A Modular Framework for Building Variable-Input-Length Tweakable Ciphers

Thomas Shrimpton and R. Seth Terashima

Dept. of Computer Science, Portland State University
{teshrim,seth}@cs.pdx.edu

Abstract. We present the Protected-IV construction (PIV) a simple, modular method for building variable-input-length tweakable ciphers. At our level of abstraction, many interesting design opportunities surface. For example, an obvious pathway to building beyond birthday-bound secure tweakable ciphers with performance competitive with existing birthday-bound-limited constructions. As part of our design space exploration, we give two fully instantiated PIV constructions, TCT_1 and TCT_2; the latter is fast and has beyond birthday-bound security, the former is faster and has birthday-bound security. Finally, we consider a generic method for turning a VIL tweakable cipher (like PIV) into an authenticated encryption scheme that admits associated data, can withstand nonce-misuse, and allows for multiple decryption error messages. Thus, the method offers robustness even in the face of certain sidechannels, and common implementation mistakes.

Keywords: tweakable blockciphers, beyond-birthday-bound security, authenticated encryption, associated data, full-disk encryption

The proceedings version of this paper appears in Asiacrypt '13. This is the full version.

1 Introduction

The main contribution of this paper is the Protected-IV construction (PIV), see Figure 1. PIV offers a simple, modular method for building length-preserving, tweakable ciphers that:

(1) may take plaintext inputs of essentially any length;
(2) provably achieves the strongest possible security property for this type of primitive, that of being a strong, tweakable-PRP (STPRP);
(3) admit instantiations from n-bit primitives that are STPRP-secure well beyond the birthday-bound of $2^{n/2}$ invocations.

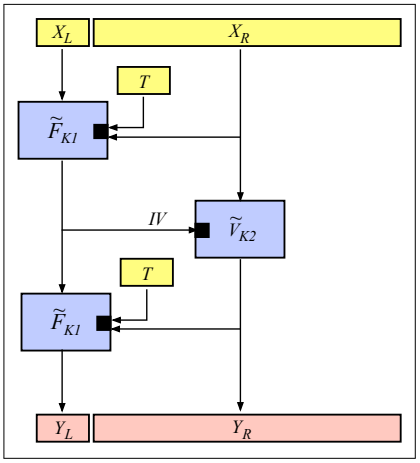

Fig. 1: The $\mathsf{PIV}[\widetilde{F}, \widetilde{V}]$ tweakable cipher. Input T is the tweak, and $X = X_L \,\|\, X_R$ is a bit string, where $|X_L| = N$ and X_R is any length accepted by $\widetilde{V}$. The filled-in box is the tweak input.

Moreover, by some measures of efficiency, beyond-birthday secure instantiations of PIV are competitive with existing constructions that are only secure to the birthday bound. (See Table 1.) We will give a concrete instantiation of PIV that has beyond birthday-bound security and, when compared to EME [20], the overhead is a few extra modular arithmetic operations for each n-bit block of input.

Tweakable ciphers with beyond birthday-bound security may have important implications for cryptographic practice. For example, in large-scale data-at-rest settings, where the amount of data that must be protected by a single key is typically greater than in settings where keys can be easily renegotiated.

At least two important applications have already made tweakable ciphers their tool-of-choice, namely full-disk encryption (FDE) and format-preserving encryption (FPE). Our work provides interesting new results for both FDE and FPE.

We also show that tweakable ciphers enable a simple mechanism for building authenticated encryption schemes with associated data (AEAD), via an extension of the encode-then-encipher approach of Bellare and Rogaway [8]. This approach has some practical benefits, for example, it securely handles the reporting of multiple types of decryption errors. It can also eliminate ciphertext expansion by exploiting any existing nonces, randomness, or redundancies appearing in either the plaintext or associated data inputs. Combined with our other results, encode-then-encipher over PIV gives a new way to build AEAD schemes with beyond birthday-bound security.

Background. Tweakable blockciphers (TBCs) were introduced and formalized by Liskov, Rivest and Wagner [24]. An n-bit TBC $\widetilde{E}$ is a family of permutations over $\{0,1\}^n$, each permutation named by specifying a key and a *tweak*. In typical usage, the key is secret and fixed across many calls, while the tweak is not secret, and may change from call to call; this allows variability in the behavior of the primitive, even though the key is fixed. A tweakable cipher[1] is the natural extension of a tweakable blockcipher to the variable-input-length (VIL) setting, forming a family of length-preserving permutations.

Since the initial work of Liskov, Rivest and Wagner, there has been substantial work on building tweakable ciphers. Examples capable of handling long inputs (required for FDE) include CMC [19], EME [20], HEH [38], HCH [14], and HCTR [41]. Loosely speaking, the common approach has been to build up the VIL primitive from an underlying n-bit blockcipher, sometimes in concert with one or more hashing operations. The security guaranteed by each of these constructions become vacuous after about $2^{n/2}$ bits have been enciphered. One of our main goals is to break through this birthday bound, i.e., to build a tweakable cipher that remains secure long after $2^{n/2}$ bits have been enciphered.

The PIV *construction.* To this end, we begin by adopting a top-down, compositional viewpoint on the design of tweakable ciphers, our PIV construction. It is a type of three-round, unbalanced Feistel network, where the left "half" of the input is of a fixed bit length N, and the right "half" has variable length. The first and third round-functions are an N-bit tweakable blockcipher ($\widetilde{F}$), where N is a parameter of the construction, e.g. $N = 128$ or $N = 256$. The middle round-function ($\widetilde{V}$) is itself a VIL tweakable cipher, whose tweak is the output of first round.

It may seem as though little has been accomplished, since we need a VIL tweakable cipher $\widetilde{V}$ in order to build our VIL tweakable cipher $\mathsf{PIV}[\widetilde{F}, \widetilde{V}]$. However, we require substantially less of $\widetilde{V}$ than we do of $\mathsf{PIV}[\widetilde{F}, \widetilde{V}]$. In particular, the target security property for PIV is that of being a strong tweakable pseudorandom permutation. Informally, being STPRP-secure means withstanding chosen-ciphertext attacks in which the attacker also has full control over all inputs. The attacker can, for example, repeat a tweak an arbitrary number of times. Our PIV security theorem (Theorem 1) says the following: given (1) a TBC $\widetilde{F}$ that is STPRP-secure over a domain of N-bit strings, and (2) a tweakable cipher $\widetilde{V}$ that is secure against attacks *that never repeat a tweak*, then the tweakable cipher $\mathsf{PIV}[\widetilde{F}, \widetilde{V}]$ is STPRP-secure. Thus, qualitatively, the PIV construction promotes security (over a large domain) against a restricted kind of attacker, into security against arbitrary chosen-ciphertext attacks.

Quantitatively, the PIV security bound contains an additive term $q^2/2^N$, where q is the number of times PIV is queried. Now, N might be the blocksize n of some underlying blockcipher; in this case the PIV composition delivers a bound comparable to those achieved by existing constructions. But $N = 2n$ presents the possibility of using an n-bit primitive to instantiate $\widetilde{F}$ and $\widetilde{V}$, and yet deliver a tweakable cipher with security well beyond beyond-birthday of $2^{n/2}$ queries.

As a small, additional benefit, the PIV proof of STPRP-security is short and easy to verify.

Impacts of modularity on instantiations. Adopting this modular viewpoint allows us to explore constructions of $\widetilde{F}$ and $\widetilde{V}$ independently. This is particularly beneficial, since building efficient and secure instantiations of

[1] Sometimes called a "tweakable enciphering scheme", or even a "large-block cipher".

VIL tweakable ciphers ($\widetilde{V}$) is relatively easy, when tweaks can be assumed not to repeat. The more difficult design task, of building a tweakable blockcipher ($\widetilde{F}$) that remains secure when tweaks may be repeated, is also made easier, by restricting to plaintext inputs of a fixed bit length N. In practice, when (say) $N = 128$ or 256, inefficiencies incurred by $\widetilde{F}$ can be offset by efficiency gains in $\widetilde{V}$.

To make thing concrete, we give two fully-specified PIV tweakable ciphers, each underlain by n-bit blockciphers. The first, TCT_1, provides birthday-bound security. It requires only one blockcipher invocation and some arithmetic, modulo a power of two, per n-bit block of input. In contrast, previous modes either require two blockcipher invocations per n-bit block, or require per-block finite field operations.

The second, TCT_2, delivers security beyond the birthday-bound. When compared to existing VIL tweakable ciphers with only birthday-bound security, like EME* construction, TCT_2 incurs only some additional, simple arithmetic operations per n bit block of input. Again, this arithmetic is performed modulo powers of two, rather than in a finite field.

In both TCT_1 and TCT_2, the VIL component is instantiated using counter-mode encryption, but over a TBC instead of a blockcipher. The additional tweak input of the TBC allows us to consider various 'tweak-scheduling' approaches, e.g. fixing a single per-message tweak across all blocks, or changing the tweak each message block.[2] We will see that the latter approach of re-tweaking on a block-by-block basis leads to a beyond birthday-bound secure PIV construction that admits strings of any length at least N.

AEAD via encode-then-(tweakable)encipher. We have already mentioned FDE and FPE as the current, main applications of tweakable ciphers and blockciphers. But the ability to construct beyond birthday-bound secure tweakable ciphers with large and flexible domains motivates us to consider their use for traditional encryption.

Specifically, we build upon the "encode-then-encipher" results of Bellare and Rogaway [8]. They show that messages endowed with randomness (or nonces) and redundancy do not need to be processed by a authenticated encryption (AE) scheme in order to enjoy privacy and authenticity guarantees; a VIL strong-PRP suffices. This is valuable when typical messages are short, as there is no need to waste bandwidth upon transmitting an AE scheme's IV and a dedicated authenticity tag.

We find that the tweakable setting gives additional advantages to the encode-then-encipher approach. First, the tweak can be used effectively to bind a message header to the ciphertext, enabling authenticated encryption with associated data (AEAD). Additionally, one can explore the effects of randomness, state or redundancy present in the message *and* the header inputs. For example, we will see that randomness and state can be shifted to the header without loss of security, potentially reducing the number of bits that must be processed cryptographically.

Our results show that AEAD schemes, built via encode-then-encipher over a tweakable cipher, can accommodate multiple decryption error messages. Multiple, descriptive error messages can be quite useful in practice, but have often empowered damaging attacks (e.g. padding-oracle attacks [40, 11, 32, 2, 16]) against AE schemes using blockcipher modes, CBC-mode in particular. These attacks don't work against our AEAD schemes because, loosely, changing any bit of a ciphertext will randomize every bit of the decrypted string.

Our work in this direction also has important implications for FPE [7, 9]. Motivated by the need to transparently insert an encryption layer into legacy systems with minimal changes to, e.g., database schemas or protocol specifications, FPE requires ciphertexts to have the same format as plaintexts; for example, both might be restricted to the set Σ^n, for some (often non-binary) alphabet Σ. This precludes inserting dedicated IVs into ciphertexts, making encryption deterministic. One compensates by using associated data to tweak encryption, making tweakable ciphers and their security definitions a natural fit. Our work formally proves the intuitive ideas that unique associated data guarantees privacy in a more standard sense, and that checking redundancies, such as credit card Issuer Identification Numbers and checksums, provides authenticated encryption.

[2] There is a natural connection between changing the tweak of a TBC, and changing the key of a blockcipher. Both can be used to boost security, but the former is cleaner because tweaks do not need to be secret.

In addition, the Tor network is considering the use of tweakable ciphers in their onion-encryption scheme [27], where multiple layers of encryption make ciphertext expansion prohibitively expensive, and our results shed light on this application.

Related work. Researchers have developed three general approach for constructing tweakable ciphers from n-bit blockciphers. Each approach has yielded a series of increasingly refined algorithms.

We contribute a new, top-down approach that leads us to the first beyond-birthday-bound secure tweakable cipher suitable for encrypting long inputs (i.e., longer than the blocksize of an underlying blockcipher). Table 1 and Figure 2 compare existing algorithms with our new TCT_1 and TCT_2 constructions in terms of computational cost and security, respectively. Note that the finite field operations counted in Table 1 take hundreds of cycles in software [25, 4], whereas their cost relative to an AES blockcipher invocation is much lower in hardware [26]. TCT_1 is the first tweakable cipher to require only a single blockcipher invocation and no extra finite field multiplications for each additional n bits of input, while TCT_2 is the first to provide beyond-birthday-bound security (and still gets away with a fixed number of finite field multiplications).

	Cost				
Cipher	[BC]	$[\mathbb{F}_{2^n}\times]$	$[\mathbb{Z}_w+]$	$[\mathbb{Z}_{2w}]$	Ref.
HCTR	ℓ	$2\ell+2$	–	–	[41]
CMC	$2\ell+1$	–	–	–	[19]
EME	$2\ell+1$	–	–	–	[20]
EME*	$2\ell+3$	–	–	–	[17]
PEP	$\ell+5$	$4\ell-6$	–	–	[13]
HCH	$\ell+3$	$2\ell-2$	–	–	[14]
TET	ℓ	2ℓ	–	–	[18]
HEH	$\ell+1$	$\ell+2$	–	–	[38, 39]
TCT_1	$\ell+1$	5	$2\ell\left(\frac{n}{w}\right)^2$	$2\ell\left(\frac{n}{w}\right)^2$	–
TCT_2	$2\ell+8$	32	$4\ell\left(\frac{n}{w}\right)^2$	$4\ell\left(\frac{n}{w}\right)^2$	–

Table 1: Tweakable ciphers and their computational costs for ℓn-bit inputs. Costs measured in n-bit blockcipher calls [BC], finite field multiplications $[\mathbb{F}_{2^n}\times]$, and ring operations $[\mathbb{Z}_w+]$ and $[\mathbb{Z}_{2w}]$, for some word size w. Typically, $\ell = 32$ for FDE, and we anticipate $n = 128$, $w = 64$.

The first approach for constructing tweakable ciphers, "encrypt-mix-encrypt", is used by CMC [19], EME [20], and EME* [17], which employ two rounds of encryption separated by a light-weight "mixing layer". CMC is the first in this line of work, and can be used to encrypt strings whose lengths are integral multiples of n. EME improves on CMC by allowing encryption and decryption to be parallelized, and EME* extends the domain to include strings of arbitrary length.

Naor and Reingold [31] proposed the "hash-ECB-hash" approach, which sandwiches a layer of ECB-mode encryption between two invertible hashes. Informally, the role of the hashing layers is to diffuse the input. The PEP [13] mode of operation employs this approach. TET [18] and HEH [38] provide various improvements, notably in terms of performance. In each case, the two hashing layers require finite field multiplications. A variant of HEH described by Sarkar [39], however, manages to halve the number of multiplications that are required.

The final approach is "hash-CTR-hash". Here, the hashing layers are not invertible, but provide the mechanism by which the first output bits become dependent on every input bit. Examples include HCTR [41], which initially offered rather poor security bounds, and HCH, which provides birthday-bound security and requires only a single blockcipher key. Chakraborty and Nandi [12] later gave a birthday-bound-security proof for HCTR.

When blockcipher calls are faster than two finite field multiplications, the encrypt-mix-encrypt approach appears most promising.

We mention the LargeBlock constructions due to Minematsu and Iwata [29], since they provide ciphers with beyond-birthday-bound security. These do not support tweaking, but it seems plausible that they

could without significant degradation of performance or security. These constructions overcome the birthday bound by using $2n$-bit blockciphers as primitives, which are in turn constructed from an n-bit TBC. To our knowledge, CLRW2 [23] is the most efficient n-bit TBC with beyond-birthday-bound security that supports the necessary tweakspace (Minematsu's TBC [28] limits tweak lengths to fewer than $n/2$ bits). Compared to TCT_2, instantiating the LargeBlock constructions with this primitive ultimately requires an extra six finite field multiplications for each n bits of input. Thus, we suspect the LargeBlock designs would be impractical even if adding tweak support proves feasible.

