Attacking the IETF/ISO Standard for Internal Re-keying CTR-ACPKM

Orr Dunkelman, Shibam Ghosh and Eran Lambooij

Department of Computer Science, University of Haifa, Haifa, Israel
orrd@cs.haifa.ac.il,sghosh03@campus.haifa.ac.il,eran@hideinplainsight.io

Abstract. Encrypting too much data using the same key is a bad practice from a security perspective. Hence, it is customary to perform re-keying after a given amount of data is transmitted. While in many cases, the re-keying is done using a fresh execution of some key exchange protocol (e.g., in IKE or TLS), there are scenarios where internal re-keying, i.e., without exchange of information, is performed, mostly due to performance reasons.

Originally suggested by Abdalla and Bellare, there are several proposals on how to perform this internal re-keying mechanism. For example, Liliya et al. offered the CryptoPro Key Meshing (CPKM) to be used together with GOST 28147-89 (known as the GOST block cipher). Later, ISO and the IETF adopted the Advanced CryptoPro Key Meshing (ACKPM) in ISO 10116 and RFC 8645, respectively.

In this paper, we study the security of ACPKM and CPKM. We show that the internal re-keying suffers from an entropy loss in successive repetitions of the re-keying mechanism. We show some attacks based on this issue. The most prominent one has time and data complexities of $O(2^{\kappa/2})$ and success rate of $O(2^{-\kappa/4})$ for a κ-bit key.

Furthermore, we show that a malicious block cipher designer or a faulty implementation can exploit the ACPKM (or the original CPKM) mechanism to significantly hinder the security of a protocol employing ACPKM (or CPKM). Namely, we show that in such cases, the entropy of the re-keyed key can be greatly reduced.

Keywords: CTR-ACPKM · Multi-user Attack · Entropy Loss · Key Collision

1 Introduction

A common security-enhancing practice is to restrict the duration in which a given key is used. This is done, for example, as a simple mitigation to attacks relying on collisions, such as Sweet32 [BL16]. It is thus, customary to change the encryption key after a set amount of time or encryptions. The maximum amount of data that can be safely encrypted under a single key in any cryptographic protocol is called the *key lifetime.*

Changing the key usually requires to run a *key-exchange* protocol, which can incur high computation and communication costs. This inspired further research on *re-keying* the existing secret key without running a key-exchange protocol.

To avoid frequent updates of the key, there are several ways of extending the key lifetime in symmetric key cryptosystems. One of these mechanisms, originally suggested by Abdalla and Bellare [AB00], is the internal re-keying mechanism. This mechanism generates a new secret key K_i for the i-th epoch based on the previous key(s).

The re-keying mechanism can occur in various protocol levels: At the block cipher level (fresh re-keying [DFH+16]), at the mode of operation level (internal re-keying [AAOS17, Smy19]), and at the protocol level (external re-keying [AAOS17, Smy19]).

In this paper, we study the security of an internal re-keying mechanism called Advanced CryptoPro Key Meshing (ACPKM) [Smy19] and its predecessor CryptoPro Key Meshing (CPKM) [PLK06]. The main idea behind these two internal re-keying mechanisms is to call a key update function after a key is used to encrypt a pre-defined number of blocks (i.e., *a section*). The ACPKM transformation generates a new key by encrypting one or more public constants with the previous key.

We provide a security analysis of CTR-ACPKM [AAS18], which combines the counter mode (CTR) operation using the ACPKM mechanism for internal re-keying. The CTR-ACPKM mode is currently passing through the last formal standardization process in IETF (CFRG) [Smy19] and was standardized by ISO (ISO 10116) [ISO]. Also, the internal re-keying technique is being widely used in the Russian variants of TLS [ANI+08] and CMS [LC06]. Thus, it is essential to analyze the security of these mechanisms as tens, if not hundreds, of millions of users are relying on their security.

