Length-preserving encryption with HCTR2

Paul Crowley, Nathan Huckleberry, and Eric Biggers

Google

June 30, 2023

Abstract

On modern processors HCTR[WFW05] is one of the most efficient constructions for building a tweakable super-pseudorandom permutation. However, a bug in the specification and another in Chakraborty and Nandi's security proof[CN08] invalidate the claimed security bound. We here present HCTR2, which fixes these issues and improves the security bound, performance and flexibility. GitHub: https://github.com/google/hctr2

1 Introduction

A *tweakable super-pseudorandom permutation* (tweakable SPRP) is a family of permutations indexed by tweak and input length, which appear to be random permutations to an adversary without the key who can make encryption and decryption queries[HR03]. [CB18] includes a detailed history of length-preserving encryption. A tweakable SPRP is a highly general and flexible cryptographic construction. One common use is in disk sector encryption: if the ciphertext must be the same size as the plaintext, with no extra room for nonce or MAC, a tweakable SPRP represents an upper bound on the achievable security. If a variable-length tweak is accepted, it can also serve as a nonce-misuse-resistant AEAD mode: concatenate the nonce and the associated data to form the tweak, and authenticate the message by appending zeroes to the plaintext which will be checked on decryption[BR00a; HKR15].

In this paper, we present a specification (section 2) and security bound (section 3) for HCTR2, a tweakable SPRP based on HCTR[WFW05] and inheriting the following advantages:

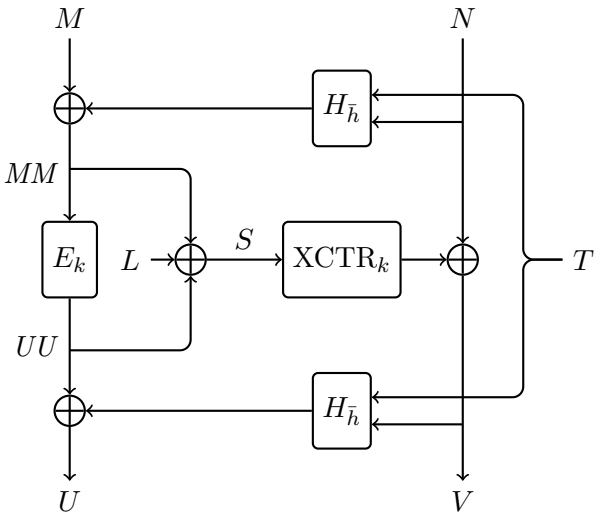

Figure 1: HCTR2

- simple
- efficient on modern processors, using a single block cipher invocation and two $\mathrm{GF}(2^{128})$ multiplications per 16-byte block
- naturally handles ciphertext of any length of 16 bytes or greater
- tight quadratic security claim

HCTR2 addresses these issues in HCTR:

- [Kum18] observes that HCTR's hash function is not almost-XOR-universal[Sti95] as claimed (subsection 4.1). HCTR2's hash function fixes this property.
- Separately, an error in the proof presented in [CN08] invalidates the quadratic security bound claimed in that paper (subsection 4.2); with our revised mode we can claim a tighter quadratic bound.
- HCTR2 supports using tweaks of any length with a single key.
- HCTR's key is a block cipher key, plus an n-bit hash key. HCTR2's key is simply the block cipher key.
- We modify the hash function to allow more precomputation for greater performance.

procedure ENCRYPT(k, T, P)
  $\bar{h} \leftarrow E_k(\text{bin}(0))$
  $L \leftarrow E_k(\text{bin}(1))$
  $M \| N \leftarrow P$, $|M| = n$
  $MM \leftarrow M \oplus H_{\bar{h}}(T, N)$
  $UU \leftarrow E_k(MM)$
  $S \leftarrow MM \oplus UU \oplus L$
  $V \leftarrow N \oplus \text{XCTR}_k(S)[0; |N|]$
  $U \leftarrow UU \oplus H_{\bar{h}}(T, V)$
  $C \leftarrow U \| V$
  return C
end procedure

Figure 2: HCTR2 encryption

procedure DECRYPT(k, T, C)
  $\bar{h} \leftarrow E_k(\text{bin}(0))$
  $L \leftarrow E_k(\text{bin}(1))$
  $U \| V \leftarrow C$, $|U| = n$
  $UU \leftarrow U \oplus H_{\bar{h}}(T, V)$
  $MM \leftarrow E_k^{-1}(UU)$
  $S \leftarrow MM \oplus UU \oplus L$
  $N \leftarrow V \oplus \text{XCTR}_k(S)[0; |V|]$
  $M \leftarrow MM \oplus H_{\bar{h}}(T, N)$
  $P \leftarrow M \| N$
  return P
end procedure

Figure 3: HCTR2 decryption

- We specify endianness and the like for interoperability.
- We provide a sample implementation and test vectors.

We discuss our design decisions (section 5) and report on its implementation on x86-64 and ARM64 (section 6). We know of no patents affecting HCTR2.

2 Specification

2.1 Notation

- $|X|$: length of $X \in \{0,1\}^*$ in bits
- λ: the empty string $|\lambda| = 0$
- $X[a; l]$: the substring of X of length l starting at the 0-based index a
- $\|$: bitstring concatenation
- $\oplus$: bitwise XOR
- n: block size in bits
- $\text{bin}_l : \{0 \dots 2^l - 1\} \to \{0,1\}^l$: little-endian conversion of integers to binary; $\text{bin}(x)$ means $\text{bin}_n(x)$

- $\mathrm{pad}(X) = X \| 0^v$ where v is the least integer ≥ 0 such that n divides $|X| + v$
- $x, x^2, \ldots$: elements of the finite field $\mathrm{GF}(2^n)$
- $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$: n-bit block cipher with keyspace $\mathcal{K}$; our concrete proposal uses AES[Dwo+01], so $n = 128$ and $\mathcal{K}$ is $\{0,1\}^{128}$, $\{0,1\}^{192}$, or $\{0,1\}^{256}$
- $\mathcal{T}$: the set of permissible tweaks $\mathcal{T} = \bigcup_{i \in \{0 \ldots 2^{n-1}-2\}} \{0,1\}^i$
- $\mathcal{M}$: the set of permissible messages $\mathcal{M} = \bigcup_{i \in \{n \ldots n+2^{n-1}-2\}} \{0,1\}^i$

We map bytes to bitstrings with bin_8. Subscripts may denote partial application; if we define $f : A \times B \to C$ and $a \in A$ then $f_a : B \to C$, and if f_a^{-1} exists then $f_a^{-1}(f_a(b)) = b$.

2.2 Polynomial hash function

We interpret n-bit blocks as little-endian field elements of $\mathrm{GF}(2^n)$, so $001 \| 0^{n-3}$ is interpreted as the element x^2. Per [GLL17; GLL19] we define

$$\begin{aligned} \mathrm{POLYVAL}(\bar{h}, \lambda) &= 0^n \\ \mathrm{POLYVAL}(\bar{h}, A \| B) &= (\mathrm{POLYVAL}(\bar{h}, A) \oplus B) \otimes \bar{h} \otimes x^{-n} \end{aligned}$$

where $|\bar{h}| = |B| = n$ and $\otimes$ is multiplication in the finite field. In our concrete proposal, $n = 128$, we reduce by $x^{128} + x^{127} + x^{126} + x^{121} + 1$, and the value of the field element x^{-n} is equal to $x^{127} + x^{124} + x^{121} + x^{114} + 1$.

For hash key $\bar{h} \in \{0,1\}^n$, tweak T and message M, we define:

$$H_{\bar{h}}(T, M)$$
$$\stackrel{\text{def}}{=} \begin{cases} \mathrm{POLYVAL}(\bar{h}, \mathrm{bin}(2|T|+2) \| \mathrm{pad}(T) \| M) & \text{if } n \text{ divides } |M| \\ \mathrm{POLYVAL}(\bar{h}, \mathrm{bin}(2|T|+3) \| \mathrm{pad}(T) \| \mathrm{pad}(M \| 1)) & \text{otherwise} \end{cases}$$

2.3 XCTR mode

HCTR and HCTR2 use an unusual mode of stream encryption, which we name *XCTR mode*:

$$\mathrm{XCTR}_k(S) = E_k(S \oplus \mathrm{bin}(1)) \| E_k(S \oplus \mathrm{bin}(2)) \| E_k(S \oplus \mathrm{bin}(3)) \| \cdots$$

Generating the first m bits $\mathrm{XCTR}_k(S)[0;m]$ takes $\lceil m/n \rceil$ block cipher calls.