A construction due to Coron, et al. [15], which we refer to as CDMS (after the authors), builds a $2n$-bit TBC from an n-bit TBC, providing beyond-birthday-bound security in n. Like PIV, CDMS uses three rounds of a Feistel-like structure. However, our middle round uses a VIL tweakable cipher, and we require a weaker security property from the round. This allows PIV to efficiently process long inputs. That said, CDMS provides an excellent way to implement a highly-secure $2n$-bit TBC, and we will use it for this purpose inside of TCT_2 to build $\widetilde{F}$. (Nesting CDMS constructions could create $(2^m n)$-bit tweakable blockciphers for any $m > 1$, but again, this would not be practical.) We note that Coron, et al. were primarily concerned with constructions indifferentiable from an ideal cipher, a goal quite different from ours.

The Thorp shuffle [30] and its successor, swap-or-not [21], are highly-secure ciphers targeting very small domains (e.g., $\{0,1\}^n$ for $n \leq 64$). Swap-or-not could almost certainly become a VIL tweakable cipher, without changing the security bounds, by using domain separation for each input length and tweak in the underlying PRF. Essentially, one would make an input-length parameterized family of (tweakable) swap-or-not ciphers, with independent round-keys for each length. While still offering reasonable performance and unmatched security for very small inputs, the result would be wildly impractical for the large domains we are considering: swap-or-not's PRF needs to be invoked at least $6b$ times to securely encipher a b-bit input (below that, the bound becomes vacuous against even $q = 1$ query), and disk sectors are often 4096 bytes. Also, to match TCT_2's security, the PRF itself would need to be secure beyond the birthday bound (with respect to n).

Finally, we note that Rogaway and Shrimpton [37] considered some forms of tweakable encode-then-encipher in the context of deterministic AE ("keywrapping"), and our work generalizes theirs.

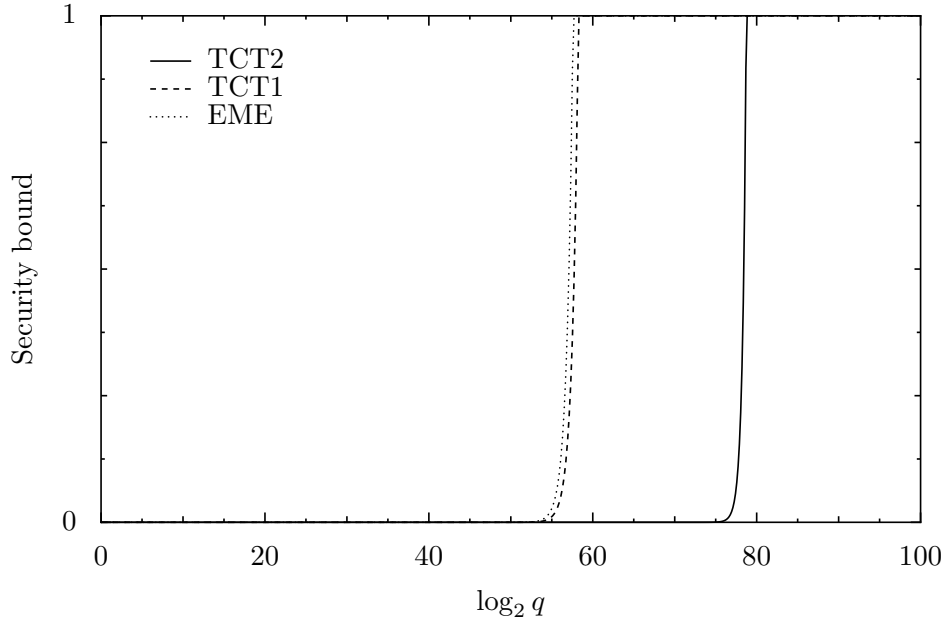

Fig. 2: Security bounds for TCT_1, EME and TCT_2, all using an underlying 128-bit primitive and 4096-bit inputs, typical for FDE. The EME curve is representative of other prior constructions.

2 Tweakable Primitives

Preliminary notation. Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of non-negative integers. For $n \in \mathbb{N}$, $\{0,1\}^n$ denotes the set of all n-bit binary strings, and $\{0,1\}^*$ denotes the set of all (finite) binary strings. We write ε for the empty string. Let $s, t \in \{0,1\}^*$. Then $|s|$ is the length of s in bits, and $|(s,t)| = |s \,\|\, t|$, where $s \,\|\, t$ denotes the string formed by concatenating s and t. If $s \in \{0,1\}^{nm}$ for some $m \in \mathbb{N}$, $s_1 s_2 \cdots s_m \xleftarrow{n} s$ indicates that each s_i should be defined so that $|s_i| = n$ and $s = s_1 s_2 \cdots s_m$. When n is implicit from context, it will be omitted from the notation. If $s = b_1 b_2 \cdots b_n$ is an n-bit string (each $b_i \in \{0,1\}$), then $s[i..j] = b_i b_{i+1} \cdots b_j$, $s[i..] = s[i..n]$, and $s[..j] = s[1..j]$. The string $s \oplus t$ is the bitwise xor of s and t; if, for example, $|s| < |t|$, then $s \oplus t$ is the bitwise xor of s and $t[..|s|]$. Given $R \subseteq \mathbb{N}$ and $n \in \mathbb{N}$ with $n \le \min(R)$, $\{0,1\}^R = \bigcup_{i \in R} \{0,1\}^i$, and by abuse of notation, $\{0,1\}^{R-n} = \bigcup_{i \in R} \{0,1\}^{i-n}$. Given a finite set $\mathcal{X}$, we write $X \xleftarrow{\$} \mathcal{X}$ to indicate that the random variable X is sampled uniformly at random from $\mathcal{X}$. Throughout, the distinguished symbol $\perp$ is assumed not to be part of any set except $\{\perp\}$. Given an integer n known to be in some range, $\langle n \rangle$ denotes some fixed-length (e.g., 64-bit) encoding of n.

Let $H : \mathcal{K} \times \mathcal{D} \to \mathcal{R} \subseteq \{0,1\}^*$ be a function. Writing its first argument as a subscripted key, H is ϵ-almost universal (ϵ-AU) if for all distinct $X, Y \in \mathcal{D}$, $\Pr[\, H_K(X) = H_K(Y)\,] \le \epsilon$ (where the probability is over $K \xleftarrow{\$} \mathcal{K}$). Similarly, H is ϵ-almost 2-XOR universal if for all distinct $X, Y \in \mathcal{D}$ and $C \in \mathcal{R}$, $\Pr[\, H_K(X) \oplus H_K(Y) = C\,] \le \epsilon$.

An adversary is an algorithm taking zero or more oracles as inputs, which it queries in a black-box manner before returning some output. Adversaries may be random. The notation $A^f \Rightarrow b$ denotes the event that an adversary A outputs b after running with oracle f as its input.

Sometimes it will be convenient to give pseudocode descriptions of the oracles with which an adversary can interact, as well as procedures that set up or otherwise relate to such interactions. These descriptions will be referred to as *games* (borrowing from the Bellare-Rogaway game-playing framework [5]). Each game consists of a set of procedures, including a special **Main** procedure that takes an adversary A as input. We define $\Pr[\, \mathsf{G}(A) \Rightarrow y\,]$ as probability that the **Main** procedure of game G outputs y when given an adversary A as input. When no **Main** procedure is explicitly given, **Main**(A) simply executes A, providing it with the remaining procedures as oracles (in some order that will be clear from context), and then returns A's output.

Syntax. Let $\mathcal{K}$ be a non-empty set, and let $\mathcal{T}, \mathcal{X} \subseteq \{0,1\}^*$. A *tweakable cipher* is a mapping $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$ with the property that, for all $(K, T) \in \mathcal{K} \times \mathcal{T}$, $\widetilde{E}(K, T, \cdot)$ is a permutation on $\mathcal{X}$. We typically write the first argument (the key) as a subscript, so that $\widetilde{E}_K(T, X) = \widetilde{E}(K, T, X)$. As $\widetilde{E}_K(T, \cdot)$ is invertible, we let $\widetilde{E}_K^{-1}(T, \cdot)$ denote this mapping. We refer to $\mathcal{K}$ as the *key space*, $\mathcal{T}$ as the *tweak space*, and $\mathcal{X}$ as the *message space*. We say that a tweakable cipher $\widetilde{E}$ is *length preserving* if $|\widetilde{E}_K(T, X)| = |X|$ for all $X \in \mathcal{X}$, $T \in \mathcal{T}$, and $K \in \mathcal{K}$. All tweakable ciphers in this paper will be length preserving. Restricting the tweak or message spaces of a tweakable cipher gives rise to other objects. When $\mathcal{X} = \{0,1\}^n$ for some $n > 0$, then $\widetilde{E}$ is a *tweakable blockcipher* with blocksize n. When $|\mathcal{T}| = 1$, we make the tweak implicit, giving a *cipher* $E \colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$, where $E_K(\cdot)$ is a (length-preserving) permutation over $\mathcal{X}$ and E_K^{-1} is its inverse. Finally, when $\mathcal{X} = \{0,1\}^n$ and $|\mathcal{T}| = 1$, we have a conventional *blockcipher* $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$.

Security notions. Let $\mathrm{Perm}(\mathcal{X})$ denote the set of all permutations on $\mathcal{X}$. Similarly, we define $\mathrm{BC}(\mathcal{K}, \mathcal{X})$ be the set of all ciphers with keyspace $\mathcal{K}$ and message space $\mathcal{X}$. When $\mathcal{X}, \mathcal{X}'$ are sets, we define $\mathrm{Func}(\mathcal{X}, \mathcal{X}')$ to be the set of all functions $f \colon \mathcal{X} \to \mathcal{X}'$. Fix a tweakable cipher $\widetilde{E} \colon \mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$. We define the tweakable pseudorandom-permutation (TPRP) and strong-TPRP (STPRP) advantage measures as

$$\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{E}(A) = \Pr\left[\, K \xleftarrow{\$} \mathcal{K} :\ A^{\widetilde{E}_K(\cdot,\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, \Pi \xleftarrow{\$} \mathrm{BC}(\mathcal{T}, \mathcal{X}) :\ A^{\Pi(\cdot,\cdot)} \Rightarrow 1 \,\right]$$

$$\mathbf{Adv}^{\widetilde{\mathrm{sprp}}}_{E}(A) = \Pr\left[\, K \xleftarrow{\$} \mathcal{K} :\ A^{\widetilde{E}_K(\cdot,\cdot), \widetilde{E}_K^{-1}(\cdot,\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, \Pi \xleftarrow{\$} \mathrm{BC}(\mathcal{T}, \mathcal{X}) :\ A^{\Pi(\cdot,\cdot), \Pi^{-1}(\cdot,\cdot)} \Rightarrow 1 \,\right]$$

where, in each case, adversary A takes the indicated number of oracles. We assume that A never makes *pointless* queries. By this we mean that for both the TPRP and STPRP experiments, the adversary never repeats a query to an oracle. For the STPRP advantage measure, this also means that if A queries (T, X) to its leftmost oracle and receives Y in return, then it never queries (T, Y) to its rightmost oracle, and vice versa. These assumptions are without loss of generality. We define a related measure, the SRND advantage, as

$$\mathbf{Adv}^{\widetilde{\mathrm{srnd}}}_{\widetilde{E}}(A) = \Pr\left[\, K \xleftarrow{\$} \mathcal{K} :\ A^{\widetilde{E}_K(\cdot,\cdot),\widetilde{E}_K^{-1}(\cdot,\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1 \,\right]$$

where the $\$(\cdot,\cdot)$ oracle always outputs a random string equal in length to its second input: $|\$(T, X)| = |X|$ for all T and X. As before, we assume that A never makes a pointless query. Here, these assumptions are not without loss of generality, but instead prevent trivial wins. Adversaries for the above three advantages are *nonce-respecting* if the transcript of their oracle queries $(T_1, X_1), \ldots, (T_q, X_q)$ does not include $T_i = T_j$ for any $i \neq j$.

Now, fix a cipher $E\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$. Then we define the pseudorandom permutation (PRP) and strong-PRP (SPRP) advantage measures

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{prp}}_{E}(A) &= \Pr\left[\, K \xleftarrow{\$} \mathcal{K} :\ A^{E_K(\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, \pi \xleftarrow{\$} \mathrm{Perm}\,(\mathcal{X}) :\ A^{\pi(\cdot)} \Rightarrow 1 \,\right] \\
\mathbf{Adv}^{\mathrm{sprp}}_{E}(A) &= \Pr\left[\, K \xleftarrow{\$} \mathcal{K} :\ A^{E_K(\cdot),E_K^{-1}(\cdot)} \Rightarrow 1 \,\right] - \\
&\qquad \Pr\left[\, \pi \xleftarrow{\$} \mathrm{Perm}\,(\mathcal{X}) :\ A^{\pi(\cdot),\pi^{-1}(\cdot)} \Rightarrow 1 \,\right].
\end{aligned}$$

where, in each case, adversary A takes the indicated number of oracles. We assume without loss of generality, that the adversary does not make pointless queries.

For all security notions in this paper, we track three adversarial resources: the time complexity t, the number of oracle queries q, and the total length of these queries μ. The time complexity of A is defined to include the complexity of its enveloping probability experiment (including sampling of keys, oracle computations, etc.), and we define the parameter t to be the maximum time complexity of A, taken over both experiments in the advantage measure.[3]

3 The PIV Construction

We begin by introducing our high-level abstraction, PIV, shown in Figure 1. Let $\mathcal{T} = \{0,1\}^t$ for some $t \geq 0$, and let $\mathcal{Y} \subseteq \{0,1\}^*$ be such that if $Y \in \mathcal{Y}$, then $\{0,1\}^{|Y|} \subseteq \mathcal{Y}$. Define $\mathcal{T}' = \mathcal{T} \times \mathcal{Y}$. Fix an integer $N > 0$. Let $\widetilde{F}\colon \mathcal{K}' \times \mathcal{T}' \times \{0,1\}^N \to \{0,1\}^N$ be a tweakable blockcipher and let $\widetilde{V}\colon \mathcal{K} \times \{0,1\}^N \times \mathcal{Y} \to \mathcal{Y}$ be a tweakable cipher. From these, we produce a new tweakable cipher $\mathsf{PIV}[\widetilde{F}, \widetilde{V}]\colon (\mathcal{K}' \times \mathcal{K}) \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$, where $\mathcal{X} = \{0,1\}^N \times \mathcal{Y}$. As shown in Figure 1, the PIV composition of $\widetilde{F}, \widetilde{V}$ is a three-round Feistel construction, working as follows. On input (T, X), let $X = X_L \,\|\, X_R$ where $|X_L| = N$. First, create an N-bit string $IV = \widetilde{F}_{K'}(T \,\|\, X_R, X_L)$. Next, use this IV to encipher X_R, creating a string $Y_R = \widetilde{V}_K(IV, X_R)$. Now create an N-bit string $Y_L = \widetilde{F}_{K'}(T \,\|\, Y_R, IV)$, and return $Y_L \,\|\, Y_R$ as the value of $\mathsf{PIV}[\widetilde{F}, \widetilde{V}]_{K',K}(T, X)$. The inverse $\mathsf{PIV}[\widetilde{F}, \widetilde{V}]^{-1}_{K',K}(T, Y)$ is computed in the obvious manner.

At first glance, it seems that nothing interesting has been accomplished: we took an N-bit TBC and a tweakable cipher, and produced a tweakable cipher with a slightly larger domain. However, the following theorem statement begins to surface what our abstraction delivers.

