRG, and an XOR addition. The security is shown under the assumptions that the underlying TBC is leak-free, the hash function and XOR have unbounded leakage (they leak all their internal values), the PRG exhibits only hard-to-invert leakage and maintains its pseudo-randomness for fresh seeds. The tweak in `MaB` consists of two parts, N, a fixed-length mandatory tweak, and A, a variable-length, potentially empty, tweak. This maps nicely to AEAD, where N can be used for the nonce and A can be used for the associated data. We also show that with a specific encoding scheme for the EtE framework, which allows early abort, the CIML2 security drop can be reduced compared to the generic case. This plain `MaB` is not CCAmL2-secure, demonstrating that the separation discussed earlier is genuine. `MaB` presents a positive solution to problem B. These results are described in Section 5.
– We define a strengthened variant of `MaB`. This strengthened variant requires an additional call to the leak-free TBC per query, and it inherits all the positive results from the plain `MaB` variant. However, with stronger assumptions on the hash function, XOR, and PRG, we can show that it also achieves CCAmL2, succeeding where the plain `MaB` fails. In particular, the plain `MaB` is designed under the assumption that all the inputs to the hash function are public, and that the hash function has unbounded leakage. This means that the adversary can detect when two related inputs are being processed by the hash function. We introduce a specific simulatable leakage assumption on

H and X. The analysis of this solution is currently informal as it requires further future work to assess both the theoretical and practical feasibility of the additional assumption.

– *Instantiations: `Qaitbay` and `Alicante`:* In the remainder of the paper, we define two particular instances of `MaB`: `Qaitbay` and `Alicante`. The former is a realization of `MaB` based on the Sponge function and a tweakable block cipher, while the latter is a realization based on an ideal cipher. We also discuss how to instantiate both of them using existing primitives, what concrete security we get, and the limitations to the current instantiations. This shows `MaB` is practical with current primitives, but it also opens the door for designing tailored primitives as future work.

2 Preliminaries

Notation We denote sets by using calligraphic letters: $\mathcal{A}$. Additionally, we will denote the cardinality of said set by $|\mathcal{A}|$. Moreover, if $\mathcal{S}$ is a finite set, we write $x \xleftarrow{\$} \mathcal{S}$ to denote that x is sampled uniformly at random from $\mathcal{S}$. We use $\emptyset$ to denote the empty set, and, abusing notation, we also use $\emptyset$ to denote the empty string. Given an algorithm $\mathcal{A}$, we denote $\mathcal{A}$ outputting a value x by $\mathcal{A} \Rightarrow x$ or $x \leftarrow \mathcal{A}$. Additionally, when defining a new variable we will use the symbol $\stackrel{\text{def}}{=}$. Let $n \in \mathbb{N}$ such that $n \geq 1$. We will abbreviate the set $\{1, \ldots, n\}$ as $[n]$. Let X be a variable-length bit string, then $|X|$ is its bit length, represented as a 64-bit string. For $n \in \mathbb{N}$, $\xleftarrow{n}$ splits an arbitrary-length bit string into n-bit blocks, with the last block potentially shorter than n bits, and $\xleftarrow{n,*}$ splits an arbitrary-length bit string of length at least n bits into an n-bit block and an arbitrary-length bit string that contains the remainder of the bits. Given a function $f : \mathcal{A} \times \mathcal{B} \mapsto \mathcal{C}$, and a fixed $a \in \mathcal{A}$, we denote the partial function $g : \mathcal{B} \mapsto \mathcal{C}$ defined by $g(b) \stackrel{\text{def}}{=} f(a, b)$ for all $b \in \mathcal{B}$ by simply $f(a, \cdot)$. Finally, we denote by $\mathsf{tperm}(\mathcal{T}, \mathcal{M})$ the set of tweakable uniformly random permutations (TURPs), that is, functions from $\mathcal{T} \times \mathcal{M} \mapsto \mathcal{M}$, so that, for any $t \in \mathcal{T}$, $R(t, \cdot)$ is an independent uniformly random permutation.

In our constructions, we use a single-input collision-resistant hash function to absorb three different strings. For that, we require an injective padding function ipad. In the case of `MaB`, we leave the exact definition of this function to the application, under the condition that it is impossible to find two sets of inputs that lead to the same string after padding, i.e., the padding must be injective, as done in [34]. Nevertheless, for our constructions, `Qaitbay` and `Alicante`, padding is explicitly defined and used when needed (cf. Algorithms 2 to 4).

Collision Resistance of Hash Functions: Let $H : \mathcal{K}_h \times \mathcal{M} \to \mathcal{X}$ be a hash function. A hash function H is called $(\epsilon_{\mathsf{cr}}, t)$-collision-resistant if for every t-bounded adversary $\mathcal{A}$ (*i.e.* running in time at most t), the probability that $\mathcal{A}(s)$ outputs a pair of distinct inputs $(M_1, M_2) \in \mathcal{M}^2$, such that $H_s(M_1) = H_s(M_2)$

and $M_1 \neq M_2$, is bounded by ϵ_{cr}, with $s \xleftarrow{\$} \mathcal{K}_h$ picked uniformly at random:

$$\Pr[s \xleftarrow{\$} \mathcal{K}_h, (M_1, M_2) \leftarrow \mathcal{A}(s) : M_1 \neq M_2, H_s(M_1) = H_s(M_2)] \leq \epsilon_{cr} .$$

The hash key is revealed to the adversary at the beginning of the security game. We omit the hash key from the notation when understood from context. We emphasize that the hash key is public, and in practice can be represented by a simple fixed-length constant IV. This is a standard definition for collision resistance of public hash functions and has been particularly useful in leakage-resilient constructions [12].

Authenticated Encryption with Associated Data (AEAD): A (nonce-based) AEAD scheme is a pair of functions $\Pi = (\mathsf{E}, \mathsf{D})$. $\mathsf{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \{0,1\}^* \to \{0,1\}^*$ is the encryption function that takes secret key $K \in \mathcal{K}$, nonce $N \in \mathcal{N}$, associated data $A \in \mathcal{A}$ and plaintext $M \in \{0,1\}^*$ and returns a ciphertext $C \in \{0,1\}^*$. The decryption function $\mathsf{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ takes as input the secret key $K \in \mathcal{K}$, nonce $N \in \mathcal{N}$, associated data $A \in \mathcal{A}$ and ciphertext $C \in \{0,1\}^*$ and returns either a plaintext $M \in \{0,1\}^*$ or the symbol $\perp$.

Nonce Misuse Resistance and Resilience: Given a nonce-based AEAD scheme Π and let's fix a security game, for example `CCA`, then we consider two variants:

- `CCA` with *nonce misuse resistance*: In this game, the adversary is allowed to reuse the nonce without limit.
- `CCA` with *nonce misuse resilience*: In this game, the adversary is allowed to reuse the nonce *during non-challenge queries*. However, during challenge queries, it may not use a previously used nonce. In practice, this corresponds to a scheme where the adversary is allowed to reuse the nonce during an "offline" phase, but is constrained during a posterior "online" phase. However, in our model, we grant more power to the adversary by allowing it to interleave "offline" and "online" queries.

Clearly, `CCA` may be replaced *mutatis mutandis* by any security notion that encompasses challenge.

Sparse Encoding. Let $\mathsf{S} = (\mathsf{encode} : \{0,1\}^* \to \{0,1\}^*, \mathsf{decode} : \{0,1\}^* \to \{0,1\}^* \cup \{\perp\})$ be an encoding scheme, such that $\mathsf{decode}(\mathsf{encode}(M)) = M$ and encode preserves length equality; for any pair of messages (M_0, M_1) such that $|M_0| = |M_1|$, we have $|\mathsf{encode}(M_0)| = |\mathsf{encode}(M_1)|$. Furthermore, we say that S is δ-dense for a certain real positive δ if the following holds:

$$\Pr\left[\bar{M} \xleftarrow{\$} \{0,1\}^n : \mathsf{decode}(\bar{M}) \neq \perp\right] \leq \delta .$$

In other words, for every possible codeword length, only a δ-fraction of all codewords of that length are valid codewords, i.e., decode to valid messages, while

the rest decode to $\perp$ (invalid). The most useful and natural example of a δ-dense encoding scheme is the following:

$$\mathsf{encode}(M) \Rightarrow 0^t \parallel M\,; \quad \mathsf{decode}(C) \Rightarrow \begin{cases} M, & \text{if } C = 0^t \parallel M\,, \\ \perp, & \text{otherwise}\,, \end{cases}$$

which is clearly a 2^{-t}-dense encoding scheme.

Encode-then-Encipher: The Encode-then-Encipher (EtE) is an AEAD construction introduced by Bellare and Rogaway [8]. It consists of an encoding scheme $\mathsf{S} = (\mathsf{encode}, \mathsf{decode})$ and an enciphering scheme Π. We denote its inverse by Π^{-1}, that is:

$$\Pi_K^{-1}(T, \Pi_K(T, M)) = M\,, \quad \forall M \in \{0,1\}^*\,.$$

EtE takes a nonce, N, associated data, A and a plaintext M as input, then EtE is defined as:

$$\mathsf{EtE}(N, A, M) = \Pi_K(N \| A, \mathsf{encode}(M))$$
$$\mathsf{EtE}^{-1}(N, A, C) = \mathsf{decode}(\Pi_K^{-1}(N \| A, C))$$

In order to guarantee AEAD security of EtE, we usually assume Π to be STPRP-secure. We will expand on this in Section 4. We remark that even though EtE takes a nonce, this is mostly for convention, and it is in fact nonce misuse resistant. Furthermore, since its introduction, the EtE construction has been a valuable tool in building robust and misuse-resistant AEAD.

Notably, EtE is a tagless AEAD construction: integrity is provided by the encoding scheme. From now on, S will be a sparse encoding scheme.

3 Security Notions for Leakage Resilience

In this section, we summarize the security notions we use in our study.

Strong Tweakable Pseudo-Random Permutation without challenge leakage ($\mathsf{STPRP}\ell 2$): This notion was introduced by Dodis and Pietrzak [28] and a non-adaptive variant was introduced by Faust *et al.* [31]. Let $\Pi : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ be a family of length-preserving keyed permutations with key space $\mathcal{K}$ and tweak space $\mathcal{T}$. Furthermore, let L be an associated leakage function.[6] Then, $\Pi_{\mathsf{L}}(K, T, \cdot)$ and $\Pi_{\mathsf{L}}^{-1}(K, T, \cdot)$ are the corresponding non-challenge leaky oracles, that is, the oracles that respond with both the output of Π (respectively Π^{-1}) and the leakage L. Let $\mathcal{R}$ be the space of length-preserving tweakable permutations, with domain $\mathcal{T} \times \mathcal{M}$ and range $\mathcal{M}$. Finally, let $\mathcal{A}$ be an adversary with access to four

[6] We keep L vague at this point, but we will make it explicit when needed.

oracles that returns a single bit. Then, the STPRPℓ2 advantage of $\mathcal{A}$ is defined as

$$\mathrm{Adv}^{\mathsf{STPRP}\ell 2}_{\Pi,\mathsf{L}}(\mathcal{A}) \stackrel{\text{def}}{=} \left|\Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{W_{\mathsf{L}},W_{\mathsf{R}}} \Rightarrow 1\right] - \Pr\left[K \xleftarrow{\$} \mathcal{K}, R \xleftarrow{\$} \mathcal{R} : \mathcal{A}^{W_{\mathsf{L}},W_{\mathsf{I}}} \Rightarrow 1\right]\right| . \tag{1}$$

where $W_{\mathsf{L}} = (\Pi_{\mathsf{L}}, {\Pi_{\mathsf{L}}}^{-1})$, $W_{\mathsf{I}} = (R, R^{-1})$ and $W_{\mathsf{R}} = (\Pi, \Pi^{-1})$ and $\mathcal{R}$ is the set of all length-preserving tweakable permutations of the same tweak and plaintext spaces. $\mathcal{A}$ is not allowed to forward queries from the challenge oracle, W_{R} or W_{I}, to the non-challenge oracle W_{L}, and vice versa.

We say that Π is $(q_e, q_d, q_{ce}, q_{cd}, t)$-STPRP$\ell$2-secure, if the advantage is negligible for any adversary that makes at most q_e and q_d leaky encryption and decryption queries and at most q_{ce} and q_{cd} challenge encryption and decryption queries and runs in time at most t. Π may be defined with respect to one or more ideal primitives. In that case, the adversary is granted access to these primitives and the number of queries made to each primitive is part of the adversarial resources.

Ciphertext Integrity with Leakage (CIML2). In the CIML2 game the goal of the adversary is to produce a valid fresh ciphertext, while given access to the leakage both in encryption and decryption. The adversary can repeat nonces even in challenge queries. The adversary $\mathcal{A}$ queries E_{L} and D_{L} with only one restriction: it is not allowed to query $\mathsf{D}_{\mathsf{L}}(K, N, A, C)$ if it has previously queried $\mathsf{E}_{\mathsf{L}}(K, N, A, M)$ for a message $M \in \mathcal{M}$ and received C. $\mathcal{A}$ wins if any query to D_{L} returns something other than $\perp$. We define the CIML2 advantage of $\mathcal{A}$ against an AEAD scheme $\Pi = (\mathsf{E}_{\mathsf{L}}, \mathsf{D}_{\mathsf{L}})$ as

$$\mathrm{Adv}^{\mathsf{CIML2}}_{\Pi,\mathsf{L}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr\left[\mathcal{A}^{\mathsf{E}_{\mathsf{L}},\mathsf{D}_{\mathsf{L}}} \text{ wins}\right] .$$

If we remove the leakag, that is if the adversary is interacting with (E, D) instead of $(\mathsf{E}_{\mathsf{L}}, \mathsf{D}_{\mathsf{L}})$, this is the classical forgery game. If we remove the leakage from the oracle, this is the classical forgery game. If the adversary is interacting with $(\mathsf{E}_{\mathsf{L}}, \mathsf{D})$, this is the CIML1 game.

Ciphertext Integrity with reforgeability and Leakage (r-CIML2). We also define the reforgeability game r-CIML2, which is a stateful game. The r-CIML2 games differs from the CIML2 game in that it is stateful, the adversary interacts with $(\mathsf{E}_{\mathsf{L}}, \mathsf{D}_{\mathsf{L}})$ and the challenger counts the number of queries to D_{L} that are succesful, i.e., return values other than $\perp$. When this count reaches a parameter r, the challenger replaces D_{L} with D. The adversary $\mathcal{A}$ wins if $r+1$ decryption queries return values other than $\perp$. In other words, the adversary must produce *one more forgery* after the decryption leakage is removed.

Chosen Ciphertext Attack with Leakage (CCA). Since CCAML2 is widely believed to be infeasible [10], except in trivial settings, we target more realistic alternatives. In particular, the notions of CCAMℓ2 and CCAmL2 are achievable [38].

The combination of CCAMℓ2 and CIML2 notions is similar[7] to the notion of [6], we do not cover them since they are implied by the notions of this work

CCAmL2: Here all oracles are leaky, but challenge queries may only use fresh nonces. The game is defined as follows:

1. At the beginning of the game, the challenger selects a random bit $b \leftarrow\$ \{0,1\}$.
2. For every non-challenge query made by the adversary, the challenger queries either E_L or D_L, and returns the answer with the corresponding leakage.
3. For each challenge query, the adversary sends N, A, M_0, M_1 with $|M_0| = |M_1|$ and the challenger returns $\mathsf{E}_\mathsf{L}(K, N, A, M_b)$, with the restriction that N must have not appeared in any query previous to E or E_L.

Formally, we define the advantage of an adversary, $\mathcal{A}$ in this game as

$$\mathrm{Adv}^{\mathsf{CCAmL2}}_{\Pi,\mathsf{L}}(\mathcal{A}) \stackrel{\text{def}}{=} |\Pr[\mathcal{A} \Rightarrow 1 | b = 1] - \Pr[\mathcal{A} \Rightarrow 1 | b = 0]| .$$

The adversary does not repeat queries, does not forward the outcomes of the encryption queries to its decryption oracle, and does not use any of the non-challenge queries in its challenge queries.

CCAMℓ2: In this games, challenge oracles are *not* leaky, but nonce reuse is allowed at any point. The game is defined as follows:

1. At the beginning of the game, the challenger selects a random bit $b \leftarrow\$ \{0,1\}$.
2. For every non-challenge query made by the adversary, the challenger queries either E_L or D_L, and returns the answer with the corresponding leakage.
3. For each challenge query, the adversary sends N, A, M_0, M_1 with $|M_0| = |M_1|$ and the challenger returns $\mathsf{E}(K, N, A, M_b)$.

Formally, we define the advantage of an adversary, $\mathcal{A}$ in this game as

$$\mathrm{Adv}^{\mathsf{CCAM}\ell 2}_{\Pi,\mathsf{L}}(\mathcal{A}) \stackrel{\text{def}}{=} |\Pr[\mathcal{A} \Rightarrow 1 | b = 1] - \Pr[\mathcal{A} \Rightarrow 1 | b = 0]| .$$

Leakage-Resilient Pseudo-Random Generator (PRGℓ1, PRGL1, and GuessL1): Let $X : \mathcal{S} \times \mathcal{N} \times \mathbb{Z}^+ \to \{0,1\}^*$ be a function that takes a random seed $S \in \mathcal{S}$, a public parameter $N \in \mathcal{N}$ and a positive integer l and returns a bit-string of length l. The leakage function L is chosen non-adaptively at the beginning of the game. Let X_L be the implementation of X that also returns non-trivial leakage. The PRGℓ1 game is defined as follows:

1. The challenger samples a random bit $w \leftarrow\$ \{0,1\}$ at the beginning of the game.
2. An adversary $\mathcal{A}$ makes a query $(N, l, b) \in \mathcal{N} \times \mathbb{Z}^+ \times \{0,1\}$.
3. The challenger samples a uniformly random seed $S \leftarrow\$ \mathcal{S}$.
4. If $b = 0$, the challenger returns $X_\mathsf{L}(S, N, l)$.
5. If $b = 1$ and $w = 0$, the challenger returns $X(S, N, l)$.

[7] We do not give it explicitely, but they are equivalent by a simple reduction.