2.4 HCTR2

HCTR2 encryption, defined in Figure 2, takes a tweak and a plaintext, and returns a ciphertext of the same length as the plaintext. HCTR2 decryption (Figure 3) recovers the plaintext given the same tweak and the ciphertext, ie for $k \in \mathcal{K}$, $T \in \mathcal{T}$ and $P \in \mathcal{M}$, $\textsc{Decrypt}(k, T, \textsc{Encrypt}(k, T, P)) = P$.

3 Security of HCTR2

Following the approach described in [Bel+97], we prove that if the underlying block cipher is secure, then HCTR2 has good security properties. The security bound proven appears in subsection 3.5.

3.1 Definitions

We use $x \xleftarrow{\$} S$ to mean "x is sampled from S uniformly at random", and we write $A^{\mathcal{O},\mathcal{O}'} \Rightarrow 1$ to refer to the event "adversary A, given access to oracles $\mathcal{O}$ and $\mathcal{O}'$, returns 1".

Let $\mathrm{Perm}(n)$ denote the set of all permutations on $\{0,1\}^n$. Per [Bel+97], for a block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ the distinguishing advantage of an adversary A is:

$$\mathsf{Adv}_E^{\pm\mathrm{prp}}(A) \stackrel{\text{def}}{=} \Big|\Pr_{k \xleftarrow{\$} \mathcal{K}}\left[A^{E_k, E_k^{-1}} \Rightarrow 1\right] - \Pr_{\pi \xleftarrow{\$} \mathrm{Perm}(n)}\left[A^{\pi,\pi^{-1}} \Rightarrow 1\right]\Big|$$

Define

$$\mathsf{Adv}_E^{\pm\mathrm{prp}}(q,t) \stackrel{\text{def}}{=} \max_{A \in \mathcal{A}(q,t)} \mathsf{Adv}_E^{\pm\mathrm{prp}}(A)$$

where $\mathcal{A}(q,t)$ is the set of all adversaries that make at most q queries and take at most t time.

Let $\mathrm{Perm}^{\mathcal{T}}(\mathcal{M})$ denote the set of all tweakable length-preserving permutations $\boldsymbol{\pi} : \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ such that for all $T, M \in \mathcal{T} \times \mathcal{M}$, $|\boldsymbol{\pi}(T,M)| = |M|$, and for all $T \in \mathcal{T}$, $\boldsymbol{\pi}_T$ is a permutation on $\mathcal{M}$. In an

abuse of notation we use $\boldsymbol{\pi}^{-1}$ to refer to the function such that $\boldsymbol{\pi}^{-1}(T, \boldsymbol{\pi}(T, M)) = M$ ie $(\boldsymbol{\pi}^{-1})_T = (\boldsymbol{\pi}_T)^{-1}$.

Per [HR03], for a tweakable super-pseudorandom permutation $\boldsymbol{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ the distinguishing advantage of an adversary A is:

$$\mathsf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\boldsymbol{E}}(A) \stackrel{\text{def}}{=} \Big|\mathrm{Pr}_{k \leftarrow\$ \mathcal{K}}\Big[A^{\boldsymbol{E}_k, \boldsymbol{E}_k^{-1}} \Rightarrow 1\Big] - \mathrm{Pr}_{\boldsymbol{\pi} \leftarrow\$ \mathrm{Perm}^{\mathcal{T}}(\mathcal{M})}\Big[A^{\boldsymbol{\pi}, \boldsymbol{\pi}^{-1}} \Rightarrow 1\Big]\Big|$$

Define

$$\mathsf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\boldsymbol{E}}(q, \sigma, t) \stackrel{\text{def}}{=} \max_{A \in \mathcal{A}(q,\sigma,t)} \mathsf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\boldsymbol{E}}(A)$$

where $\mathcal{A}(q, \sigma, t)$ is the set of all adversaries that make at most q queries and take at most t time, such that the total number of blocks sent in all queries is at most σ ie

$$\sum_s \lceil |T^s|/n \rceil + \lceil |P^s|/n \rceil \leq \sigma$$

where $|T^s|, |P^s|$ are the length of the tweak and the message presented in query s.

We use $\mathrm{HCTR2}[\pi]$ to refer to HCTR2 in which invocation of the block cipher is replaced with invocation of the permutation $\pi \in \mathrm{Perm}(n)$. XCTR_π refers to a similar substitution. $\mathrm{HCTR2}[E]$ refers to HCTR2 using the block cipher E, ie $\mathrm{HCTR2}[E_k]$ for $k \leftarrow\$ \mathcal{K}$, while $\mathrm{HCTR2}[\mathrm{Perm}(n)]$ refers to $\mathrm{HCTR2}[\pi]$ for $\pi \leftarrow\$ \mathrm{Perm}(n)$.

3.2 Hash function

Define $\mathrm{poly}(M)$ to refer to the formal polynomial $\mathrm{poly}(M_0 \| \cdots \| M_{l-1}) \stackrel{\text{def}}{=} M_0 h^{l-1} \oplus \cdots \oplus M_{l-1}$. While for example $h + 2$ and $2h + 1$ can be equal in value if $h = 1$, they are not equal as formal polynomials; two formal polynomials are only equal if every coefficient is equal. Thus $\mathrm{poly}(M) = \mathrm{poly}(M')$ only if $M = 0^{ln} \| M'$ for some l or vice versa.

Define $H(T, M)$ as the formal polynomial in h given by that tweak and message. $H_{\bar{h}}(T, M)$ is then evaluation of this polynomial at $h = x^{-n}\bar{h}$;

POLYVAL evaluates at this point for performance reasons.

$$H(T,M) \stackrel{\text{def}}{=} \begin{cases} \text{poly}(\text{bin}(2|T|+2) \| \text{pad}(T) \| M \| 0^n) & \text{if } n \text{ divides } |M| \\ \text{poly}(\text{bin}(2|T|+3) \| \text{pad}(T) \| \text{pad}(M \| 1) \| 0^n) & \text{otherwise} \end{cases}$$

We depend on the following properties of this map onto formal polynomials:

- The map is injective
- The polynomial is never 0 or $x^n h$
- The constant term is always zero
- The polynomial is of degree at most

$$d(T,M) \stackrel{\text{def}}{=} 1 + \lceil |T|/n \rceil + \lceil |M|/n \rceil$$

For the first property, see Appendix A. For the second, observe that $H(T,M)$ can be of degree 1 only if $|T| = |M| = 0$, in which case the polynomial is xh. Since $x^{n-1} \neq 1$ we have that $x^n \neq x$.

For any nonzero polynomial $p(h)$ in $\text{GF}(2^n)$, there are at most $\deg(p)$ values h such that $p(h) = 0$, and therefore $\Pr_{h \leftarrow \$\{0,1\}^n}[p(h) = 0] \leq \deg(p)/2^n$. Since multiplication by a nonzero field element is a bijection of the field onto itself, it follows that $\Pr_{\bar{h} \leftarrow \$\{0,1\}^n}\left[p(x^{-n}\bar{h}) = 0\right] \leq \deg(p)/2^n$. From this we infer three properties of $H_{\bar{h}}(T,M)$:

Property 1 For any T, M and any $g \in \{0,1\}^n$,

$$\Pr_{\bar{h} \leftarrow \$\{0,1\}^n}[H_{\bar{h}}(T,M) = g] \leq d(T,M)/2^n$$

Proof: since $H(T,M)$ is nonzero and has a zero constant term, the polynomial $H(T,M) \oplus g$ is nonzero and has the same degree, at most $d(T,M)$.

Property 2 For any $(T_1,M_1) \neq (T_2,M_2)$ and any $g \in \{0,1\}^n$

$$\begin{aligned} &\Pr_{\bar{h} \leftarrow \$\{0,1\}^n}[H_{\bar{h}}(T_1,M_1) \oplus H_{\bar{h}}(T_2,M_2) = g] \\ &\leq \max(d(T_1,M_1), d(T_2,M_2))/2^n \end{aligned}$$

Proof: H is injective onto polynomials and the constant term is zero, therefore $H(T_1, M_1) \oplus H(T_2, M_2) \oplus g$ is not the zero polynomial and has degree at most $\max(d(T_1, M_1), d(T_2, M_2))$. This is the almost-XOR-universal property.

Property 3 For any T, M and any $g \in \{0,1\}^n$

$$\Pr_{\bar{h} \leftarrow\$ \{0,1\}^n}\left[H_{\bar{h}}(T, M) \oplus \bar{h} = g\right] \leq d(T, M)/2^n$$

Proof: $H(T, M)$ has a zero constant term and cannot be equal to the polynomial $x^n h$. $H(T, M) \oplus g \oplus \bar{h} = H(T, M) \oplus g \oplus x^n h$ thus cannot be the zero polynomial and has degree at most $d(T, M)$.