Theorem 1. *Let sets $\mathcal{T}, \mathcal{Y}, \mathcal{T}', \mathcal{X}$ and integer N be as above. Let $\widetilde{F}\colon \mathcal{K}' \times \mathcal{T}' \times \{0,1\}^N \to \{0,1\}^N$ be a tweakable blockcipher, and let $\widetilde{V}\colon \mathcal{K} \times \{0,1\}^N \times \mathcal{Y} \to \mathcal{Y}$ be a tweakable cipher. Let $\mathsf{PIV}[\widetilde{F}, \widetilde{V}]$ be as just*

[3] We do this simply to make our theorem statements easier to read. A more explicit accounting of time resources in reductions, e.g. separating the running time of A from the time to run cryptographic objects "locally", would not significantly alter any of our results.

described. Let A be an adversary making $q < 2^N/4$ queries totaling μ bits and running in time t. Then there exist adversaries B and C, making q and $2q$ queries, respectively, and both running in $O(t)$ time such that $\mathbf{Adv}^{\widetilde{\text{sprp}}}_{\mathsf{PIV}[\widetilde{F},\widetilde{V}]}(A) \le \mathbf{Adv}^{\widetilde{\text{srnd}}}_{\widetilde{V}}(B) + \mathbf{Adv}^{\widetilde{\text{sprp}}}_{\widetilde{F}}(C) + \frac{4q^2}{2^N}$, *where B is* nonce-respecting *and whose queries total $\mu - qN$ bits in length.*

The first thing to notice is that the VIL portion of the PIV composition, $\widetilde{V}$, need be SRND-secure against *nonce-respecting* adversaries only. As we will see in the next section, it is easy to build efficient schemes meeting this requirement. Only the FIL portion, $\widetilde{F}$, needs to be secure against STPRP adversaries that can use arbitrary querying strategies. Thus the PIV composition promotes nonce-respecting security over a large domain into full STPRP security over a slightly larger domain.

The intuition for why this should work is made clear by the picture. Namely, if $\widetilde{F}$ is a good STPRP, then if any part of T or X is "fresh", then the string IV should be random. Hence it is unlikely that an IV value is repeated, and so nonce-respecting security of the VIL component is enough. Likewise when deciphering, if any part of T, Y is "fresh".

The term $4q^2/2^N$ accounts for collisions in IV and the difference between $\widetilde{F}$ and a random function. This is a birthday-bound term in N, the blocksize of $\widetilde{F}$. Since most TBC designs employ (one or more) underlying blockciphers, we have deliberately chosen the notation N, rather than n, to stress that the blocksize of $\widetilde{F}$ can be larger than that of some underlying blockcipher upon which it might be built. Indeed, we'll see in the next section that, given an n-bit blockcipher (and a hash function), we can build $\widetilde{F}$ with $N = 2n$. This gives us hope of building beyond birthday-bound secure VIL STPRPs in a modular fashion; we will do so, and with relatively efficient constructions, too.

It will come as no surprise that, if one does away with the lower $\widetilde{F}$ invocation and returns $IV \parallel Y_R$, the resulting composition does not generically deliver a secure STPRP. On the other hand, it *is* secure as a TPRP (just not a *strong* TPRP). This can be seen through a straight-forward modification of the PIV security proof.

Proof. Fix a message space $\{0,1\}^S$ ($S \subseteq \mathbb{N}$), a tweakspace $\mathcal{T}$, and a non-negative integer $n \le \min(S)$. Let A be an adversary making at most q queries and running in time t. Halevi and Rogaway [20] show that

$$\mathbf{Adv}^{\widetilde{\text{sprp}}}_{\mathsf{PIV}[\widetilde{F},\widetilde{V}]}(A) \le \mathbf{Adv}^{\widetilde{\text{srnd}}}_{\mathsf{PIV}[\widetilde{F},\widetilde{V}]}(A) + \frac{q(q-1)}{2^{\min(S)+1}}.$$

This result is essentially a PRF–PRP switching lemma for TBCs, and reduces our problem to that of bounding $\mathbf{Adv}^{\widetilde{\text{srnd}}}_{\mathsf{PIV}[\widetilde{F},\widetilde{V}]}(A)$.

We begin in the information-theoretic setting, and consider $\mathcal{E}[\widetilde{V}] = \mathsf{PIV}[\widetilde{V}, \Pi]$, where $\Pi \xleftarrow{\$} \mathrm{BC}(N)$ is an ideal cipher. The oracles in Game 1 simulate $\mathcal{E}[\widetilde{V}]$ and $\mathcal{E}[\widetilde{V}]^{-1}$ using lazy sampling to deinfe Π, so $\Pr\left[\, A^{\mathcal{E}[\widetilde{V}],\mathcal{E}[\widetilde{V}]^{-1}} \Rightarrow 1 \,\right] = \Pr\left[\, \mathrm{G1}(A) \Rightarrow 1 \,\right]$.

In Game 2, we no longer resample "illegal" values when defining Π. The only changes in the code occur after a boolean "bad" flag is set to true; by the Fundamental Lemma of Game-Playing,

$$\mathbf{Adv}^{\widetilde{\text{srnd}}}_{\mathcal{E}[\widetilde{V}]}(A) \le \left(\Pr\left[\, \mathrm{G2}(A) \Rightarrow 1 \,\right] - \Pr\left[\, A^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1 \,\right]\right) + \Pr\left[\, \mathrm{G2}(A)\,;\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \vee \mathsf{bad}_3 \,\right]$$

Note that in Game G2, $\widetilde{V}$ is never queried using the same tweak twice. Hence we may consider a third game (not shown), Game G3, in which $\widetilde{V}$ is replaced by an oracle $\$(\cdot,\cdot)$ that always returns a random string equal in length to its second input. By a standard arument, there exists some nonce-respecting adversary B making q queries and running in $\mathcal{O}(t)$ time such that

$$\Pr\left[\, \mathrm{G2}(A) \Rightarrow 1 \,\right] - \Pr\left[\, \mathrm{G3}(A) \Rightarrow 1 \,\right] \le \mathbf{Adv}^{\widetilde{\text{srnd}}}_{\widetilde{V}}(B).$$

We now have

$$\begin{aligned}\mathbf{Adv}^{\widetilde{\text{srnd}}}_{\mathcal{E}[\widetilde{V}]}(A) \le &\left(\Pr\left[\, \mathrm{G3}(A) \Rightarrow 1 \,\right] - \Pr\left[\, A^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1 \,\right]\right) \\ &+ \Pr\left[\, \mathrm{G2}(A)\,;\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \vee \mathsf{bad}_3 \,\right] + \mathbf{Adv}^{\widetilde{\text{srnd}}}_{\widetilde{V}}(B).\end{aligned}$$

However, note that now each the first N bits of each oracle output (corresponding to Z'_i) are always uniformly random in Game G2, and when we switch from $\widetilde{V}$ to $\$(\cdot,\cdot)$ in the next game, the remaining bits also become uniformly random. Hence $\Pr[\,\mathrm{G3}(A) \Rightarrow 1\,] = \Pr[\,A^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1\,]$.

GAMES $\boxed{G1}$, $G2$

Oracle $\mathcal{E}_\ell(T, X)$:

200 $j \leftarrow j + 1$
201 $T_j \leftarrow T \,\|\, X[N+1..]$
202 $IV_j \xleftarrow{\$} \{0,1\}^N \setminus \mathsf{range}(\Pi[T_j])$
203 $\Pi[T_j](X[1..N]) \leftarrow IV_j$
204 $IV \leftarrow IV_j$
205 **if** $IV_j \in \{IV_i : i < j\}$ **then**
206 $\quad \mathsf{bad}_1 \leftarrow \mathsf{true}$
207 $\quad IV_j \xleftarrow{\$} \{0,1\}^N \setminus \{IV_i : i < j\}$
208 $\quad \boxed{IV_j \leftarrow IV}$
209 $Z_i \leftarrow \widetilde{V}[IV_j](X[N+1..])$
210 $T'_j \leftarrow T \,\|\, Z_i$
211 $Z'_i \xleftarrow{\$} \{0,1\}^N$
212 **if** $IV_j \in \mathsf{dom}(\Pi[T'_j])$ **then**
213 $\quad \mathsf{bad}_2 \leftarrow \mathsf{true}$
214 $\quad \boxed{Z'_i \leftarrow \Pi[T'_j](IV_j)}$
215 **else if** $Z'_i \in \mathsf{range}(\Pi[T'_j])$
216 $\quad \mathsf{bad}_3 \leftarrow \mathsf{true}$
217 $\quad \boxed{Z'_i \xleftarrow{\$} \{0,1\}^N \setminus \mathsf{range}(\Pi[T'_j])}$
218 **return** $Z'_i \,\|\, Z'_i$

Oracle $\mathcal{E}_\ell^{-1}(T, Y)$:

219 $j \leftarrow j + 1$
220 $T_j \leftarrow T \,\|\, Y[N+1..]$
221 $IV_j \xleftarrow{\$} \{0,1\}^N \setminus \mathsf{dom}(\Pi[T_j])$
222 $\Pi[T_j]^{-1}(Y[1..N]) \leftarrow IV_j$
223 $IV \leftarrow IV_j$
224 **if** $IV_j \in \{IV_i : i < j\}$ **then**
225 $\quad \mathsf{bad}_1 \leftarrow \mathsf{true}$
226 $\quad IV_j \xleftarrow{\$} \{0,1\}^N \setminus \{IV_i : i < j\}$
227 $\quad \boxed{IV_j \leftarrow IV}$
228 $Z_i \leftarrow \widetilde{V}[IV_j]^{-1}(Y[N+1..])$
229 $T'_j \leftarrow T \,\|\, Z_i$
230 $Z'_i \xleftarrow{\$} \{0,1\}^N$
231 **if** $IV_j \in \mathsf{range}(\Pi[T'_j])$ **then**
232 $\quad \mathsf{bad}_2 \leftarrow \mathsf{true}$
233 $\quad \boxed{Z'_i \leftarrow \Pi[T'_j]^{-1}(IV_j)}$
234 **else if** $Z'_i \in \mathsf{dom}(\Pi[T'_j])$
235 $\quad \mathsf{bad}_3 \leftarrow \mathsf{true}$
236 $\quad \boxed{Z'_i \xleftarrow{\$} \{0,1\}^N \setminus \mathsf{range}(\Pi[T'_j])}$
237 **return** $Z'_i \,\|\, Z'_i$

Fig. 3: Game 1, which includes the boxed statements, simulates $\mathsf{PIV}[\widetilde{V}, \Pi]$ by defining Π through lazy sampling. Game 2, which does *not* include the boxed statements, never invokes $\widetilde{V}$ with the same tweak twice, and the oracles in this game always return values with a random n-bit prefix. All boolean variables are silently initialized to false.

Our final task is to bound the probability that A sets a bad flag in Game G2. The probability that bad_1 is set during query j is less than $j/(2^N - 2j)$. Similarly, the probabilities of bad_2 and bad_3 being set are at most $2j/(2^N - 2j)$ and $2^j/2^N$, respectively. Therefore the probability that at least one flag is set during query j is at most $3j/(2^N - 2j) + 2j/2^N$.

Taking the union bound over $j \in 1, 2, \ldots, q$ gives us $\Pr[\,\mathrm{G1}(A)\,;\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \vee \mathsf{bad}_3\,] \le q^2 \left(\frac{1.5}{2^N - 2q} + \frac{1}{2^N}\right)$. Since $q < 2^N/4$, $1.5/(2^N - 2q) < 3/2^N$. Collecting our previous results and using a standard argument to return to the computational setting completes the proof:

$$\mathbf{Adv}^{\widetilde{\mathrm{sprp}}}_{\mathsf{PIV}[\widetilde{F},\widetilde{V}]}(A) \le \mathbf{Adv}^{\widetilde{\mathrm{srnd}}}_{\widetilde{V}}(B) + \mathbf{Adv}^{\widetilde{\mathrm{sprp}}}_{\widetilde{F}}(C) + \frac{4q^2}{2^N},$$

where C makes $2q$ queries, B makes q queries of total length $\mu - qN$ bits without repeating a tweak, and both run in $\mathcal{O}(t)$ time. □

4 Concrete Instantiations of PIV

Instantiating a PIV composition requires two objects, a (fixed-input-length) tweakable blockcipher $\widetilde{F}$ with an N-bit blocksize, and a variable-input-length tweakable cipher $\widetilde{V}$. In this section we explore various ways to instantiate these two objects, under the guidance of Theorem 1 and practical concerns.

Theorem 1 suggests setting N to be as large as possible, so that the final term is vanishingly small for any realistic number of queries. But for this to be useful, one must already know how to build a TBC $\widetilde{F}$ with domain $\{0,1\}^N$ for a large N, and for which $\mathbf{Adv}^{\widetilde{\text{sprp}}}_{\widetilde{F}}(C)$ approaches $q^2/2^N$. To our knowledge, there are no efficient constructions that permit $\mathbf{Adv}^{\widetilde{\text{sprp}}}_{\widetilde{F}}(C)$ to be smaller than $\mathcal{O}(q^3/2^{2n})$ when using an n-bit blockcipher as a starting point. (A recent result by Lampe and Seurin [22] shows how to beat this security bound, but at a substantial performance cost.) A construction by Coron, et al., which will be discussed in more detail shortly, does meet this bound[4] while providing $N = 2n$.

So we restrict our attention to building TBC $\widetilde{F}$ with small N. In particular, we follow the common approach of building TBCs out of blockciphers. Letting n be the blockcipher blocksize, we will consider $N = n$, and $N = 2n$. In the former case, Theorem 1 only promises us security up to roughly $q = 2^{n/2}$, which is the birthday bound with respect to the blockcipher. With this security bound in mind, we can use simple and efficient constructions of both $\widetilde{F}$ and the VIL tweakable cipher $\widetilde{V}$. On the other hand, when $N = 2n$, Theorem 1 lets us hope for security to roughly $q = 2^n$ queries. To realize this hope we will need a bit more from both $\widetilde{F}$ and $\widetilde{V}$, but we will still find reasonably efficient constructions delivering beyond birthday bound security.

In what follows, we will sometimes refer to objects constructed in other works. These are summarized for convenience in Figure 10, found in Appendix A.

An efficient VIL tweakable cipher. We will start by considering general methods for constructing the VIL tweakable cipher, $\widetilde{V}$. Recall that $\widetilde{V}$ need only be secure against adversaries that never repeat a tweak. In Figure 4, we see an analogue of conventional counter-mode encryption, but over an n-bit TBC $\widetilde{E}$ instead of a blockcipher. Within a call (T, X) to TCTR, each n-bit block X_i of the input X is processed using a

procedure $\text{TCTR}[\widetilde{E}]_K(T, X)$:	**procedure** $\text{TCTR}[\widetilde{E}]_K^{-1}(T, Y)$:
$X_1, X_2, \ldots, X_\nu \xleftarrow{n} X$	$Y_1, Y_2 \ldots, Y_\nu \xleftarrow{n} Y$
for $i = 1$ to ν	for $i = 1$ to ν
$\quad T_i \leftarrow g(T, i)$; $Z_i \leftarrow \langle i \rangle$	$\quad T_i \leftarrow g(T, i)$; $Z_i \leftarrow \langle i \rangle$
$\quad Y_i \leftarrow \widetilde{E}_K(T_i, Z_i) \oplus X_i$	$\quad X_i \leftarrow Y_i \oplus \widetilde{E}_K(T_i, Z_i)$
Return $Y_1, Y_2, \ldots, Y_\nu$	Return $X_1, \ldots, X_\nu$

Fig. 4: The TCTR VIL tweakable cipher.

per-block tweak T_i, this being determined by a function $g\colon \mathcal{T}' \times \mathbb{N} \to \mathcal{T}$ of the input tweak T and the block index i.

Consider the behavior of TCTR when $g(T, i) = T$. The following result is easily obtained using standard techniques.

Theorem 2. *Let $\widetilde{E}\colon \{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher, and let $\text{TCTR}[\widetilde{E}]_K$ and $\text{TCTR}[\widetilde{E}]_K^{-1}$ be defined as above, with $g(T, i) = T \in \mathcal{T}$. Let A be a nonce-respecting adversary that runs in time t, and asks q queries, each of length at most ℓn bits (so, $\mu \le q\ell n$). Then for some adversary B making at most $q\ell$ queries and running in time $\mathcal{O}(t)$, $\mathbf{Adv}^{\widetilde{\text{srnd}}}_{\text{TCTR}[\widetilde{E}]}(A) \le \mathbf{Adv}^{\widetilde{\text{prp}}}_{\widetilde{E}}(B) + 0.5q\ell^2/2^n$.*

[4] However, nesting this construction to provide a VIL tweakable cipher is prohibitively inefficient.