1.1   Related Work

The re-keying mechanism was first introduced by Abdalla and Bellare in [AB00]. The idea is that frequent re-keying increases the security as well the key lifetime. An interesting proposal for a re-keying mechanism is CPKM [PLK06], used with the GOST 28147-89 cipher (which today is called MAGMA). This method generates a new key in the following manner:

$$\mathsf{K}_{i+1} = E_{\mathsf{K}_i}^{-1}(D_1)\|E_{\mathsf{K}_i}^{-1}(D_2)\|E_{\mathsf{K}_i}^{-1}(D_3)\|E_{\mathsf{K}_i}^{-1}(D_4)$$

for some constants D_1, D_2, D_3 and D_4. Its security properties when combined with the CTR mode of operation are analyzed by Liliya *et al.* in [AAO+16].

Since counter mode does not use decryption, the use of decryption in the CPKM mechanism is considered a drawback. Another drawback is that the inputs to the block cipher may collide with the constants used for the key generation. When this happens (i.e., when D_i is equal to some output of the block cipher), information about the key leaks. This would of course result in a trivial attack.

The ACPKM (advanced CPKM) mechanism, proposed in CTCrypt'2016, addresses one of these problems. The new method generates a new key in the following way:

$$\mathsf{K}_{i+1} = E_{\mathsf{K}_i}(D_1)\|\cdots\|E_{\mathsf{K}_i}(D_r)$$

for some carefully chosen constants $D_1, D_2, D_3, ..., D_r$. As a result there is no need to implement decryption, and if ACPKM is used in a mode of operation where the user cannot control the input to the cipher, such as in CTR-mode, then the second problem is avoided.

The security of CTR-ACPKM was analyzed in [AAS18]. In [AASO20], Liliya *et al.*, analyse ACPKM in combination with the GCM mode of operation (GCM-ACPKM). The analysis of CTR-ACPKM in [AAS18] claims IND-CPNA security (indistinguishability under Chosen Plaintext and Nonce Attack) model. The claimed bound on the IND-CPNA advantage is

$$\frac{(\sigma_1 + r)^2 + \ldots + (\sigma_{\ell-1} + r)^2 + \sigma_\ell^2}{2^{n+1}}$$

where n is the block size, κ is the key size, σ_j is the total length of data blocks processed under the j-th key (called a *section*), ℓ is the maximal number of sections, and $r = \left\lceil \frac{\kappa}{n} \right\rceil$. This led both IETF and ISO to accept ACPKM as a mechanism for internal re-keying. In this paper, we show that despite the security proof, the proposed update causes an entropy drop in the key space. We show that this entropy loss impacts the security of CTR-ACPKM and puts users at risk. We note that entropy loss of repeated invocations of random functions has been studied before [FO89, FS09], as it also impacts the analysis of MACs and hash functions [PvO96, LPW13, DL14]. A survey covering these attacks can be found in [BGW18]. We build on these results and offer a somewhat more delicate

analysis of the entropy (specifically, we consider Shannon entropy rather than the H_0 entropy considered and used in most previous works).

In addition, if the designer of a cipher can successfully embed a high probability differential in the design (e.g., such as in the MALICIOUS framework [PW20]), then this entropy loss can be significantly accelerated.[1]

1.2 The Multi User Security Setting

In the multi-user security setting the adversary is allowed to distribute its online queries adaptively across multiple instances of the construction. This setting is originally proposed in Biham's technical report [Bih96, Bih02] and formalized by Bellare *et al.* [BBM00] in the context of public-key encryption. In other words, the adversary can obtain the encryption of a message with respect to a large number of users under independent keys $K_1, K_2, ..., K_u$. For a nonce-based encryption algorithm, the adversary gets an encryption oracle that takes an index $i \in \{1, 2, ..., u\}$, a message, and a nonce, and returns an encryption of these under K_i. The attacker can fix a pair of nonce and message (N, M), and send them to different users. Consequently, the attacker receives $C_i = E_{K_i}(N, M)$ for each $i = 1, 2, ..., u$. Upon receiving the responses, the attacker's goal is to recover at least one of the users' keys in a key-recovery attack. In a distinguishing game the adversary gets an encryption oracle that takes an index $i \in \{1, 2, ..., u\}$, a message, and a nonce, after which it returns either an encryption of these under K^i in the *real world* or a random string of the same length in the *ideal world*.