3.3 H-coefficient technique

The H-coefficient technique was introduced by Patarin in 1991 [Pat91; Pat09]. We highly recommend the exposition of [CS14] Section 3, "The H-coefficient Technique in a Nutshell"; we here present a simpler exposition that does not cover the technique in its full generality but only our use of it. Our use of the symbol $\mathcal{T}$ and the term "compatible" differ from [CS14].

We wish to bound the adversary's ability to distinguish between two "worlds", world X (the "real world") and world Y (the "ideal world"). Each world is a probability distribution over deterministic oracles the adversary interacts with.

We consider only deterministic adversaries. A randomized adversary can be considered as a random draw from a population of deterministic adversaries, so a bound on the advantage achievable by a deterministic adversary bounds the whole population and therefore the advantage of the randomized adversary. In what follows we consider the adversary A fixed; only the world, and the particular oracles drawn from that world, vary.

When the adversary interacts with the oracle, a transcript τ of queries and responses is created. $\mathcal{T}_c$ is the set of "compatible transcripts": if $\tau \in \mathcal{T}_c$ then for the fixed adversary, there is some oracle that results in its creation. For example, since the adversary is deterministic, the first query will always be the same; a transcript that does not start with this query is not a compatible transcript. For a given $\tau \in \mathcal{T}_c$, a deterministic adversary must always return the same answer; call this answer $A(\tau)$.

Let random variables X and Y represent the distribution of transcripts in world X and world Y respectively, so that each transcript τ has a probability $\Pr[X = \tau]$ of arising in world X, and similarly $\Pr[Y = \tau]$ in world Y. The adversary's distinguishing advantage is then $|\Pr[A(Y) = 1] - \Pr[A(X) = 1]|$. Without loss of generality, we assume that $\Pr[A(Y) = 1] \geq \Pr[A(X) = 1]$. We further assume that $A(\tau)$ is optimal: $A(\tau) = 1$ when $\Pr[Y = \tau] > \Pr[X = \tau]$ and 0 otherwise.

In subsection 3.4, we partition $\mathcal{T}_{\mathrm{c}}$ into $\mathcal{T}_{\mathrm{good}}$ and $\mathcal{T}_{\mathrm{bad}}$, and prove that:

- $\Pr[Y = \tau] \leq \Pr[X = \tau]$ for all $\tau \in \mathcal{T}_{\mathrm{good}}$
- $\Pr[Y \in \mathcal{T}_{\mathrm{bad}}] \leq \epsilon$

It follows that $A(\tau) = 0$ for all $\tau \in \mathcal{T}_{\mathrm{good}}$, and therefore that $\Pr[A(Y) = 1] \leq \epsilon$, from which we bound the distinguishing advantage: $\Pr[A(Y) = 1] - \Pr[A(X) = 1] \leq \epsilon$.

With this technique, only the first proof need consider the probability distribution of world X at all, and this proof need only consider good transcripts. The bulk of the work, proving $\Pr[Y \in \mathcal{T}_{\mathrm{bad}}] \leq \epsilon$, involves only world Y, which is far simpler to reason about.

3.4 Main lemma

In what follows, we take world X (the "real world") to be $\mathrm{HCTR2}[\mathrm{Perm}(n)]$, ie HCTR with all calls to the block cipher replaced with calls to a random permutation, and world Y (the "ideal world") to be $\pm\widetilde{\mathrm{rnd}}$, which maps every query to a random response such that all responses of the appropriate length are equally likely; we then use the H-coefficient technique to bound the distinguishing advantage between them for a fixed adversary $A \in \mathcal{A}(q, \sigma, t)$ as defined in subsection 3.1.

We use superscripts to distinguish between queries; where we refer to r, s, we assume that $r < s$. An encryption query (T^s, P^s) yields ciphertext C^s while a decryption query (T^s, C^s) yields plaintext P^s. We forbid "pointless queries": encryption queries (T^s, P^s) such that $(T^r, P^r) = (T^s, P^s)$ for some $r < s$, or decryption queries (T^s, C^s) such that $(T^r, C^r) = (T^s, C^s)$ for some $r < s$, whether query r was an encryption or decryption query.

For each query s, let $m^s \stackrel{\text{def}}{=} \lceil |P^s|/n \rceil = \lceil |C^s|/n \rceil$ be the number of blocks in the response, and let $d^s \stackrel{\text{def}}{=} m^s + \lceil |T^s|/n \rceil$ be the number of blocks in the query. Note that d^s is the degree of the hash function polynomial used in query s (since all but one block of the message is hashed), and that $\sum_s d^s \leq \sigma$. We give the adversary some extra information which is included in the transcript. In world X, this information is:

- the "leftover block" for each query: where a query has plaintext/ciphertext that is not a multiple of the block size, this is the extra output from the last block cipher call that is not used. For query s, this is

$$D^s = \mathrm{XCTR}_\pi(S^s)[|P^s| - n; nm^s - |P^s|]$$

- the hash key $\bar{h}$, given after all queries are complete
- the mask L, given after all queries are complete

In world Y, random output of the expected length is substituted. Since the adversary can always ignore this information, giving it to them cannot make their performance worse.

3.4.1 Good and bad transcripts

For $j \in \{1 \dots m^s - 1\}$ we define $S^s_j = S^s \oplus \mathrm{bin}(j)$, the block cipher inputs used in XCTR, and Y^s_j the corresponding outputs, so that in world X $Y^s_j = \pi(S^s_j)$ and $Y^s_1 \| \cdots \| Y^s_{m^s-1} = \mathrm{XCTR}_\pi(S^s)[0; n(m^s - 1)]$.

Given the full transcript, including $\bar{h}$ and L, we can infer all block cipher plaintexts and ciphertexts. For each query (omitting the query superscript s for readability) we know T, P, C and D and so can infer:

$$\begin{aligned} M \| N &= P \\ U \| V &= C \\ MM &= M \oplus H_{\bar{h}}(T, N) \\ UU &= U \oplus H_{\bar{h}}(T, V) \\ S &= MM \oplus UU \oplus L \\ S_j &= S \oplus \mathrm{bin}(j) \\ Y_1 \| \cdots \| Y_{m-1} &= (N \oplus V) \| D \end{aligned}$$

This gives us multisets $\mathcal{D}$ and $\mathcal{R}$ of inferred block cipher plaintexts and ciphertexts:

$$\mathcal{D} \stackrel{\text{def}}{=} [\text{bin}(0), \text{bin}(1)] \uplus \biguplus_s [MM^s, S_1^s, \ldots, S_{m^s-1}^s]$$

$$\mathcal{R} \stackrel{\text{def}}{=} [\bar{h}, L] \uplus \biguplus_s [UU^s, Y_1^s, \ldots, Y_{m^s-1}^s]$$

We therefore infer $|\mathcal{D}| = |\mathcal{R}| = \sigma_m \stackrel{\text{def}}{=} 2 + \sum_s m^s$ block cipher plaintexts/ciphertexts from $\sigma_m n$ bits of response (including the extra information D^s, $\bar{h}$, L). A transcript is "bad" ($\tau \in \mathcal{T}_{\text{bad}}$) iff any entry in $\mathcal{D}$ or $\mathcal{R}$ has multiplicity greater than one, ie if any pair of inferred plaintexts are the same, or if any pair of inferred ciphertexts are the same.

Since responses in world Y are coin flips, the probability of a particular $\tau \in \mathcal{T}_{\text{c}}$, good or bad, in world Y is always simply $2^{-\sigma_m n}$. For a transcript $\tau \in \mathcal{T}_{\text{good}}$, the probability in world X is the probability of all of those plaintext/ciphertext pairs being part of a given random permutation. This is $\prod_{i=0}^{\sigma_m - 1} 1/(2^n - i)$. Thus $\Pr[Y = \tau] \leq \Pr[X = \tau]$ for all $\tau \in \mathcal{T}_{\text{good}}$ as required (subsection 3.3).

3.4.2 Case analysis of collisions

Next we bound $\Pr[Y \in \mathcal{T}_{\text{bad}}]$. We consider a case by case analysis of possible collisions, in either inferred plaintexts ($\mathcal{D}$) or inferred ciphertexts ($\mathcal{R}$), and bound the probability in world Y each case.