We note that the bound displays birthday-type behavior when $\ell = o(\sqrt{q})$, and is tightest when ℓ is a small constant. An important application with small, constant ℓ is full-disk encryption. Here plaintexts X would typically be 4096 bytes long, so if the underlying TBC has blocksize $n = 128$, we get $\ell = 256$ blocks.[5]

Extending tweakspaces. In PIV, the TBC $\widetilde{F}$ will need to handle long tweaks. Fortunately, a result by Coron, et al. [15] shows that one can compress tweaks using an ϵ-AU hash function at the cost of adding a $q^2\epsilon$ term to the tweakable cipher's TPRP security bound. In particular, we will use (a slight specialization of) the NH hash, defined by Black, et al. [10]; $\mathrm{NH}[r,s]_L$ takes r-bit keys ($|L| = r$), maps r-bit strings to s-bit strings, and is $2^{s/2}$-AU. Please see Table 10 for the description. Given a TBC $\widetilde{E}$, $\widetilde{E}^{\mathrm{NH}}$ denotes the resulting TBC, whose tweakspace is now the domain of NH, rather than its range.

4.1 Targeting efficiency at birthday-type security: TCT_1

Let us begin with the case of $N = n$. To instantiate the n-bit TBC $\widetilde{F}$ in PIV we refer to the pioneering TBC work of Liskov, Rivest and Wagner [24], from which we draw the LRW2 TBC; please refer to Figure 10 for a description.

Before we give the TCT_1 construction, a few notes. In Figure 10 we see that in addition to a blockcipher E, $\mathrm{LRW2}[H,E]$ uses an ϵ-AXU_2 hash function, H, and so, in theory, it could natively accommodate large tweaks. But for practical purposes, it will be more efficient to implement LRW2 with a small tweakspace, and then extend this using a fast ϵ-AU hash function.[6] For the ϵ-AXU_2 hash function itself, we use the polynomial hash polyH (also described in Table 10).

Now are ready to give our TCT_1 construction, which is birthday-bound secure for applications with small plaintext messages (e.g. FDE).

The TCT_1 Construction. Fix $k, n > 0$, and let $N = n$. Let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, and let polyH^{mn}, and NH be as defined in Table 10. Then define $\mathsf{TCT}_1 = \mathsf{PIV}[\widetilde{F}, \widetilde{V}]$, where to obtain a τn-bit tweakspace and domain $\{0,1\}^{\{n,n+1,\ldots,\ell n\}}$ we set:

1. n-bit TBC $\widetilde{F} = \mathrm{LRW2}[\mathrm{polyH}^{2n}, E]^{\mathrm{NH}[(\ell+\tau)n,2n]}$, i.e. LRW2 with its tweakspace extended using NH. The keyspace for $\widetilde{F}$ is $\{0,1\}^k \times \{0,1\}^{2n} \times \{0,1\}^{(\ell+\tau)n}$, with key K' partitioning into keys for E, polyH^{2n}, and $\mathrm{NH}[(\ell+\tau)n, 2n]$. (Since NH supports only fixed length inputs, we implicitly pad NH inputs with a 1 and then as many 0s as are required to reach a total length of $(\ell+\tau)n$ bits.) The tweakspace for $\widetilde{F}$ is $\{0,1\}^{\{0,1,2,\ldots,(\ell+\tau-1)n\}}$.
2. VIL tweakable cipher $\widetilde{V} = \mathrm{TCTR}\,[\mathrm{LRW2}[\mathrm{polyH}^n, E]]$ with the TCTR function $g\colon \{0,1\}^n \times \mathbb{N} \to \{0,1\}^n$ as $g(T,i) = T$. The keyspace for $\widetilde{V}$ is $\{0,1\}^k \times \{0,1\}^n$, with key K partitioning into keys for E and polyH^n. The tweakspace for $\widetilde{V}$ is $\{0,1\}^n$, and its domain is $\{0,1\}^{\{0,1,\ldots,(\ell-1)n\}}$.

Putting together Theorems 1,2, and results from previous works [10, 24], we have the following security bound.

Theorem 3 (STPRP-security of TCT_1). *Define TCT_1 as above, and let A be an adversary making $q < 2^n/4$ queries and running in time t. Then there exist adversaries B and C, both running in time $\mathcal{O}(t)$ and making $(\ell-1)q$ and $2q$ queries, respectively, such that* $\mathbf{Adv}^{\widetilde{\mathrm{sprp}}}_{\mathsf{TCT}_1[E]}(A) \le \mathbf{Adv}^{\mathrm{prp}}_{E}(B) + \mathbf{Adv}^{\mathrm{sprp}}_{E}(C) + \frac{32q^2}{2^n} + \frac{4q^2(\ell-1)^2}{2^n}$.

[5] Actually, slightly less than this when used in the PIV composition, since the first N bits are enciphered by $\widetilde{F}$.

[6] Indeed, one can show composing an ϵ-AU hash function with an ϵ'-AXU_2 hash function yields an $(\epsilon+\epsilon')$-AXU_2 hash function; however, we prefer to work on a higher level of abstraction.

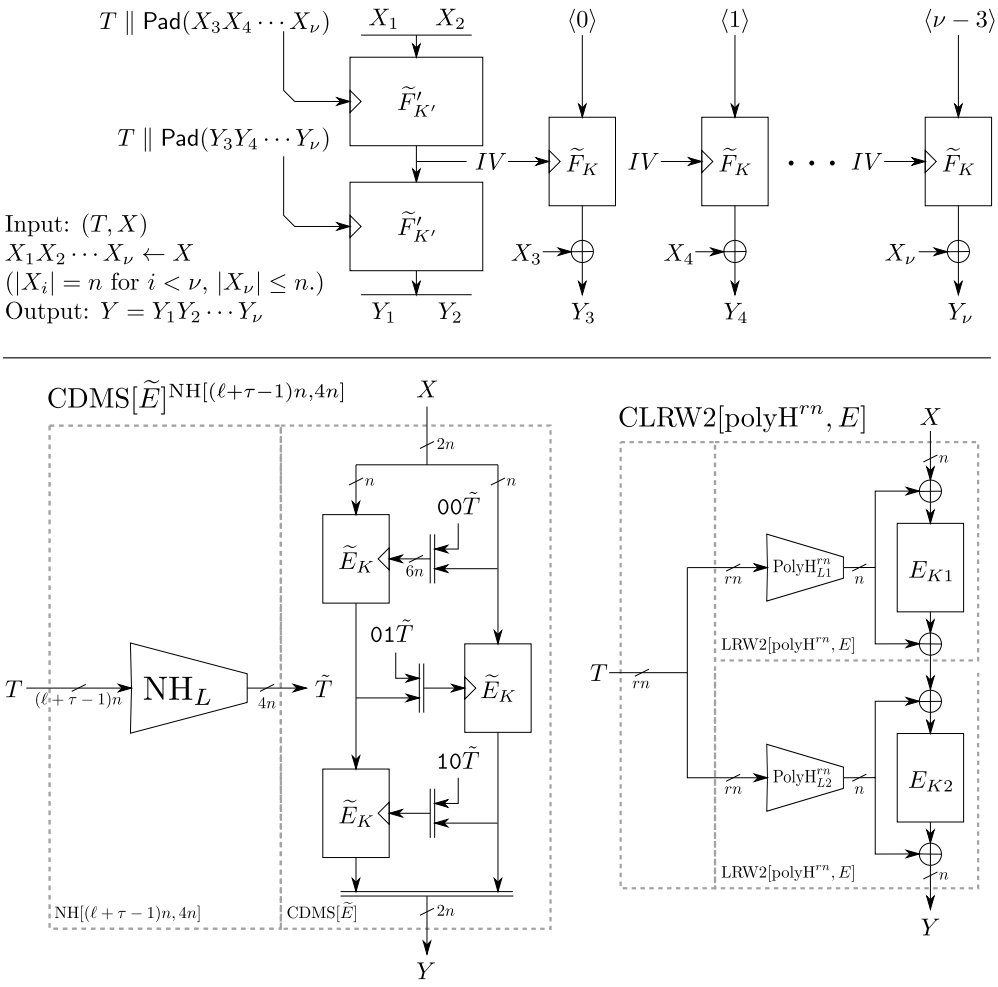

Fig. 5: The TCT_2 construction (top). TCT_2 takes τn-bit tweaks, and the input length is between $2n$ and ℓn bits, inclusive. Here, $\widetilde{F}$ is implemented using the $2n$-bit CDMS construction coupled with the NH hash function (bottom left). Both $\widetilde{V}$ and the TBC $\widetilde{E}$ used inside of CDMS are implemented using $\mathrm{CLRW2}[\mathrm{polyH}^{rn}, E]$ (bottom right), with $r = 6$ and $r = 2$, respectively. The function Pad maps s to $s \parallel 10^{(\ell+1)n-1-|s|}$. In the diagram for CDMS, the strings $00\widetilde{T}$, $01\widetilde{T}$, and $10\widetilde{T}$ are padded with 0s to length $5n$ before being used.

Proof. Using Theorem 1 and security bounds from prior works,

$$\begin{aligned}
\mathbf{Adv}^{\widetilde{\text{sprp}}}_{\mathsf{TCT}_1[E]}(A) &\le \frac{4q^2}{2^n} + \mathbf{Adv}^{\widetilde{\text{srnd}}}_{\widetilde{V}}(t', q) + \mathbf{Adv}^{\text{sprp}}_{\widetilde{F}}(t', 2q) \\
&\le \frac{4q^2}{2^n} + \left[\frac{q(\ell-1)^2}{2^n} + \mathbf{Adv}^{\widetilde{\text{prp}}}_{\text{LRW2}[\text{polyH}^n, E]}(t', (\ell-1)q)\right] + \left[\frac{24q^2}{2^n} + \frac{4q^2}{2^n} + \mathbf{Adv}^{\text{sprp}}_{E}(t', 2q)\right] \\
&\le \frac{4q^2}{2^n} + \left[\frac{q(\ell-1)^2}{2^n} + \frac{3q^2(\ell-1)^2}{2^n} + \mathbf{Adv}^{\text{prp}}_{E}((\ell-1)q, t')\right] + \left[\frac{28q^2}{2^n} + \mathbf{Adv}^{\text{sprp}}_{E}(t', 2q)\right] \\
&\le \frac{32q^2}{2^n} + + \frac{q(\ell-1)^2}{2^n} + \frac{3q^2(\ell-1)^2}{2^n} + \mathbf{Adv}^{\text{prp}}_{E}((\ell-1)q, t') + \mathbf{Adv}^{\text{sprp}}_{E}(t', 2q).
\end{aligned}$$

□

This algorithm requires $2k + (3 + \tau + \ell)n$ bits of key material, including two keys for $\widetilde{E}$. As we show at the end of this section, we can get away with a single key for E with no significant damage to our security bound, although this improvement is motivated primarily by performance concerns.

Thus TCT_1 retains the security of previous constructions (see Figure 2 for a visual comparison), uses arithmetic in rings with powers-of-two moduli, rather than in a finite field. This may potentially improve performance in some architectures.

4.2 Aiming for beyond birthday-bound security: TCT_2

Now let us consider the PIV composition with $N = 2n$. For the FIL component, we can use Coron et al.'s [15] CDMS construction to get a $2n$-bit TBC from an n-bit TBC, and implement the latter using the CLRW2, a recent beyond-birthday-bound secure construction by Landecker, Shrimpton, and Terashima [23]. Table 10 describes both constructions.[7] We again extend the tweakspace using NH. (To stay above the birthday bound, we set the range of NH to $\{0,1\}^{2n}$). Ultimately, setting $\widetilde{F} = \text{CDMS}[\text{CLRW2}]^{\text{NH}}$ is secure against up to around $2^{2n/3}$ queries.

CLRW2 also gives us a way to realize a beyond birthday-bound secure VIL component, namely $\widetilde{V} = \text{TCTR}[\text{CLRW2}[E, H]$, at least for $\ell = o(q^{1/4})$. (We'll see how to avoid this restriction, if desired, in a moment.)

We are now ready to give our second fully concrete PIV composition, TCT_2, targeted at applications that would benefit from beyond birthday-bound security. This algorithm requires us to nest four layers of other constructions, so we provide an illustration in Figure 5. Again we emphasize that the (admittedly significant) cost of $\widetilde{F}$ can be amortized.

TCT_2 supports τn-bit tweaks and has domain $\{0,1\}^{\{2n, 2n+1, \ldots, \ell n\}}$.

The TCT_2 Construction. Fix $k, \ell, n, \tau > 0$, and let $N = 2n$. Let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, and let $\text{polyH}^{\ell n}$, and NH be as defined in Table 10. Then define $\mathsf{TCT}_2 = \mathsf{PIV}[\widetilde{F}, \widetilde{V}]$, where:

1. $\widetilde{F} = \text{CDMS}\left[\text{CLRW2}[\text{polyH}^{6n}, E]\right]^{\text{NH}[(\ell+\tau-1)n, 4n]}$, that is, the $2n$-bit TBC CDMS $\left[\text{CLRW2}[\text{polyH}^{6n}, E]\right]$ with its tweakspace extended using NH. The keyspace for $\widetilde{F}$ is $\{0,1\}^{2k} \times \{0,1\}^{12n} \times \{0,1\}^{(\ell+\tau-1)n}$, with key K' partitioning into two keys for E, two keys for polyH^{6n}, and a key for $\text{NH}[\ell n, 4n]$. The tweakspace for $\widetilde{F}$ is $\{0,1\}^{\tau n}$.
2. $\widetilde{V} = \text{TCTR}\left[\text{CLRW2}[\text{polyH}^{2n}, E]\right]$, with the TCTR function $g\colon \{0,1\}^n \times \mathbb{N} \to \{0,1\}^n$ as $g(T, i) = T$. The keyspace for $\widetilde{V}$ is $\{0,1\}^{2k} \times \{0,1\}^{4n}$ with key K partitioning into two keys for E and two keys for polyH^{2n}. The tweakspace for $\widetilde{V}$ is $\{0,1\}^{2n}$, and its domain is $\{0,1\}^{\{0,1,2,\ldots,(\ell-2)n\}}$.

[7] We note that for CDMS$[\widetilde{E}]$, we enforce domain separation via $\widetilde{E}$'s tweak, whereas the authors of [15] use multiple keys for $\widetilde{E}$. The proof of our construction follows easily from that of the original.

TCT_2 requires $4k + (\ell + \tau + 15)n$ bits of key material. Putting together Theorems 1, 5, and results from previous works [10, 15, 23], we have the following security result.

Theorem 4 (STPRP-security of TCT_2). *Define TCT_2 as above, and let A be an adversary making q queries and running in time t, where $6q, \ell q < 2^{2n}/4$. Then there exist adversaries B and C, both running in $\mathcal{O}(t)$ time and making $(\ell-1)q$ and $6q$ queries, respectively, such that* $\mathbf{Adv}^{\widetilde{\mathrm{sprp}}}_{\mathsf{TCT}_2}(A) \le 2\mathbf{Adv}^{\mathrm{prp}}_{E}(B) + 2\mathbf{Adv}^{\mathrm{sprp}}_{E}(C) + \frac{12q^2}{2^{2n}} + \frac{q(\ell-1)^2}{2^n} + \frac{6\ell^3q^3}{2^{2n-2}-\ell^3q^3} + \frac{6^4q^3}{2^{2n-2}-6^3q^3}$.