The key recovery attack in the multi-user setting is an important aspect to consider from a mass-surveillance perspective, where billions of users use the same cryptographic algorithm with independent keys (e.g., CTR-ACPKM [Smy19] or AES-GCM [BT16] is now widely used in TLS [ANI+08] protocol to protect web traffic and is currently used by billions of users daily). Due to its practical relevance, key recovery attacks in multi-user setting have been studied in a series of work in the last few years [BBT16, BT16, BMS05, BHT18, CMS11, HTT18, LMP17, ML15, MPS20].

1.3 Our Contributions

In this work, we propose several attacks on the CTR-CPKM and on the CTR-ACPKM mode. These modes use the CPKM or the ACPKM internal re-keying technique to update the key (i.e. generating a new key for each section of a hard drive). Our work studies the entropy loss due to frequent re-keying using these schemes. Based on this entropy loss, we propose three attacks on the CTR-ACPKM (or the CTR-CPKM) mode. Our first attack is an improved exhaustive search. Secondly, we propose a distinguishing attack based on *key collisions*, which are more frequent due to the entropy loss. Our third attack is a key-recovery attack, motivated by [Bih96]. In the last two attacks, the adversary is modeled in the multi-user security setting. Furthermore, we show that these attacks may be more efficient when the key size is greater than the block size. Furthermore, our experiments on several ciphers showed that the H_1-entropy loss in the ACPKM transformation is much more effective than the H_0-entropy loss. Based on this, we propose an improved master-key recovery attack.

In addition we propose two different attacks based on differential properties of the underlying block cipher. The crucial part of these attacks is that updated keys can be

---

[1]One can argue that such ciphers, where we do not trust the designer, should not be accepted as standards. However, as evident by reality, many times such ciphers are being selected for widely deployed standards — e.g., the GEA-1 algorithm, the DUAL-EC DRBG, or Speck (which was standardized by ISO in RFID communication standards). Furthermore, some people, e.g., residents of the Russian Federation, must use some cryptographic standards that were not openly negotiated. Hence, it is important to understand what level of security they can expect from a given system.

directly recovered if a differential property exists in the underlying block cipher. This may occur either accidentally or if the cipher was backdoored. We want to mention that all the attacks we have presented here are equally applicable to GCM-ACPKM [AASO20] due to its similarity in the encryption part with CTR-ACPKM mode.

Next, we show that a malicious designer can further harm the mode, if a specific high probability related-key differential or linear property exists in the underlying block cipher. We show that such a property leads to a significantly higher entropy loss. Furthermore, we show that even an innocent implementation error can lead to devastating attacks. This issue may even be triggered when following the current ACPKM standard with a block size that is not a power of two. Finally, we proposes a related-key distinguisher on the CTR-ACPKM mode.

We discuss the attacks on the CTR-ACPKM mode, but these attacks trivially carry over to the CTR-CPKM mode.

1.4   Organization of the Paper

We develop necessary notations and conventions in Section 2. In Section 3 we apply the existing analysis to the ACPKM construction and analyse its security. Afterwards, in Section 4, we refine the analysis of the loss of entropy and introduce new attacks based on this analysis. In Section 5, we consider the misuse resistance of the ACPKM construction. In Section 6 we discuss the security of ACPKM in the multi-user setting. Finally, we conclude the paper in Section 7.

2   Preliminaries

For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. For any positive integer N and p, $(\mathsf{N})_p$ denotes $\mathsf{N}(\mathsf{N}-1)...(\mathsf{N}-p+1)$. We refer to an element of $\{0,1\}^n$ as block. For $X \in \{0,1\}^*$, if $\|X\|$ is a multiple of n, $\|X\|_n$ denotes the length of X in n-bit blocks, i.e., $\|X\|_n = \lceil \|X\|/n \rceil$. For any non-empty binary string X, $(X[1], \ldots, X[k]) \xleftarrow{n} X$ denotes the n-bit block parsing of X, where $\|X[i]\| = n$ for $1 \le i \le k-1$, and $1 \le \|X[k]\| \le n$. For $X \in \{0,1\}^*$ and a positive integer $t \le \|X\|$, we use $\mathsf{MSB}_t(X)$ and $\mathsf{LSB}_t(X)$ to denote the bit string of the t most significant bits and the t least significant bits of X, respectively. For any non-negative integer I and t, we denote the t-bit representation of I with $\mathsf{str}_t[I]$ where the least significant bit is on the right. Similarly, from a bit string $X \in \{0,1\}^t$, we write $\mathsf{int}(X)$ to denote the integer representation of X. For a positive integer $c < n$, let $\mathsf{INC}_c(X)$ be the function which takes the input $X \in \{0,1\}^n$ and outputs the increment of X as