6. If $b = 1$ and $w = 0$, the challenger samples a uniformly random string $Z \leftarrow\!\!\$\ \{0,1\}^l$ and returns Z.

The adversary can also make q_p queries to X with a chosen seed, we call this oracle X_{P}. The adversary makes q_l queries with $b = 0$ and q_e queries with $b = 1$. The PRGℓ1 advantage of $\mathcal{A}$ is defined as

$$\mathrm{Adv}^{\mathsf{PRG}\ell\mathsf{1}}_{X,\mathsf{L}}(\mathcal{A}) \stackrel{\text{def}}{=} |\Pr\left[\mathcal{A}^{X_{\mathsf{P}},X_{\mathsf{L}},X} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{X_{\mathsf{P}},X_{\mathsf{L}},\$} \Rightarrow 1\right]|,$$

where \$ is the oracle that returns uniformly random bit strings. The leakage-resistant variant of this notion is defined by the interaction of the adversary with $(X_{\mathsf{P}}, X_{\mathsf{L}})$ or $(X_{\mathsf{P}}, \$_{\mathsf{L}})$, where \$ outputs random strings but with the correct leakage of X. The PRGL1 advantage of $\mathcal{A}$ is defined as

$$\mathrm{Adv}^{\mathsf{PRGL1}}_{X,\mathsf{L}}(\mathcal{A}) \stackrel{\text{def}}{=} \left|\Pr\left[\mathcal{A}^{X_{\mathsf{P}},X_{\mathsf{L}}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{X_{\mathsf{P}},\$_{\mathsf{L}}} \Rightarrow 1\right]\right|,$$

Clearly, PRGL1 implies PRGℓ1, as the adversary can simply ignore the leakage from some queries. A PRGL1 adversary can simulate a PRGℓ1 game using only primitive queries for the PRGℓ1 leakage queries and use its second oracle for the challenge queries, ignoring the leakages. We also define the GuessL1 game as follows:

1. An adversary $\mathcal{B}$ makes a query $(N, l) \in \mathcal{N} \times \mathbb{Z}^+$.
2. The challenger samples a uniformly random seed $S \leftarrow\!\!\$\ \mathcal{S}$.
3. The challenger returns $X_{\mathsf{L}}(S, N, l)$.

The adversary can also make q_p queries to X with a chosen seed, we call this oracle X_{P}. At the end of the game, $\mathcal{B}$ returns a set $\mathcal{G}$ of size q_g. $\mathcal{B}$ wins if $S \in \mathcal{G}$. We define $\mathcal{B}$'s advantage as

$$\mathrm{Adv}^{\mathsf{GuessL1}[q_g]}_{X,\mathsf{L}}(\mathcal{B}) \stackrel{\text{def}}{=} \Pr[\mathcal{G} \leftarrow \mathcal{B}^{X_{\mathsf{P}},X_{\mathsf{L}}} : S \in \mathcal{G}].$$

We note that this notion can be trivially broken with a brute-force attack, where $\mathcal{B}$ observes a collision between S and a query to X_{P}. This occurs with probability about $q_p/|\mathcal{S}|$. In the presence of leakage, this probability can be improved. We will refer to a function λ as hard-to-invert if both $\mathrm{Adv}^{\mathsf{PRGL1}}_{X}(\mathcal{A})$ and $\mathrm{Adv}^{\mathsf{GuessL1}[q_g]}_{X}(\mathcal{B})$ are negligible for any adversaries $\mathcal{A}$ and $\mathcal{B}$.

Key Unpredictability with two traces (2-up): Berti *et al.* [11] introduced the 2-up security notion as a notion for security against 2-trace attacks on ideal ciphers. GuessL1 can be seen as a generalization of 2-up. The game works almost identically to the GuessL1 game except the adversary is given two queries to the ideal cipher with the same secret key S. Then, the adversary wins if it can guess the key. Against this game, the advantage of the adversary is defined as

$$\mathrm{Adv}^{\mathsf{2\text{-}up}[q_g]}_{X}(\mathcal{B}) \stackrel{\text{def}}{=} \Pr[\mathcal{G} \leftarrow \mathcal{B}^{\mathsf{IC}_{\mathsf{P}},\mathsf{IC}_{\mathsf{L}}} : S \in \mathcal{G}]\,.$$

3.1 Adversarial Resources

Whenever we consider an adversary $\mathcal{A}$, said adversary is constrained in the following way: we denote by $\Xi_{\mathcal{A}} = (t_{\mathcal{A}}, \overline{Q}_{\mathcal{A}})$ its resource tuple, where $t_{\mathcal{A}}$ is the total time complexity and $Q_{\mathcal{A}}$ is a tuple encoding how many queries $\mathcal{A}$ can make to its corresponding oracle. We will drop the subindex when it can be inferred from the context. In the case where there are challenge oracles, we define

$$\overline{Q}_{\mathcal{A}} = (q_p, q_e, q_d, q_{ce}, q_{cd})\,, \tag{2}$$

where q_p is the number of primitive queries (see next paragraph), q_e, q_d are respectively the number of queries to the non-challenge encryption and decryption oracles. Respectively, q_{ce}, q_{cd} are the number of queries to the challenge encryption and decryption oracles. Additionally, we denote $q = q_e + q_{ce} + q_d + q_{cd}$ and $Q = q + q_p$. Furthermore, in the case where there is no decryption oracle but a forgery oracle (for example $\texttt{CIML2}$), we replace q_d by q_f in (2).

In the case where $\mathcal{A}$ sits in a game where there are no challenge oracles, then we simply omit q_{ce}, q_{cd}.

Access to inner primitives : In the remainder of the paper, we will further assume that H is a $(\epsilon_{\mathsf{CR}}, t')$-collision-resistant hash function, that $\tilde{E}$ is a leak-free TBC, and X is a leakage-resilient random function. Queries to X are called offline queries and the total number of evaluations of H is denoted t_H. Additionally, $\tilde{E}$ and H are sampled independently of X.

Choice of leakage function : Furthermore, the adversary chooses a leakage function non-adaptively at the beginning of the game, in particular, all security notions in this section use the same leakage function.

4 Generic Relations: Building EtE from STPRPℓ2

In this section, we elevate EtE to the realm of leakage resilience. Instead of assuming the inner enciphering scheme, Π, is STPRP-secure, we now assume it is STPRPℓ2-secure. Then, we provide security reductions for EtE, from STPRPℓ2 security to CIML1 (Theorem 1), a relaxed reforgeability notion r-CIML2 (Theorem 2), CIML2 with linear security drop (Theorem 3), and CCAMℓ2 (Theorem 4). We close the section by discussing that STPRPℓ2 does not imply CCAmL2.

For the remainder of this section, for the sake of exposition, we will fix an EtE with a δ-dense encoding scheme, for a fixed positive real δ, and using as a primitive a variable-length TBC Π and we denote it by $\Psi = \mathsf{EtE}[\delta, \Pi]$.

Theorem 1. *Let $\mathcal{A}$ be a CIML1 adversary with resources as in Section 3.1. There exists an STPRPℓ2 adversary $\mathcal{B}$ with the same resources, such that*

$$\mathrm{Adv}^{\mathsf{CIML1}}_{\Psi,\mathsf{L}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{STPRP}\ell 2}_{\Pi,\mathsf{L}}(\mathcal{B}) + 2q_f\delta\,.$$

The proof of Theorem 1 is deferred to Appendix B.1. In a nutshell, it is a reduction where all the encryption queries are non-challenge queries, and all the forgery attempts are simulated using challenge queries. Should the forgery be successful, then $\mathcal{A}$ is very likely in the real world. For r-CIML2 (Theorem 2), we split $q_f = q_f^a + q_f^b$, where q_f^b is the number of forgery attempts made before the r^{th} successful forgery, and q_f^a the remainder forgery attempts.

Theorem 2. *Let $r \in \mathbb{Z}^+$. Let $\mathcal{A}$ be an r-CIML2 adversary with resources as above and Section 3.1. Then, there exists an STPRPℓ2 adversary $\mathcal{B}$ against Π making $q = q_e + q_f^b + q_f^a$ queries, where $q_e + q_f^b$ is the number of non-challenge queries and q_f^a is the number of challenge queries, such that*

$$\mathrm{Adv}^{r\text{-CIML2}}_{\Psi,\mathsf{L}}(\mathcal{A}) \leq \mathrm{Adv}^{\text{STPRP}\ell 2}_{\Pi,\mathsf{L}}(\mathcal{B}) + 2q_f^a\delta\,.$$

The proof of Theorem 2 is almost the same as the proof of Theorem 1, except $\mathcal{A}$ uses the non-challenge leaky decryption oracle at first, and keeps track of how many successful forgeries have been made. Once r successful forgeries have occurred, $\mathcal{A}$ switches to the challenge oracle. Furthermore, it is easy to see that 0-CIML2 is equivalent to CIML1. However, since there is no r for which r-CIML2 is equivalent to CIML2, and due to the fact that r has to be non-adaptively chosen, CIML2 is not implied by this theorem and requires further analysis. Clearly, if the number of successful forgeries is less than r, the adversary cannot win and the claim is trivial. However, Theorems 1 and 2 give strong evidence that STPRPℓ2 offers strong integrity guarantees for ciphertext integrity in a generic sense. On the other hand, the relation between STPRPℓ2 and CIML2 is more complicated. Theorem 3 shows that STPRPℓ2 implies CIML2 with a linear gap. We do not have a matching attack that exploits this linear gap, and it may be an artefact of the proof or a limitation of the model. Later, we will show that the gap between CIML2 and STPRPℓ2 can be closed for concrete instantiations (with additional assumptions). We also note that r-CIML2 is just a weaker version of the *challenge-then-leak* philosophy. The proof of Theorem 3 is deferred to Appendix B.2.

Theorem 3. *Let $\mathcal{A}$ be a CIML2 adversary as in Section 3.1 against Ψ. Then, there exists an STPRPℓ2 adversary $\mathcal{B}$ that makes at most q_e encryption queries and at most q_f decryption queries and runs in time $O(t)$, such that*

$$\mathrm{Adv}^{\text{CIML2}}_{\Psi,\mathsf{L}}(\mathcal{A}) \leq q_f \mathrm{Adv}^{\text{STPRP}\ell 2}_{\Pi,\mathsf{L}}(\mathcal{B}) + q_f\delta.$$

Proof Sketch. Because STPRPℓ2 does not allow challenge leakage, and the leaky oracles are the same in both the real and ideal worlds, an STPRPℓ2 adversary can only win by querying its leak-free oracles. However, any CIML2 adversary is expecting exclusive access to leaky oracles. If we try to make a security reduction where a STPRPℓ2 adversary simulates the CIML2 games honestly, it can never win its own game, as it will never query its challenge oracles. Thus, we design the reduction such that the STPRPℓ2 adversary *cheats* by selecting one forgery

attempt, and feeding it to its challenge oracle. The CIML2 adversary is undisturbed up to the point of cheating, and if that forgery attempt was successful, it would still be successful in this simulation. A successful forgery at this particular point allows the STPRPℓ2 adversary to win its own game. However, we cannot say anything about the behaviour of the CIML2 adversary after this point, while everything that preceded this point does not really help the STPRPℓ2 adversary. The reduction hinges on the STPRPℓ2 adversary being able to guess which forgery attempt is actually successful. This leads to a linear drop in security from STPRPℓ2 to CIML2, where the security drops with the number of forgery attempts. ◊

Next, we show that EtE also provides confidentiality with leakage. Particularly, we consider Chosen-Ciphertext Attacks with nonce-misuse resilience and challenge leakage (CCAmL2) and Chosen-Ciphertext Attacks with nonce-misuse resistance and no challenge leakage (CCAMℓ2). We show that STPRPℓ2 also implies CCAMℓ2 in terms of privacy, but does not imply CCAmL2. The full proof can be found in Appendix B.3.

Theorem 4. *Let $\mathcal{A}$ be a CCAMℓ2 adversary as in Section 3.1 against Ψ as in Section 4. Then, there exists an STPRPℓ2 adversary $\mathcal{B}$ against Π with the same query complexity as $\mathcal{A}$, such that*

$$\mathrm{Adv}^{\mathsf{CCAM}\ell 2}_{\Psi,\mathsf{L}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{STPRP}\ell 2}_{\Pi,\mathsf{L}}(\mathcal{B}) + \frac{q(q-1)}{2^{l_{\min}+1}}\,,$$

where $l_{\min}$ is the minimum ciphertext length in $\mathcal{M}$.

Theorem 4 establishes that STPRPℓ2 generically implies CCAMℓ2. However, CCAmL2 is not implied by STPRPℓ2. MaB, under the leakage model of Theorem 5, achieves STPRPℓ2 but admits a single-query CCAmL2 distinguisher. We delay the formal treatment of this separation until after MaB is introduced (Theorem 7). We also give sufficient conditions for CCAmL2 in Theorem 8, showing that under simulatable-leakage assumptions, STPRPℓ2 can imply CCAmL2. However, we acknowledge that these are strong assumptions.

5 The Motte-and-Bailey Framework

In this section, we introduce the Motte-and-Bailey framework or MaB for short. MaB aims to provide high security guarantees, even in the presence of leakage. It achieves this through a *layered construction*, using two fully protected, fixed-size block cipher calls and a less protected PRG, but the rest remains unprotected. Motte-and-Bailey comes in two flavours, a plain, less protected version, which we simply call MaB, and a more protected version, Str-MaB, with stronger security guarantees, but at the same time less efficient. A detailed description can be found in Algorithm 1 and Fig. 1.

The tweak in MaB consists of two parts, N, a fixed-length mandatory tweak, and A, a variable-length, potentially empty, tweak. This maps nicely to AEAD, where N can be used for the nonce and A can be used for the associated data.

The framework consists of four main components: a leak-free TBC, a collision-resistant hash function, a leakage-resilient PRG, and an XOR addition. The MaB framework draws inspiration from the PIV framework [63], with two major differences:

– PIV consists of two TBCs: one has a fixed block length and variable tweak length and is STPRP-secure, while the other has a fixed tweak length and variable block length and is only STPRP-secure against adversaries that never repeat tweaks. The secret key of this primitive requires protection against leakage. On the other hand, MaB does not have a secret key in the middle layer X, which is a completely public primitive. Informally, the middle layer of MaB can be seen as eXtendable-Output Function (XOF).
– While PIV uses any variable-tweak-length TBC for its outer layers, MaB specifies these layers using a (public) collision-resistant hash function, and a fixed-tweak-length TBC.

The second difference can be seen as a special case of PIV. Indeed, if it was the only difference, then MaB could have been seen as an instance of PIV. The first difference is more subtle and deviates from PIV. In fact, the first difference qualitatively downgrades the security of PIV. It is not hard to see that in the black-box setting, the security proof of PIV can be adapted to MaB. This adaptation leads to security bound with complexity limits

$$D \cdot T < 2^n, D < 2^{n/2}$$

where D is the online data complexity and T is the offline number of queries to the primitive X. For PIV, only the second limit applies. In terms of leakage resilience, we argue that the new definition of X is an improvement. We could have built X as a keyed stream cipher. However, in order to maintain leakage resilience while keeping the cost low, we would have had to use a leakage resilient stream cipher. This would have added additional long-term secrets that need to be protected. Leakage-resilient stream ciphers that do not require full protection also suffer from similar security drops. We analyze the security of Motte-and-Bailey and state the main security claims related to it below.

Algorithm 1 Plain (left) and strengthened (right) Motte-and-Bailey. Protected TBC calls are colored in teal.

$\alpha \leftarrow \mathtt{H}(N \parallel |A| \parallel A \parallel M_r)$
$\theta \leftarrow \tilde{E}_K^{\alpha}(M_l)$
$C_r \leftarrow M_r \oplus \mathtt{X}(N \parallel \theta)$
$\alpha \leftarrow \mathtt{H}(N \parallel |A| \parallel A \parallel C_r)$
$C_l \leftarrow \tilde{E}_K^{\alpha}(\theta)$

$\gamma \leftarrow \tilde{E}_K^{1,0}(N)$
$\alpha \leftarrow \mathtt{H}(\gamma \parallel |A| \parallel A \parallel M_r)$
$\theta \leftarrow \tilde{E}_K^{0,\alpha}(M_l)$
$C_r \leftarrow M_r \oplus \mathtt{X}(N \parallel \theta)$
$\gamma' \leftarrow \tilde{E}_K^{2,0}(N)$
$\beta \leftarrow \mathtt{H}(\gamma' \parallel |A| \parallel A \parallel C_r)$
$C_l \leftarrow \tilde{E}_K^{0,\beta}(\theta)$

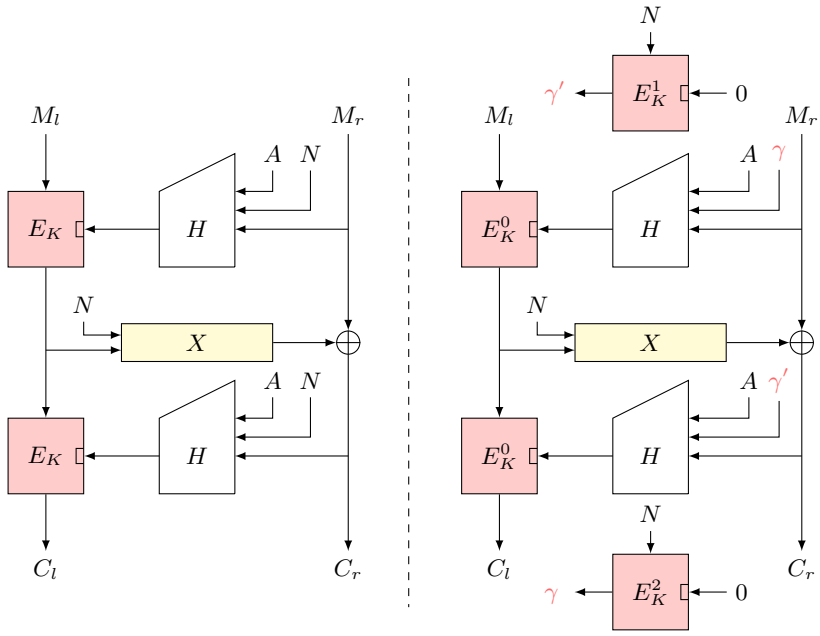


Fig. 1: The plain (left) and strengthened (right) `Motte-and-Bailey` framework

5.1 Choosing a Leakage Function for `Motte-and-Bailey`

The leakage function L is chosen non-adaptively by the adversary, with the following restrictions. On input (K, N, A, M, C) it returns (T_1, T_2, ζ) where:

$$L_1 = \mathsf{L}_H(A, N, M_r)\,, \quad \zeta = \lambda(N, \theta)\,, \quad L_2 = \mathsf{L}_H(A, N, C_r)\,,$$

where $\theta = E_K(T_1, M_l)$ computed leak-free (respectively $\theta = E_K^{-1}(T_1, C_l)$ for inverse queries) with $T_1 = H(A, N, M_r)$. L_H is chosen freely by the adverary, but λ has to be chosen among the set of hard-to-invert and oracle-free functions, in the sense of Bellizia et al. [10, Section 3.1].