Proof. Using Theorem 1 and security bounds from prior works,

$$\begin{aligned}
\mathbf{Adv}^{\widetilde{\mathrm{sprp}}}_{\mathsf{TCT}_2}(A) &\le \frac{4q^2}{2^{2n}} + \mathbf{Adv}^{\widetilde{\mathrm{srnd}}}_{\widetilde{V}}(t', q) + \mathbf{Adv}^{\mathrm{sprp}}_{\widetilde{F}}(t', 2q) \\
&\le \frac{4q^2}{2^{2n}} + \left[\frac{q(\ell-1)^2}{2^n} + \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\mathrm{CLRW2}[H^{2n},E]}(t', (\ell-1)q,)\right] \\
&\qquad + \left[\frac{4q^2}{2^{2n}} + \frac{4q^2}{2^{2n}} + \mathbf{Adv}^{\widetilde{\mathrm{sprp}}}_{\mathrm{CLRW2}[H^{6n},E]}(t', 6q,)\right] \\
&\le \frac{12q^2}{2^{2n}} + \frac{q(\ell-1)^2}{2^n} + \frac{6\ell^3q^3}{2^{2n-2}-\ell^3q^3} + \frac{6^4q^3}{2^{2n-2}-6^3q^3} + 2\mathbf{Adv}^{\mathrm{prp}}_{E}(t', (\ell-1)q) + 2\mathbf{Adv}^{\mathrm{sprp}}_{E}(t', 6q).
\end{aligned}$$

□

Some of the constants in this bound are rather significant. However, as Figure 2 shows, TCT_2 nevertheless provides substantially better security bounds than TCT_1 and previous constructions.

4.3 Additional practical considerations

Several variations and optimizations on TCT_1 and TCT_2 are possible. We highlight a few of them here. None of these changes significantly impact the above security bounds, unless otherwise noted.

Reducing the number of blockcipher keys. In the case of TCT_1, we can use a single key for both LRW2 instances provided we enforce domain separation through the tweak. This allows us to use a single key for the underlying blockcipher, which in some situations may allow for significant implementation benefits (for example, by allowing a single AES pipeline). One method that accomplishes this is to replace $\mathrm{LRW2}[\mathrm{polyH}^{2n}, E]^{\mathrm{NH}[(\ell+1)n,2n]}$ with $\mathrm{LRW2}[\mathrm{polyH}^{3n}, E]^{f(\varepsilon,\cdot)}$ and $\mathrm{LRW2}[\mathrm{polyH}^{n}, E]$ with $\mathrm{LRW2}[\mathrm{polyH}^{3n}, E]^{f(\cdot,\varepsilon)}$. Here, f is a 2^{-n}-AU function with keyspace $\{0,1\}^{3n} \times \{0,1\}^{\ell n}$, taking inputs of the form (X, ε) (for some $X \in \{0,1\}^n$) or (ε, Y) (for some $Y \in \{0,1\}^{\{0,1,\ldots,\ell n\}}$), and outputting a $3n$-bit string. Let $f_L(X, \varepsilon) = 0^{2n} \,\|\, X$ and $f_L(\varepsilon, Y) = 1^n \,\|\, \mathrm{NH}[(\ell+1)n, 2n]_L(Y)$. The function f described here is a mathematical convenience to unify the signatures of the two LRW2 instances, thereby bringing tweak-based domain separation into scope; in practice, we imagine the two instances would be implemented independently, save for a shared blockcipher key. We note that TCT_2 can be modified in a similar manner to require only two blockcipher keys.

Performance optimizations. If we need only a tweakable (FIL) blockcipher, we can use $\mathrm{NH}[\ell n, 2n]$ in place of $\mathrm{NH}[(\ell+1)n, 2n]$ by adjusting our padding scheme appropriately. We emphasize that in the TCTR portion, the polyH functions only need to be computed once, since each LRW2 invocation uses the same tweak. The corresponding optimizations apply to TCT_2, as well.

A naïve implementation of TCT_2 would make a total 72 finite field multiplications during the two FIL phases (a result of evaluating polyH^{6n} twelve times). We can cache an intermediate value of the polyH^{6n} hash used inside of CDMS (four n-bit tweak blocks are constant per invocation), and this saves 32 finite field multiplications. Precomputing the terms of the polynomial hash corresponding to the domain-separation constants eliminates 12 more multiplications, leaving 28 in total. Four more are required during the VIL phase, giving the count of 32 reported in Table 1.

Handling large message spaces. Both TCT_1 and TCT_2 are designed with FDE applications in mind. In particular, they require ℓ to be fixed ahead of time, and require more than ℓn bits of key material.

These limitations are a consequence of using the NH hash function; however, a simple extension to NH (described by the original authors [10]) accommodates arbitrarily long strings. Fix a positive integer r and define $\mathrm{NH}^*_L(M_1 M_2 \cdots M_\nu) = \mathrm{NH}_L(M_1) \,\|\, \mathrm{NH}_L(M_2) \,\|\, \cdots \,\|\, \mathrm{NH}_L(M_\nu) \,\|\, \langle |M| \bmod rn \rangle$, where $|M_i| = rn$ for $i < \nu$, $|M_\nu| \leq rn$, and NH_L abbreviates $\mathrm{NH}_L[rn, 2N]$. Thus defined, NH^* is 2^{-N}-almost universal, has domain $\{0,1\}^*$, and requires rn bits of key material. This modification shifts some of the weight to the polyH hash; we now require eight extra finite field multiplications for each additional rn bits of input. As long as $r > 4$, however, we require fewer of these multiplications when compared to previous hash-ECB-hash or hash-CTR-hash constructions.

With these modifications, the final two terms in TCT_1's security bound (Theorem 3) would become $8q^2/2^n + 600q^2\ell^2/r^2 2^n + 4q^2(\ell-1)^2/2^n$, where ℓn is now the length of the adversary's longest query, $\ell > 2.5r$, and the remaining terms measure the (S)PRP security of the underlying blockcipher. We also assume $2^n \geq rn$, so that $|M| \bmod rn$ can be encoded within a single n-bit block. Although the constant of 600 is large, we note that setting $r = 16$, for example, reduces it to a more comfortable size — in this case to less than three. The bound for TCT_2 changes in a similar manner. (Note that if $2^{n-2} \geq rn$, we can use a single n-bit block for both the tweak domain-separation constants and $\langle |M| \bmod rn \rangle$.)

Beyond birthday-bound security for long messages. When ℓ is not bounded to some small or moderate value, TCT_2 no longer provides beyond-birthday-bound security. The problematic term in the security bound is $q(\ell-1)^2/2^n$. To address this, we return to TCTR (Figure 4) and consider a different per-block tweak function.

In particular, $g(T, i) = T \,\|\, \langle i \rangle$. In the nonce-respecting case, the underlying TBC $\widetilde{E}$ is then retweaked with a never-before-seen value on each message block. Again, think about what happens when $\widetilde{E}$ is replaced by an ideal cipher Π: in the nonce-respecting case, every *block* of plaintext is masked by the output of a fresh random permutation.[8] In other words, every block returned will be uniformly random. Thus we expect a tight bound, in this case. Formalizing this logic yields the following theorem.

Theorem 5. *Let* $\widetilde{E}\colon \{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ *be a tweakable blockcipher, and let* $\mathrm{TCTR}[\widetilde{E}]_K$ *and* $\mathrm{TCTR}[\widetilde{E}]_K^{-1}$ *be defined as above, with* $g\colon \mathcal{T}' \times \mathbb{N} \to \mathcal{T}$ *an arbitrary injective mapping. Let* A *be a nonce-respecting adversary that runs in time* t*, and asks* q *queries of total length at most* $\mu = \sigma n$ *bits. Then there exists some adversary* B *making at most* σ *queries and running in time* $\mathcal{O}(t)$ *such that* $\mathbf{Adv}^{\widetilde{\mathrm{srnd}}}_{\mathrm{TCTR}[\widetilde{E}]}(A) \leq \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(B)$.

Consequently, using this variation of TCTR in Theorems 3 and 4 would remove the $q(\ell-1)^2$ term from the bounds, thereby lifting message length concerns. Note that if this change is made, $g(T, i)$ needs to be computed up to ℓ times per invocation, rather than just once. This problem may be mitigated by using the XEX [33] TBC in place of LRW2, which makes incrementing the tweak extremely fast without significantly changing our security bound.

When the above change are made, TCT_1 and TCT_2 offer efficient tweakable ciphers on an unbounded domain, losing security guarantees only after $\mathcal{O}(2^{n/2})$ (resp., $\mathcal{O}(2^{2n/3})$) bits have been enciphered.

4.4 Instantiating PIV with conventional encryption

To further highlight the flexibility surfaced by our compositional approach, we point out that the VIL component $\widetilde{V}$ can be realized directly using conventional blockcipher-based encryption. Consider the implementation of $\widetilde{V}, \widetilde{V}^{-1}$ shown in Figure 6. We recognize this immediately as counter-mode encryption, but with the initial value T surfaced as an input to make it a tweakable cipher. (Rogaway [34] formalized this as a "nonce-based encryption scheme".) Unfortunately, we cannot use the main PIV security statement, Theorem 1, with this $\widetilde{F}$, because it is not SRND-security against nonce-respecting adversaries. Nonetheless,

[8] Notice that one could use (say) $Z_i \leftarrow 0^n$ and the same would be true. We present it as $Z_i \leftarrow \langle i \rangle$ for expositional purposes.

procedure $\widetilde{V}_K(T,X)$:	**procedure** $\widetilde{V}_K^{-1}(T,Y)$:
$X_1, X_2, \ldots, X_b \xleftarrow{n} X$	$Y_1, Y_2 \ldots, Y_b \xleftarrow{n} Y$
for $i = 1$ to b	for $i = 1$ to b
$\quad Y_i \leftarrow E_K(\langle T + i \rangle) \oplus X_i$	$\quad X_i \leftarrow Y_i \oplus E_K(\langle T + i \rangle)$
Return $Y_1, Y_2, \ldots, Y_b$	Return $X_1, \ldots, X_b$

procedure $\widetilde{G}_K(T,X)$:	**procedure** $\widetilde{G}_K^{-1}(T,Y)$:
$X_1, X_2, \ldots, X_b \xleftarrow{n} X$	$Y_1, Y_2 \ldots, Y_b \xleftarrow{n} Y$
$\overline{K} \leftarrow f_K(T)$	$\overline{K} \leftarrow f_k(T)$
for $i = 1$ to b	for $i = 1$ to b
$\quad Y_i \leftarrow E_{\overline{K}}(\langle T + i \rangle) \oplus X_i$	$\quad X_i \leftarrow Y_i \oplus E_{\overline{K}}(\langle T + i \rangle)$
Return $Y_1, Y_2, \ldots, Y_b$	Return $X_1, \ldots, X_b$

Fig. 6: Two VIL tweakable cipher constructions, based on conventional counter-mode encryption over an n-bit blockcipher E.

examination of the proof of Theorem 1 shows that it can be modified to work. To that end, we introduce a new notion of security $\mathbf{Adv}^{\widetilde{\text{srnd\$}}}_{\widetilde{E}}(A) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\widetilde{\mathbb{E}}_K(\cdot), \widetilde{\mathbb{E}}_K^{-1}(\cdot)} \Rightarrow 1 \right] - \Pr\left[A^{\$(\cdot),\$(\cdot)} \Rightarrow 1 \right]$. Given $\widetilde{E}$ with tweakspace $\mathcal{T}$ and message space $\mathcal{X}$, the oracle $\mathbb{E}_K$ takes a query $X \in \mathcal{X}$ and: (1) samples $T \xleftarrow{\$} \mathcal{T}$, (2) returns $E_K(T, X)$. Oracle $\mathbb{E}_K^{-1}$ behaves likewise on input $Y \in \mathcal{X}$, sampling $T \xleftarrow{\$} \mathcal{T}$ and returning $E_K^{-1}(T, Y)$. With this, one can state an alternative composition theorem.

Lemma 1. *Let sets $\mathcal{T}, \mathcal{Y}, \mathcal{T}', \mathcal{X}$, integer N, TBC $\widetilde{F}$ and tweakable cipher $\widetilde{V}$ be as described in Theorem 1. Let A be an adversary making $q < 2^N/4$ queries and running in time t. Then there exist adversaries B and C, making q and $2q$ queries, respectively, and both running in $O(t)$ time such that $\mathbf{Adv}^{\widetilde{\text{sprp}}}_{\mathsf{PIV}[\widetilde{F},\widetilde{V}]}(A) \leq \frac{5q^2}{2^N} + \mathbf{Adv}^{\widetilde{\text{srnd\$}}}_{\widetilde{V}}(B) + \mathbf{Adv}^{\widetilde{\text{sprp}}}_{\widetilde{F}}(C)$, where B is nonce-respecting.*

Then a standard proof shows that the TBC in Figure 6 is SRND\$-secure, up to a birthday bound, if E is secure as a PRP.

Lemma 2. *Let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, and let $\widetilde{V}$ and $\widetilde{V}^{-1}$ be as in Figure 6. Let A be an adversary running in time t, and ask asking q queries, these totalling at most $\mu = \sigma n$ bits. Then $\mathbf{Adv}^{\widetilde{\text{srnd\$}}}_{\widetilde{V}}(A) \leq \sigma^2/2^n + \mathbf{Adv}^{\text{prp}}_E(B)$, where B asks at most σ queries, and runs in time $\mathcal{O}(t)$.*

Thus, one could compose counter-mode encryption with $\widetilde{F}$ based on LRW2 to build an efficient, birthday-bound-secure STPRP.

Here we point out that there is much similarity between the SIV construction of Rogaway and Shrimpton [37], and PIV using this counter-mode $\widetilde{V}$. Indeed, one can view $\widetilde{F}_{K'}(T \,\|\, X[N+1..], X[1..N))$ as a special kind of PRF (one with invertibility properties), that takes input (T, X) and returns a "synthetic IV" for use in counter-mode. The second application of $\widetilde{F}_{K'}$ then serves to hide this synthetic IV in way that leaves it recoverable to those in possession of key K'. The SIV construction achieves both privacy and authenticity by using the IV as a plaintext authenticator, too. In the next section, we'll look at generic ways to build authenticated encryption with the PIV composition.

As a final topic in this section, we examine a folklore suggestion for increasing the security of this counter-mode construction, namely to "rekey" the underlying blockcipher, using a PRF $f\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^k$ to generate per-message keys, as shown in the construction of $\widetilde{G}, \widetilde{G}^{-1}$ in Figure 6. The idea of rekeying formally addressed by Abdalla and Bellare [1]. Following their lead, we notice that $\widetilde{G}_K(T, X)$ can be computed as (1) $\overline{K} \leftarrow f_K(T)$, (2) Return $\widetilde{F}_{\overline{K}}(T, X)$. Recalling that, for any function $f : \mathcal{K} \times \mathcal{S} \to \mathcal{S}'$, the pseudo-random function (PRF) advantage of adversary A against f is defined to be $\mathbf{Adv}^{\text{prf}}_f(A) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{f_K(\cdot)} \Rightarrow 1 \right] - \Pr\left[\rho \xleftarrow{\$} \text{Func}(\mathcal{S}, \mathcal{S}') : A^{\rho(\cdot)} \Rightarrow 1 \right]$, we have the following theorem.

Theorem 6 (Rekeyed CTR). *Let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, and let $\widetilde{V}$ and $\widetilde{V}^{-1}$ be as just defined. Let $f\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^k$ be a function, and let $\widetilde{G}$ and $\widetilde{G}^{-1}$ be as just defined. Let A be a nonce-respecting adversary that runs in time t, and asks q queries, each of length at most mn*

bits (so, $\mu \le qmn$). Then $\mathbf{Adv}^{\widetilde{\text{srnd\$}}}_{\widetilde{G}}(A) \le q\left(m^2/2^n + \mathbf{Adv}^{\text{prp}}_E(B)\right) + \mathbf{Adv}^{\text{prf}}_f(C)$ *where: (1) B asks at most m queries, and runs in time $\mathcal{O}(t)$, and (2) C asks q queries, and runs in time at most $\mathcal{O}(t)$.*

Thus, if the maximum number of blocks in a message m is small, this rekeying approach seems to have the potential to give beyond birthday-bound SRND-security against nonce-respecting adversaries. In the case of full-disk encryption, say, it seems reasonable to believe that $\mathbf{Adv}^{\text{prp}}_E(t', 32)$ is very small when $n = 128$, so security for $q \gg 2^{64}$ seems possible. Although one needs a PRF f that remains secure for any such q, which is no small assumption.