Responses are random in world Y, but some caution is required. If we know the adversary's query s, then conditioning on that, we cannot treat the response to query $r < s$ as uniformly random; if the choice of later query depends on the earlier response, knowing the later query is information about the earlier response. However, conditioning on a query and all prior queries and responses, we still have that $\bar{h}$, L, and the query response are uniformly random and independent, and so we can consider them in any order.

Consider for example the case $S_i^r \stackrel{?}{=} MM^s$: for a given r, s, i, we want to evaluate $\Pr[S_i^r = MM^s]$. From 3.4.1, $S_i^r = MM^s$ iff $L = M^r \oplus H_{\bar{h}}(T^r, N^r) \oplus U^r \oplus H_{\bar{h}}(T^r, V^r) \oplus \text{bin}(i) \oplus M^s \oplus H_{\bar{h}}(T^s, N^s)$. $\bar{h}$ and L are given at the end of the transcript and so are independent of

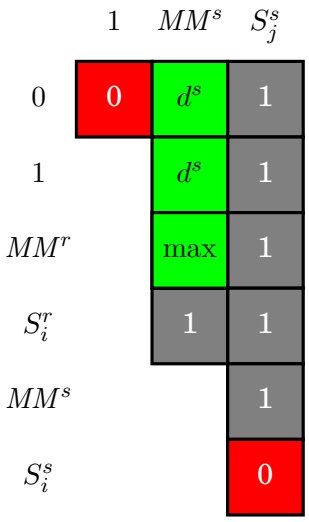

Figure 4: Block cipher plaintext collision cases ($\mathcal{D}$)

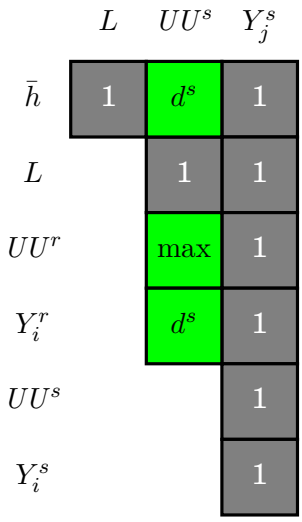

Figure 5: Block cipher ciphertext collision cases ($\mathcal{R}$)

all other queries and responses. If we knew the entire transcript except L, we would know the entire right hand side of this equation. In world X, we would also know for example that $L \neq \bar{h}$, but in world Y, conditioning on the rest of the transcript, all values of L are equally likely; therefore this equation holds with probability exactly $1/2^n$.

There are twenty-two cases to consider, ten of which arise because we use the block cipher to generate $\bar{h}$ and L. In fourteen cases, the probability of a collision between two specific blocks is $1/2^n$:

- $\bar{h} \stackrel{?}{=} L$, $L \stackrel{?}{=} UU^s$, $L \stackrel{?}{=} Y_j^s$, $\mathrm{bin}(0) \stackrel{?}{=} S_j^s$, $\mathrm{bin}(1) \stackrel{?}{=} S_j^s$, $S_i^r \stackrel{?}{=} MM^s$, $MM^r \stackrel{?}{=} S_j^s$, $MM^s \stackrel{?}{=} S_j^s$: Given $\bar{h}$ and all queries and responses, there is exactly one value of L which causes the equation to hold.
- $\bar{h} \stackrel{?}{=} Y_j^s$, $UU^r \stackrel{?}{=} Y_j^s$, $Y_i^r \stackrel{?}{=} Y_j^s$, $Y_i^s \stackrel{?}{=} Y_j^s$ where $i < j$: If query s is an encryption query, then given $\bar{h}$, query s, all prior queries and responses, and $C^s[0; jn]$, there is exactly one value of $C^s[jn; n]$ that results in the equation holding. If s is a decryption query, the same reasoning holds with P^s, C^s swapped.
- $UU^s \stackrel{?}{=} Y_j^s$: If query s is a decryption query, the exact argument above applies. If it is an encryption query, then given $\bar{h}$, T^s, P^s,

and V^s, there is exactly one value of U^s that results in the equation holding.

- $S_i^r \stackrel{?}{=} S_j^s$: If query s is an encryption query, then given $\bar{h}$, L, T^s, P^s, V^s and all prior queries and responses, there is exactly one response value U^s that results in the equation holding. For a decryption query, given $\bar{h}$, L, T^s, C^s, N^s and all prior queries and responses, there is exactly one response value M^s that results in the equation holding.

In two cases, a collision is impossible:

- $\mathrm{bin}(0) \stackrel{?}{=} \mathrm{bin}(1)$: Trivially impossible.
- $S_i^s \stackrel{?}{=} S_j^s$: This is impossible; $S_i^s \oplus S_j^s = \mathrm{bin}(i) \oplus \mathrm{bin}(j)$.

There are six cases where the probability may be greater than $1/2^n$. Considering first collisions with MM^s where query s is an encryption query:

- $MM^r \stackrel{?}{=} MM^s$: This holds iff $M^r \oplus H_{\bar{h}}(T^r, N^r) = M^s \oplus H_{\bar{h}}(T^s, N^s)$. Since pointless queries are forbidden, we have that $(T^r, M^r, N^r) \neq (T^s, M^s, N^s)$. If $(T^r, N^r) = (T^s, N^s)$ then $M^r \neq M^s$, and the equation does not hold. Otherwise, by hash function property 2, the equation holds with probability at most $\max(d^r, d^s)/2^n$.
- $\mathrm{bin}(0) \stackrel{?}{=} MM^s$: This holds iff $\mathrm{bin}(0) = M^s \oplus H_{\bar{h}}(T^s, N^s)$; by hash function property 1, this holds with probability at most $d^s/2^n$.
- $\mathrm{bin}(1) \stackrel{?}{=} MM^s$: As above.

In each case, if query s is a decryption query, then given $\bar{h}$, T^s, C^s, N^s, and all prior queries and responses, all values of M^s are equally likely and a single value causes the equation to hold, for a probability of $1/2^n$.

Similarly, considering collisions with UU^s where query s is a decryption query:

- $UU^r \stackrel{?}{=} UU^s$: This holds iff $U^r \oplus H_{\bar{h}}(T^r, V^r) = U^s \oplus H_{\bar{h}}(T^s, V^s)$; as with the case of $MM^r \stackrel{?}{=} MM^s$, this holds with probability at most $\max(d^r, d^s)/2^n$.

- $Y_i^r \stackrel{?}{=} UU^s$: This holds iff $Y_i^r = U^s \oplus H_{\bar{h}}(T^s, V^s)$; by hash function property 1, this holds with probability at most $d^s/2^n$.
- $\bar{h} \stackrel{?}{=} UU^s$: This holds iff $\bar{h} = U^s \oplus H_{\bar{h}}(T^s, V^s)$; by hash function property 3, this holds with probability at most $d^s/2^n$.

As above, if query s is an encryption query, then given $\bar{h}$, T^s, P^s, V^s, and all prior queries and responses, all values of U^s are equally likely and a single value causes the equation to hold, for a probability of $1/2^n$.

Figure 4 illustrates the various cases we consider for inferred block cipher plaintext collisions. Rows represent the terms on the left hand side of the collision, while columns represent the terms on the right; for example, the top left box represents $\mathrm{bin}(0) \stackrel{?}{=} \mathrm{bin}(1)$. Where a square is left blank it is either because it represents comparing a term to itself (eg $MM^s \stackrel{?}{=} MM^s$) or because it represents something that is already handled elsewhere (eg considering $\mathrm{bin}(1) \stackrel{?}{=} MM^s$ handles the $MM^r \stackrel{?}{=} \mathrm{bin}(1)$ case). A square is colored red and marked 0 if the probability of a particular collision of that kind is zero, grey and marked 1 if the probability is always $1/2^n$, and green if the probability may be greater than $1/2^n$ and depends on the number of solutions to a particular polynomial; d^s when there are at most d^s solutions, $\max$ where there are at most $\max(d^r, d^s)$ solutions. Even where a square is green, if query s is a decryption query, the probability of a particular collision of that kind is $1/2^n$. Figure 5 covers block cipher ciphertext collisions; in this case, it is only decryption queries where probabilities may be above $1/2^n$.