Theorem 5. *Let H be an (ϵ_{CR}, t')-collision-resistant hash function with unbounded leakage, $\tilde{E}$ be a leak-free TBC, and X be a leakage-resilient random function, where $\tilde{E}$ and H are sampled independently of X. Let L and L' be as Section 5.1. Then, for any adversary $\mathcal{A}$ against the $\mathsf{STPRP}\ell 2$ game with resources as in Section 3.1, there exist adversaries $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ in their corresponding games such that*

$$\begin{aligned}\mathrm{Adv}^{\mathsf{STPRP}\ell 2}_{\mathit{MaB}[\tilde{E},H,X],\mathsf{L}}(\mathcal{A}) \le\ & 2\mathrm{Adv}^{\mathsf{STPRP}}_{\tilde{E}}(\mathcal{A}_1) \\ & + \mathrm{Adv}^{\mathsf{PRG}\ell 1}_{X,\mathsf{L}}(\mathcal{A}_3) + (q_e + q_d)\mathrm{Adv}^{\mathsf{GuessL1}[2q+q_p]}_{X,\mathsf{L}'}(\mathcal{A}_2) \\ & + \epsilon_{CR} + \frac{14q^2}{2^n} + \frac{(q_{ce} + q_{cd})q_p}{2^n}\,,\end{aligned}$$

where $t' > t + 2qt_H$, and t_H is the time of one call to H.

Clarifying the proof model: In the proof of Theorem 5, the adversary has access to the ideal primitive X. As such, this is classified as a Random-Oracle-Model (ROM) proof. However, in the proof it is emphasized that the two standard-model primitives $\tilde{E}$ and H (especially $\tilde{E}$) do not have access to X, and are sampled independently. If H is not independent of X, then this will affect the GuessL1 and PRGℓ1 reductions, but this can be taken care of by a dedicated analysis. Similarly, If $\tilde{E}$ is not independent of X, a dedicated hybrid analysis is needed to ensure that the dependency does not lead to a security break.

The GuessL1, PRGℓ1 and PRGL1 security of X cannot be quantified except in the black-box model. In the leakage-resilient model, the security relies on both the cryptanalysis of X and scrutinizing it using SPA-style side-channel analysis. The proof is deferred to Appendix B.4.

The reduction can be made simpler by assuming that H is also a random oracle, independent of X, rather than just a collision-resistant hash function. We avoid that for two reasons. A random oracle assumption is a much stronger assumption and in fact several standardized hash functions are not indifferentiable from a random oracle, such as SHA2. Besides, X is also not exactly a random oracle but something slightly weaker, since the online queries only include random inputs. Adding a random oracle assumption on H is stronger than that, as the adversary can control and choose the inputs during online queries.

5.2 The necessity of hard-to-invert leakage

It may be tempting to consider the possibility of defining X with unbounded leakage. However, leaking the intermediate states of a wide-block cipher has led to negative results in the literature. For instance, in the case of Feistel ciphers, it was shown that leaking the intermediate states mandates a super-logarithmic number of rounds in order to prove security [28]. In our context, leaking the input to X is a *disastrous* idea. In order to illustrate this, assume that $\mathcal{A}$ is an adversary that makes a query (N, A, M_l, M_r) and receives the ciphertext (C_l, C_r) and the input to X, which we denote by θ. Next, the adversary makes an offline query to X with C_l as input and an online challenge query to the construction with (θ, C_r) as the plaintext and receives an answer (C'_l, C'_r). In the real world, $C'_r = C_r \oplus X(C_l)$, while in the ideal world this equality, with high probability, does not hold. Hence, the adversary can clearly tell whether the challenge query has been generated by the real construction using X or by an independent random permutation. The attack does not work if we use different TBCs for different rounds of the construction, making it harder for the adversary to force the input of X to a predictable value. It is worth exploring whether using different TBCs can lead to relaxed assumptions on the leakage of X for STPRPℓ2 security. However, since we need this assumption later for other security notions anyway, we leave this exploration outside the scope of this paper. Besides, even if the attack is inapplicable, the PRGℓ1 assumption takes into consideration the uniformity and secrecy of the inputs.

5.3 The infeasibility of challenge leakage with unbounded hash leakage

As discussed in [10, 11], it is considered impossible to achieve confidentiality with challenge leakage in the deterministic setting (nonce-misuse in the case of AEAD) without having strong countermeasures. Consider for instance an adversary that makes the challenge query $(N, A, (M_l, M_r))$. The adversary can observe the leakage for that query and try to match it with the leakage from a previous query $(N, A, (M_l, M_r^\star))$ where M_r and $M_r^\star$ differ in only a few bits. Thus, achieving a notion stronger than $\mathsf{STPRP}\ell 2$ is considered impossible.

5.4 Improved CIML2 security

While Theorem 3 shows that $\mathsf{STPRP}\ell 2$ implies $\mathsf{CIML2}$, the reduction is quite loose. For instance, the reduction implies the terms

$$q_f \epsilon_{\mathsf{CR}} \text{ and } q_f \mathrm{Adv}^{\mathsf{STPRP}}_{\tilde{E}}(\mathcal{A}).$$

These terms are uncharacteristic and indeed not tight. In this section, we give a more dedicated security proof for $\mathsf{CIML2}$. We split M as

$$(M_l, M_r) \xleftarrow{n,*} M,$$

where M_l is the first (leftmost) n bits of the plaintext, and M_r is the remainder of the plaintext. Thus, in the remainder of the paper, we fix the following encoding scheme: $\mathsf{encode}(M) = (M_l, 0^t \| M_r)$.

Theorem 6. *Let H be an (ϵ_{CR}, t')-collision-resistant hash function with unbounded leakage, $\tilde{E}$ is a leak-free TBC, and X is a leakage-resilient random function, where $\tilde{E}$ and H are sampled independently of X, and let I be as Section 5.1. Then, for any adversary $\mathcal{A}$ against $\mathtt{MaB}[\tilde{E}, H, X]$ that makes q_e encryption queries, q_f forgery attempts, and q_p offline queries to X, such that $\mathcal{A}$ runs in time at most t and $t' > t + (2q_e + q_f)t_H$, there exist adversaries $\mathcal{A}_1, \mathcal{A}_3$ and $\mathcal{A}_4$ in their respective games, such that*

$$\begin{aligned}\mathrm{Adv}^{\mathit{CIML2}}_{\mathsf{EtE}[\mathtt{MaB}[\tilde{E},H,X]]}(\mathcal{A}) \leq\ & \mathrm{Adv}^{\mathit{STPRP}}_{\tilde{E}}(\mathcal{A}_1) + \epsilon_{\mathit{CR}} \\ & + \frac{3(2q_e + q_f)^2 + q_f q_p}{2^n} + \frac{q_f}{2^t} \\ & + 2q_e \mathrm{Adv}^{\mathit{GuessL1}[q_f]}_{X,\mathsf{L}}(\mathcal{A}_3) + \mathrm{Adv}^{\mathit{PRGL1}}_{X,\mathsf{L}}(\mathcal{A}_4).\end{aligned}$$

$t' > t + (2q_e + q_f)t_H$, where t_H is the time of one call to H. $\mathcal{A}_1$ makes $2q_e + q_f$ queries, while $\mathcal{A}_3$ and $\mathcal{A}_4$ make $q_e + q_f$ online queries and q_p offline queries.

The proof of Theorem 6 follows a similar structure to that of Theorem 5. The goal is to show that as long as θ is not leaked in any query, the values that are compared to 0^t are indistinguishable from random under the $\mathsf{PRGL1}$ assumption. The complete proof is deferred to Appendix B.5.

Tightness of the CIML2 bound. In the proof of CIML2, we are being generous by considering any repetition of θ to be a win for the adversary, whether this repetition comes from two online queries, or an online query and an offline query to X. While this approach is tight against STPRPℓ2, it is not clear that it is always tight against CIML2. For instance, the adversary can guess θ using offline queries, but it still needs to find the corresponding ciphertext to produce this θ, and that ciphertext needs to satisfy that the first t bits are the same as the first t bits when X is queried with that θ and the same nonce. One possible attack under the assumption that X has unbounded leakage is to guess a θ such that the first t bits of $X(\theta, N)$ are 0^t. Then, query a message with N as the nonce and $M_l = \theta$. We can then leak the input of X and we get a valid forgery. This attack is non-trivial, since the first step requires 2^t offline computations and a weaker assumption on X. However, t can be a lot smaller than n, and if $t < n/2$ then the attack is better than generic attacks. Thus, we do not expect a significant improvement with a finer analysis than what we provide.

On the other hand, we note that Theorem 6 is indeed an improvement over Theorem 3. To see that, consider $t \leq n/2$, which is quite a reasonable assumption if $n = 256$ and $t = 128$. Consider also that the hash output is 256 bits. Theorem 3 provides strictly less than 86 bits of security, due to the existence of cubic terms. Theorem 6 ensures 128-bit security, with only quadratic terms with respect to n and the hash output, but only a term linear in t.

Last but not least, we note that the term $\mathrm{Adv}_X^{\mathsf{PRGL1}}(\mathcal{A}_4)$ may also not be tight. The adversary cannot use the leakage from challenge queries (in this case forgeries) except for future queries. Thus, PRGℓ1 maybe sufficient. However, a proof with PRGℓ1 would give a bound $q_f \mathrm{Adv}_X^{\mathsf{PRG}\ell 1}(\mathcal{A}_4)$ using a hybrid argument. Schemes such as the Sponge construction can target both assumptions, and it is likely that the bound we get from PRGL1 is better since it does not rely on this hybrid argument.

6 Security Requirements for Challenge Leakage

In this section, we present a strengthened variant that achieves all the same security goals, as well as CCAmL2. Thus, it achieves the best possible AEAD security (up to the birthday bound).

6.1 Generic Results: Continued

First, we present a theorem that captures the plain MaB insecurity against CCAmL2 adversaries, and the separation between STPRPℓ2 and CCAmL2.

Theorem 7. *(STPRPℓ2 $\not\Rightarrow$ CCAmL2) There is an adversary that breaks the CCAmL2 security of Encode-then-Encipher using MaB, under the construction and assumptions in Theorem 5, using a single query.*

Proof. The adversary chooses a pair (N, A) and two messages M^0, M^1, such that $M_r^0 \neq M_r^1$. The adversary makes a challenge query and receives the unbounded

leakage of H which includes its input M_r^b. The adversary returns 0 if $M_r^b = M_r^0$, and 1 otherwise. Such an adversary distinguishes the two worlds with probability 1 under the unbounded leakage assumption of H. □

Definition 1 (Simulatable leakage). *Let $\Pi[\mathsf{prim}] : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ be a TWBC built from the public primitives* prim *and possibly secret primitives, such that $\mathcal{T} = \mathcal{N} \times \mathcal{A}$, where $\mathcal{N}$ is a fixed-length nonce space, and $\mathcal{A}$ is a variable-length associated data space. Let $\mathsf{S}[\mathsf{prim}]$ be a probabilistic polynomial time simulator that takes as input a query N, $|A|$ and $|M|$, but not A or M, and returns simulated leakage for $\Pi_K^{(N,A)}(M)$. The simulator can make queries to* prim*. We define the following game:*

1. *At the beginning of the game, the challenger selects K uniformly at random from the key space and a bit b uniformly at random.*
2. *The adversary makes two types of queries:*
   - Non-challenge queries*: These are queries to Π_K, its inverse, or the public primitives* prim*. All the non-challenge queries to Π and its inverse are made to the leaky oracles.*
   - Challenge queries*: These are queries to Π_K of the form: $((N, A), M)$, where N was not used in any previous query.*
     - *If $b = 1$, the challenger sends back $\Pi_K^{(N,A)}(M)$ and $\mathsf{L}(\Pi_K^{(N,A)}(M))$, i.e., it queries the leaky Π oracle. Otherwise, it queries the leak-free oracle $\Pi_K^{(N,A)}(M)$ and returns the output as well as $\mathsf{S}(N, |A|, |M|)$.*

   Challenge and non-challenge queries can be interleaved.
3. *The adversary returns a bit b'.*

We say that Π is $(\epsilon_{\mathsf{sim}}, t, \mathsf{S})$-leakage simulatable (or $(\epsilon_{\mathsf{sim}}, t, \mathsf{S})$-$\mathsf{LS}$ for short) if, for any adversary $\mathcal{A}$ that makes q_e non-challenge encryption queries, q_d non-challenge decryption queries, q_{ce} encryption queries, and q_p prim *queries, and runs in time at most t,*

$$|\Pr[\mathcal{A} \Rightarrow 1|b = 1] - \Pr[\mathcal{A} \Rightarrow 1|b = 0]| \leq q_{ce}\epsilon_{\mathsf{sim}}.$$

Theorem 7 shows that $\mathsf{STPRP}\ell 2$ is not enough to imply $\mathsf{CCAmL2}$, yet, if we combine STPRP and LS then it does imply $\mathsf{CCAmL2}$ security, more formally:

Theorem 8. *($\mathsf{STPRP}\ell 2 \wedge \mathsf{LS} \Rightarrow \mathsf{CCAmL2}$) For any $\mathsf{CCAmL2}$ adversary $\mathcal{A}$ against Ψ, built from an $(\epsilon_{\mathsf{sim}}, t, \mathsf{S})$-leakage-simulatable TWBC Π that runs in time t_a, there exists an $\mathsf{STPRP}\ell 2$ adversary $\mathcal{B}$ that makes the same number of challenge and non-challenge queries, and runs in time poly(t_a), such that*

$$\mathrm{Adv}^{\mathsf{CCAmL2}}_{\Psi,\mathsf{L}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{STPRP}\ell 2}_{\Pi,\mathsf{L}}(\mathcal{B}) + 2q_{ce}\epsilon_{\mathsf{sim}},$$

provided $t > t_a$ and q_{ce} is the number of challenge (encryption) queries.

In particular, Theorem 7 implies that the plain `MaB` is not leakage-simulatable. The proof of Theorem 8 can be found in Appendix B.6. In the next section, we introduce a strengthened variant of `MaB` that is leakage-simulatable, and thus achieves $\mathsf{CCAmL2}$ security.

6.2 Strengthened MaB

The strengthened variant is described in Algorithm 1 (right) and Figure 1 (right). A pair of leak-free calls to the TBC are added, such that for each nonce value N, two independent secret blocks are generated. These blocks replace the nonce as the first blocks absorbed by the hash function, where γ is used on the plaintext side, and γ' is used on the ciphertext side. We note that since the strengthening is particularly for CCAmL2, applying it only on the plaintext side is plausible. However, we maintain the symmetry such that a user can instantiate the encode-then-encipher construction in either direction. First, we show that the strengthened `Strengthened-MaB` (`Str-MaB`) maintains STPRPℓ2 and CIML2 security. Then, we argue informally that it satisfies the assumptions of Theorem 8. The relation between `Str-MaB` and Theorem 8 requires further practical analysis of side-channel attacks that is beyond the scope of this paper and shall be the basis for future work.

6.3 Choosing a Leakage Function for `Strengthened-MaB`

The leakage function L is chosen non-adaptively by the adversary, with the following restrictions. On input (K, N, A, M, C) it returns (T_1, T_2, ζ) where:

$$L_1 = \mathsf{L}_H(A, N, M_r)\,, \quad \zeta = \lambda(N, \theta)\,, \quad L_2 = \mathsf{L}_H(A, N, C_r)\,,$$

where $\theta = E_K(T_1, M_l)$ computed leak-free (respectively $\theta = E_K^{-1}(T_1, C_l)$ for inverse queries) with $T_1 = H(A, N, M_r)$. both L_H and λ has to be chosen among the set of hard-to-invert and oracle-free functions, in the sense of Bellizia et al. [10, Section 3.1]. Note that the choice of L_H is more restrictive than that of Section 5.1.

Theorem 9. *Let H be an (ϵ_{CR}, t')-collision-resistant hash function with unbounded leakage, $\tilde{E}$ be a leak-free TBC, and X be a leakage-resilient random function, where $\tilde{E}$ and H are sampled independently of X, and let L and L' be as Section 6.3. Then, for any adversary $\mathcal{A}$ against the STPRPℓ2 game with resources as in Section 3.1, there exist adversaries $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ in their corresponding games such that*

$$\begin{aligned}\mathrm{Adv}^{\mathit{STPRP\ell2}}_{\mathit{StrMaB}[\tilde{E},H,X],\mathsf{L}}(\mathcal{A}) \le\ & 2\mathrm{Adv}^{\mathit{STPRP}}_{\tilde{E}}(\mathcal{A}_1) \\ & + (q_e + q_d)\mathrm{Adv}^{\mathit{GuessL1}[q_p]}_{X,\mathsf{L}}(\mathcal{A}_2) \\ & + \mathrm{Adv}^{\mathit{PRG\ell1}}_{X,\mathsf{L}'}(\mathcal{A}_3) \\ & + \epsilon_{CR} + \frac{15q^2}{2^n} + \frac{(q_{ce} + q_{cd})q_p}{2^n}\,,\end{aligned}$$

where $t' > t + 2qt_H$, and t_H is the time of one call to H. $\mathcal{A}_1$ makes $4q$ queries.