In any case, the construction of the VIL tweakable cipher $\widetilde{G}$ violates a core principle of the tweakable primitive abstraction, namely that tweak changes should be fast, and much faster than changing the key. But this example offers further support for the thesis put forward by Liskov, Rivest and Wagner, that if one wants to build a tweakable primitive, it is better to start with a tweakable primitive.

5 AEAD from Tweakable Ciphers

We now turn our attention to a new use of PIV (and other tweakable ciphers), that of building authenticated encryption with associated data (AEAD) [35] via a generalization of Bellare and Rogaway's "encode-then-encipher" [8]. Bellare and Rogaway show that when messages are augmented with a nonce and redundancy, one can obtain authenticated encryption [6] simply by passing these encoded messages directly through an SPRP-secure blockcipher with a sufficiently large domain. (Similar tricks do not work if the primitive is merely IND-NM or IND-CCA secure [3].) We revisit encode-then-encipher in the tweakable setting. In particular, we precisely identify the salient properties of the mapping from header-message pairs (H, M) to tweakable cipher inputs (T, X), and explore where best to apportion state, randomness, and redundancy in this encoding.

Our results answer natural questions regarding the relationship between tweakable ciphers and nonce-based encryption. But there remains the question of *why* one would adopt this method, given the existence of highly efficient AEAD schemes, such as OCB [36, 33]. In addition to its simplicity, there are two important, practical advantages of our approach:

(1) It admits the use of multiple decryption error messages, while remaining robust against side-channel attacks that seek to exploit them (e.g. padding oracles).
(2) It can eliminate bandwidth overhead by leveraging nonces, randomness and redundancy already present in plaintext inputs.

The first point holds because, loosely, changing any ciphertext bit results in randomizing every resulting plaintext bit. Thus, descriptive error messages, e.g. `bad_padding` or `bad_seq_number`, cannot be leveraged to forge useful future ciphertexts. We also remark that, under our approach, nonce repetitions only leak equality of plaintexts.

As an example of the second point, messages in protocols that include sequence numbers, human-readable fields, or padding likely contain useful nonces and redundancy. In these cases, the encoding step of encode-then-encipher is implicit, acting as an assumed *model* for how information is encoded into bit strings before being passed down the stack to encrypt.

Before moving on to formal definitions for our encode-then-encipher scheme, we will provide a brief example of what we have in mind in order to motivate certain departures from the standard AEAD formulation. Given a header H and a message M, we need to encode (H, M) into a tweakable cipher input, (T, X). What if we choose a nonce N of some fixed length, set $T = N \,\|\, H$, and $X = N \,\|\, M$? We are already sending H down the wire, but our ciphertext cannot simply be $C = \widetilde{E}_K(T, X)$ because we also need N to decrypt. So our ciphertext should be (N, C). In this example, $T = N \,\|\, H$ was an encoded header, and $X = N \,\|\, M$ was an encoded message. However, we wish to consider encoding schemes separately from tweakable ciphers, so we will discard the symbols T and X in favor of $\overline{H}$ and $\overline{M}$, respectively. The mapping from H to $\overline{H} = N \,\|\, H$ is non-deterministic, but can be completely reproduced provided one knows N. We therefore refer to N as

our *reconstruction information*. (Note that the message encoding function needs the reconstruction information, and therefore has a different signature than the header encoding function; further, the reconstruction information should not depend on M). The recipient computes $\widetilde{E}_K^{-1}(\overline{H}, C)$ and verifies that N is a prefix.

Authenticated encryption with associated data. An *authenticated encryption scheme with associated data* is a tuple $\Psi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consisting of a key-generation algorithm $\mathcal{K}$, an encryption algorithm $\mathcal{E}$, and a decryption algorithm $\mathcal{D}$. For simplicity, we assume the key-generation algorithm $\mathcal{K}$ samples a key from a non-empty set of the same name. The encryption algorithm, which may be randomized or stateful, is a mapping $\mathcal{E} : \mathcal{K} \times \mathcal{H} \times \mathcal{M} \to \mathcal{R} \times \{0,1\}^*$. Thus, encryption takes a key $K \in \mathcal{K}$, a *header* $H \in \mathcal{H} \subseteq \{0,1\}^*$, and a message $M \in \mathcal{M} \subseteq \{0,1\}^*$, and returns some *reconstruction information* $R \in \mathcal{R}$, along with a ciphertext C. We write $(R, C) \xleftarrow{\$} \mathcal{E}_K(H, M)$ to mean running $\mathcal{E}_K$ on (H, M), this returning (R, C). The deterministic decryption algorithm is a mapping $\mathcal{D} : \mathcal{K} \times \mathcal{H} \times \mathcal{R} \times \{0,1\}^* \to \mathcal{M} \cup \mathsf{Errors}$, where Errors is a set such that $\mathcal{M} \cap \mathsf{Errors} = \emptyset$. (We do not insist that $|\mathsf{Errors}| = 1$.) For proper operation, we require that $\Pr[\, \mathcal{D}_K(H, \mathcal{E}_K(H, M)) = M \,] = 1$ for all $K \in \mathcal{K}$, $H \in \mathcal{H}$ and $M \in \mathcal{M}$. If $\mathcal{E}$ is stateful, this must hold for all states.

Let us discuss this formalization of AEAD. First, in practice the header H will be sent in the clear along with the ciphertext (e.g., when the header is needed for routing), but our encode-then-encipher AEAD schemes may encode H into some related $\overline{H}$ for internal use. If this encoding is non-deterministic, we use the reconstruction information R to deliver whatever is needed by decryption to properly reconstruct this $\overline{H}$ from H. For example, R may consist of a counter, some random bits, or some redundancy. (It may also be the empty string.)

To avoid constructions that are trivially insecure by virtue of writing message or key bits into R, we require the following. Given any sequence of inputs $\{(H_i, M_i)\}_{i \le q}$ and any two sequences of possible outcomes $\{(R_i, C_i)\}_{i \le q}$ and $\{(R'_i, C'_i)\}_{i \le q}$, we must have that for any $H \in \mathcal{H}$, $M, M' \in \mathcal{M}$, $K, K' \in \mathcal{K}$, and $r \in \mathcal{R}$, $\Pr[\, R = r \mid \mathcal{E}_K(H_i, M_i) = (R_i, C_i) \text{ for } i \le q \,] = \Pr[\, R' = r \mid \mathcal{E}_{K'}(H'_i, M'_i) = (R'_i, C'_i) \text{ for } i \le q \,]$ where $(R, C) \xleftarrow{\$} \mathcal{E}_K(H, M)$ and $(R', C') \xleftarrow{\$} \mathcal{E}_K(H, M')$, the states of $\mathcal{E}_K$ and $\mathcal{E}_{K'}$ being conditioned on the two transcripts, respectively. That is, R (hence, R') can depend only on H, q, and coins tossed by $\mathcal{E}_K$ on the query that generates R.

Second, by allowing $|\mathsf{Errors}| > 1$, we let our AEAD schemes return multiple, distinct error messages. This can be useful in practice for, say, diagnostics within a protocol session. Often, allowing decryption to return multiple error messages has been problematic in practice; witness the various "padding oracle" attacks on SSL/TLS [40, 11, 32]. For our encode-then-encipher AEAD schemes, such attacks will not be a concern.

AEAD security notions. Our desired privacy notion is indistinguishability of ciphertexts from random bits. However, we do not require the recovery information to be random (e.g., R might be a counter), so we modify the usual IND$-CPA notion slightly. To be specific, we measure the privacy of an AEAD scheme Ψ via the following advantage: $\mathbf{Adv}^{\mathrm{priv}}_{\Psi}(A) = \Pr\left[K \xleftarrow{\$} \mathcal{K} ;\, A^{\mathcal{E}_K(\cdot,\cdot)} \Rightarrow 1 \right] - \Pr\left[K \xleftarrow{\$} \mathcal{K} ;\, A^{\$_K(\cdot,\cdot)} \Rightarrow 1 \right]$. Here, $\$_K(\cdot,\cdot)$ is an oracle that on input (H, M), computes $(R, C) \xleftarrow{\$} \mathcal{E}_K(H, C)$, samples $Y \xleftarrow{\$} \{0,1\}^{|C|}$, and returns (R, Y).

The authenticity goal for our AEAD scheme is integrity of ciphertexts, INT-CTXT [6]. Namely, we define $\mathbf{Adv}^{\text{int-ctxt}}_{\Psi}(A) = \Pr\left[K \xleftarrow{\$} \mathcal{K} ;\, A^{\mathcal{E}_K(\cdot,\cdot), \mathcal{D}_K(\cdot,\cdot)} \text{ } \mathsf{Forges} \right]$ where the boolean event Forges is true if and only if A asks a query (H, R, C) to its $\mathcal{D}_K$ oracle such that (1) $\mathcal{D}_K(H, R, C) \notin \mathsf{Errors}$, and (2) no prior query to $\mathcal{E}_K(\cdot,\cdot)$ returned (H, R, C). Without loss of generality, A halts as soon as Forges becomes true.

5.1 Encoding schemes

Informally, an encoding algorithm is responsible for reformatting its input, injecting randomness, state or redundancy, while decoding validates and distills out the original input data.

Fix a message space $\mathcal{M}$, a header space $\mathcal{H}$, an encoded message space $\overline{\mathcal{M}} \subseteq \{0,1\}^*$, and an encoded header space $\overline{\mathcal{H}} \subseteq \{0,1\}^*$. All of these sets must be non-empty (but could equal $\{\varepsilon\}$). Also fix a set Errors such that $\mathsf{Errors} \cap \mathcal{M} = \emptyset$.

As mentioned earlier, we need two types of encoding functions. A *header encoding function* $\text{Encode}_{\mathsf{H}} : \mathcal{H} \to \overline{\mathcal{H}}$ maps headers to encoded headers, possibly in some random or stateful fashion. We require there to be a non-empty set $\mathcal{R}$ and a bijection $\langle\cdot,\cdot\rangle : \mathcal{R} \times \mathcal{H} \to \overline{\mathcal{H}}$ with the property that for all H, $\text{Encode}_{\mathsf{H}}(H) = \langle R, H\rangle$ for some $R \in \mathcal{R}$. In other words, we should always be able to recover H from $\text{Encode}_{\mathsf{H}}(H)$, and any particular output of $\text{Encode}_{\mathsf{H}}(H)$ can be reconstructed from H given the corresponding R. We call $\text{Encode}_{\mathsf{H}}$ an $(\mathcal{H}, \mathcal{R}, \overline{\mathcal{H}})$-encoder (leaving $\langle\cdot,\cdot\rangle$ implicit).

A *message encoding scheme* $\mathsf{EncodeMsg} = (\text{Encode}_{\mathsf{M}}, \text{Decode}_{\mathsf{M}})$ consists of a *message encoding function* $\text{Encode}_{\mathsf{M}} : \overline{\mathcal{H}} \times \mathcal{M} \to \overline{\mathcal{M}}$ and a *message decoding function* $\text{Decode}_{\mathsf{M}} : \overline{\mathcal{H}} \times \overline{\mathcal{M}} \to \mathcal{M} \cup \mathsf{Errors}$. We allow $\text{Encode}_{\mathsf{M}}$ to be randomized or stateful. We require $\text{Decode}_{\mathsf{M}}(\overline{H}, \overline{M}) = M \in \mathcal{M}$ if and only if $\Pr\left[\,\text{Encode}_{\mathsf{M}}(\overline{H}, M) = \overline{M}\,\right] > 0$ for some state of $\text{Encode}_{\mathsf{M}}$; otherwise, $\text{Decode}_{\mathsf{M}}(\overline{H}, \overline{M}) \in \mathsf{Errors}$ (note that we allow, for example, $\text{Decode}_{\mathsf{M}}(\overline{H}, \overline{M}) = (\bot, \overline{H}, \overline{M}) \in \mathsf{Errors}$). The $\text{Decode}_{\mathsf{M}}$ algorithm must be deterministic, and all algorithms should run in linear time. We call $\mathsf{EncodeMsg}$ a $(\mathcal{H}, \mathcal{M}, \overline{\mathcal{H}}, \overline{\mathcal{M}}, \mathsf{Errors})$-encoding scheme.

We only consider encoding functions having an associated *maximal stretch*, defined to be the smallest $s \in \mathbb{N}$ such that, for all $H \in \mathcal{H}$, $\overline{H} \in \overline{\mathcal{H}}$, and $\overline{M} \in \overline{\mathcal{M}}$, $|\text{Encode}_{\mathsf{H}}(H)| \le |H| + s$ and $\left|\text{Encode}_{\mathsf{M}}(\overline{H}, M)\right| \le |M| + s$.

Encoding scheme properties. Our encode-then-encipher AEAD security theorems will surface two key properties of the encoding mechanisms. The first property speaks to the likelihood that an encoding scheme can be made to repeat outputs.

Let A be an adversary that asks q queries (not necessarily distinct) to an oracle f, receiving $\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_q$ in response, and that halts by outputting these values. (Think of f as a message– or header-encoding function.) Without loss of generality, we can assume A is deterministic. Let $\delta\colon \mathbb{N} \to [0,1]$ be a function. Generalizing the definition from [8], we say f is (d,δ)-colliding if

$$\Pr\left[\, A^f \Rightarrow (\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_q)\colon \exists i_1 < i_2 < \ldots < i_d \text{ such that } \overline{Y}_{i_1} = \overline{Y}_{i_2} = \cdots = \overline{Y}_{i_d}\,\right] \le \delta(q).$$

This notion is only defined for $d \ge 2$. Given a (d,δ)-colliding oracle, we may assume without loss of generality that $\delta(0) = \delta(1) = 0$.

The second property captures the probability that a random string (of a given length) is a valid encoding. One can think of this as a measure of the density of encodings. Thus, let $\epsilon \in \mathbb{R}$ be a real number. Then the $(\mathcal{H}, \mathcal{M}, \overline{\mathcal{H}}, \overline{\mathcal{M}}, \mathsf{Errors})$-encoding scheme $\mathsf{EncodeMsg} = (\text{Encode}_{\mathsf{M}}, \text{Decode}_{\mathsf{M}})$ is *ϵ-sparse* if for all positive integers n and all $H \in \overline{\mathcal{H}}$, $\left|\{C \in \{0,1\}^n \,:\, \text{Decode}(H, C) \notin \mathsf{Errors}\}\right| / 2^n \le \epsilon$.

5.2 AEAD via encode-then-encipher

Now, let $\widetilde{E}\colon \mathcal{K} \times \overline{\mathcal{H}} \times \overline{\mathcal{M}} \to \overline{\mathcal{M}}$ be a tweakable cipher. Let $\text{Encode}_{\mathsf{H}}$ be a $(\mathcal{H}, \mathcal{R}, \overline{\mathcal{H}})$-encoder, and let $\mathsf{EncodeMsg} = (\text{Encode}_{\mathsf{M}}, \text{Decode}_{\mathsf{M}})$ be an $(\mathcal{H}, \mathcal{M}, \overline{\mathcal{H}}, \overline{\mathcal{M}}, \mathsf{Errors})$-encoding scheme, for some non-empty sets $\mathcal{H}$, $\mathcal{M}$, and $\mathcal{R}$. From these, we define an encode-then-encipher AEAD scheme $\Psi[\text{Encode}_{\mathsf{H}}, \mathsf{EncodeMsg}, \widetilde{E}]$ in Figure 7. As a simple example, let $\text{Encode}_{\mathsf{H}}$ prepend an 64-bit counter R to the header H, and $\text{Encode}_{\mathsf{M}}(\overline{H}, M)$ return $M \,\|\, 0^{80}$. Then $\text{Encode}_{\mathsf{H}}$ is $(2,0)$-colliding, and $\text{Encode}_{\mathsf{M}}$ is 2^{-80}-sparse (but $(2,1)$-colliding). We point out that all authenticity checks implicitly take place inside of the $\text{Decode}_{\mathsf{M}}$ function.

Security theorems and discussion. Here we give the privacy and authenticity security statements for our encode-then-encipher AEAD scheme. We'll give the statements first, and then discuss what they imply. Proofs follow the discussion.