$$\mathsf{INC}_c(X) = \mathsf{MSB}_{n-c}(X) \| \mathsf{str}_c(\mathsf{int}(\mathsf{LSB}_c(X)) + 1 \mod 2^c).$$

Consequently, we denote i consecutive increments as $\mathsf{INC}_c^i(X)$ which is the composition of the INC_c function i times.

Definition 1. (H_0-entropy) Let X be a discrete random variable such that $\Pr[X = x] > 0$ for all $x \in \mathcal{A}$. The H_0-entropy is defined as

$$\mathsf{H}_0[\mathsf{X}] = \log(|\mathcal{A}|).$$

Definition 2. (H_1-entropy) Let X be a discrete random variable with possible outcomes in the set $\mathcal{A} = \{x_1, x_2, ..., x_n\}$. The H_1-entropy is defined as

$$\mathsf{H}_1[\mathsf{X}] = \sum_{x \in \mathcal{A}} \Pr[X = x] \log_2 \left(\frac{1}{\Pr[X = x]} \right).$$

2.1   The CTR-ACPKM Internal Re-keying Mode

CryptoPro Key Meshing (CPKM) is an internal re-keying mode introduced in [PLK06]. However, several issues with performance and security led to an improved version of CPKM, called ACPKM introduced in [Smy19]. RFC 8645 [Smy19] defines two kinds of internal re-keying mechanisms, namely **ACPKM** and **ACPKM-Master**, where ACPKM stands for *Advanced Cryptographic Prolongation of Key Material.* The ACPKM mode does not use a master key, and the ACPKM-Master uses a master key during the key transformation. In our discussion, we focus on the Counter mode of encryption with ACPKM transformation. For a detailed discussion on the ACPKM-Master, we refer the interested reader to [Smy19].

The CTR-ACPKM encryption mode takes a key $K \in \{0,1\}^\kappa$, a nonce $IV \in \{0,1\}^{\frac{n}{2}}$ and a message $M \in \{0,1\}^*$ and returns a ciphertext $C \in \{0,1\}^{|M|}$. We use the increment function $INC_{\frac{n}{2}}$ as used in [AAS18] to increment the counter. However, RFC 8645 documentation [Smy19] uses INC_c with $32 \le c \le \frac{3n}{4}$. In that case, $IV \in \{0,1\}^{n-c}$. Here let us recall the notion of a *section* from [Smy19].

– A set of consecutive blocks encrypted under the same key is called a *section.* We use s to denote the number of blocks in a *section.*

The CTR-ACPKM works as follows. At first the message M is divided into $w = \lceil \|M\|_n / s \rceil$ sections as

$$(M[1], M[2], ... M[s], M[s+1], ..., M[(w-1)s+r]) \xleftarrow{n} M,$$

where $\|M[i]\| = n$ for each $i \in \{1, ..., (w-1)s+(r-1)\}$ and $1 \le \|M[(w-1)s+r]\| \le n$ for some $r \le s$. So the α-th block in the β-th section is $M[\beta s + \alpha]$ for all $\beta \in [0, w-1]$ and $\alpha \in [1, s]$. In this mechanism the 0-th section of each message is processed by the CTR mode of operation with the initial counter $IV\|0^{\frac{n}{2}}$ and $K_0 = K$ where K is the input key to the process. The β-th section is encrypted by the CTR mode with the section key $K_{\beta-1}$ without initializing the counter to 0, i.e., with the counter starting from βs. This section key is calculated using the ACPKM transformation as follows:

$$K_\beta = ACPKM(K_{\beta-1}) = MSB_\kappa(E_{K_{\beta-1}}(D_1)| \cdots |E_{K_{\beta-1}}(D_r))$$

where κ is the size of the key used in the block cipher E_K and $r = \lceil \kappa/n \rceil$. Thus we can view the ACPKM transformation as a function from $\{0,1\}^\kappa$ to itself. Moreover, $D_1, D_2, ..., D_r$ are pairwise different constants in $\{0,1\}^n$ such that the $\frac{n}{2}$-th bit (counting from the least significant bit) of each D_i is equal to 1. The plaintext length is at most $2^{\frac{n}{2}-1}$ blocks. Also, the section size in bits must be divisible by the block size.