Theorem 10. *Let H be an (ϵ_{CR}, t')-collision-resistant hash function with unbounded leakage, $\tilde{E}$ is a leak-free TBC, and X is a leakage-resilient random*

function, where $\tilde{E}$ and H are sampled independently of X, and let L *be as Section 6.3. Then, for any adversary $\mathcal{A}$ against $\mathtt{StrMaB}[\tilde{E}, H, X]$ that makes q_e encryption queries, q_f forgery attempts, and q_p offline queries to X, such that $\mathcal{A}$ runs in time at most t and $t' > t + (2q_e + q_f)t_H$, there exist adversaries $\mathcal{A}_1$ and $\mathcal{A}_3$ in their respective games, such that*

$$\begin{aligned}\mathrm{Adv}^{\mathit{CIML2}}_{\mathsf{EtE}[\mathtt{StrMaB}[\tilde{E},H,X]],\mathsf{L}}(\mathcal{A}) \leq\ & \mathrm{Adv}^{\mathit{STPRP}}_{\tilde{E}}(\mathcal{A}_1) + \epsilon_{\mathit{CR}} \\ & + \frac{3(2q_e + q_f)^2 + q_f q_p}{2^n} + \frac{q_f}{2^t} \\ & + \mathrm{Adv}^{\mathit{PRGL1}}_{X,\mathsf{L}}(\mathcal{A}_3).\end{aligned}$$

$t' > t + (2q_e + q_f)t_H$, where t_H is the time of one call to H. $\mathcal{A}_1$ makes $4q_e + 2q_f$ queries, while $\mathcal{A}_3$ makes $q_e + q_f$ online queries and q_p offline queries.

bhe proofs of Theorems 9 and 10 follow the same structure as Theorems 5 and 6 with minor differences: we need to ensure that there are no collisions between γ and γ' in any pair of queries (or in the same query), and the GuessL1 bad event is now restricted to collisions with offline queries, since the TURPs used in the top and bottom layer are now domain separated.

StrMaB is leakage-simulatable under reasonable assumption. We emphasize that this is not a formal proof, but a sequence of observations, that raise our confidence in the security of StrMaB. The reason we present this section informally is two-fold:

1. The simulatable-leakage framework has been shown to capture formally by defining good simulators for physical leakage [53]. We acknowledge this limitation and, as mentioned earlier, we leave a more in-depth study for future work.
2. A more formal study would require a lot of theoretical and physical analyses of H and X that are beyond the scope of this already dense work.

The STPRPℓ2 security proof ensures several nice properties in the range where the bound is valid:

– There is no collision on θ and there are no cross-collision between γ and γ'.
– If H is a sequential hash function built from a smaller primitive, paramaterized by block size $|\gamma|$, then we can instantiate it such that the first input block in any challenge query is indistinguishable from random.

This gives us a recipe for designing a simulator: for each fresh nonce in each challenge query, we sample three random values γ, γ' and θ, and simulate the leakage using the executions $H(\gamma, |A_\$|, A_\$, 0^t \| M_\$)$, $H(\gamma', |A_\$|, A_\$, C_r)$, $X(\theta, N)$, such that N is the fresh nonce, $A_\$$ and $M_\$$ are random strings of the same length of A and M_r^0, respectively, and C_r is the right-side ciphertext. The simulator does not reveal the outcome of these executions, only the associated leakage. Distinguishing the simulated leakage requires distinguishing:

– two executions of H with random fresh state.
– two executions of X with random fresh seeds.
– two executions of $\oplus$ with random input.

For well established hash functions and leakage-resilient PRGs, these tasks are expected to be hard as long as the implementation is protected against SPA. Under this assumption, the strengthened `MaB` achieves the strongest practically feasible combination of notions:

$$\mathsf{STPRP}\ell 2 \wedge \mathsf{CIML2} \wedge \mathsf{CCAmL2} \wedge \mathsf{CCAM}\ell 2.$$

7 `Qaitbay`: A Sponge-Based Instantiation

`Qaitbay` is the sponge-based instance of the `MaB` framework. Its two variants, `Qaitbay-1` and `Qaitbay-2`, instantiate plain `MaB` (Algorithm 1, left) and StrMaB (Algorithm 1, right), respectively. In both, the hash function H and the public primitive X are realized by TurboSHAKE-256 [13] over a permutation P. The tweakable block cipher $\tilde{E}$ is left abstract in Construction 1 and instantiated three different ways.

Construction 1 *Let n, k, b, r, c, h be positive integers with $b = r + c$ and $h \geq c$. Let*

$$\tilde{E} : \{0,1\}^k \times \big(\{0,1,2\} \times (\{0,1\}^h \setminus \mathcal{S}_b)\big) \times \{0,1\}^n \to \{0,1\}^n$$

be a leak-free *tweakable block cipher, where $B \in \{0,1,2\}$ is a domain separator and $T \in \{0,1\}^h \setminus \mathcal{S}_b$ is the main tweak. We denote its evaluation by $\tilde{E}_K^{B,T}(M)$. The set $\mathcal{S}_b \subset \{0,1\}^h$ is a (possibly empty) set of reserved tweak values determined by the realization of $\tilde{E}$. Additionally, let $P : \{0,1\}^b \to \{0,1\}^b$ be a publicly accessible random permutation sampled independently of $\tilde{E}$, and let $\mathsf{Sponge}_{b,r,c} : \{0,1\}^* \times \mathbb{Z}^+ \to \{0,1\}^*$ denote the sponge construction of [14] over P with rate r and capacity c, whose second input specifies the output length. An algorithmic description of the `Qaitbay` family of accordion modes is given by Algorithm 2.*

Security. `Qaitbay-1` inherits the security of plain `MaB`. `Qaitbay-2` inherits the security of StrMaB (including the informal simulatable leakage assumption). A proof sketch for `Qaitbay-1` is given in Appendix C.1. In the remainder of this section, we give plausible practical instantiations.

7.1 Practical Instantiations

For simplicity, we give the practical instantiations for `Qaitbay-1`, and leave their extension to `Qaitbay-2` out of scope. In particular, we give three instantiations. All instantiations use `TurboSHAKE-256` [13] (i.e. with 256-bit capacity). They differ in the instantiation of $\tilde{E}$.

Qaitbay [PHOTON,TurboSHAKE]: The TBC is instantiated using the XPX TBC [56], using the PHOTON-256 permutation. The instantiation requires Galois field multiplication over $\mathrm{GF}(2^{256})$ and is defined by $K' \leftarrow K \odot T; Y \leftarrow \texttt{PHOTON}(K' \oplus X) \oplus K'$. Mennink [56] defines a set of valid tweaks for which the construction is secure. In our case, the relevant condition is $T \notin \{0, 1\}$.

Qaitbay [Rijndael-IC,TurboSHAKE]: We can use an ideal cipher with 256-bit key and 256-bit block to instantiate a TBC with 256-bit key, 256-bit tweak, and 256-bit block, while providing 128-bit security with a straightforward construction

$$C \leftarrow \texttt{IC}(K \oplus T, M).$$

In practice, a candidate for replacing this ideal cipher is Rijndael-256-256.

Qaitbay [Rijndael-PRP,TurboSHAKE]: We can construct a TBC 128-bit key, 256-bit tweak, and 256-bit block using a block cipher with 128-bit key and 256-bit block, using the seminal LRW2 construction [51].

$$L \leftarrow E_{K \oplus 1}(T);\ C \leftarrow E_K(M \oplus L) \oplus L\,.$$

This can be instantiated using Rijndael-256-128. In terms of security, this instantiation provides the same bit security as the rest (128), although it is more immune against key recovery attacks, and requires less key material. On the flip side, it requires double the number of TBC queries. We need 4 calls to the block cipher.

A discussion regarding the design rationale for Qaitbay can be found in Appendix D.

Algorithm 2 Qaitbay-1 (left) and Qaitbay-2 (right). Lines 4, 10, on the left-hand side, and lines 1, 5, 8 and 12 on the right-hand side, are leak-free. For further clarity, this lines have been coloured teal.

1: $L \leftarrow 0 \,\|\, N \,\|\, |A| \,\|\, A \,\|\, M_r$
2: $L \leftarrow L \,\|\, 1 \,\|\, 0^{r-1-(|L| \bmod r)}$
3: $\alpha \leftarrow \mathsf{Sponge}^{b,r,c}(L, h)$
4: $\theta \leftarrow \tilde{E}_K^{0,\alpha}(M_l)$
5: $\Delta \leftarrow \mathsf{Sponge}^{b,r,c}(1 \,\|\, N \,\|\, \theta, |M_r|)$
6: $C_r \leftarrow M_r \oplus \Delta$
7: $L \leftarrow 0 \,\|\, N \,\|\, |A| \,\|\, A \,\|\, C_r$
8: $L \leftarrow L \,\|\, 1 \,\|\, 0^{r-1-(|L| \bmod r)}$
9: $\alpha \leftarrow \mathsf{Sponge}^{b,r,c}(L, h)$
10: $C_l \leftarrow \tilde{E}_K^{0,\alpha}(\theta)$
11: return $C_l \,\|\, C_r$

1: $\gamma \leftarrow \tilde{E}_K^{1,0}(N)$
2: $L \leftarrow 0 \,\|\, \gamma \,\|\, |A| \,\|\, A \,\|\, M_r$
3: $L \leftarrow L \,\|\, 1 \,\|\, 0^{r-1-(|L| \bmod r)}$
4: $\alpha \leftarrow \mathsf{Sponge}^{b,r,c}(L, h)$
5: $\theta \leftarrow \tilde{E}_K^{0,\alpha}(M_l)$
6: $\Delta \leftarrow \mathsf{Sponge}^{b,r,c}(1 \,\|\, N \,\|\, \theta, |M_r|)$
7: $C_r \leftarrow M_r \oplus \Delta$
8: $\gamma' \leftarrow \tilde{E}_K^{2,0}(N)$
9: $L \leftarrow 0 \,\|\, \gamma' \,\|\, |A| \,\|\, A \,\|\, C_r$
10: $L \leftarrow L \,\|\, 1 \,\|\, 0^{r-1-(|L| \bmod r)}$
11: $\alpha \leftarrow \mathsf{Sponge}^{b,r,c}(L, h)$
12: $C_l \leftarrow \tilde{E}_K^{0,\alpha}(\theta)$
13: return $C_l \,\|\, C_r$

8 Alicante: A TBC-Based Instantiation

Alicante is a TBC-based instantiation of the MaB framework. It is analyzed in the ideal cipher model, uses of MDPH (Merkle-Damgård with Hirose's compression function [36]) as a hash function and uses a block cipher-based PRG: 2PRG. The Alicante family is described in Algorithm 3 and similarly to Qaitbay, its security follows mostly from Section 5. The proof sketch is delegated to Appendix C.2.

8.1 Concrete Instantiation: Deoxys-128-640

Deoxys-128-640 is a *large-tweak* TBC proposed by Cogliati *et al.* [19] as a variant of the Deoxys-TBC [44]. The details of the design are not relevant to our paper. Nevertheless, at a high level, its round function design is based on the AES round function and consists of 20 such rounds, alongside a tweakey schedule that takes 640 bits as input and generates 21 128-bit round keys. In this section, we provide two instances of Alicante with $n = k = 128$, and $t_w = 637$. The security is maintained as long as the total amount of data encrypted blocks D and the total number of primitive queries T satisfy $DT < 2^{128}$, and $D < 2^{64}$. The complete instantiation is given in Algorithm 4. We note that we can optimize the scheme further by choosing different variants for Deoxys-TBC in different purposes, instead of always padding the tweak parameters to 640 bits. However, what we present in this subsection is merely a proof of concept on the instantiability Alicante, so for the sake of simplicity of the presentation, we use only one variant.

8.2 Comparison

Table 1 summarizes the parameters selected for Qaitbay-1 and Alicante-1, and the target black-box security level, compared to HCTR2 and Kravatte-WBC. We note that HCTR2 and Kravatte-WBC require full implementation protection.

Algorithm 3 Alicante-1 (left) and Alicante-2 (right). Protected TBC calls are colored in teal.

$L \leftarrow N \parallel |A| \parallel A \parallel M_r$
$\delta \leftarrow t_w - 1 - (|L| \bmod (t_w))$
$\alpha \leftarrow \texttt{MDPH}[\tilde{E}^1](L \parallel 1 \parallel 0^\delta)$
$\theta \leftarrow \tilde{E}_K^{0,\alpha}(M_l)$
$C_r \leftarrow M_r \oplus \texttt{2PRG}[\tilde{E}^{2,N}](\theta)$
$L \leftarrow N \parallel |A| \parallel A \parallel C_r$
$\delta \leftarrow t_w - 1 - (|L| \bmod (t_w))$
$\alpha \leftarrow \texttt{MDPH}[\tilde{E}^1](L \parallel 1 \parallel 0^\delta)$
$C_l \leftarrow \tilde{E}_K^{0,\alpha}(\theta)$

$\gamma \leftarrow \tilde{E}_K^{3,0}(N)$
$L \leftarrow \gamma \parallel |A| \parallel A \parallel M_r$
$\delta \leftarrow t_w - 1 - (|L| \bmod (t_w))$
$\alpha \leftarrow \texttt{MDPH}[\tilde{E}^1](L \parallel 1 \parallel 0^\delta)$
$\theta \leftarrow \tilde{E}_K^{0,\alpha}(M_l)$
$C_r \leftarrow M_r \oplus \texttt{2PRG}[\tilde{E}^{2,N}](\theta)$
$\gamma' \leftarrow \tilde{E}_K^{4,0}(N)$
$L \leftarrow \gamma' \parallel |A| \parallel A \parallel C_r$
$\delta \leftarrow t_w - 1 - (|L| \bmod (t_w))$
$\alpha \leftarrow \texttt{MDPH}[\tilde{E}^1](L \parallel 1 \parallel 0^\delta)$
$C_l \leftarrow \tilde{E}_K^{0,\alpha}(\theta)$

Table 1: Summary of the plausible instantiations and their security. # Calls refers to the number of unprotected/lightly protected permutation or TBC calls, where l is the bit-length of M. In case of `XPX`, we count the field multiplication and `PHOTON-256` calls separately as two calls. $\star$ abstracts the cost of the `UHF`. We remark that `TurboSHAKE-128` has a permutation size of 1600 bits and a capacity of 256 bits, while `Deoxys-128-640` has a block size of 128 bits and a key + tweak size of 640 bits.

Scheme	TBC	H/X	Sec.	Key Size	Prot. Calls	# Calls
`Qaitbay-1`	`XPX[PHOTON-256]`	`TurboSHAKE-128`	128	256	4	$3\lceil\frac{(l-256)}{1344}\rceil$
`Qaitbay-1`	`Rijndael-IC`	`TurboSHAKE-128`	128	256	2	$3\lceil\frac{(l-256)}{1344}\rceil$
`Qaitbay-1`	`Rijndael-PRP`	`TurboSHAKE-128`	128	128	4	$3\lceil\frac{(l-256)}{1344}\rceil$
`Alicante-1`	`Deoxys-128-640`	`Deoxys-128-640`	64	128	2	$4\lceil\frac{(l-256)}{509}\rceil+\lceil\frac{(l-256)}{64}\rceil$
`HCTR2`	`AES`	`AES/UHF`	64	256	$\lceil\frac{l}{128}\rceil+\star$	0
`HCTR2`	`Rijndael`	`Rijndael/UHF`	128	> 256	$\lceil\frac{l}{256}\rceil+\star$	0
`Kravatte-WBC`	`Kecccak-p[6]`	`Kecccak-p[6]`	128	128	$3\lceil\frac{l}{800}\rceil$	0

We would like to emphasize that the efficiency of our approach comes from the assumption of the secure implementation, and not the unprotected performance. Thus, we find that comparing practical, yet unprotected, implementations is of little value. However, as a simple proof of concept, we have implemented the unprotected `Qaitbay-1` in C, and compared it to the `Kravatte-WBC`, since they both use the same permutation family. We used the following libraries for the cryptographic primitives: XKCP for `Kravatte-WBC` and for `TurboShake` [2], and Rhys Weatherley's NIST LWC library for `PHOTON-256` [1]. For 256-bit keys, 128-bit nonces, 1024-bit ADs, and 4096-bit plaintexts (the typical memory page size): `Kravatte-WBC` takes 13 cycles per byte. while `Qaitbay-1` takes 36.5 cycles per byte. These numbers consider only the plaintext size in the denominator, and are taken as the average over 1000000 calls on an Apple Silicone M1 processor. This more or less matches our estimates.

9 Conclusion

In this work we have studied the relation between STPRPℓ2 security and CIML2, CCAMℓ2, and CCAmL2 security using the generic EtE framework. We have introduced `Motte-and-Bailey`, a novel, generic framework to build secure leakage-resilient accordion modes. Furthermore, we have shown said framework with two competitive instantiations, with strong security properties for STPRPℓ2, CIML2 and CCAMℓ2, and can potentially provide CCAmL2 security with additional assumptions. Finally, we have argued that these instantiations, `Qaitbay` and `Alicante`, are more efficient for the same level of security than previous works in the leakage-resilience setting.

While `Motte-and-Bailey` is a promising framework, there are still future research directions to be explored. One clear problem is scrutinizing the assumptions needed for CCAmL2 and formalizing the security of `Qaitbay-2` and

Alicante-2 in that regard. Another problem is building Qaitbay using a single permutation. Not only would the proof have to be changed, but also this permutation has to be easily invertible to be used in XPX. We also note that XPX may not be the only approach to instantiate $\tilde{E}$ from a public permutation. The second problem is studying the security of MaB when $\tilde{E}$ is not leak-free, but only STPRPℓ2 secure. This problem in particular will be a step further in this domain. Last but not least, both Qaitbay and Alicante can benefit from a masking-friendly TBC with 128-bit key, 256-bit block and more than 256-bit tweak.

Acknowledgement

This work is partially funded by the Wallenberg-NTU Presidential Postdoctoral Fellowship. The project was performed while Mario Marhuenda Beltrán was on a research attachment at NTU, Singapore. This work is also funded by the Singapore MoE Tier 1 grant RG111/25.