Theorem 7. [Privacy.] *Let $\Psi = \Psi[\text{Encode}_{\mathsf{H}}, \mathsf{EncodeMsg}, \widetilde{E}]$ be defined as in Figure 7. Let s be the maximal stretch of $\mathsf{EncodeMsg}$, s' be the maximal stretch of $\text{Encode}_{\mathsf{H}}$, and define m as the length of the shortest $\overline{M} \in \overline{\mathcal{M}}$ satisfying $\text{Decode}_{\mathsf{M}}(\overline{H}, \overline{M}) \ne \mathsf{Errors}$ for some $\overline{H} \in \overline{\mathcal{H}}$. Let A be an adversary making q queries totaling μ bits, and running in time t. Then if $\text{Encode}_{\mathsf{H}}$ is (d, δ_H)-colliding and $\text{Encode}_{\mathsf{M}}$ is $(2, \delta_M)$-colliding for some δ_M that is increasing and convex on $\{0, 1, \ldots, q\}$, there is an adversary B such that*

$$\mathbf{Adv}^{\text{priv}}_{\Psi}(A) \le \mathbf{Adv}^{\widetilde{\text{prp}}}_{\widetilde{E}}(B) + \left(\delta_M(d-1) + \frac{(d-1)(d-2)}{2^{m+1}}\right)\left\lceil \frac{q}{d-1} \right\rceil + \left(\delta_M(q) + \frac{q(q-1)}{2^{m+1}}\right)\delta_H(q)$$

procedure $\mathcal{E}_K(H, M)$:	**procedure** $\mathcal{D}_K(H, R, C)$:
$\overline{H} \leftarrow \langle R, H\rangle \xleftarrow{\$} \mathrm{Encode}_{\mathsf{H}}(H)$	$\overline{H} \leftarrow \langle R, H\rangle$
$\overline{M} \xleftarrow{\$} \mathrm{Encode}_{\mathsf{M}}(\overline{H}, M)$	$\overline{M} \leftarrow \widetilde{E}_K^{-1}(\overline{H}, C)$
Return $R, \widetilde{E}_K(\overline{H}, \overline{M})$	Return $\mathrm{Decode}_M(\overline{H}, \overline{M})$

Fig. 7: The AEAD scheme $\Psi[\mathrm{Encode}_{\mathsf{H}}, \mathsf{EncodeMsg}, \widetilde{E}]$. Reconstruction information R allows decryption to reconstruct $\overline{H}$ from the header. (This is in lieu of sending $\overline{H}$ as part of the ciphertext, or forcing the calling application to replace H with $\overline{H}$.) Note that all authenticity checks are implicitly carried out by Decode_M.

where B makes q queries of total length at most $\mu + q(s + s')$ bits and runs in time $\mathcal{O}(t)$.

Theorem 8. [Authenticity.] *Let $\Psi[\widetilde{E}] = \Psi[\mathrm{Encode}_{\mathsf{H}}, \mathsf{EncodeMsg}, \widetilde{E}]$ be defined as in Figure 7. Let s be the stretch of* $\mathsf{EncodeMsg}$*, s' be the maximal stretch of* $\mathrm{Encode}_{\mathsf{H}}$*, and define m as the length of the shortest $\overline{M} \in \overline{\mathcal{M}}$ satisfying $\mathrm{Decode}_{\mathsf{M}}(\overline{H}, \overline{M}) \neq \mathsf{Errors}$ for some $\overline{H} \in \overline{\mathcal{H}}$. Let A be an adversary making $q_{\mathcal{E}}$ (resp., $q_{\mathcal{D}}$) queries totaling $\mu_{\mathcal{E}}$ (resp., $\mu_{\mathcal{D}}$) bits to its encryption (resp., decryption) oracle, and running in time t. Then if* $\mathrm{Encode}_{\mathsf{M}}$ *is ϵ-sparse and $q_{\mathcal{E}} + q_{\mathcal{D}} < 2^{m-1}$, there is an adversary B such that*

$$\mathbf{Adv}^{\text{int-ctxt}}_{\Psi[\widetilde{E}]}(A) \leq \mathbf{Adv}^{\widetilde{\text{sprp}}}_{\widetilde{E}}(B) + 2q_{\mathcal{D}}\epsilon.$$

where B makes $q_{\mathcal{E}}$ forward-queries of total length $(\mu_{\mathcal{E}} + q_{\mathcal{E}}s)$ bits, $q_{\mathcal{D}}$ inverse-queries of total length $(\mu_{\mathcal{D}} + q_{\mathcal{D}}s')$ bits, and runs in $\mathcal{O}(t)$ time.

To begin our discussion of these results, consider the case that $\mathrm{Encode}_{\mathsf{H}}$ is $(2, 0)$-colliding. Then the privacy bound (Theorem 7) simplifies to $\mathbf{Adv}^{\text{priv}}_{\Psi}(A) \leq \mathbf{Adv}^{\widetilde{\text{prp}}}_{\widetilde{E}}(B)$. This is intuitive, since if the tweak $\overline{H}$ never repeats, the outputs $\widetilde{E}_K(\overline{H}, \overline{M})$ are uniformly random strings for *any* valid encoding of (H, M) into $\overline{M}$; $\mathrm{Encode}_{\mathsf{H}}(H, M) = M$ suffices. Thus, good encodings of the header H can substantially reduce the burden placed upon encoding of (H, M) into $\overline{M}$.

This case generalizes nicely. Say that we can assume that the probability of $\mathrm{Encode}_{\mathsf{H}}$ producing any $\overline{H}$ more than d times, for some small constant $d \ll q$, is negligible. Then the final term in the bound can effectively be ignored. The second term is roughly $q\delta_M(d) + q/2^{m+1}$. Now, notice that δ_M is evaluated at d, not q, and so $q\delta_M(d)$ can be made negligible by encoding a reasonable amount of randomness into $\overline{M}$ (e.g. $\log(q)$ bits). For some natural choices of $\mathsf{EncodeMsg}$ then, $q/2^{m+1}$ will be the dominating term, where m is the shortest length of $\overline{M}$. But to achieve good authenticity bounds, which we will turn to momentarily, m is unlikely to let $q/2^{m+1}$ ruin the bound.

We point out that in the un-tweakable setting considered in [8], privacy must be achieved by encoding randomness or state into $\overline{M}$. The presence of the tweak allows us to shift these "extra" bits into the encoding of the header, which potentially reduces the number of bits that must be cryptographically processed.

In the extreme case that $\overline{H}$ is fixed across all queries (perhaps by design, or perhaps a result of a faulty implementation), the construction reverts to the un-tweakable setting. In this case, $\mathrm{Encode}_{\mathsf{H}}$ is $(2, 1)$-colliding, and our bound essentially reverts to the one in [8], although we consider indistinguishability from random bits, which is stronger than the privacy notion considered there.

Turning now to the authenticity bound (Theorem 8), note that if $\mathrm{Encode}_{\mathsf{M}}$ inserts b redundant bits (so $\epsilon \approx 2^{-b}$) and $q_{\mathcal{E}} + q_{\mathcal{D}} \ll 2^m$, the second term of our authenticity bound is approximately $q_{\mathcal{D}}/2^b$. Thus, if the tweakable cipher $\widetilde{E}$ has STPRP-security up to (say) 2^{80} queries (e.g., an appropriately instantiated PIV with $N = 256$), then encoding the header with an nonce, and the message with 80 bits of redundancy, yields an AEAD scheme with roughly 80-bit privacy and authenticity guarantees, and one that can tolerate nonce-misuse.

We note that the proof of Theorem 8 can be easily modified to show that the stated bound holds even if the adversary controls the coins and state of $\mathrm{Encode}_{\mathsf{M}}$ and $\mathrm{Encode}_{\mathsf{H}}$. Additionally, we assume only that

decryption oracle queries are not redundant — adversaries are *not* assumed to respect any nonces encoded into the headers or messages.

Relationship to deterministic authenticated encryption. Motivated by the key-wrapping problem, Rogaway and Shrimpton [37] introduce *deterministic authenticated encryption* (DAE). The encryption and decryption algorithms of a DAE scheme take a header string as an auxiliary input, as in the AEAD case. However, both algorithms are required to be deterministic. The corresponding security notion considers only adversaries that never repeat queries (or make redundant queries to a decryption oracle).

Our encode-then-encipher AEAD scheme may be viewed as a DAE scheme, provided the $\mathrm{Encode}_{\mathsf{M}}$ and $\mathrm{Encode}_{\mathsf{H}}$ algorithms are deterministic. One can easily show that the DAE security of a scheme is upper bounded by the sum of its privacy and authenticity bounds, as given in Theorems 7 and 8. We note that under the assumption that adversaries do not repeat queries, the privacy bound from Theorem 7 reduces to $\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(B) + q^2/2^{m+1}$ for some adversary B. Obtaining this result requires trivial proof modifications, and generalizes a result from [37], which considers $\mathrm{Encode}_{\mathsf{M}}(\overline{H}, M) = M \,\|\, 0^s$.

5.3 Proof of Theorem 7

Proof. Let $\Pi \xleftarrow{\$} \mathrm{BC}(\overline{\mathcal{H}}, \overline{\mathcal{M}})$ be a random cipher. Let $\Psi[\Pi]$ be the AEAD scheme obtained by replacing $\widetilde{E}_K$ and $\widetilde{E}_K^{-1}$ with Π and Π^{-1}, respectively, everywhere in the algorithms of $\Psi[\mathrm{Encode}_{\mathsf{H}}, \mathsf{EncodeMsg}, \widetilde{E}]$. Let $\mathcal{E}_\Pi$ be the corresponding encryption algorithm. We begin by observing that $\Pr\left[\, A^{\mathcal{E}_K(\cdot,\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\mathcal{E}_\Pi(\cdot,\cdot)} \Rightarrow 1 \,\right] \leq \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(B)$, for the standard adversary B that simulates $\mathcal{E}_K$ or $\mathcal{E}_\Pi$, depending on its own oracle. This leaves us with $\mathbf{Adv}^{\mathrm{priv}}_{\Psi}(A) \leq \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(B) + \Pr\left[\, A^{\mathcal{E}_\Pi(\cdot,\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, K \xleftarrow{\$} \mathcal{K}\,;\, A^{\$_K(\cdot,\cdot)} \Rightarrow 1 \,\right]$.

Now we proceed by a sequence of games. Let s be the stretch of $\mathrm{Encode}_{\mathsf{M}}$. Game G1 implements $\mathcal{E}_\Pi$, using lazy-sampling to define Π. In particular, on the i-th query (H_i, M_i), Game G1 computes the encodings $\overline{H}_i, \overline{M}_i$, and then samples a potential value S_i to assign to $\Pi(\overline{H}_i, \overline{M}_i)$. This value will be valid so long as $\Pi(\overline{H}_i, \overline{M}_i)$ has not already been set, and as long as S_i has not been used with $\overline{H}_i$ before; otherwise, Game G1 sets bad_1 or bad_2, and then reassigns S_i an appropriate value.

Since Game G2 is identical to Game G1 until bad_1 or bad_2 is set, we have

$$\begin{aligned}
\Pr\left[\, A^{\mathcal{E}_\Pi(\cdot,\cdot)} \Rightarrow 1 \,\right] &= \Pr\left[\, \mathrm{G1}(A) \Rightarrow 1 \,\right] \\
&= \Pr\left[\, \mathrm{G1}(A) \Rightarrow 1 \ \wedge\ (\mathsf{bad}_1 \vee \mathsf{bad}_2) \,\right] + \Pr\left[\, \mathrm{G1}(A) \Rightarrow 1 \ \wedge\ \neg(\mathsf{bad}_1 \vee \mathsf{bad}_2) \,\right] \\
&\leq \Pr\left[\, \mathrm{G2}(A)\,;\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right] + \Pr\left[\, \mathrm{G2}(A) \Rightarrow 1 \ \wedge\ \neg(\mathsf{bad}_1 \vee \mathsf{bad}_2) \,\right]
\end{aligned}$$

where in the final line we use an alternative formulation of the fundamental lemma of game-playing [5]. Now, notice that in Game G2, the value of S_i is always uniformly random in $\{0,1\}^{|M_i|+s}$, and $\mathcal{E}_\Pi$'s outputs are independent of each M_i. Consequently we can postpone assigning values to each M_i until *after* A halts. This in turn allows us to postpone checking to see if bad_1 or bad_2 should be set without altering the probability that they will be. We make both these changes to create Game 3 (so $\Pr\left[\, \mathrm{G2}(A)\,;\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right] = \Pr\left[\, \mathrm{G3}(A)\,;\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right]$). Thus,

$$\begin{aligned}
\Pr\left[\, \mathrm{G2}(A) \Rightarrow 1 \ \wedge\ \neg(\mathsf{bad}_1 \vee \mathsf{bad}_2) \,\right] &= \Pr\left[\, \mathrm{G3}(A) \Rightarrow 1 \ \wedge\ \neg(\mathsf{bad}_1 \vee \mathsf{bad}_2) \,\right] \\
&\leq \Pr\left[\, \mathrm{G3}(A) \Rightarrow 1 \,\right] = \Pr\left[\, K \xleftarrow{\$} \mathcal{K}\,;\, A^{\$_K(\cdot,\cdot)} \Rightarrow 1 \,\right]
\end{aligned}$$

where the final equality follows from the fact that in Game G3, each S_i is sampled independently and uniformly at random from the set of appropriately sized strings. To recap, at this point we have

$$\mathbf{Adv}^{\mathrm{priv}}_{\Psi}(A) \leq \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(B) + \Pr\left[\, \mathrm{G3}(A)\,;\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right]$$

We need an upper bound for $\Pr\left[\, \mathrm{G3}(A)\,;\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right]$. Therefore suppose that A has just generated its output after running in G3. We will first bound the probability that bad_1 gets set. Let $N = \left|\left\{\overline{H}_i \,:\, i \leq q\right\}\right|$

be the number of distinct tweak encodings generated during the course of the game. Let $R_1, R_2, \ldots, R_N \subseteq \{1, 2, \ldots, q\}$ be equivalence classes characterized by the property that i and j are in the same class if and only if $\overline{H}_i = \overline{H}_j$. The probability that bad_1 will be set is at most $\sum_k \delta_M(|R_k|)$.

Note that the upper bound is obtained by summing the values of the increasing convex function δ_M at the points $|R_1|, |R_2|, \ldots, |R_N|$ where $|R_1| + |R_2| + \cdots + |R_N| = q$.