3.4.3 Summing collision bounds

To establish an upper bound on the probability that any pair will collide, we sum collision probabilities for all pairs in the multiset $\mathcal{D}$ and all pairs in the multiset $\mathcal{R}$. To make summing easier, we define a

"correction" c:

$$\begin{aligned}
&\Pr[Y \in \mathcal{T}_{\text{bad}}] \\
=&\Pr[\exists [a,b] \subseteq \mathcal{D} : a = b \vee \exists [a,b] \subseteq \mathcal{R} : a = b] \\
\leq&\left(\sum_{[a,b]\subseteq\mathcal{D}} \Pr[a=b]\right) + \left(\sum_{[a,b]\subseteq\mathcal{R}} \Pr[a=b]\right) \\
=&\left(2\binom{\sigma_m}{2} + c\right)\Big/ 2^n
\end{aligned}$$

where

$$c \stackrel{\text{def}}{=} \left(\sum_{[a,b]\subseteq\mathcal{D}} 2^n \Pr[a=b] - 1\right) + \left(\sum_{[a,b]\subseteq\mathcal{R}} 2^n \Pr[a=b] - 1\right)$$

This rearrangement is so that the fourteen cases above which have a probability of colliding of $1/2^n$ make zero contribution to c and so need not be considered further; only the remaining eight need to be considered. Define $c = c_b + c_f + c_w + c_a$ where

- c_b covers collisions within $\{\text{bin}(0), \text{bin}(1)\}$ and within $\{\bar{h}, L\}$
- c_f covers collisions for all s between $\{\text{bin}(0), \text{bin}(1)\}$ and $\{MM^s, S_1^s, \ldots, S_{m^s-1}^s\}$ and between $\{\bar{h}, L\}$ and $\{UU^s, Y_1^s, \ldots, Y_{m^s-1}^s\}$
- c_w covers collisions for all s within $\{MM^s, S_1^s, \ldots, S_{m^s-1}^s\}$ and within $\{UU^s, Y_1^s, \ldots, Y_{m^s-1}^s\}$
- c_a covers collisions for all $r < s$ between $\{MM^r, S_1^r, \ldots, S_{m^r-1}^r\}$ and $\{MM^s, S_1^s, \ldots, S_{m^s-1}^s\}$ and between $\{UU^r, Y_1^r, \ldots, Y_{m^r-1}^r\}$ and $\{UU^s, Y_1^s, \ldots, Y_{m^s-1}^s\}$

$c_b = -1$, since $\text{bin}(0) \stackrel{?}{=} \text{bin}(1)$ is impossible.

For c_f: if query s is an encryption query, the only nonzero contributions come from the pairs $\text{bin}(0) \stackrel{?}{=} MM^s$ and $\text{bin}(1) \stackrel{?}{=} MM^s$. In each of these cases the probability bound is not $1/2^n$ but $d^s/2^n$, implying a correction of at most $2(d^s - 1)$ for each encryption query. If query s is a decryption query, we need only consider the pair $\bar{h} \stackrel{?}{=} UU^s$ for a correction of at

most $d^s - 1$. Summing across all queries, we conclude that

$$\begin{aligned} c_f \leq & \sum_s \max(2(d^s - 1), d^s - 1) \\ = & \sum_s 2(d^s - 1) \\ \leq & 2\sigma \end{aligned}$$

For c_w: we need only consider $S_i^s \stackrel{?}{=} S_j^s$, which is impossible:

$$\begin{aligned} c_w = & \sum_s -\binom{m^s - 1}{2} \\ \leq & 0 \end{aligned}$$

For c_a: if query s is an encryption query, $MM^r \stackrel{?}{=} MM^s$ gives a correction of at most $\max(d^r, d^s) - 1$; if it is a decryption query, $UU^r \stackrel{?}{=} UU^s$ and $Y_i^r \stackrel{?}{=} UU^s$ give a correction of at most $\max(d^r, d^s) - 1 + (m^r - 1)(d^s - 1)$. Summing across queries, we find

$$\begin{aligned} c_a \leq & \sum_{r<s} \max(\max(d^r, d^s) - 1, \max(d^r, d^s) - 1 + (m^r - 1)(d^s - 1)) \\ = & \sum_{r<s} \max(d^r, d^s) - 1 + (m^r - 1)(d^s - 1) \\ \leq & \sum_{r<s} d^r + d^s - 1 + (m^r - 1)(d^s - 1) \\ \leq & (q-1)\sigma + \sum_{r<s} (m^r - 1)(d^s - 1) \\ \leq & (q-1)\sigma + \sum_{r<s} (d^r - 1)(d^s - 1) \\ \leq & (q-1)\sigma + \binom{\sigma}{2} \end{aligned}$$

Applying the H-coefficient technique, we conclude that

$$\begin{aligned}
&\mathsf{Adv}^{\widetilde{\pm\mathrm{rnd}}}_{\mathrm{HCTR2}[\mathrm{Perm}(n)]}(q,\sigma,t)\\
&\leq \Pr[Y \in \mathcal{T}_{\mathrm{bad}}]\\
&\leq \left(2\binom{\sigma_m}{2} + c_b + c_f + c_w + c_a\right)\Big/ 2^n\\
&\leq \left(2\binom{\sigma_m}{2} - 1 + 2\sigma + (q-1)\sigma + \binom{\sigma}{2}\right)\Big/ 2^n\\
&\leq \left(2\binom{\sigma+2}{2} - 1 + 2\sigma + (q-1)\sigma + \binom{\sigma}{2}\right)\Big/ 2^n\\
&= \left(2\left(\binom{\sigma}{2} + 2\sigma + 1\right) - 1 + 2\sigma + (q-1)\sigma + \binom{\sigma}{2}\right)\Big/ 2^n\\
&= (3\sigma(\sigma-1)/2 + q\sigma + 5\sigma + 1)/2^n\\
&= \big(3\sigma^2 + 2q\sigma + 7\sigma + 2\big)/2^{n+1}
\end{aligned}$$

3.5 Security bound

By a standard substitution argument [BKR94; Bel+97] we have that

$$\mathsf{Adv}^{\mathrm{HCTR2}[\mathrm{Perm}(n)]}_{\mathrm{HCTR2}[E]}(q,\sigma,t) \leq \mathsf{Adv}^{\pm\mathrm{prp}}_{E}(\sigma+2, t+\sigma t')$$

where t' is a small constant representing the per-block cost of simulating HCTR2, and $\sigma+2$ bounds the number of block cipher calls made by the simulator.

Halevi and Rogaway's PRP-RND lemma [HR03, Appendix C, Lemma 6] tells us that

$$\mathsf{Adv}^{\widetilde{\pm\mathrm{prp}}}_{\widetilde{\pm\mathrm{rnd}}}(q,\sigma,t) \leq \binom{q}{2}\Big/ 2^n \leq q^2/2^{n+1}$$

Putting these together with our main lemma, we conclude

$$\begin{aligned}
&\mathsf{Adv}^{\widetilde{\pm\mathrm{prp}}}_{\mathrm{HCTR2}[E]}(q,\sigma,t)\\
\leq\ &\mathsf{Adv}^{\widetilde{\pm\mathrm{prp}}}_{\widetilde{\pm\mathrm{rnd}}}(q,\sigma,t)\\
&+ \mathsf{Adv}^{\widetilde{\pm\mathrm{rnd}}}_{\mathrm{HCTR2}[\mathrm{Perm}(n)]}(q,\sigma,t)\\
&+ \mathsf{Adv}^{\mathrm{HCTR2}[\mathrm{Perm}(n)]}_{\mathrm{HCTR2}[E]}(q,\sigma,t)\\
\leq\ &\mathsf{Adv}^{\pm\mathrm{prp}}_{E}(\sigma+2, t+\sigma t')\\
&+ \big(3\sigma^2 + 2q\sigma + q^2 + 7\sigma + 2\big)/2^{n+1}
\end{aligned}$$

4 HCTR issues

Our presentation of HCTR2 uses names that differ from those used to present HCTR.

- HCTR and HCTR2 use an unusual mode of stream encryption. [WFW05; CN08] refer to this mode as "CTR mode", but note the differences between this mode and standard CTR mode[LRW00]. For the avoidance of ambiguity we name this mode *XCTR mode*.
- What [WFW05; CN08] refer to as C, CC, D we refer to as U, UU, V so that we can use P, C to refer to plaintext and ciphertext. We will use our names in what follows.
- Because of our use of POLYVAL, HCTR2 draws a distinction between the raw hash key $\bar{h}$ and the value at which the polynomial is evaluated h. HCTR has no such distinction and we use h in our discussion of HCTR.

Two errors in previous work on HCTR are addressed in HCTR2.

4.1 Hash function

HCTR uses a hash function based on the polynomial

$$H(X) = \mathrm{poly}(\mathrm{pad}(X) \| \mathrm{bin}(|X|) \| 0^n)$$

Because it assumes a fixed-length tweak it simply sets $X = M \| T$. However, HCTR requires that the resulting polynomial be nonzero even when $X = \lambda$, so as a special case $H(\lambda) = h$.