We note that both ISO 10116 [ISO] and RFC 8645 [Smy19] suggest the use of D_i's which are a sequence of bytes whose most significant bits are set to 1: $80_x, 81_x, 82_x, 83_x, \ldots, FF_x$. This was done to ensure a complete domain separation between calls to E made during encryption and the key update. As we discussed in Section 5.1 this separation does not hold when a non-power of 2 block size is used.

3   Security Issues with the ACPKM Transformation

In this section we look at the security of multiple invocations of the ACPKM transformation. First we note that a cipher with a fixed plaintext and a random key behaves like a random function. This is a well known result and has, most notably, been used in the analysis of Hellman's time-memory trade-off attack [Hel80].

Another well-known fact is that the iterative application of a random function is expected to reduce the H_0-entropy of the image set [FO89]. We analyse how this loss of H_0-entropy impacts the security of the ACPKM construction and show how to use this to recover section keys.

We denote the ACPKM transformation with key and block size κ by $f : \{0,1\}^{\kappa} \to \{0,1\}^{\kappa}$. We can now construct the functional graph G_f corresponding to the function f. Let $G_f = (V_f, E_f)$ where $V_f = \{0,1\}^{\kappa}$ is the set of vertices and the set of edges, E_f, consists of all the ordered pairs of the form $(v, f(v))$ for every vertex $v \in \{0,1\}^{\kappa}$. Statistical analysis of a random function's functional graph reveals some information about the function's multiple invocations. One of the most important properties we study here is the H_0-entropy loss on multiple invocations of the ACPKM transformation. Notably, we are interested in the approximate size of the output set after, say, ν iterations of the ACPKM transformation over $\{0,1\}^{\kappa}$. We recall the definition of the ν-th iterate image point from [FO89]:

Definition 3 (ν-th iterate image point [FO89])**.** A vertex in the functional graph G_f is called an ν-th iterate image point if this is an image of the ν-th iterate f^{ν} of the function f. We denote the set of all the ν-th iterate image points as I^{ν}.

Flajolet and Odlyzko analyzed many properties of the functional graphs of random functions [FO89]. We recall one result from [FO89] to motivate our analysis in Theorem 1.

Theorem 1 ([FO89])**.** *If $f : D \to D$ be random function over a domain D of size $N = 2^{\kappa}$ with a functional graph G_f. Then as $N \to \infty$, the expected size of I^{ν} is*

$$E(|I^{\nu}|) = (1 - \tau_{\nu})N, \ \text{where } \tau_0 = 0, \tau_{\nu+1} = e^{-(1-\tau_{\nu})}.$$

The above theorem (Theorem 1) suggests that the number of ν-th iterate image points decreases, i.e., there is H_0-entropy loss as H_0 is the codomain size. Hong *et al.* used this idea to find collisions in the state of the MICKEY stream cipher [HK05]. Moreover, based on their experiments, Hong *et al.* proposed the following conjecture: As ν goes to infinity, $(1 - \tau_{\nu})$ goes to $2^{1-\log_2(\nu)}$. This conjecture was proven to be true for $\nu \leq 2^{\frac{\kappa}{2}}$ by Bao *et al.* [BGW18]. To conclude, in the ACPKM construction with key size κ, the H_0 entropy of the key-space after ν iterations is approximately $\kappa + 1 - \log_2(\nu)$ where $\nu \leq 2^{\frac{\kappa}{2}}$.