Consequently the bound is largest (for fixed q) when $N = 1$ and $R_1 = \{1, 2, \ldots, q\}$. But suppose that each $|R_k| < d$, an event we shall denote as Capped; then $\Pr[\,\mathrm{G3}(A)\,;\,\neg\mathsf{Capped}\,] \le \delta_H(q)$. Given that Capped occurs, $\sum_k \delta_M(|R_k|)$ is largest when $N = \lceil q/(d-1) \rceil$ and $|R_k| = d - 1$ for $k = 1, 2, \ldots, N-1$. We have

$$\begin{aligned}\Pr[\,\mathrm{G3}(A)\,;\,\mathsf{bad}_1\,] &\le \Pr[\,\mathrm{G3}(A)\,;\,\mathsf{bad}_1 \mid \mathsf{Capped}\,] + \Pr[\,\mathrm{G3}(A)\,;\,\mathsf{bad}_1 \mid \neg\mathsf{Capped}\,]\Pr[\,\mathrm{G3}(A)\,;\,\neg\mathsf{Capped}\,] \\ &\le \delta_M(d-1)\lceil \frac{q}{d-1} \rceil + \delta_M(q)\delta_H(q).\end{aligned}$$

By a similar argument,

$$\begin{aligned}\Pr[\,\mathrm{G3}(A)\,;\,\mathsf{bad}_2\,] &\le \Pr[\,\mathrm{G3}(A)\,;\,\mathsf{bad}_2 \mid \mathsf{Capped}\,] + \Pr[\,\mathrm{G3}(A)\,;\,\mathsf{bad}_2 \mid \neg\mathsf{Capped}\,]\Pr[\,\mathrm{G3}(A)\,;\,\neg\mathsf{Capped}\,] \\ &\le \frac{(d-1)(d-2)}{2^{m+1}}\lceil \frac{q}{d-1} \rceil + \frac{q(q-1)}{2^{m+1}}\delta_H(q),\end{aligned}$$

since the standard $i \mapsto i(i-1)/2^{m+1}$ birthday bound (for the probability of a collision among i independent random variables sampled uniformly from $\{0,1\}^m$) meets the criterion of an increasing convex function. This completes the proof. □

Games [*G1*], *G2*	Game *G3*
Oracle $\mathcal{E}_\Pi(H, M)$:	**procedure Main(A)**:
$i \leftarrow i + 1$; $M_i \leftarrow M$	$b \xleftarrow{\$} A^{\mathcal{E}(\cdot,\cdot)}$
$\overline{H}_i \leftarrow \langle r, H_i \rangle \xleftarrow{\$} \mathrm{Encode}_T(H_i)$	**for** $i \leftarrow 1$ to q **do**
$\overline{M}_i \xleftarrow{\$} \mathrm{Encode}_M(\overline{H}_i, M_i)$	$\overline{M}_i \xleftarrow{\$} \mathrm{Encode}_M(\overline{H}_i, M_i)$
$S_i \xleftarrow{\$} \{0,1\}^{\lvert\overline{M}\rvert}$	**if** $\overline{M}_i \in \mathsf{dom}(\Pi(\overline{H}_i, \cdot))$ **then** $\mathsf{bad}_1 \leftarrow \mathsf{true}$
if $S_i \in \mathsf{range}(\Pi(\overline{H}_i, \cdot))$ **then**	**if** $S_i \in \mathsf{range}(\Pi(\overline{H}_i, \cdot))$ **then** $\mathsf{bad}_2 \leftarrow \mathsf{true}$
$\mathsf{bad}_2 \leftarrow \mathsf{true}$	$\Pi(\overline{H}_i, \overline{M}_i) \leftarrow S_i$
[$S_i \xleftarrow{\$} \{0,1\}^{\lvert\overline{M}\rvert} \setminus \mathsf{range}(\Pi(\overline{H}_i, \cdot))$]	**return** b
if $\overline{M}_i \in \mathsf{dom}(\Pi(\overline{H}_i, \cdot))$ **then**	
$\mathsf{bad}_1 \leftarrow \mathsf{true}$	**Oracle** $\mathcal{E}_\Pi(H, M)$:
[$S_i \leftarrow \Pi(\overline{H}_i, \overline{M}_i)$]	$i \leftarrow i + 1$; $M_i \leftarrow M$
$\Pi(\overline{H}_i, \overline{M}_i) \leftarrow S_i$	$\overline{H}_i \leftarrow \langle r, H_i \rangle \xleftarrow{\$} \mathrm{Encode}_T(H_i)$;
return $r \,\|\, S_i$	$S_i \xleftarrow{\$} \{0,1\}^{\lvert M\rvert + s}$
	return $r \,\|\, S_i$

Fig. 8: Games for the proof of Theorem 7. Boxed commands are omitted in Game G2, causing the $\mathcal{E}_\Pi$ oracle to always return random strings. We use Game G3 to bound the probability that this change can be detected by an adversary (as measured by the probability that a bad will be set).

5.4 Proof of Theorem 8

Proof. Let B be the STPRP adversary that simulates the INT-CTXT experiment for A and outputs 1 if A would set Forges to true. Then $\mathbf{Adv}^{\text{int-ctxt}}_{\Psi}(A) \leq \mathbf{Adv}^{\widetilde{\text{sprp}}}_{\widetilde{E}}(B) + \mathbf{Adv}^{\text{int-ctxt}}_{\Psi[\Pi]}(A)$, where $\Psi[\Pi]$ is the scheme obtained by replacing $\widetilde{E}_K$ and $\widetilde{E}_K^{-1}$ with Π and Π^{-1}, respectively, everywhere in the algorithms of Ψ, and $\mathbf{Adv}^{\text{int-ctxt}}_{\Psi[\Pi]}(A)$ is defined in the natural way (with probabilities over the random choice of Π, rather than K).

GAME *G4*

Oracle $\mathcal{E}_\Pi(H, M)$:

$\overline{H} \leftarrow \langle r, H\rangle \xleftarrow{\$} \text{Encode}_T(H)$
$\overline{M} \xleftarrow{\$} \text{Encode}_M(\overline{H}, M)$
if $\overline{M} \notin \mathsf{dom}(\Pi(\overline{H}, \cdot))$ **then**
  $\Pi(\overline{H}, \overline{M}) \xleftarrow{\$} \{0,1\}^{|\overline{M}|} \setminus \mathsf{range}(\Pi(\overline{H}, \cdot))$
return $r \,\|\, \Pi(\overline{H}, \overline{M})$

Oracle $\mathcal{D}_\Pi(H, r, C)$:

$\overline{H} \leftarrow \langle r, H\rangle$
$\overline{M} \xleftarrow{\$} \{0,1\}^{|C|} \setminus \mathsf{dom}(\Pi(\overline{H}, \cdot))$
$\Pi(\overline{H}, \overline{M}) \leftarrow C$
$M \leftarrow \text{Decode}_{\mathsf{M}}(\overline{H}, \overline{M})$
if $M \in \mathsf{Errors}$ **then** $\mathsf{Forges} \leftarrow \mathsf{true}$
return M

Fig. 9: Game G4 simulates the IND-CTXT experiment for $\mathcal{SE}[\Pi]$.

Consider Game G4. By defining Π through lazy sampling, we have $\mathbf{Adv}^{\text{int-ctxt}}_{\Psi[\Pi]}(A) = \Pr[\,\text{G4}(A)\,;\,\mathsf{Forges}\,]$. Note that in the code for the $\mathcal{D}_\Pi$ oracle, we do not need to check if $\Pi^{-1}(\overline{H}, Y)$ has already been defined; this possibility is excluded by the fact that A does not repeat queries to $\mathcal{D}_\Pi$, and does not send $\mathcal{D}_\Pi$ a value previously returned by $\mathcal{E}_\Pi$ (while preserving the header).

Fix some query to $\mathcal{D}_\Pi$. The probability that A forges on this query is equal to the probability that the corresponding, randomly chosen value of $\mathcal{M}$ is a valid encoding. There are at most $2^{|Y|}\epsilon$ valid encodings of the correct length, and $\mathcal{M}$ is sampled from a set of size at least $2^{|Y|} - (q_\mathcal{E} + q_\mathcal{D})$. Consequently, A forges on this query with probability at most $2^{|Y|}\epsilon/(2^{|Y|} - (q_\mathcal{E} + q_\mathcal{D})) < 2^m\epsilon/(2^m - 2^{m-1}) = 2\epsilon$. A union bound completes the proof. □

6 Acknowledgements

Portions of this work were carried out while Terashima was visiting Voltage Security. We thank them, especially Terence Spies, for their support. We also thank the CRYPTO and Asiacrypt 2013 reviewers for their diligence and useful feedback. Both Terashima and Shrimpton were supported by NSF grants CNS-0845610 and CNS-1319061.

References

1. Michel Abdalla and Mihir Bellare. Increasing the lifetime of a key: A comparative analysis of the security of re-keying techniques. In *Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '00, pages 546–559, London, UK, UK, 2000. Springer-Verlag.
2. N AlFardan and KG Paterson. Lucky 13: Breaking the TLS and DTLS record protocols. In *IEEE Symposium on Security and Privacy*, 2013.

3. Jee Hea An and Mihir Bellare. Does encryption with redundancy provide authenticity? In *Advances in Cryptology–Eurocrypt*, pages 512–528. Springer, 2001.
4. D. Aranha, J. López, and D. Hankerson. Efficient software implementation of binary field arithmetic using vector instruction sets. *Progress in Cryptology–LATINCRYPT 2010*, pages 144–161, 2010.
5. M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. In *Advances in Cryptology–EUROCRYPT*, volume 4004, page 10, 2006.
6. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Advances in Cryptology–ASIACRYPT 2000*, pages 531–545, 2000.
7. Mihir Bellare, Thomas Ristenpart, Phillip Rogaway, and Till Stegers. Format-preserving encryption. In *Selected Areas in Cryptography*, pages 295–312. Springer, 2009.
8. Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In *ASIACRYPT*, pages 317–330, 2000.
9. Mihir Bellare, Phillip Rogaway, and Terence Spies. The FFX mode of operation for format-preserving encryption. Unpublished NIST proposal, 2010.
10. John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: Fast and secure message authentication. In *CRYPTO*, pages 216–233, 1999.
11. Brice Canvel, Alain Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a SSL/TLS channel. *Advances in Cryptology–CRYPTO 2003*, pages 583–599, 2003.
12. Debrup Chakraborty and Mridul Nandi. An improved security bound for HCTR. In Kaisa Nyberg, editor, *Fast Software Encryption*, volume 5086 of *Lecture Notes in Computer Science*, pages 289–302. Springer Berlin / Heidelberg, 2008.
13. Debrup Chakraborty and Palash Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. In Matthew Robshaw, editor, *Fast Software Encryption*, volume 4047 of *Lecture Notes in Computer Science*, pages 293–309. Springer Berlin / Heidelberg, 2006.
14. Debrup Chakraborty and Palash Sarkar. HCH: A new tweakable enciphering scheme using the hash-counter-hash approach. *IACR Cryptology ePrint Archive*, 2007:28, 2007.
15. Jean-Sébastien Coron, Yevgeniy Dodis, Avradip Mandal, and Yannick Seurin. A domain extender for the ideal cipher. In *TCC*, pages 273–289, 2010.
16. Jean Paul Degabriele and Kenneth G Paterson. On the (in)security of IPsec in MAC-then-encrypt configurations. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 493–504. ACM, 2010.
17. Shai Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In *INDOCRYPT*, pages 315–327, 2004.
18. Shai Halevi. Invertible universal hashing and the TET encryption mode. In *CRYPTO*, pages 412–429, 2007.
19. Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In *Advances in Cryptology — CRYPTO 2003*, pages 482–499, 2003.
20. Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In *CT-RSA*, pages 292–304, 2004.
21. Viet Tung Hoang, Ben Morris, and Phillip Rogaway. An enciphering scheme based on a card shuffle. In *Advances in Cryptology–CRYPTO 2012*, pages 1–13. Springer, 2012.
22. Rodolphe Lampe and Yannick Seurin. Tweakable blockciphers with asymptotically optimal security. In *FSE*, 2013.
23. Will Landecker, Thomas Shrimpton, and R. Terashima. Tweakable blockciphers with beyond birthday-bound security. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology — CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 14–30. Springer Berlin / Heidelberg, 2012.
24. Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '02, pages 31–46, London, UK, UK, 2002. Springer-Verlag.
25. J. Luo, K.D. Bowers, A. Oprea, and L. Xu. Efficient software implementations of large finite fields GF(2^n) for secure storage applications. *ACM Transactions on Storage (TOS)*, 8(1):2, 2012.
26. Cuauhtemoc Mancillas-Lopez, Debrup Chakraborty, and Francisco Rodriguez-Henriquez. Reconfigurable hardware impementations of tweakable enciphering schemes. *IEEE Transactions on Computers*, 59:1547–1561, 2010.
27. Nick Mathewson. Cryptographic challenges in and around Tor. Talk given at the Workshop on Real-World Cryptography, Jan 2013.
28. Kazuhiko Minematsu. Beyond-birthday-bound security based on tweakable block cipher. In *FSE*, pages 308–326, 2009.
29. Kazuhiko Minematsu and Tetsu Iwata. Building blockcipher from tweakable blockcipher: Extending FSE 2009 proposal. In *IMA International Conference*, pages 391–412, 2011.

30. Ben Morris, Phillip Rogaway, and Till Stegers. How to encipher messages on a small domain. In Shai Halevi, editor, *Advances in Cryptology–CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 286–302. Springer Berlin Heidelberg, 2009.
31. M. Naor and O. Reingold. On the construction of pseudo-random permutations: Luby-rackoff revisited. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 189–199. ACM, 1997.
32. Kenneth Paterson and Arnold Yau. Padding oracle attacks on the ISO CBC mode encryption standard. *Topics in Cryptology–CT-RSA 2004*, pages 1995–1995, 2004.
33. P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. *Advances in Cryptology–ASIACRYPT 2004*, pages 55–73, 2004.
34. P. Rogaway. Nonce-based symmetric encryption. In *Fast Software Encryption*, pages 348–358. Springer, 2004.
35. Phillip Rogaway. Authenticated-encryption with associated-data. In *ACM Conference on Computer and Communications Security*, pages 98–107, 2002.
36. Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 196–205. ACM, 2001.
37. Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In *EUROCRYPT*, pages 373–390, 2006.
38. Palash Sarkar. Improving upon the TET mode of operation. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *Information Security and Cryptology - ICISC 2007*, volume 4817 of *Lecture Notes in Computer Science*, pages 180–192. Springer Berlin / Heidelberg, 2007.
39. Palash Sarkar. Efficient tweakable enciphering schemes from (block-wise) universal hash functions. *IEEE Trans. Inf. Theor.*, 55(10):4749–4760, October 2009.
40. Serge Vaudenay. Security flaws induced by CBC paddingapplications to SSL, IPSEC, WTLS.... In *Advances in Cryptology–EUROCRYPT 2002*, pages 534–545. Springer, 2002.
41. P. Wang, D. Feng, and W. Wu. HCTR: A variable-input-length enciphering mode. In *Information Security and Cryptology*, pages 175–188. Springer, 2005.
42. M.N. Wegman and J.L. Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(3):265–279, 1981.

A Components for TCT1 and TCT2

LRW2 [24]: Birthday-bound TBC. Needs blockcipher E, ϵ-AXU$_2$ function H.

$$\mathrm{LRW2}[H,E]_{(K,L)}(T,X) = E_K(X \oplus H_L(T)) \oplus H_L(T)$$

CLRW2[23]: TBC with beyond-birthday-bound security. Requires blockcipher E and ϵ-AXU$_2$ function H.

$$\mathrm{CLRW2}[H,E]_{(K_1,K_2,L_1,L_2)}(T,X) = \mathrm{LRW2}[H,E]_{(K_2,L_2)}(T, \mathrm{LRW2}[H,E]_{(K_1,L_1)}(T,X))$$

polyH^{mn} [42]: ϵ-AXU$_2$ function with domain $(\{0,1\}^n)^m$ and $\epsilon = m/2^n$. All operations in $\mathbb{F}_{2^n}$.

$$\mathrm{polyH}^{mn}_L(T_1T_2\cdots T_m) = \bigoplus_{i=1}^{m} T_i \otimes L^i,$$

NH($\nu w, 2tw$) [10]: ϵ-AU hash function with $\epsilon = 1/2^{tw}$. Inputs are νw bits, where ν is even and $w > 0$ is fixed.

$$\mathrm{NH}[\nu,t]_{K_1 \,\|\, \cdots \,\|\, K_{\nu+2(t-1)}}(M) = H_{K_1\cdots K_\nu}(M) \,\|\, H_{K_3\cdots K_{\nu+2}}(M) \,\|\, \cdots \,\|\, H_{K_{2t-1}\cdots K_{\nu+2t-2}}(M)$$

where $H_{K_1 \,\|\, \cdots \,\|\, K_\nu}(X_1\cdots X_\nu) = \sum_{i=1}^{\nu/2}(K_{2i-1} +_w X_{2i-1}) \cdot (K_{2i} +_w X_{2i}) \bmod 2^{2w}$.

CDMS [15]: Feistel-like domain extender for TBC $\widetilde{E}$.

$$\mathrm{CDMS}[\widetilde{E}]_K(T, L \,\|\, R) = \widetilde{E}_K(10 \,\|\, T \,\|\, R', L') \,\|\, R'$$

where $R' = \widetilde{E}_K(01 \,\|\, T \,\|\, L', R)$ and $L' = \widetilde{E}_K(00 \,\|\, T \,\|\, R, L)$.

Fig. 10: TCT_1 and TCT_2 use these constructions as components.