Unfortunately, as [Kum18] observes this is no longer an injective map from X—we also have $H(0) = h$. This breaks the almost-XOR-universal property relied on in the security bound and straightforwardly leads to an attack in which two encryption queries are presented, one of a block width, and the second extending the first with a single zero (and assuming a zero length tweak).

4.2 Security bound

HCTR was initially presented in [WFW05] with a security bound cubic in the total size of all queries combined. This is a little low for comfort; if a 128-bit block cipher is used, it suggests a distinguisher can be

effective given tens of terabytes of queries, which can quickly be reached over a fast link. [CN08] presents a much more satisfactory quadratic bound, but the proof has an error that invalidates the claimed bound.

While presented in a different way, the proof of [CN08] is very similar to that of this paper: in the game RAND2, all queries get random responses, the block cipher inputs are inferred, and a collision in either the inferred plaintext or inferred ciphertext of the block cipher sets the "bad" flag. Where HCTR2 has $S = MM \oplus UU \oplus L$, HCTR simply has $S = MM \oplus UU$. HCTR uses a fixed-length tweak; for simplicity we assume a zero-length tweak in what follows.

For equation 17, the paper observes that the collision $Y_i^r \stackrel{?}{=} UU^s$ occurs iff h is one of the zeroes of the polynomial $Y_i^r \oplus U^s \oplus H(T^s, V^s)$. This polynomial has degree at most m^s, and so can have at most that many solutions. From this the paper infers a quadratic bound on the probability of any such collision given σ input blocks.

In equation 21, the paper considers collisions of the form $S_i^r \stackrel{?}{=} MM^s$ and asserts that they are quadratically bounded for the same reason. However this equation is crucially different: unlike with Y_i^r, the value we infer for S_i^r depends on h. The values of h for which this collision occurs are the zeroes of the polynomial

$$M^r \oplus H(T^r, N^r) \oplus U^r \oplus H(T^r, V^r) \oplus \mathrm{bin}(i) \oplus M^s \oplus H(T^s, N^s)$$

This polynomial can have degree up to $\max(m^r, m^s)$, and so the bound of equation 17 does not apply. If queries are permitted to be of any length, this leads to a cubic security bound. Consider an adversary who sends a single query with $x + 1$ blocks, followed by x queries of one block. For each $i \in \{1 \ldots x\}$ and for each $s \in \{2 \ldots x + 1\}$, we have at best $\Pr[S_i^r = MM^s] \leq (x + 1)/2^n$. Summing all these bounds for each such pair, we find that for queries with $\sigma = 2x + 1$ this technique yields an upper bound on the total probability of such a collision of $(x^3 + x^2)/2^n$.

[Nan21] observes that a quadratic bound can be recovered if a bound $l_{\max}$ is set on the maximum size of a single query; in this case we can prove a bound which is some small multiple of $l_{\max} q\sigma/2^n$.

5 Design of HCTR2

HCTR2 is intended as a successor to HCTR and retains several of the features that make it an attractive design:

- An unbalanced Feistel-like network based on universal hashing, with a single block encryption on the narrow side. This gives excellent performance and parallelizability, as well as natural handling of messages that are not a multiple of the block size. It also efficiently handles messages as small as the block size; our work on HCTR2 is motivated by filename encryption for Linux's *fscrypt* module[21], where short messages will be commonplace.
- Use of $MM \oplus UU$ in generating S, which means that an adversary's control over S is very limited for both encryption and decryption queries; this is used in the proof of security to bound $S_i^r \stackrel{?}{=} S_j^s$ and avert a cubic term in the security bound.
- The CTR mode variant XCTR. Because of the extra constant L, it would be straightforward to prove secure an HCTR2 variant that used CTR mode. However, unlike CTR, XCTR never needs to maintain a counter larger than needed for the message size; when a 128-bit nonce is used, CTR must use a 128-bit counter, and is at risk of implementations whose flaws only manifest on the rare occasions that a counter overflows into the next word. GCM[MV04] uses a variant of CTR in which only 32 bits are incremented for the same reason; XCTR seems a more elegant solution.

HCTR2 differs from HCTR in the following ways:

- We introduce the extra key-dependent constant L, so that where HCTR has $S = MM \oplus UU$ we have $S = MM \oplus UU \oplus L$, fixing the issue described in subsection 4.2 and restoring the quadratic security bound.
- We redesign the polynomial hash input format, as described in subsubsection 5.2.2, fixing the issue described in subsection 4.1.
- We accept a variable-length tweak. This increases flexibility, and eliminates the risk of attacks where two users of the same key have different ideas of what the tweak length is.

- We derive $\bar{h} \leftarrow E_k(\mathrm{bin}(0))$ and $L \leftarrow E_k(\mathrm{bin}(1))$ from the block cipher key; this makes HCTR2 more convenient to use.
- We specify POLYVAL[GLL17; GLL19] as the polynomial evaluation function, for reasons set out in subsubsection 5.2.1.
- We present a new proof, based on the H-coefficient technique, with a tighter bound.
- We specify endianness and the like so that implementations can be interoperable. We use little-endian representation everywhere, since this is faster on nearly all modern platforms. This is another difference between XCTR and CTR, since CTR is defined to be big-endian.
- We rename some variables in our exposition and proof to allow some more standard usage.
- We provide a sample implementation and test vectors.

5.1 Comparison of SPRP modes

We considered a number of modes aiming to provide tweakable super-pseudorandom permutations as the basis for our design before settling on HCTR. HCTR is simpler than all of these modes except HHFHFH; each also has specific qualities that led us to choose an HCTR variant in preference.

- CMC[HR03], EME[HR04], and EME*[Hal05] require two block cipher calls per input block.
- PEP[CS06], TET[Hal07], and HEH[Sar07] are complex, and are either unable to handle messages that are not multiples of the block size, or require extra ciphertext-stealing like tricks to handle such messages. In addition, they require five passes over the data, or three if passes are combined. Thanks to the simplicity of the unbalanced Feistel network, HCTR and HCTR2 require three passes, or two if combined.
- HCH[CS08] is similar to HCTR but uses $S = E_k(MM \oplus UU)$. With this change the authors were able to prove a quadratic security bound. Our modification, $S = MM \oplus UU \oplus E_k(\mathrm{bin}(1))$, saves a block cipher call per invocation.

- HSE[MM07] achieves similar performance to this mode, but is significantly more complicated, and accepts only an n-bit tweak.
- HMC[Nan08] allows the encryption of the first block to run in parallel with subsequent blocks, but at a cost of significant complication of decryption, which does not gain this advantage; at key setup time, the multiplicative inverse of the hash key must be calculated. In addition, like HCTR (subsection 4.1) HMC's hash function is not correctly injective onto polynomials.
- FAST[Cha+20] uses only the encryption direction of the block cipher. However it is fairly complex, and the minimum message size is twice the width of the block cipher; for our application we need efficient handling of small messages.
- HHFHFH[Ber16] is a particularly clean design based on a four-round Feistel network, but requires a 2^{4n}-bit message size for n-bit security; again this doesn't meet our small-message needs.

5.2 Hash function design

5.2.1 POLYVAL

We aim to specify HCTR2 in sufficient detail for implementations to be interoperable, so we must be precise about endianness and the like in $\mathrm{GF}(2^{128})$ polynomial evaluation. The most widely used convention is that of GCM's GHASH[MV04]. However, GHASH is not consistent in its endianness conventions, which increases implementation complexity and reduces efficiency.

Instead, we use the POLYVAL function defined in [GLL17; GLL19]. POLYVAL incurs a small cost in specification and proof complexity because the polynomial is evaluated not at the parameter $\bar{h}$ but at $x^{-n}\bar{h}$ so that Montgomery multiplication[Mon85] can be key-agile. However it is carefully designed, efficient on processors with carryless multiply instructions (1.2x faster than GHASH according to [GLL17]) and offers an efficient conversion between POLYVAL and GHASH hashing which allows code/hardware for one to be used for the other.

5.2.2 Input formatting

The formatting of inputs to the polynomial hash in HCTR2 is significantly different from that in HCTR. Our design goals are:

- fix the flaw described in [Kum18]
- allow a variable-length tweak
- guarantee $H(T, M) \neq \bar{h}$, required because $\bar{h} \leftarrow E_k(\mathrm{bin}(0))$
- allow implementations to precompute as much as possible, to reduce $\mathrm{GF}(2^n)$ multiplications

See subsection 3.2 for the properties we require of the hash function.

To fix the flaw described in [Kum18], we eliminate the zero-length special case by adding one to the length before encoding it.