References

1. Lightweight Cryptography Primitives (2026), https://github.com/rweather/lightweight-crypto
2. XKCP: The eXtended Keccak Code Package (2026), https://github.com/XKCP/XKCP
3. Andreeva, E., Bhati, A.S., Preneel, B., Vizár, D.: 1, 2, 3, Fork: Counter Mode Variants based on a Generalized Forkcipher. IACR Trans. Symmetric Cryptol. **2021**(3), 1–35 (2021). https://doi.org/10.46586/TOSC.V2021.I3.1-35, https://doi.org/10.46586/tosc.v2021.i3.1-35
4. Bacuieti, N., Daemen, J., Hoffert, S., Assche, G.V., Keer, R.V.: Jammin' on the Deck. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13792, pp. 555–584. Springer (2022). https://doi.org/10.1007/978-3-031-22966-4_19, https://doi.org/10.1007/978-3-031-22966-4_19
5. Bao, Z., Chakraborti, A., Datta, N., Guo, J., Nandi, M., Peyrin, T., Yasuda, K.: PHOTON-beetle authenticated encryption and hash family. NIST Lightweight Compet. Round **1**, 115 (2019), https://csrc.nist.gov/csrc/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/PHOTON-Beetle-spec.pdf
6. Barwell, G., Martin, D.P., Oswald, E., Stam, M.: Authenticated Encryption in the Face of Protocol and Side Channel Leakage. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10624, pp. 693–723. Springer (2017). https://doi.org/10.1007/978-3-319-70694-8_24, https://doi.org/10.1007/978-3-319-70694-8_24

7. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1976, pp. 531–545. Springer (2000). https://doi.org/10.1007/3-540-44448-3_41, https://doi.org/10.1007/3-540-44448-3_41
8. Bellare, M., Rogaway, P.: Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In: Okamoto, T. (ed.) Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1976, pp. 317–330. Springer (2000). https://doi.org/10.1007/3-540-44448-3_24, https://doi.org/10.1007/3-540-44448-3_24
9. Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., Leander, G., Leurent, G., Levi, I., Momin, C., Pereira, O., Peters, T., Standaert, F.X., Udvarhelyi, B., Wiemer, F.: Spook: Sponge-Based Leakage-Resistant Authenticated Encryption with a Masked Tweakable Block Cipher. IACR Trans. Symmetric Cryptol. **2020**(S1), 295–349 (2020). https://doi.org/10.13154/TOSC.V2020.IS1.295-349, https://doi.org/10.13154/tosc.v2020.iS1.295-349
10. Bellizia, D., Bronchain, O., Cassiers, G., Grosso, V., Guo, C., Momin, C., Pereira, O., Peters, T., Standaert, F.: Mode-Level vs. Implementation-Level Physical Security in Symmetric Cryptography - A Practical Guide Through the Leakage-Resistance Jungle. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12170, pp. 369–400. Springer (2020). https://doi.org/10.1007/978-3-030-56784-2_13, https://doi.org/10.1007/978-3-030-56784-2_13
11. Berti, F., Guo, C., Pereira, O., Peters, T., Standaert, F.: TEDT, a Leakage-Resist AEAD Mode for High Physical Security Applications. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1), 256–320 (2020). https://doi.org/10.13154/TCHES.V2020.I1.256-320, https://doi.org/10.13154/tches.v2020.i1.256-320
12. Berti, F., Guo, C., Peters, T., Standaert, F.: Efficient Leakage-Resilient MACs Without Idealized Assumptions. In: Tibouchi, M., Wang, H. (eds.) 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. Lecture Notes in Computer Science, vol. 13091, pp. 95–123. Springer (2021). https://doi.org/10.1007/978-3-030-92075-3_4, https://doi.org/10.1007/978-3-030-92075-3_4
13. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Assche, G.V., Keer, R.V., Viguier, B.: TurboSHAKE. IACR Cryptol. ePrint Arch. p. 342 (2023), https://eprint.iacr.org/2023/342
14. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge Functions. In: ECRYPT hash workshop. vol. 2007 (2007), https://csrc.nist.rip/groups/ST/hash/documents/JoanDaemen.pdf
15. Bhati, A.S., Verbauwhede, M., Andreeva, E.: Breaking, Repairing and Enhancing XCBv2 into the Tweakable Enciphering Mode GEM. IACR Cryptol. ePrint Arch. p. 1554 (2024), https://eprint.iacr.org/2024/1554

16. Chakraborty, D., Nandi, M.: An Improved Security Bound for HCTR. In: Nyberg, K. (ed.) Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5086, pp. 289–302. Springer (2008). https://doi.org/10.1007/978-3-540-71039-4_18, https://doi.org/10.1007/978-3-540-71039-4_18
17. Chen, Y.L., Davidson, M., Dworkin, M., Kang, J., Kelsey, J., Sasaki, Y., Turan, M.S.n., Chang, D., Mouha, N., Largo, M., et al.: Proposal of Requirements for an Accordion Mode. In: Discussion Draft for the NIST Accordion Mode Workshop 2024 (2024), https://csrc.nist.gov/files/pubs/other/2024/04/10/proposal-of-requirements-for-an-accordion-mode-dis/iprd/docs/proposal-of-requirements-for-an-accordion-mode-discussion-draft.pdf
18. Chen, Y.L., Hiraga, Y., Mouha, N., Naito, Y., Sasaki, Y., Sugawara, T.: Beyond-Birthday-Bound Security with HCTR2: Cascaded Construction and Tweak-Based Key Derivation. In: Hanaoka, G., Yang, B. (eds.) Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part I. Lecture Notes in Computer Science, vol. 16245, pp. 3–34. Springer (2025). https://doi.org/10.1007/978-981-95-5018-0_1, https://doi.org/10.1007/978-981-95-5018-0_1
19. Cogliati, B., Jean, J., Peyrin, T., Seurin, Y.: A Long Tweak Goes a Long Way: High Multi-user Security Authenticated Encryption from Tweakable Block Ciphers. IACR Communications in Cryptology **1**(2) (2024). https://doi.org/10.62056/a3qjp2fgx
20. Crowley, P., Biggers, E.: Adiantum: Length-Preserving Encryption for Entry-Level Processors. IACR Trans. Symmetric Cryptol. **2018**(4), 39–61 (2018). https://doi.org/10.13154/TOSC.V2018.I4.39-61, https://doi.org/10.13154/tosc.v2018.i4.39-61
21. Crowley, P., Huckleberry, N., Biggers, E.: Length-Preserving Encryption with HCTR2. IACR Cryptol. ePrint Arch. p. 1441 (2021), https://eprint.iacr.org/2021/1441
22. Daemen, J., Rijmen, V.: The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition. Information Security and Cryptography, Springer (2020). https://doi.org/10.1007/978-3-662-60769-5, https://doi.org/10.1007/978-3-662-60769-5
23. Datta, N., Dutta, A., Ghosh, S., List, E., Nandi, H.: HCTR+: An Optimally Secure TBC-based Accordion Mode. Cryptology ePrint Archive, Paper 2024/2053 (2024), https://eprint.iacr.org/2024/2053
24. Datta, N., Dutta, A., List, E., Mandal, S.: On the Security of Triplex- and Multiplex-Type Constructions with Smaller Tweaks. In: Chattopadhyay, A., Bhasin, S., Picek, S., Rebeiro, C. (eds.) Progress in Cryptology - INDOCRYPT 2023 - 24th International Conference on Cryptology in India, Goa, India, December 10-13, 2023, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14459, pp. 25–47. Springer (2023). https://doi.org/10.1007/978-3-031-56232-7_2, https://doi.org/10.1007/978-3-031-56232-7_2
25. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., Unterluggauer, T.: ISAP v2.0. IACR Trans. Symmetric Cryptol. **2020**(S1), 390–416 (2020). https://doi.org/10.13154/TOSC.V2020.IS1.390-416, https://doi.org/10.13154/tosc.v2020.iS1.390-416

26. Dobraunig, C., Matusiewicz, K., Mennink, B., Tereschenko, A.: Efficient Instances of Docked Double Decker with AES, and Application to Authenticated Encryption. In: Fehr, S., Fouque, P. (eds.) Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part I. Lecture Notes in Computer Science, vol. 15601, pp. 62–92. Springer (2025). https://doi.org/10.1007/978-3-031-91107-1_3, https://doi.org/10.1007/978-3-031-91107-1_3
27. Dobraunig, C., Mennink, B.: Leakage Resilience of the Duplex Construction. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11923, pp. 225–255. Springer (2019). https://doi.org/10.1007/978-3-030-34618-8_8, https://doi.org/10.1007/978-3-030-34618-8_8
28. Dodis, Y., Pietrzak, K.: Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks. In: Rabin, T. (ed.) Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6223, pp. 21–40. Springer (2010). https://doi.org/10.1007/978-3-642-14623-7_2, https://doi.org/10.1007/978-3-642-14623-7_2
29. Dutta, A., Nandi, M.: Tweakable HCTR: A BBB Secure Tweakable Enciphering Scheme. In: Chakraborty, D., Iwata, T. (eds.) Progress in Cryptology - INDOCRYPT 2018 - 19th International Conference on Cryptology in India, New Delhi, India, December 9-12, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11356, pp. 47–69. Springer (2018). https://doi.org/10.1007/978-3-030-05378-9_3, https://doi.org/10.1007/978-3-030-05378-9_3
30. Dziembowski, S., Pietrzak, K.: Leakage-Resilient Cryptography. In: 49th Annual IEEE Symposium on Foundations of Computer Science , FOCS 2008, Philadelphia, PA, USA, October 25-28, 2008. pp. 293–302. IEEE Computer Society (2008). https://doi.org/10.1109/FOCS.2008.56, https://doi.org/10.1109/FOCS.2008.56
31. Faust, S., Pietrzak, K., Schipper, J.: Practical Leakage-Resilient Symmetric Cryptography. In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7428, pp. 213–232. Springer (2012). https://doi.org/10.1007/978-3-642-33027-8_13, https://doi.org/10.1007/978-3-642-33027-8_13
32. Gjiriti, E., Reyhanitabar, R., Vizár, D.: Power Yoga: Variable-Stretch Security of CCM for Energy-Efficient Lightweight IoT. IACR Trans. Symmetric Cryptol. **2021**(2), 446–468 (2021). https://doi.org/10.46586/TOSC.V2021.I2.446-468, https://doi.org/10.46586/tosc.v2021.i2.446-468
33. Gunsing, A., Daemen, J., Mennink, B.: Deck-Based Wide Block Cipher Modes and an Exposition of the Blinded Keyed Hashing Model. IACR Transactions on Symmetric Cryptology **2019**(4), 1–22 (Jan 2020). https://doi.org/10.13154/tosc.v2019.i4.1-22, https://tosc.iacr.org/index.php/ToSC/article/view/8451
34. Guo, C., Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Romulus v1.3. Submission to NIST Lightweight Cryptography (2021), https://romulusae.github.io/romulus/docs/Romulusv1.3.pdf

35. Guo, C., Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Security Proof for Romulus-T (2022), https://romulusae.github.io/romulus/docs/Romulus_T_proof.pdf
36. Guo, C., Iwata, T., Minematsu, K.: New Indifferentiability Security Proof of MDPH Hash Function. IET Inf. Secur. **16**(4), 262–281 (2022). https://doi.org/10.1049/ISE2.12058, https://doi.org/10.1049/ise2.12058
37. Guo, C., Khairallah, M., Peyrin, T.: AET-LR: Rate-1 Leakage-Resilient AEAD based on the Romulus Family (2020), https://thomaspeyrin.github.io/web/assets/docs/slides/Guo-etal-NIST2020_slides.pdf
38. Guo, C., Pereira, O., Peters, T., Standaert, F.: Authenticated Encryption with Nonce Misuse and Physical Leakage: Definitions, Separation Results and First Construction - (Extended Abstract). In: Schwabe, P., Thériault, N. (eds.) Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11774, pp. 150–172. Springer (2019). https://doi.org/10.1007/978-3-030-30530-7_8, https://doi.org/10.1007/978-3-030-30530-7_8
39. Guo, C., Pereira, O., Peters, T., Standaert, F.: Towards Low-Energy Leakage-Resistant Authenticated Encryption from the Duplex Sponge Construction. IACR Trans. Symmetric Cryptol. **2020**(1), 6–42 (2020). https://doi.org/10.13154/TOSC.V2020.I1.6-42, https://doi.org/10.13154/tosc.v2020.i1.6-42
40. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON Family of Lightweight Hash Functions. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6841, pp. 222–239. Springer (2011). https://doi.org/10.1007/978-3-642-22792-9_13, https://doi.org/10.1007/978-3-642-22792-9_13
41. Halevi, S., Rogaway, P.: A Tweakable Enciphering Mode. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2729, pp. 482–499. Springer (2003). https://doi.org/10.1007/978-3-540-45146-4_28, https://doi.org/10.1007/978-3-540-45146-4_28
42. Halevi, S., Rogaway, P.: A Parallelizable Enciphering Mode. In: Okamoto, T. (ed.) Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings. Lecture Notes in Computer Science, vol. 2964, pp. 292–304. Springer (2004). https://doi.org/10.1007/978-3-540-24660-2_23, https://doi.org/10.1007/978-3-540-24660-2_23
43. Hazay, C., López-Alt, A., Wee, H., Wichs, D.: Leakage-Resilient Cryptography from Minimal Assumptions. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7881, pp. 160–176. Springer (2013). https://doi.org/10.1007/978-3-642-38348-9_10, https://doi.org/10.1007/978-3-642-38348-9_10
44. Jean, J., Nikolic, I., Peyrin, T., Seurin, Y.: The Deoxys AEAD Family. J. Cryptol. **34**(3), 31 (2021). https://doi.org/10.1007/S00145-021-09397-W, https://doi.org/10.1007/s00145-021-09397-w

45. Jimale, M.A., Abdullah, N.A., Kiah, M.L.M., Idris, M.Y.I.B., Z'aba, M.R., Jamil, N., Rohmad, M.S.: Sponge-Based Parallel Authenticated Encryption With Variable Tag Length and Side-Channel Protection. IEEE Access **11**, 59661–59674 (2023). https://doi.org/10.1109/ACCESS.2023.3267161, https://doi.org/10.1109/ACCESS.2023.3267161
46. Khairallah, M.: Security of COFB against Chosen Ciphertext Attacks. IACR Trans. Symmetric Cryptol. **2022**(1), 138–157 (2022). https://doi.org/10.46586/TOSC.V2022.I1.138-157, https://doi.org/10.46586/tosc.v2022.i1.138-157
47. Khairallah, M.: A note on -Tweakable HCTR: A BBB Secure Tweakable Enciphering Scheme-. IACR Cryptol. ePrint Arch. p. 600 (2024), https://eprint.iacr.org/2024/600
48. Khairallah, M.: CCA Security with Short AEAD Tags. IACR Commun. Cryptol. **1**(1), 11 (2024). https://doi.org/10.62056/AEVUA69P1, https://doi.org/10.62056/aevua69p1
49. Khairallah, M.: Revisiting Leakage-Resilient MACs and Succinctly-Committing AEAD More Applications of Pseudo-Random Injections. IACR Trans. Symmetric Cryptol. **2025**(1), 211–239 (2025). https://doi.org/10.46586/TOSC.V2025.I1.211-239, https://doi.org/10.46586/tosc.v2025.i1.211-239
50. Kumar, M.: Security of XCB and HCTR. CoRR **abs/2308.06082** (2023). https://doi.org/10.48550/ARXIV.2308.06082, https://doi.org/10.48550/arXiv.2308.06082
51. Liskov, M.D., Rivest, R.L., Wagner, D.A.: Tweakable Block Ciphers. In: Yung, M. (ed.) Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2442, pp. 31–46. Springer (2002). https://doi.org/10.1007/3-540-45708-9_3, https://doi.org/10.1007/3-540-45708-9_3
52. List, E.: TEDT2 - Highly Secure Leakage-Resilient TBC-Based Authenticated Encryption. In: Longa, P., Ràfols, C. (eds.) Progress in Cryptology - LATINCRYPT 2021 - 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6-8, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12912, pp. 275–295. Springer (2021). https://doi.org/10.1007/978-3-030-88238-9_14, https://doi.org/10.1007/978-3-030-88238-9_14
53. Longo, J., Martin, D.P., Oswald, E., Page, D., Stam, M., Tunstall, M.: Simulatable Leakage: Analysis, Pitfalls, and New Constructions. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 223–242. Springer (2014). https://doi.org/10.1007/978-3-662-45611-8_12, https://doi.org/10.1007/978-3-662-45611-8_12
54. Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM Journal on Computing **17**(2), 373–386 (1988), https://epubs.siam.org/doi/abs/10.1137/0217022
55. Martin, D.P.: Leakage-Resilient Symmetric Cryptography: Theory and Practice. Ph.D. thesis, University of Bristol, UK (2016), https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.702904
56. Mennink, B.: XPX: Generalized Tweakable Even-Mansour with Improved Security Guarantees. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Bar-

bara, CA, USA, August 14-18, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 64–94. Springer (2016). https://doi.org/10.1007/978-3-662-53018-4_3, https://doi.org/10.1007/978-3-662-53018-4_3
57. Mennink, B.: Understanding the Duplex and Its Security. IACR Trans. Symmetric Cryptol. **2023**(2), 1–46 (2023). https://doi.org/10.46586/TOSC.V2023.I2.1-46, https://doi.org/10.46586/tosc.v2023.i2.1-46
58. Mohajerani, K., Beckwith, L., Abdulgadir, A., Ferrufino, E., Kaps, J.P., Gaj, K.: SCA Evaluation and Benchmarking of Finalists in the NIST Lightweight Cryptography Standardization Process. Cryptology ePrint Archive, Paper 2023/484 (2023), https://eprint.iacr.org/2023/484
59. Momin, C., Standaert, F., Verhamme, C.: Primitive-Level vs. Implementation-Level DPA Security: a Certified Case Study (Pleading for Standardized Leakage-Resilient Cryptography). IACR Trans. Cryptogr. Hardw. Embed. Syst. **2025**(3), 717–744 (2025). https://doi.org/10.46586/TCHES.V2025.I3.717-744, https://doi.org/10.46586/tches.v2025.i3.717-744
60. Safavi-Naini, R., Lisý, V., Desmedt, Y.: Economically Optimal Variable Tag Length Message Authentication. In: Kiayias, A. (ed.) Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10322, pp. 204–223. Springer (2017). https://doi.org/10.1007/978-3-319-70972-7_11, https://doi.org/10.1007/978-3-319-70972-7_11
61. Shen, Y., Peters, T., Standaert, F.: Multiplex: TBC-Based Authenticated Encryption with Sponge-Like Rate. IACR Trans. Symmetric Cryptol. **2024**(2), 1–34 (2024). https://doi.org/10.46586/TOSC.V2024.I2.1-34, https://doi.org/10.46586/tosc.v2024.i2.1-34
62. Shen, Y., Peters, T., Standaert, F., Cassiers, G., Verhamme, C.: Triplex: an Efficient and One-Pass Leakage-Resistant Mode of Operation. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(4), 135–162 (2022). https://doi.org/10.46586/TCHES.V2022.I4.135-162, https://doi.org/10.46586/tches.v2022.i4.135-162
63. Shrimpton, T., Terashima, R.S.: A Modular Framework for Building Variable-Input-Length Tweakable Ciphers. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I. Lecture Notes in Computer Science, vol. 8269, pp. 405–423. Springer (2013). https://doi.org/10.1007/978-3-642-42033-7_21, https://doi.org/10.1007/978-3-642-42033-7_21
64. Standaert, F., Pereira, O., Yu, Y.: Leakage-Resilient Symmetric Cryptography under Empirically Verifiable Assumptions. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 335–352. Springer (2013). https://doi.org/10.1007/978-3-642-40041-4_19, https://doi.org/10.1007/978-3-642-40041-4_19
65. Standaert, F., Pereira, O., Yu, Y., Quisquater, J., Yung, M., Oswald, E.: Leakage Resilient Cryptography in Practice. In: Sadeghi, A., Naccache, D. (eds.) Towards Hardware-Intrinsic Security - Foundations and Practice, pp. 99–134. Information Security and Cryptography, Springer (2010). https://doi.org/10.1007/978-3-642-14452-3_5, https://doi.org/10.1007/978-3-642-14452-3_5
66. Wang, P., Feng, D., Wu, W.: HCTR: A Variable-Input-Length Enciphering Mode. In: Feng, D., Lin, D., Yung, M. (eds.) Information Security and Cryp-

tology, First SKLOIS Conference, CISC 2005, Beijing, China, December 15-17, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3822, pp. 175–188. Springer (2005). https://doi.org/10.1007/11599548_15, https://doi.org/10.1007/11599548_15