The above analysis holds for any key and block size. However, as is evident in the case of GOST, it may be that $k > n$. Suppose that the size of the key, κ is rn where n is the size of the block cipher. Then the ACPKM method performs the key transformation as

$$K_j = E_{K_{j-1}}(D_1)||E_{K_{j-1}}(D_2)||\ldots||E_{K_{j-1}}(D_r).$$

These $D_1, D_2, \ldots, D_r$ are pairwise different constants. Then a key is valid only if all the parts are different. So there are at most $2^n(2^n - 1)\ldots(2^n - r + 1) = (2^n)_r$ valid keys in each section from the second section onward. In this case, the entropy of the key-space in the s-th section reduces more than in the previous case. From the Theorem 1 we can obtain the entropy in the s-th section is reduced to $(2^n)_r 2^{1-\log_2(s)}$.

4   Attack Based on Entropy Loss

Next we look at how we can use the reduction in entropy to improve exhaustive search of later section keys. Since the entropy drops with each iteration, we need to try about $2^{\kappa+1-\log_2 \nu}$ keys for the ν-th section. For example, consider the following scenario with a key size of $\kappa = 128$ bits, a block size of $n = 128$ bits and a section size of 1024 bits (i.e., 8 blocks of 128 bits). To find the key for the 2^{64}-th section, we need to try, as Theorem 1 suggests, about 2^{65} keys.

While there are indeed only 2^{65} possible keys, we do not get a simple list of them. For example, if we choose an arbitrary key, it is, with high probability, not a valid key for this section. Thus, to make use of the entropy loss of the function we need an efficient way to enumerate all the valid keys for the ν-th section.

The first way to enumerate all ν-th section keys is by using a naive precomputation approach: Take all possible 2^{κ} keys and update them $\nu - 1$ times. Store the resulting

values sorted by their frequency in a dictionary D_ν. Once the sorted dictionary D_ν is prepared, we can use it to find keys for the ν-th section. This approach takes $\nu 2^\kappa$ time and requires about $2^{\kappa+1-\log_2 \nu}\kappa$ memory.

The second approach is an immediate optimization of the naive approach. We can use $O(E(|\mathsf{l}^1|)\kappa)$ memory to reduce the time complexity of the preprocessing to $\sum_{i=0}^{\nu-1} E(|\mathsf{l}^i|)$ updates.[2] To do so we use the set of keys that were valid keys for the $(i-1)$-th section to compute the keys that are valid for the i-th section. Naturally, when $i = 0$ we use the full key space.

In both cases we need to build a dictionary D_ν of keys which are valid for the ν-th section only once. This means that we can amortize the cost of its construction cost over many applications of the brute-force phase (like in Hellman's time memory trade-off attacks [Hel80]). To conclude, given D_ν, containing all valid keys for the ν-th section (and their frequency), one can find the keys of the ν-th section by an exhaustive search. Next we look at an improved exhaustive search algorithm to find the ν-th section key.

4.1   Improved Exhaustive Search

We now discuss improving the exhaustive search for the ν-th section keys. The main idea is that all keys in the functional graph of f, which have ν predecessors, are valid section keys (even if they are in a cycle). Thus by exploring the functional graph of f, we can find a large number of valid section keys for the ν-th section.

Using the above fact, we present Algorithm 1 to find valid section keys for the ν-th section. Algorithm 1 takes a random key K as an input and outputs a list $\mathbb{L}$ of candidate keys that are valid for the ν-th section. As long as we do not detect a cycle, we update the key K using ACPKM. We note that detecting a cycle is easy, as we maintain a list $\mathbb{L}$ of keys we encountered in the traversal of the graph. After ν calls to update, we start recording the keys we encounter. Once we identify a recorded key, we know we are in a cycle and stop. We note that even if we entered the cycle *before* we start recording keys – there is nothing wrong, as we need to enumerate the entire cycle to find all the candidate ν-section keys (as each value in the cycle has ν predecessors for sure). One can slightly optimize our algorithm by detecting if we entered a cycle *before* we start recording the keys (and then traverse the cycle once more to record it). Note that if it updates the key ν times without detecting a cycle, all subsequent updates produce valid section keys for the ν-th section which are stored in $\mathbb{L}$.