We process the tweak before the message, so that implementations need only process the tweak once for each encryption/decryption, instead of twice.

With the introduction of the constant L our security proof no longer relies on the hash function having property 2 of [WFW05, Section 3.3]. This allows us to move the length block first, so that implementations need only process it once per encryption/decryption. It is never zero, so its position in the polynomial can be inferred from the degree.

This change means that $\lceil |T|/n \rceil + \lceil |M|/n \rceil$ can be inferred from the degree of the polynomial. If we append a 1 to the message before padding with zeroes, we need only encode only the tweak length in the length block, and the message length can then be inferred. For users whose tweaks are always the same length this means the length block is always the same, so the multiplication with h can be precomputed.

However, in the very common case where the message length is a multiple of the block size, we don't want an extra multiplication for an extra block containing only the appended 1 bit. Borrowing from [BR00b], we don't append a 1 bit to such messages. Instead, we indicate whether the message length is a multiple of the block size in the least significant bit of the length block. If all tweaks are of length t, implementations can cache $\mathrm{bin}(2t+2)h$ and $\mathrm{bin}(2t+3)h$ and use one of

these to start hashing as appropriate, XORing this value directly with the first block of the tweak.

5.2.3 Alternatives considered

Like HCTR, HCTR2's almost-XOR-universal hash function uses a standard polynomial evaluation in $\mathrm{GF}(2^n)$. This uses an n-bit key and requires l field multiplications where l is the number of blocks. We considered several alternatives:

BRW polynomials: BRW polynomials[Ber07][Sar09] are theoretically attractive since they only need $\lfloor l/2 \rfloor$ multiplications to evaluate. However, they pose a number of difficulties. [GS19] gives a nonrecursive algorithm that handles variable-length messages, but it is complex, uses temporary space that grows logarithmically with the message length, and does not handle incremental computation well. Standard polynomials avoid these issues; fast and correct implementations are easier to write, and implementers have much more control over code size, precomputation, instruction-level parallelism, number of reductions, and so forth. Finally, preserving our guarantees of injectivity on variable-length tweaks and messages, and the other hash function properties we need to guarantee, proved challenging.

Hash2L: Hash2L[CGS17] solves two issues with BRW polynomials. First, it limits the depth of recursion, and thus the space needed, by replacing the uppermost levels by a simpler Horner based evaluation. This slightly increases the number of multiplications per block but solves several implementation issues. Secondly, it adds an extra multiplication at the end to include length information so that the whole construction is injective on variable-length messages. Where most messages are large, such as for disk encryption, a variant of HCTR2 that uses Hash2L could be attractive; however since performance on small messages is key to our application we prefer the simplicity and optimization potential of Horner evaluation.

Polynomials over non-binary fields: When CPU instructions for carryless multiplication are unavailable, hashes using non-binary fields such as Poly1305[Ber05] tend to be faster than hashes using binary fields. However, HCTR2 primarily targets processors that support carryless multiplication, and on such processors hashes using

binary fields tend to be faster and simpler.

Multivariate hashes: Adiantum[CB18] builds an almost-Δ-universal hash function using the multivariate hash NH[Bla+99] combined with polynomial evaluation. Where NH is faster than polynomial evaluation, this increases performance. However, this adds complexity, and NH requires a long key which needs to be derived and cached. HCTR2 primarily targets processors where polynomial evaluation is fast, so we do not add an NH layer.

6 Implementation

At https://github.com/google/hctr2 we provide a (very slow) reference implementation of HCTR2 in Python, a portable C implementation, and assembly implementations for x86-64 and ARM64 making use of AES acceleration and carryless multiplication instructions. We adapted our assembly implementation of XCTR mode from the Linux kernel's CTR mode implementation, retaining all parallelism.

References

[21] *Filesystem-level encryption (fscrypt)*. Linux kernel documentation. 2021. URL: https://www.kernel.org/doc/html/v5.15/filesystems/fscrypt.html.

[Bel+97] Mihir Bellare et al. "A Concrete Security Treatment of Symmetric Encryption". In: *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*. FOCS '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 394–. ISBN: 0-8186-8197-7. DOI: 10.1109/SFCS.1997.646128. URL: https://web.cs.ucdavis.edu/~rogaway/papers/sym-enc.pdf.

[Ber05] Daniel J. Bernstein. "The Poly1305-AES message-authentication code". In: *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21–23, 2005, revised selected papers*. Ed. by Henri Gilbert and Helena Handschuh. Vol. 3557. Lecture Notes in Computer Science. Springer, 2005,

pp. 32–49. ISBN: 3–540–26541–4. URL: https://cr.yp.to/papers.html#poly1305.

[Ber07] Daniel J. Bernstein. “Polynomial evaluation and message authentication”. In: *Contemporary Mathematics* (2007). URL: https://cr.yp.to/antiforgery/pema-20071022.pdf.

[Ber16] Daniel J. Bernstein. *Some challenges in heavyweight cipher design*. Tech. rep. University of Illinois at Chicago and Technische Universiteit Eindhoven, 2016. URL: https://cr.yp.to/talks/2016.01.15/slides-djb-20160115-a4.pdf.

[BKR94] Mihir Bellare, Joe Kilian, and Phillip Rogaway. “The Security of Cipher Block Chaining”. In: *Advances in Cryptology — CRYPTO ’94*. Ed. by Yvo G. Desmedt. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 341–358. ISBN: 978-3-540-48658-9. DOI: 10.1006/jcss.1999.1694. URL: https://cseweb.ucsd.edu/~mihir/papers/cbc.pdf.

[Bla+99] John Black et al. “UMAC: Fast and Secure Message Authentication”. In: *Advances in Cryptology — CRYPTO’ 99*. Ed. by Michael Wiener. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 216–233. ISBN: 978-3-540-48405-9. DOI: 10.1007/3-540-48405-1_14. URL: https://fastcrypto.org/umac/umac_proc.pdf.

[BR00a] Mihir Bellare and Phillip Rogaway. “Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography”. In: *Advances in Cryptology — ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 317–330. ISBN: 978-3-540-44448-0. DOI: https://doi.org/10.1007/3-540-44448-3_24. URL: https://web.cs.ucdavis.edu/~rogaway/papers/encode.pdf.

[BR00b] John Black and Phillip Rogaway. “CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions”. In: *Advances in Cryptology — CRYPTO 2000*. Ed. by Mihir Bellare. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 197–215. ISBN: 978-3-540-44598-2. URL: https://www.cs.ucdavis.edu/~rogaway/papers/3k.pdf.

[CB18] Paul Crowley and Eric Biggers. “Adiantum: length-preserving encryption for entry-level processors”. In: *IACR Transactions on Symmetric Cryptology* 2018.4 (2018-12), pp. 39–61. DOI: 10.13154/tosc.v2018.i4.39-61. URL: https://tosc.iacr.org/index.php/ToSC/article/view/7360.

[CGS17] Debrup Chakraborty, Sebati Ghosh, and Palash Sarkar. “A Fast Single-Key Two-Level Universal Hash Function”. In: *IACR Transactions on Symmetric Cryptology* 2017.1 (2017-03), pp. 106–128. DOI: 10.13154/tosc.v2017.i1.106-128. URL: https://tosc.iacr.org/index.php/ToSC/article/view/586.

[Cha+20] Debrup Chakraborty et al. “FAST: Disk encryption and beyond”. In: *Advances in Mathematics of Communications* 0.1930-5346_2019_0_92 (2020). ISSN: 1930-5346. DOI: 10.3934/amc.2020108. URL: http://aimsciences.org//article/id/ac2cbcad-a848-4d90-9c63-981709c4f988.

[CN08] Debrup Chakraborty and Mridul Nandi. “An Improved Security Bound for HCTR”. In: *Fast Software Encryption*. Ed. by Kaisa Nyberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 289–302. ISBN: 978-3-540-71039-4. DOI: 10.1007/978-3-540-71039-4_18. URL: https://www.iacr.org/cryptodb/archive/2008/FSE/paper/15611.pdf.

[CS06] Debrup Chakraborty and Palash Sarkar. “A New Mode of Encryption Providing a Tweakable Strong Pseudo-random Permutation”. In: *Fast Software Encryption: 13th International Workshop, Graz, Austria, March 15–17, 2006, Revised Selected Papers*. Ed. by Matthew Robshaw. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 293–309. ISBN: 978-3-540-36598-3. DOI: 10.1007/11799313_19. URL: https://ia.cr/2006/275.