67. Yu, Y., Standaert, F., Pereira, O., Yung, M.: Practical leakage-resilient pseudorandom generators. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010. pp. 141–151. ACM (2010). https://doi.org/10.1145/1866307.1866324, https://doi.org/10.1145/1866307.1866324

Supplementary Material

A Leakage Functions

Throughout the paper, we refer to the leakage function of different primitives, and different leakage models. These models can be summarized by three concepts:

- Leak-free functions: a function or a primitive is leak-free if an adversary/observer cannot learn anything about its execution except what they can infer from its inputs and outputs defined in its syntax and revealed by the oracle. In other words, if the oracle executes a leak-free function and the inputs and/or outputs are only used internally by the oracle, then the adversary learns nothing about the function execution, except what they can infer from the interaction with the oracle.
- Leakage-resilient functions: a function or a primitive is leakage-resilient if its execution leaks some information that is not inferred from the oracle interaction, but is assumed to still be secure despite the leaked information. The leaked information can be defined by an associated leakage function if needed. Leakage resilience of a primitive is typically an assumption: we assume that a TBC is still unpredictable even with leakage, or we assume that a value comparison function is still secure with leakage. In this case, the security is captured by a notion that embodies this assumption. For larger constructions, we need a security proof that reduces the security to smaller primitive that are themselves either leak-free, leakage-resilient or have unbounded leakage. For instance, we can assume that a block cipher has a non-trivial leakage functions. This makes it harder to assume it is indistinguishable from a random permutation, but it maintains a security property such as unpredictability and/or resistance against key recovery despite the leakage.
- Functions with unbounded leakage: These functions are assumed to leak all their internal values, execution flow, etc. Formally, some functions appear in the security proof as monolithic functions where the internals are not specified, but saying a function has unbounded leakage means that we make no assumptions at all about what the adversary can learn when the function is executed. Note that this also means that even if the function's inputs and outputs are hidden inside the oracle, the adversary is assumed to learn them if the function has unbounded leakage.

We refer to [10] for a more in-depth discussion on the different levels and types of leakage assumptions in this context.

Classification of leakage functions

- *Bounded Leakage*: Introduced by Pietrzak in [30], it works by assuming leakage is independent and keeps tracks of decrease of the computation entropy cause by knowledge of the leakage. We do not use bounded leakage in this work.

– *Oracle-free Hard-to-invert Leakage*: Introduced by Yu et al. [67], the goal of this methodology is similar to that of bounded leakage, limitting the power of the leakage function. This is achieved by first making the leakage functions oracle-free, that is, it has no access to the other primitives or the general constructions, only to the inputs of the leaky primitive itself, and second by assuming it's underlying primitives behave in some idealized way. In our particular case, this idealized behaviour might be PRGL1 and PRGℓ1, or 2-up In practice, hard-to-invert leakage can be easily attained with low protections in hardware, and it is not exceedingly challenging in software either.
– *Simulatable Leakage*: Introduced by Standaert [64], the goal is to produce weak and empirically verifiable assumptions from which we can build more robust security proofs. The idea is that if it is possible to build a simulator that can imitate fake leakages, without knowledge of the secret key, then leakage cannot contain secret-revealing information. Nevertheless, proposing simulator for real circuits is for the most part, an open problem.
– *Unbounded Leakage*: In this model everything is leaked, in particular, the adversary is allowed to choose the leakage function, clearly this model can only provide security in primitives that have no direct access to the key. Nevertheless, that does not mean that these primitives cannot handle sensitive inputs. For example, in MaB, the hash functions are modeled with unbounded leakage, but their inputs contain parts of the plaintext and ciphertext, yet CCAM12 is achievable, however it also explains why CCAmL2 is not attainable without reinforcing the construction. Further discussion can be found in [10, 11].

It is worth remarking that despite their different approaches, all models attempt to capture security against SPA and DPA attacks under different practical assumptions.

B Deferred Proofs

B.1 Proof of Theorem 1

Let $\mathcal{A}$ be a CIML1 adversary, we build a STPRPℓ2 adversary $\mathcal{B}$ which runs $\mathcal{A}$. Assume, without loss of generality, that $\mathcal{A}$ terminates after the first successful forgery. If $\mathcal{A}$ makes an encryption query, $\mathcal{B}$ makes the non-challenge after encoding encryption query and returns the response and corresponding leakage to $\mathcal{A}$. Similary, if $\mathcal{A}$ makes a decryption query, except $\mathcal{B}$ drops the leakage.

If $\mathcal{A}$ makes a forgery query, $\mathcal{B}$ first makes a (leak-free) challenge decryption query. Then, it decodes the response. If the response is a valid codeword, $\mathcal{B}$ returns 1 and the game ends. If it is not a valid codeword, then $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. where each attempt is bounded by probability at most δ, and we apply a standard union bound. Then by (1) we have:

$$\mathrm{Adv}^{\mathsf{STPRP}\ell 2}_{\tilde{\Pi}}(\mathcal{B}) \geq \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{B}^{W_{\mathsf{L}},W_{\mathsf{R}}} \Rightarrow 1\right] - \Pr\left[K \xleftarrow{\$} \mathcal{K}, R \xleftarrow{\$} \mathcal{R} : \mathcal{B}^{W_{\mathsf{L}},W_{\mathsf{I}}} \Rightarrow 1\right]. \tag{3}$$

By construction of $\mathcal{B}$, if $\mathcal{A}$ makes a successful forgery, then $\mathcal{B} \Rightarrow 1$, which implies:

$$\Pr\Big[K \xleftarrow{\$} \mathcal{K} : \mathcal{B}^{W_{\mathsf{L}}, W_{\mathsf{R}}} \Rightarrow 1\Big] \geq \mathrm{Adv}^{\mathsf{CIML1}}_{\mathsf{EtE}[\delta, \tilde{\Pi}]}(\mathcal{A})\,. \tag{4}$$

On the other hand, if $\mathcal{B}$ is placed in the real world, we denote by $\mathbf{Forge}^i$ the event that $\mathcal{A}$ forges succesfully on the i-th forgery attempt, and correspondingly, we denote by $\mathbf{Forge}$ the union event that $\mathcal{A}$ forges succesfully, and $\mathbf{Forge}^{<i} = \vee_{j<i}\, \mathbf{Forge}^j$. Then, by a simple union bound argument, we get:

$$\Pr[\mathbf{Forge}] \leq \sum_{i=1}^{q_f} \Pr\big[\mathbf{Forge}^i \;\big|\; \neg\mathbf{Forge}^{<i}\big] = \sum_{i=1}^{q_f} \frac{2^n\delta}{2^n - i} \leq q_f \frac{2^n\delta}{2^n - q_f} \leq 2q_f\delta\,, \tag{5}$$

where the last inequality comes from $q_f \leq 2^{n-1}$. Then, combining (3) to (5) we conclude

$$\mathrm{Adv}^{\mathsf{STPRP}\ell 2}_{\tilde{\Pi}}(\mathcal{B}) \geq \mathrm{Adv}^{\mathsf{CIML1}}_{\mathsf{EtE}[\delta, \tilde{\Pi}]}(\mathcal{A}) - 2q_f\delta\,,$$

or equivalently

$$\mathrm{Adv}^{\mathsf{CIML1}}_{\mathsf{EtE}[\delta, \tilde{\Pi}]}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{STPRP}\ell 2}_{\tilde{\Pi}}(\mathcal{B}) + 2q_f\delta\,,$$

which concludes the proof.

B.2 Proof of Theorem 3

Proof. For $i \in \mathbb{Z}^+$, let $\mathcal{B}_i$ be an $\mathsf{STPRP}\ell 2$ adversary that works as follows:

1. $\mathcal{B}_i$ runs $\mathcal{A}$.
2. As long as $\mathcal{A}$ has made less than i forgery attempts, $\mathcal{B}_i$ simulates the $\mathsf{CIML2}$ game using the encoding scheme and its leaky oracles.
3. At the i^{th} forgery attempt, $\mathcal{B}_i$ sends the forgery to its challenge leak-free decryption oracle.
4. $\mathcal{B}_i$ decodes the response.
5. If the forgery is successful, $\mathcal{B}_i \Rightarrow 1$. Otherwise, $\mathcal{B}_i \Rightarrow 0$.
6. $\mathcal{B}_i$ terminates the game.

First, we consider the ideal world. Consider that the challenge oracle is simulated using lazy sampling. $\mathcal{B}_i$ queries the challenge oracles once and gets a response sampled from a uniform distribution. Thus, the probability that the response is a valid codeword is exactly δ. In other words,

$$\Pr\Big[\mathcal{B}_i^{W_{\mathsf{L}}, W_{\mathsf{I}}} \Rightarrow 1\Big] \leq \delta\,.$$

Next, we analyze the real world. It is clear that if $\mathcal{A}$ wins, then $\exists i,\ s.t.\ 1 \leq i \leq q_f$ so that $\mathcal{B}_i \Rightarrow 1$. Thus,

$$\sum_{i=1}^{q_f} \Pr\Big[\mathcal{B}_i^{W_{\mathsf{L}}, W_{\mathsf{R}}} \Rightarrow 1\Big] \geq \Pr[\mathcal{A} \text{ wins}] = \mathrm{Adv}^{\mathsf{CIML2}}_{\Psi}(\mathcal{B}).$$

Combining the two bounds, we get,

$$\sum_{i=1}^{q_f} \mathrm{Adv}_{\Pi}^{\mathsf{STPRP}\ell 2}(\mathcal{B}_i) = \sum_{i=1}^{q_f} \Pr\left[\mathcal{B}_i^{W_{\mathsf{L}},W_{\mathsf{R}}} \Rightarrow 1\right] - \Pr\left[\mathcal{B}_i^{W_{\mathsf{L}},W_{\mathsf{I}}} \Rightarrow 1\right]$$
$$\geq \mathrm{Adv}_{\Psi}^{\mathsf{CIML2}}(\mathcal{A}) - q_f\delta\,.$$

Let $\mathcal{B}$ be an adversary such that $\mathrm{Adv}_{\Pi}^{\mathsf{STPRP}\ell 2}(\mathcal{B}) = \max_{1\leq j\leq q_f} \mathrm{Adv}_{\Pi}^{\mathsf{STPRP}\ell 2}(\mathcal{B}_j)$. Then,

$$q_f \mathrm{Adv}_{\Pi}^{\mathsf{STPRP}\ell 2}(\mathcal{B}) \geq \sum_{i=1}^{q_f} \mathrm{Adv}_{\Pi}^{\mathsf{STPRP}\ell 2}(\mathcal{B}_i) \geq \mathrm{Adv}_{\Psi}^{\mathsf{CIML2}}(\mathcal{B}) - q_f\delta\,.$$

Thus,

$$\mathrm{Adv}_{\Psi}^{\mathsf{CIML2}}(\mathcal{A}) \leq q_f \mathrm{Adv}_{\Pi}^{\mathsf{STPRP}\ell 2}(\mathcal{B}) + q_f\delta\,.$$

□

B.3 Proof of Theorem 4

Let $\mathcal{B}$ be an adversary. When $\mathcal{A}$ makes an encryption non-challenge query, $\mathcal{B}$ codes the message and forwards it to its leaky encryption oracle, then forwards the response to $\mathcal{A}$. When $\mathcal{A}$ makes a decryption non-challenge query, $\mathcal{B}$ makes the corresponding leaky decryption query, and checks whether the response is a valid codeword. If it is, $\mathcal{B}$ forwards the decoded response to $\mathcal{A}$. Otherwise, it sends $\perp$. In all the previous cases, $\mathcal{B}$ also sends the corresponding leakage to $\mathcal{A}$. When $\mathcal{A}$ makes a challenge query (N, A, M), $\mathcal{B}$ makes the query $(N\|A, \mathsf{encode}(M))$ to its encryption challenge oracle, and forwards the response back to $\mathcal{A}$. At the end of the game, $\mathcal{A}$ returns b'. $\mathcal{B}$ returns the same value.

We note that the difference between the real world and the ideal world is only in the challenge queries. The $\mathsf{CCAM}\ell 2$ adversary $\mathcal{A}$ expects uniformly random strings in the ideal world, while the ideal world of $\mathcal{B}$ returns uniformly random strings up to injectivity (since they are sampled using a TURP). On the other hand, in the reduction $\mathcal{B}$ only queries the forward challenge queries. Thus, we can switch from the ideal world of $\mathcal{B}$ to an intermediate world that returns uniformly random strings as response to challenge queries with a simple PRP-PRF switching hybrid argument with the transition bound

$$\frac{q_c(q_c-1)}{2^{l_{\min}+1}} \leq \frac{q(q-1)}{2^{l_{\min}+1}},$$

where $l_{\min}$ is the minimum ciphertext size, q_c is the number of challenge queries, and q is the total number of queries. This transition means that we can consider the case where the output of the challenge oracle is indistinguishable from random strings. The game is now identical to the $\mathsf{CCAM}\ell 2$ game. We consider an adversary $\mathcal{B}'$, which distinguishes between the real world and the intermediate

ideal world by running the same reduction. We label the security game of $\mathcal{B}'$ as $\mathsf{STPRP}\ell 2^{\star}$. Thus,

$$\mathrm{Adv}_{\Pi}^{\mathsf{STPRP}\ell 2^{\star}}(\mathcal{B}') = \mathrm{Adv}_{\Psi}^{\mathsf{CCAM}\ell 2}(\mathcal{A}).$$

and for any $\mathcal{B}'$, there exists $\mathcal{B}$ such that

$$\mathrm{Adv}_{\Pi}^{\mathsf{STPRP}\ell 2^{\star}}(\mathcal{B}') \leq \mathrm{Adv}_{\Pi}^{\mathsf{STPRP}\ell 2}(\mathcal{B}) + \frac{q(q-1)}{2^{l_{\min}+1}}.$$

Thus,

$$\mathrm{Adv}_{\Psi}^{\mathsf{CCAM}\ell 2}(\mathcal{A}) \leq \mathrm{Adv}_{\Pi}^{\mathsf{STPRP}\ell 2}(\mathcal{B}) + \frac{q(q-1)}{2^{l_{\min}+1}}.$$

B.4 Proof of Theorem 5

We build a sequence of hybrid games. Let E_i be the event that the adversary returns 1 in game G_i.

G0: This game represents the real world, that is, the adversary has access to the construction, with real leakage in the non-challenge queries. The construction is described in Algorithm 1 (left).

G1: We replace the leak-free TBC $\tilde{E}_K$ with a TURP. There exists an adversary $\mathcal{A}_1$ that makes $2q$ queries, such that

$$|\Pr[E_0] - \Pr[E_1]| \leq \mathrm{Adv}_{\tilde{E}}^{\mathsf{STPRP}}(\mathcal{A}_1)\,.$$

G2: We consider an adversary $\mathcal{A}_{\mathsf{cr}}$ that has access to the construction, and forwards $\mathcal{A}$'s queries to the construction and sends back the responses to $\mathcal{A}$. For every query, $\mathcal{A}_{\mathsf{cr}}$ makes two queries to the hash function H: $H(N\|\|A\|\|A\|\|M_r)$ and $H(N\|\|A\|\|A\|\|C_r)$ and adds the input-output pairs to a hash table $\mathcal{H}$, initialized as empty. At the end of the game, $\mathcal{A}_{\mathsf{cr}}$ checks $\mathcal{H}$ and terminates the game if it includes a collision. From the assumption on H,

$$|\Pr[E_1] - \Pr[E_2]| \leq \epsilon_{\mathsf{CR}}.$$

where H is $(\epsilon_{\mathsf{CR}}, t')$-collision-resistant, such that $t' > t + 2qt_H$ and t_H is the time of one call to H. In the remainder of the proof, we set T_w to be the input to H for each hash query.

G3: We replace the composition of H and the TURP with a function $\tilde{F}$ that operates as follows: For each forward query (T_w, Y) (resp. backward query (T_w, Z)), it checks if there is a previous query $\tilde{F}(T_w, Y)$ with an assigned output Z, in which case it returns Z (resp. Y). Otherwise, it returns a uniformly random value. This transition is covered by the PRP-PRF-switch-like result of Halevi and Rogaway [41, Lemma 1], with

$$|\Pr[E_2] - \Pr[E_3]| \leq \frac{4q^2}{2^n}.$$

G4: We terminate the game if one of the following occurs: There exist two queries to $\tilde{F}$ such that

– $\tilde{F}(T_w, Y) = \tilde{F}(T'_w, Y')$ such that $(T_w, Y) \neq (T'_w, Y')$, or
– $\tilde{F}^{-1}(T_w, Z) = \tilde{F}^{-1}(T'_w, Z')$ such that $(T_w, Z) \neq (T'_w, Z')$.

The probability of this event is at most $\binom{2q}{2}/2^n$. Thus,

$$|\Pr[E_3] - \Pr[E_4]| \leq \frac{\binom{2q}{2}}{2^n} \leq \frac{4q^2}{2^n}.$$

We note that this transition is needed although some of the collisions are already covered in the previous transition. This is because the previous transition only deals with collisions of the form $\tilde{F}(T_w, Y) = \tilde{F}(T_w, Y')$ or $\tilde{F}^{-1}(T_w, Z) = \tilde{F}^{-1}(T_w, Z')$, but does not deal with collisions that involve different tweaks.

G5: This game is constructed such that the inputs to X are: (a) uniformly distributed, and (b) collision-free. Let $(N_i, T_i, M_{r_i} \| M_{l_i}, C_{r_i} \| C_{l_i}, \theta_i)$ be the tuple of the i^{th} non-challenge query, augmented with the corresponding θ_i. We define the following bad events:

– $\mathsf{B5_F}$: there exists an encryption (challenge or non-challenge) query (N, A, M_l, M_r) where $(N_i, A_i, \theta_i, C_{r_i}) = (N, A, M_l, M_r)$.
– $\mathsf{B5_B}$: there exists a decryption (challenge or non-challenge) query (N, A, C_l, C_r) where $(N_i, A_i, \theta_i, M_{r_i}) = (N, A, C_l, C_r)$.
– $\mathsf{B5_P}$: there exists an offline query (χ_j, ι_j) such that $\chi_j = (N_i, \theta_i)$.

Additionally, we denote $\mathsf{B5} = \mathsf{B5_F} \vee \mathsf{B5_B} \vee \mathsf{B5_P}$. Intuitively, if an adversary can make some query, where it knows the intermediate value θ, then it can compute the right value, C_r (respectively M_r). In the real world, $C_r = X(\theta_i) \oplus M_r$ (respectively vice-versa), while C_r is sampled randomly in the ideal world, which makes distinguishing trivial. Hence, the game terminates if any of these events occurs.

If either of these events occurs, then there is an adversary $\mathcal{A}_2$ against the $\mathsf{GuessL1}$ security of X. $\mathcal{A}_2$ has offline access to X, runs $\mathcal{A}$, and $\mathcal{A}_2$ is given $X_{\mathsf{L}}(S)$ and its goal is to output some set of size $2q + q_p$ such that S is in this set. We build $\mathcal{A}_2$ as follows:

1. $\mathcal{A}_2$ chooses $i \xleftarrow{\$} \{1, \ldots, q_e + q_d\}$.
2. Next, $\mathcal{A}$ can make a challenge or non-challenge query.
   – If $\mathcal{A}$ makes a challenge query, $\mathcal{A}_2$ merely simulates Π or Π^{-1}, maintaining consistency with G4.
   – Otherwise, $\mathcal{A}$ makes its j-th non-challenge query:
     • If $j{=}i$ then $\mathcal{A}$ changes the output of X to $X(S)$ when simulating Π (or Π^{-1}).
     • If $j{\neq}i$, then $\mathcal{A}$ just simulates Π_{L} or Π_{L}^{-1} faithfully according to G4.

$\mathcal{A}_2$ outputs the set of all the values of M_l, C_l and the inputs corresponding to θ in offline queries to X_{P} as its set of guesses. Unless S appears as the input to X in any faithfully simulated query, the two games (G4 and G5) are identical. Thus, the probability that $\mathsf{B5}$ is triggered in G4 or G5 is identical unless such collision occurs.

For the sake of the proof, we introduce the following notation:

– $\mathsf{SB5}$: the event that $\mathsf{B5}$ is triggered in the simulated game.
– coll_θ: the event that a seed sampled by the challenger of $\mathsf{GuessL1}$ collides with a simulated seed.

We have the following,

$$\begin{aligned}|\Pr[E_4] - \Pr[E_5]| &\leq \sum_{i=1}^{q_e+q_d} \Pr\left[\mathsf{B5}^{i,G4}\right] \\ &\leq (q_e+q_d)\Pr[\mathsf{coll}_\theta] + \sum_{i=1}^{q_e+q_d} \Pr\left[\mathsf{SB5}^{i,G4}\right] \\ &\leq \frac{(q_e+q_d)^2}{2^n} + (q_e+q_d)\mathrm{Adv}^{\mathsf{GuessL1}[2q+q_p]}(\mathcal{A}_2),\end{aligned}$$

where $\mathsf{B5}^{i,G4}$ denotes that the bad event $\mathsf{B5}$ was triggered in game $G4$ at precisely the i-th query, and the same holds for $\mathsf{SB5}$. Essentially, we assume that the first non-challenge query for which $\mathcal{A}$ triggers such bad event is i and $\mathcal{A}_2$ attempts to guess i.

G6: We define the bad event $\mathsf{B6}$: Let θ_i be the input to X in the i^{th} challenge query. $\exists 1 \leq j \leq q_p$, s.t. $\chi_j = \theta_i$, where the adversary has made the offline query (χ_j, N) to X_P prior to the i^{th} challenge query. From the description of the game up to this point, θ_i is sampled uniformly at random and never appeared before. Thus,

$$|\Pr[E_5] - \Pr[E_6]| \leq \sum_{1\leq i\leq q_{ce}+q_{cd}} \Pr[\mathsf{B6}] \leq \frac{(q_{ce}+q_{cd})q_p}{2^n}.$$

Collisions with offline queries issued after the challenge query are handled in G7.

G7: We replace the outputs of X in the challenge queries with random strings. This is equivalent to \$. Thus, there exists a $\mathsf{PRG}\ell 1$ adversary $\mathcal{A}_3$ such that

$$|\Pr[E_6] - \Pr[E_7]| \leq \mathrm{Adv}_X^{\mathsf{PRG}\ell 1}(\mathcal{A}_3),$$

where $\mathcal{A}_3$ makes $q_{ce}+q_{cd}$ challenge queries, q_e+q_d non-challenge queries and q_p offline queries to X_P.

G8: We note that all the responses to the challenge queries are uniformly distributed in G7, and we can replace them with random strings.

$$|\Pr[E_7] - \Pr[E_8]| = 0.$$

G9: We replace the random strings in challenge queries with strings sampled according to the ideal world TURP $\tilde{R}$. We apply [41, Lemma 1]

$$|\Pr[E_8] - \Pr[E_9]| \leq \frac{q(q-1)}{2^{l_{\min}}+1} \leq \frac{q^2}{2^n},$$

where $l_{\min} \geq n$ is the minimum plaintext length in challenge queries. We also use the fact that $(q_{ce}+q_{cd}) \leq q$.

G10, G11 and G12: Now the challenge queries are sampled according to the ideal world. Nevertheless, when we changed $\tilde{E}_K$, first to a TURP in G1 and then to a random function in G3, we modified the leakage oracles. Therefore, G9 is not identical to the ideal world. We only need to undo these changes in the non-challenge queries. This is possible since θ is collision-free. Arguing exactly the same as in those games we get:

$$\begin{aligned} |\Pr[E_9] - \Pr[E_{12}]| &\leq \mathsf{Adv}^{\mathsf{stprp}}_{\tilde{E}}(\mathcal{A}_5) + \frac{(2(q_e+q_d))^2}{2^n} \\ &\leq \mathsf{Adv}^{\mathsf{stprp}}_{\tilde{E}}(\mathcal{A}_1) + \frac{(2(q_e+q_d))^2}{2^n}\,, \end{aligned}$$

where the last inequality holds because $\mathcal{A}_5$ makes a subset of the queries made by $\mathcal{A}_1$. We can simplify the bound by noting that

$$\frac{9q^2}{2^n} + \frac{(2(q_e+q_d))^2}{2^n} + \frac{(q_e+q_d)^2}{2^n} \leq \frac{14q^2}{2^n}\,.$$

Finally, G12 *is* identical to the (collision-free) ideal world, which concludes the proof.

B.5 Proof of Theorem 6

We build a sequence of hybrid games. Let E_i be the event that the adversary forges against game G_i. W.l.o.g., we assume $\mathcal{A}$ terminates after the first successful forgery, if any.

G0: The real-world construction with early abort: during a forgery attempt, the verifier checks if the first t bits of the output of X are equal to the first t bits of C_r. If not, it terminates the algorithm.

G1: We replace the leak-free TBC $\tilde{E}_K$ with a TURP. There exists an adversary $\mathcal{A}_1$, such that

$$|\Pr[E_0] - \Pr[E_1]| \leq \mathrm{Adv}^{\mathsf{STPRP}}_{\tilde{E}}(\mathcal{A}_1).$$

G2: We consider an adversary $\mathcal{A}_{\mathsf{cr}}$ that has access to the construction, and forwards $\mathcal{A}$'s queries to the construction and sends back the responses to $\mathcal{A}$. For every encryption query, $\mathcal{A}_{\mathsf{cr}}$ makes two queries to the hash function H: $H(N\|\|A\|\|A\|\|0^t\|\|M_r)$ and $H(N\|\|A\|\|A\|\|C_r)$ and adds the input-output pairs to a hash table $\mathcal{H}$. For every forgery attempt, $\mathcal{A}_{\mathsf{cr}}$ makes a query $H(N\|\|A\|\|A\|\|C_r)$ and adds it to $\mathcal{H}$. At the end of the game, $\mathcal{A}_{\mathsf{cr}}$ checks $\mathcal{H}$ and terminates the game if it includes a collision. From the assumption on H,

$$|\Pr[E_1] - \Pr[E_2]| \leq \epsilon_{\mathsf{CR}}.$$

where H is $(\epsilon_{\mathsf{CR}}, t')$-collision-resistant, such that $t' > t + (2q_e + q_f)t_H$ and t_H is the time of one call to H. In the remainder of the proof, we set T_w to be the input to H for each hash query.

G3: We replace the composition of H and the TURP with a function $\tilde{F}$ that operates as follows: For each forward query (T_w, Y) (resp. backward query (T_w, Z)), it checks if there is a previous query such that $\tilde{F}(T_w, Y) = Z$, in which case it returns Z (resp. Y). Otherwise, it returns a uniformly random value. This transition is covered by the result of [41, Lemma 1], with

$$|\Pr[E_2] - \Pr[E_3]| \leq \frac{(2q_e + q_f)^2}{2^n}.$$

G4: We terminate the game if there exist two queries to $\tilde{F}$ such that $\tilde{F}(T_w, Y) = \tilde{F}(T'_w, Y')$ such that $(T_w, Y) \neq (T'_w, Y')$, or $\tilde{F}^{-1}(T_w, Z) = \tilde{F}^{-1}(T'_w, Z')$ such that $(T_w, Z) \neq (T'_w, Z')$. The probability of this event is at most $\binom{2q_e+q_f}{2}/2^n$. Thus,

$$|\Pr[E_3] - \Pr[E_4]| \leq \frac{\binom{2q_e+q_f}{2}}{2^n}.$$

We note that this transition is needed although some of the collisions are already covered in the previous transition. This is because the previous transition only deals with collisions of the form $\tilde{F}(T_w, Y) = \tilde{F}(T_w, Y')$ or $\tilde{F}^{-1}(T_w, Z) = \tilde{F}^{-1}(T_w, Z')$, but does not deal with collisions that involve different tweaks.

G5: This game is constructed such that the inputs to X are: (a) uniformly distributed, and (b) collision-free. Let $i \leq q_f$ be the i^{th} forgery attempt. We define the bad events:

- $\mathsf{B5a}$: there exists an encryption query (N, A, M_l, M_r) where

$$(N_i, A_i, \theta_i, C_{r_i}) = (N, A, M_l, M_r)\,.$$

- $\mathsf{B5b}$: Let θ_i be the input to X in the i^{th} forgery attempt. $\exists 1 \leq j \leq q_p$, s.t. $\theta_j = \theta_i$, where the adversary has made the offline query (θ_j, N) to X_{P} prior to the i^{th} forgery attempt.

The game terminates if any of these events occurs. If either of these events occur, then there is an adversary $\mathcal{A}_3$ against the $\mathsf{GuessL1}$ security of X. $\mathsf{B5a_i}$ means that $\mathsf{B5a}$ occurs at the i^{th} forgery attempt. $\mathcal{A}_3$ has offline access to X, and can simulate $\tilde{F}$. $\mathcal{A}_3$ simulates G5, except it replaces one of the simulated queries to X with the query to its X $\mathsf{GuessL1}$ challenger. If $\mathsf{B5a}$ occurs in this simulated game, then the correct guess for the replaced query is M_l of the encryption query that triggered the event.

The analysis of $\mathsf{B5a}$ game is similar to the analysis of G5 in the proof of Theorem 5, where the $\mathsf{GuessL1}$ adversary replaces one of the encryption queries to include the challenge, and outputs guesses based on the forgery attempts. On the other hand, $\mathsf{B5b}$ can only occur by a random collision, as θ_i is sampled randomly since the adversary has no leakage corresponding to the forgery attempt before generating it. Thus,

$$|\Pr[E_4] - \Pr[E_5]| \leq \frac{q_f q_p}{2^n} + 2q_e \mathrm{Adv}_X^{\mathsf{GuessL1}[q_f]}(\mathcal{A}_3) + \frac{(q_e + q_f)^2}{2^n}\,.$$

G6: We can construct a $\mathsf{PRGL1}$ adversary $\mathcal{A}_4$ that returns 1 if and only if any forgery attempt is successful. We replace all instances of X during encryption and forgery queries such that X is replaced by a random string. Note that we still return the leakage from X. This is equivalent to $\$_{\mathsf{L}}$, thus,

$$|\Pr[E_5] - \Pr[E_6]| \leq \mathrm{Adv}_X^{\mathsf{PRGL1}}(\mathcal{A}_4),$$

where $\mathcal{A}_4$ makes $q_e + q_f$ challenge queries and q_p offline queries to X_{P}. Finally, the probability that the i^{th} forgery attempt is successful is 2^{-t}, since the first t bits are uniformly distributed. Thus,

$$\Pr[E_6] \leq \frac{q_f}{2^t}.$$

B.6 Proof of Theorem 8

We note that in the $\mathsf{CCAmL2}$ game, all the challenge queries must use fresh nonces. Let (N_i, A_i, M_i^0, M_i^1) be the i^{th} challenge query. We construct a reduction as follows: Let $\mathcal{C}$ be an adversary playing the game in Definition 1 with access to Π. At the beginning of the game, it samples its own random bit b'. It receives queries from $\mathcal{A}$. For non-challenge encryption queries, it applies $\mathsf{encode}(M)$ and queries $((N, A), \mathsf{encode}(M))$, then forwards the response and leakage to $\mathcal{A}$. For non-challenge decryption queries, it forwards the query to its non-challenge decryption oracle and, checks if the response if a valid codeword,then forwards the output of decode and leakage to $\mathcal{A}$. For challenge queries, it applies the encoding scheme of Ψ to $M_{b'}$. It forwards $((N, A), \mathsf{encode}(M_{b'}))$ as challenge queries, and forwards the response back to $\mathcal{A}$. If $\mathcal{C}$ is dealing with real challenge leakage, then it is simulating $\mathcal{A}$'s game faithfully. At the end of the game, it checks if $\mathcal{A}$ returns $b^\star = b'$ and returns 1, otherwise it returns 0.