[CS08] Debrup Chakraborty and Palash Sarkar. “HCH: A New Tweakable Enciphering Scheme Using the Hash-Counter-Hash Approach”. In: *IEEE Transactions on Information Theory* 54.4 (2008-04), pp. 1683–1699. ISSN: 0018-9448. DOI: 10.1109/TIT.2008.917623. URL: https://ia.cr/2007/028.

[CS14] Shan Chen and John Steinberger. “Tight Security Bounds for Key-Alternating Ciphers”. In: *Advances in Cryptology – EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 327–350. ISBN: 978-3-642-55220-5. DOI: 10.1007/978-3-642-55220-5_19. URL: https://ia.cr/2013/222.

[Dwo+01] Morris Dworkin et al. *Advanced Encryption Standard (AES)*. en. 2001-11. DOI: https://doi.org/10.6028/NIST.FIPS.197. URL: https://www.nist.gov/publications/advanced-encryption-standard-aes.

[GLL17] Shay Gueron, Adam Langley, and Yehuda Lindell. *AES-GCM-SIV: Specification and Analysis*. Cryptology ePrint Archive, Report 2017/168. 2017. URL: https://ia.cr/2017/168.

[GLL19] Shay Gueron, Adam Langley, and Yehuda Lindell. *AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption*. RFC 8452. RFC Editor, 2019-04. URL: https://www.rfc-editor.org/rfc/rfc8452.txt.

[GS19] Sebati Ghosh and Palash Sarkar. “Evaluating Bernstein-Rabin-Winograd polynomials”. In: *Designs, Codes and Cryptography* 87 (2019), pp. 527–546. DOI: 10.1007/s10623-018-0561-7. URL: https://ia.cr/2017/328.

[Hal05] Shai Halevi. “EME*: Extending EME to Handle Arbitrary-Length Messages with Associated Data”. In: *Progress in Cryptology - INDOCRYPT 2004: 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004. Proceedings*. Ed. by Anne Canteaut and Kapaleeswaran Viswanathan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 315–327. ISBN: 978-3-540-30556-9. DOI: 10.1007/978-3-540-30556-9_25. URL: https://ia.cr/2004/125.

[Hal07] Shai Halevi. “Invertible Universal Hashing and the TET Encryption Mode”. In: *Advances in Cryptology - CRYPTO 2007: 27th Annual International Cryptology Conference,*

Santa Barbara, CA, USA, August 19-23, 2007. Proceedings. Ed. by Alfred Menezes. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 412–429. ISBN: 978-3-540-74143-5. DOI: 10.1007/978-3-540-74143-5_23. URL: https://ia.cr/2007/014.

[HKR15] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. "Robust Authenticated-Encryption AEZ and the Problem That It Solves". In: *Advances in Cryptology – EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 15–44. ISBN: 978-3-662-46800-5. DOI: https://doi.org/10.1007/978-3-662-46800-5_2. URL: https://ia.cr/2014/793.

[HR03] Shai Halevi and Phillip Rogaway. "A Tweakable Enciphering Mode". In: *Advances in Cryptology - CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings*. Ed. by Dan Boneh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 482–499. ISBN: 978-3-540-45146-4. DOI: 10.1007/978-3-540-45146-4_28. URL: https://ia.cr/2003/148.

[HR04] Shai Halevi and Phillip Rogaway. "A Parallelizable Enciphering Mode". In: *Topics in Cryptology – CT-RSA 2004: The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*. Ed. by Tatsuaki Okamoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 292–304. ISBN: 978-3-540-24660-2. DOI: 10.1007/978-3-540-24660-2_23. URL: https://ia.cr/2003/147.

[Kum18] Manish Kumar. "Security of XCB and HCTR". MA thesis. Indian Statistical Institute, 2018-07. URL: http://library.isical.ac.in:8080/jspui/handle/10263/6953.

[LRW00] Helger Lipmaa, Phillip Rogaway, and David Wagner. *CTR-Mode Encryption*. Comments to NIST concerning AES modes of operation. 2000. URL: https://www.cs.ucdavis.edu/~rogaway/papers/ctr.pdf.

[MM07] Kazuhiko Minematsu and Toshiyasu Matsushima. “Tweakable Enciphering Schemes from Hash-Sum-Expansion”. In: *Progress in Cryptology – INDOCRYPT 2007*. Ed. by K. Srinathan, C. Pandu Rangan, and Moti Yung. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 252–267. ISBN: 978-3-540-77026-8. DOI: 10.1007/978-3-540-77026-8_19.

[Mon85] Peter L. Montgomery. “Modular Multiplication Without Trial Division”. In: *Mathematics of Computation* 44 (1985), pp. 519–521. ISSN: 0025–5718. DOI: https://doi.org/10.1090/S0025-5718-1985-0777282-X. URL: https://www.ams.org/journals/mcom/1985-44-170/S0025-5718-1985-0777282-X/S0025-5718-1985-0777282-X.pdf.

[MV04] David McGrew and John Viega. *The Galois/Counter Mode of Operation (GCM)*. Submission to NIST Modes of Operation Process. 2004. URL: https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf.

[Nan08] Mridul Nandi. *Improving upon HCTR and matching attacks for Hash-Counter-Hash approach*. Cryptology ePrint Archive, Report 2008/090. 2008. URL: https://ia.cr/2008/090.

[Nan21] Mridul Nandi. Private communication. 2021-09.

[Pat09] Jacques Patarin. “The “Coefficients H” Technique”. In: *Selected Areas in Cryptography*. Ed. by Roberto Maria Avanzi, Liam Keliher, and Francesco Sica. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 328–345. ISBN: 978-3-642-04159-4. DOI: 10.1007/978-3-642-04159-4_21. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.702.3488.

[Pat91] Jacques Patarin. “Pseudorandom permutations based on the D.E.S. scheme”. In: *EUROCODE ’90*. Ed. by Gérard Cohen and Pascale Charpin. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 193–204. ISBN: 978-3-540-47546-0. DOI: 10.1007/3-540-54303-1_131.

[Sar07] Palash Sarkar. “Improving Upon the TET Mode of Operation”. In: *Information Security and Cryptology—ICISC 2007: 10th International Conference, Seoul, Korea, November 29–30, 2007. Proceedings*. Ed. by Kil-Hyun Nam and Gwangsoo Rhee. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 180–192. ISBN: 978-3-540-76788-6. DOI: 10.1007/978-3-540-76788-6_15. URL: https://ia.cr/2007/317.

[Sar09] Palash Sarkar. “Efficient Tweakable Enciphering Schemes From (Block-Wise) Universal Hash Functions”. In: *IEEE Transactions on Information Theory* 55.10 (2009), pp. 4749–4760. DOI: 10.1109/TIT.2009.2027487. URL: https://ia.cr/2008/004.

[Sti95] Douglas R. Stinson. “On the Connections Between Universal Hashing, Combinatorial Designs and Error-Correcting Codes”. In: *Electronic Colloquium on Computational Complexity (ECCC)* 2.52 (1995). URL: http://eccc.hpi-web.de/eccc-reports/1995/TR95-052/index.html.

[WFW05] Peng Wang, Dengguo Feng, and Wenling Wu. “HCTR: A Variable-Input-Length Enciphering Mode”. In: *Information Security and Cryptology: First SKLOIS Conference, CISC 2005, Beijing, China, December 15-17, 2005. Proceedings*. Ed. by Dengguo Feng, Dongdai Lin, and Moti Yung. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 175–188. ISBN: 978-3-540-32424-9. DOI: 10.1007/11599548_15. URL: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.470.5288.

A Injectivity of H onto polynomials

To demonstrate injectivity, the following algorithm recovers T and M given a binary string X of length $|X| = n(1 + \deg(H(T, M)))$ representing the coefficients of the polynomial $H(T, M)$ in binary form, starting with the greatest nonzero power; thus $13h^3$ would encode as $1011\|0^{4n-4}$.

1: **procedure** GETTM(X)
2: **assert** $|X| \bmod n = 0$

```
    assert |X| ≥ 2n
    assert X[|X| − n; n] = 0^n
    t ← bin^{−1}_{n−1}(X[1; n − 1])
    assert t > 0
    t ← t − 1
    w ← n(1 + ⌈t/n⌉)
    if X[0; 1] = 0 then
        assert w + n ≤ |X|
        M ← X[w; |X| − w − n]
    else
        assert w + 2n ≤ |X|
        assert X[|X| − 2n + 1; n − 1] ≠ 0^{n−1}
        i ← |X| − n − 1
        while X[i; 1] = 0 do
            i ← i − 1
        end while
        M ← X[w; i − w]
    end if
    T ← X[n; t]
    return T, M
end procedure
```