If $\mathcal{C}$ is in its real world (interacting with a leaky oracle), then the game is faithful, $\mathcal{C}$ returns 1 if $\mathcal{A}$ guesses b'. If $\mathcal{C}$ is interacting with the simulator, then $\mathcal{A}$'s game is disturbed, but since Π is $(\epsilon_{\mathsf{sim}}, t, \mathsf{S})$-leakage-simulatable, then we can perform a hybrid transition between the two settings. Let G_0 be the game in which $\mathcal{C}$ is interacting with the leaky challenge oracle, and G_1 be the game in which $\mathcal{C}$ is interacting with the simulator, then

$$\mathrm{Adv}_{\Psi}^{\mathsf{CCAmL2}}(\mathcal{A}) = \mathrm{Adv}_{\Psi}^{\mathsf{CCAmL2}}(\mathcal{A}|G_0) \leq \mathrm{Adv}_{\Psi}^{\mathsf{CCAmL2}}(\mathcal{A}|G_1) + 2q_{ce}\epsilon_{\mathsf{sim}}.$$

The factor two comes from the observation that

$$\Pr[\mathcal{C} \Rightarrow 1|G_x] = 1/2(1 - \Pr[\mathcal{A} \Rightarrow 1|b' = 0 \wedge G_x]) + 1/2\Pr[\mathcal{A} \Rightarrow 1|b' = 1 \wedge G_x]$$

for $x \in \{0, 1\}$ and reordering the terms. Next, we construct an $\mathsf{STPRP}\ell 2$ adversary $\mathcal{B}$ that simulates G_1, and runs S itself (note that S only uses public primitives and no secret information, so any adversary can run it itself). We note that in this game, $\mathcal{B}$ plays its game faithfully by using its challenge oracles to simulate $\mathcal{A}$'s challenge oracles, with polynomial time overhead. We define G_2

as the game where all the challenge queries are made to an independent random TURP $\tilde{R}$. Then,

$$\mathrm{Adv}^{\mathsf{CCAmL2}}_{\Psi}(\mathcal{A}|G_1) \leq \mathrm{Adv}^{\mathsf{CCAmL2}}_{\Psi}(\mathcal{A}|G_2) + \mathrm{Adv}^{\mathsf{STPRP}\ell 2}_{\Pi}(\mathcal{B}).$$

Finally, given all the challenge queries of $\mathcal{A}$ use fresh nonces, then the output of each query to $\tilde{R}$ can be sampled using lazy sampling as a uniformly random string of appropriate length, independent of all the other queries. Thus, $\mathcal{A}$ receives uniformly random challenge ciphertexts that are independent of b', and has no advantage over random guessing:

$$\mathrm{Adv}^{\mathsf{CCAmL2}}_{\Psi}(\mathcal{A}|G_2) = |\Pr[\mathcal{A} \Rightarrow 1|b' = 1] - \Pr[\mathcal{A} \Rightarrow 1|b' = 0]| = 0.$$

C STPRP security of Qaitbay and Alicante

C.1 STPRP security of Qaitbay

Corollary 1. *Let the* ***Qaitbay-2*** *construction be defined as described in Construction 1. Let σ be the sum of the number of r-bit blocks processed by* $\mathtt{Sponge}(0\|\cdot)$ *in all queries. Let σ' be the number of r-bit blocks in M_r in all queries. Let q_p be the number of primitive calls to the permutation used in* Sponge*. Then, for any adversary $\mathcal{A}$ that runs in time at most t and makes q_e, q_d, q_{ce}, q_{cd} queries, as defined in Theorem 5, there exists an adversary $\mathcal{A}_1$ that makes $2(q_e+q_d)$ queries to $\tilde{E}$, and four adversaries $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_4$, and $\mathcal{A}_5$, such that*

$$\begin{aligned}\mathrm{Adv}^{\mathit{STPRP}\ell 2}_{\mathtt{Qaitbay\text{-}2}[\tilde{E},\mathtt{Sponge}]}(\mathcal{A}) \leq\ & 2\mathrm{Adv}^{\mathit{STPRP}}_{\tilde{E}}(\mathcal{A}_1) + \sigma \mathrm{Adv}^{\mathit{GuessL1}[q_g]}_{\mathsf{P}}(\mathcal{A}_2) \\ & + \mathrm{Adv}^{\mathsf{coll}}_{\mathsf{sponge}}(\mathcal{A}_3) + (|\mathcal{S}_b| + 1)\mathrm{Adv}^{\mathsf{pre}}_{\mathsf{sponge}}(\mathcal{A}_4) \\ & + \mathrm{Adv}^{\mathsf{PRG}}_{\mathsf{sponge}}(\mathcal{A}_5) + \frac{22q^2}{2^n},\end{aligned}$$

where $\mathcal{A}_3$ and $\mathcal{A}_4$ are collision and preimage finding adversaries, respectively, for $\mathtt{Sponge}(0\|\cdot)$*, they run in time $O(t)$ and make $2(q_e + q_d)$ queries to* Sponge*. $\mathcal{A}_5$ is a black-box* PRG *adversary against the full-state keyed sponge PRF that runs in time $O(t)$ and makes $2\sigma + 2 + q_p$ primitives queries to P. Where, $q_g = q_p + 2\sigma + 2(q_e + q_d)$, $q = q_e + q_d + q_{ce} + q_{cd}$ and $2\sigma + \sigma' + 3q + q_p < \min\{2^{n/2}, 2^{c/2}\}$.*

Proof. Corollary 1 follows from Theorem 5 as follows: First, we build an adversary that simulates the game given oracle access to P and $\tilde{E}_K$. This adversary terminates the game if any hash output is either 1 or in $\mathcal{S}_b$. This is a preimage-finding adversary against the hash function. Next, we replace $\tilde{E}$ with a TURP. Next, we terminate the game if a collision is found at the output of the hash function. Next, we consider an adversary that simulates the construction except X (the middle application of Sponge), and wins if it guesses any of the secret inputs to P. This follows from Guo et al. [39]. Finally, if all the secret inputs to P never appears in the offline queries to P or during the hash function, then any adversary that breaks the PRGL1 security of X also breaks its black-box security. The advantage is given by [57, Theorem 5]. □

Corollary 1 gives us an indication of what security to expect. As long as $|\mathcal{S}_b|$ is small, P is protected against single-trace attacks, and $\tilde{E}$ is leak-free, then `Qaitbay` is secure as long as (roughly) $(3D+T) < \min\{2^{n/2}, 2^{c/2}\}$. Setting $n = c = 256$ and $|\mathcal{S}_b| \in \{0,1,2\}$, we get 128-bit security. We note that `Qaitbay-1`'s proof for STPRPℓ2 security follows exactly the same, with $|S_b|$ instead of $|S_b|+1$, as the constant 1 just captures the condition that the output of the hash function collides with the tweak of the first TBC call. We note that the LORL2 security of Sponge also follows from the analysis of Guo *et al.* [39]. Thus, `Qaitbay-1` and `Qaitbay-2` have the same level of CIML2 and CCAMℓ2 security (up to 2^t forgeries), and `Qaitbay-2` has the same level of CCAmL2 security.

C.2 STPRP security of `Alicante`

In order to study the security of `Alicante` we need the following assumption:

Assumption 1 *Let $\tilde{E} : \{0,1,2,3\} \times \{0,1\}^{t_w} \times \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be an n-bit ideal cipher with the following leakage functions:*

1. *If $B \in \{0,3\}$, $L(\tilde{E}_X^{B,Y}(M)) = \perp$.*
2. *If $B \in \{1\}$, $L(\tilde{E}_X^{B,Y}(M)) = (B, Y, X, M)$.*
3. *If $B \in \{2\}$, $L(\tilde{E}_X^{B,Y}(M)) = L_{\mathsf{NI}}(B, Y, X, M)$.*

Let $c > 1$ be an integer, and Ω the set of tweakable permutations from $\{0,1\}^{cn} \times \{0,1\}^k \times \{0,1\}^n$ to $\{0,1\}^n$. Additionally, let $\pi_0, \pi_3 \in \Omega$ be independent leak-free tweakable permutations. Let C be a construction parametrized by $B \in \{0,3\}$ and a unique key K sampled uniformly at random and that it also uses $B \in \{1,2\}$ with varying keys. Then, we assume that for any adversary $\mathcal{A}$

$$\left|\Pr\left[K \xleftarrow{\$} \{0,1\}^k \ : \ \mathcal{A}^{C[\mathsf{O}_R^K], \mathsf{O}_R^f} \Rightarrow 1\right] - \Pr\left[\pi_0, \pi_3 \xleftarrow{\$} \Omega \ : \ \mathcal{A}^{\mathsf{O}_I, \mathsf{O}_R^f}\right]\right| \leq \frac{q_p}{2^k},$$

where $\mathsf{O}_R^K = (\tilde{E}_K^0, \tilde{E}^1, \tilde{E}^2, \tilde{E}_K^3)$, $\mathsf{O}_R^f = (\tilde{E}^0, \tilde{E}^1, \tilde{E}^2, \tilde{E}^3)$ and $\mathsf{O}_I = (\pi_0, \tilde{E}^1, \tilde{E}^2, \pi_3)$ and q_p is the number of queries the adversary makes to all its oracles, where the adversary also gets the leakage of their queries.

Corollary 2. *Let the `Alicante-2` construction be defined as described in Algorithm 3 and 4. Let σ_H be the total number of t_w-bit blocks processed by H in all queries. Let σ_X be the number of n-bit blocks in M_r in all queries. Let q_p be the number of primitive calls to the ideal cipher. Then, for any adversary $\mathcal{A}$ that runs in time at most t and makes q_e, q_d, q_{ce}, q_{cd} queries, as defined in Theorem 5,*

$$\begin{aligned}\mathrm{Adv}^{\textit{STPRP}\ell 2}_{\textit{Alicante-2}[\tilde{E}]}(\mathcal{A}) \leq \frac{q_p}{2^k} + \frac{22q^2 + qq_p}{2^n} \\ + \frac{6(6\sigma_H + (3+n)q_p)}{(2^n - 4\sigma_H - 2q_p)^2} + \frac{24n\sigma_H + 12nq_p}{2^n - 4\sigma_H - 2q_p} + \frac{(nq_p + 2q)^2}{2^{2n}} \\ + (q_e + q_d)\mathrm{Adv}^{\textit{2-up}[q_g]}_{\tilde{E}}(\mathcal{A}_1) + 2\sigma_X \mathrm{Adv}^{\textit{2-up}[q_g]}_{\tilde{E}}(\mathcal{A}_2)\,,\end{aligned}$$

where $q_g = q_{IC} + 4\sigma_H$ and $q = q_e + q_d + q_{ce} + q_{cd}$.

The proof follows from Theorem 5 and the techniques of Berti *et al.* [11].

Proof. In G_0, the challenger implements Algorithm 3. In G_1, we replace all the calls where the first parameter of the tweak is 0 and 3 by π_0 and π_3, respectively. The transition advantage is bounded by $q_p/2^k$, according to Assumption 1.

G_2: Similar to the proof of Theorem 5, we now simulate the challenger with oracle access to π_0, π_3, and the ideal cipher. We terminate the game if a collision is found at the output of `MDPH`. The transition advantage follows from [36, Theorem 3.1]. Let the total number of construction queries be q, the total number of $(t_w - n)$-bit blocks processed by H be σ_H, then the probability of such collision occurring is upper bounded by

$$\frac{6(6\sigma_H + (3+n)q_p)}{(2^n - 4\sigma_H - 2q_p)^2} + \frac{24n\sigma_H + 12nq_p}{2^n - 4\sigma_H - 2q_p} + \frac{(nq_p + 2q)^2}{2^{2n}}.$$

Note that each construction calls includes two hash calls, and double the number of blocks.

G_3 and G_4 are identical to the corresponding game in the proof of Theorem 5, with transition advantage bounded by

$$\frac{8q^2}{2^n}.$$

G_5, G_6, G_7: The analysis of these transitions follows from Yu *et al.* [11, Lemma 6]. The transition advantage to G_5 is bounded by

$$\frac{q_e + q_d}{2^n} + (q_e + q_d)\mathrm{Adv}_{\tilde{E}}^{\text{2-up}[q_g]}(\mathcal{A}_1)$$

$q_g = q_{IC} + 4\sigma_h$: q_{IC} is the number of offline ideal cipher calls, and σ_H is the number of blocks absorbed by `MDPH`. The transition advantage to G_6 is bounded by

$$\frac{(q_{ce} + q_{cd})q_p}{2^n}.$$

The transition advantage to G_7 is bounded by

$$(2\sigma_X)\mathrm{Adv}_{\tilde{E}}^{\text{2-up}[q_g]}(\mathcal{A}_2).$$

where σ_X is the number of n-bit blocks of M_r in all queries. The rest of the proof follows from the proof of Theorem 5.

Comments on the proof. This assumption is practical and is common in the proofs of leveled schemes. For instance, the authors of [11] use exactly the same assumption in their analysis where they assume that the keyed, leak-free instances of the TBC can be replaced by ideal random tweakable permutations. In their case, if the key appears in any of the queries to other TBC calls it can be problematic, which leads to a increasing the offline power of the adversaries. We avoid this by domain separation, where even if the key value appeared in a

different query, the domains will be different, and the adversary cannot detect such occurrence using the queries. In principle, the adversary can still detect this occurrence, for example by matching the leakages from the key scheduling function. This is not possible if we restrict The usage of static secret keys to the leak-free domain. Under the ideal cipher assumption, replacing $\tilde{E}_K^{0,T}(M)$ by $\pi_0(K,T,M)$ can be done with a hybrid argument, where the distinguishing game is terminated if the adversary uses the key in any offline query to the ideal cipher, which occurs with probability at most $p/2^k$ after p queries to the ideal cipher. Such assumption is referred to in [11] as the oracle-free leakage assumption; the leakage from one oracle query should not leak the response to a different oracle query.

In the next part of the analysis, we model the leakage function when $B = 1$, such that it leaks all the ideal cipher inputs and outputs. Alternatively, we define the leakage function when $B = 2$ according to the non-invertible leakage assumption of [11, Section 6]. In fact, we use exactly the same stream cipher as [11, Section 6]. Thus, [11, Lemma 6] applies to the middle layer of our design. This gives us a $\mathsf{PRG}\ell\mathsf{1}$ term in the order of $O(D^2/(c \cdot 2^n))$, where D is the total data complexity counted in blocks and c is a small constant based on the side-channel protection of $\tilde{E}$ when $B = 1$. The non-invertible leakage assumption on the ideal cipher in the `2PRG` scheme requires that any key to the ideal cipher appears in only two leaky queries. As we argued in the description `Qaitbay`, the generic analysis of `MaB` does not hold directly as the ideal cipher may leak useful information about the keys even if the tweaks are different. Thus, during the $\mathsf{GuessL1}$, $\mathsf{PRG}\ell\mathsf{1}$ and $\mathsf{PRGL1}$ games, we have to consider the total number of queries made to the ideal cipher and not just the ideal cipher queries in X.

Finally, we rely on the independence between the ideal cipher calls with $B = 1$ vs $B = 2$, and instantiate H using the `MDPH` hash function [36]. Note that the remainder of the analysis follows directly from the security proof of the `Motte-and-Bailey` framework.

□

D Rationale for `Qaitbay`

The `PHOTON` family of hash functions was introduced by Guo *et al.* [40] and has been standardized under ISO/IEC 29192-5:2016. It is based on the Sponge function and has been widely analyzed. The underlying permutation, particularly `PHOTON-256` which we use, was also used as part of `PHOTON-Beetle` [5], a finalist in the NIST lightweight cryptography project. Thus, we believe it is a trusted cryptographic permutation. On the other hand, it does not suffer from some of the implementation shortcomings of other cryptography permutations such as `Keccak-p`, where the inverse of the permutation is not efficient to implement. While it is not particularly masking-friendly, its secure implementations have been studied as part of the NIST lightweight project [58].

`Rijndael` [22] is the precursor to the AES block cipher, and in the initial family of block ciphers, one of the variants uses 256-bit key and block. Despite

not being standardized, it is based on the same internal primitives and a similar internal structure. The block size suits the use case of `Qaitbay` perfectly, and the fact that it is based on the AES primitives means that its masked implementation inherits a lot of the analysis already performed on AES.

Despite their nice features, instantiations based on `PHOTON` and `Rijndael` suffer from two limitations. Both these primitives were not designed to be masking-friendly, and incur a large cost for protecting against side-channel attacks. Besides, all these instantiations provide 128-bit security yet require a 256-bit key. We believe this is acceptable given that these are the first practical leakage-resilient Accordion modes. However, in the long term, we believe a `Qaitbay` variant with a dedicated TBC is an interesting design choice to explore. That TBC would need to be masking-friendly and with parameters: 128-bit key, 256-bit tweak, and 256-bit block.

E Detailed `Alicante-2`

Algorithm 4 `Alicante-2` [`Deoxys-128-640`]. In the case of `Alicante-1`, the first two TBC calls are removed and all instances of γ and γ' are replaced by N. Protected TBC calls are colored in teal. While implicitly increments i.

$\gamma \leftarrow \tilde{E}_K(011 \parallel 0^{509}, N)$
$L \leftarrow \gamma \parallel |A| \parallel A \parallel M_r$
$L_1, \ldots, L_l \leftarrow L \parallel 1 \parallel 0^{508-(|L| \bmod 509)}$ where $|L_i| = 508$
$i \leftarrow 1$
$(H_0, H_1) \leftarrow (0^{128}, 0^{128})$
while $i < l$:
  $Z \leftarrow \tilde{E}_{H_1}(001 \parallel L_i, H_0) \oplus H_0$
  $H_1 \leftarrow \tilde{E}_{H_1}(001 \parallel L_i, H_0 \oplus 0^{127} \parallel 1) \oplus H_0 \oplus 0^{127} \parallel 1$
  $H_0 \leftarrow Z$
$H_0 \leftarrow H_0 \oplus 0^{126} \parallel 10$
$Z \leftarrow \tilde{E}_{H_1}(001 \parallel L_l, H_0) \oplus H_0$
$H_1 \leftarrow \tilde{E}_{H_1}(001 \parallel L_l, H_0 \oplus 0^{127} \parallel 1) \oplus H_0 \oplus 0^{127} \parallel 1$
$H_0 \leftarrow Z$
$\theta \leftarrow \tilde{E}_K(000 \parallel 0^{253} \parallel H_0 \parallel H_1, M_l)$
$M_1, \ldots, M_m \xleftarrow{128} M_r$
$\theta' \leftarrow \theta$
$i \leftarrow 1$
while $i < m$:
  $C_i \leftarrow M_i \oplus \tilde{E}_{\theta'}(101 \parallel 0^{381} \parallel N, 0^{128})$
  $\theta' \leftarrow \tilde{E}_{\theta'}(010 \parallel 0^{381} \parallel N, 0^{127} \parallel 1)$
$C_m \leftarrow M_m \oplus \lfloor \tilde{E}_{\theta'}(101 \parallel 0^{381} \parallel N, 0^{128}) \rfloor_{|M_m|}$
$C_r \leftarrow C_1 \parallel \ldots \parallel C_m$
$\gamma' \leftarrow \tilde{E}_K(100 \parallel 0^{509}, N)$
$L \leftarrow \gamma' \parallel |A| \parallel A \parallel C_r$
$L_1, \ldots, L_l \leftarrow L \parallel 1 \parallel 0^{508-(|L| \bmod 509)}$ where $|L_i| = 508$
$i \leftarrow 1$
$(H_0, H_1) \leftarrow (0^{128}, 0^{128})$
while $i < l$:
  $Z \leftarrow \tilde{E}_{H_1}(001 \parallel L_i, H_0) \oplus H_0$
  $H_1 \leftarrow \tilde{E}_{H_1}(001 \parallel L_i, H_0 \oplus 0^{127} \parallel 1) \oplus H_0 \oplus 0^{127} \parallel 1$
  $H_0 \leftarrow Z$
$H_0 \leftarrow H_0 \oplus 0^{126} \parallel 10$
$Z \leftarrow \tilde{E}_{H_1}(001 \parallel L_l, H_0) \oplus H_0$
$H_1 \leftarrow \tilde{E}_{H_1}(001 \parallel L_l, H_0 \oplus 0^{127} \parallel 1) \oplus H_0 \oplus 0^{127} \parallel 1$
$H_0 \leftarrow Z$
$C_l \leftarrow \tilde{E}_K(000 \parallel 0^{253} \parallel H_0 \parallel H_1, \theta)$
return (C_l, C_r)