If the size of the key is κ, we can expect, by the birthday paradox, to get a cycle after $2^{\kappa/2}$ updates. Once we get a list $\mathbb{L}$, we can choose another key outside of this list and start the algorithm again.

4.2   The H_1-Entropy of the ACPKM Transformation

Previously we discussed the H_0-entropy loss which is based on the cardinality of the section keys. We now focus on the H_1-entropy or the Shannon entropy of the section keys. For any key K we define the following set

$$P_\mathsf{K}^\nu = \{k \in \{0,1\}^\kappa : f^\nu(k) = \mathsf{K}\},$$

where f is the ACPKM transformation. As we can see, P_K^ν is the set of master-keys that after ν sections can reach the section key K. Let $\mathrm{Pr}_\nu(\mathsf{K})$ be the probability that a key K is a valid key for the ν-th section. We can define this probability as:

$$\mathrm{Pr}_\nu(\mathsf{K}) = \frac{|P_\mathsf{K}^\nu|}{2^\kappa}.$$

---

[2]The approximate value of $\sum_{i=0}^{\nu-1} E(|\mathsf{l}^i|)$ can be computed as $\sum_{i=0}^{\nu-1} E(|\mathsf{l}^i|) = 2^\kappa + \sum_{i=1}^{\nu-1} E(|\mathsf{l}^i|) = 2^\kappa + \sum_{i=1}^{\nu-1} 2^{\kappa+1}/i \approx 2^{\kappa+1}(1 + \log(\nu-1))$.

Algorithm 1 Algorithm to find section keys for the ν-th section

---

Require: A key K
Ensure: A list $\mathbb{L}$ of keys
  1: $\mathbb{L} = \emptyset$;
  2: **if** $\nu == 0$ **then**
  3:     $\mathbb{L} = \mathbb{L} \cup \{K\}$;
  4:     **return** $\mathbb{L}$;
  5: $Counter = 0$;
  6: **while** True **do**
  7:     $K = ACPKM(K)$;
  8:     **if** $K \in \mathbb{L}$ **then** $\triangleright$ Cycle Detected
  9:         **return** $\mathbb{L}$;
 10:     $Counter = Counter + 1$;
 11:     **if** $Counter \geq \nu$ **then**
 12:         $\mathbb{L} = \mathbb{L} \cup \{K\}$;

Table 1: H_0 and H_1-entropy for 32-bit truncated AES-128 (key size is 32 bits).

		AES: Key Size = 32, Block Size = 32			
Section	H_0	H_1	$\log_2(\kappa) - H_0$	$\log_2(\kappa) - H_1$	$H_1 - H_0$
1	31.338262	31.172745	0.661738	0.827255	-0.165517
2	30.906223	30.654303	1.093777	1.345697	-0.251920
4	30.319969	29.974669	1.680031	2.025331	-0.345300
8	29.596806	29.167126	2.403194	2.832874	-0.429680
16	28.769438	28.274307	3.230562	3.725693	-0.495131
32	27.871677	27.331245	4.128323	4.668755	-0.540432
64	26.930389	26.361420	5.069611	5.638580	-0.568969
128	25.963864	25.377889	6.036136	6.622111	-0.585975
256	24.982210	24.386729	7.017790	7.613271	-0.595481

Based on this definition, we can compute the Shannon entropy for the valid keys of the ν-th section as

$$H_1(I^\nu) = \sum_{K \in I^\nu} \Pr_\nu(K) \log\left(\frac{1}{\Pr_\nu(K)}\right)$$

The loss of H_1-entropy suggests that the distribution of the master-keys is not uniform across the valid section keys. We ran an experiment to study the H_1-entropy loss for truncated versions of AES-128 [DR02] and Simon [BSS+15]. The details of these ciphers are given in Appendix A. The results of the experiments are given in Table 1 and Table 2. We can observe that the entropy loss for the H_1-entropy is higher than H_0-entropy.

4.3   Attack Motivated by H_1-entropy Loss

We now propose a section key recovery attack motivated by the H_1-entropy loss. The loss of the H_1-entropy on key update suggests one crucial issue: the master keys are not uniformly distributed over the section keys. This fact suggests that not only one can cover many master keys by a ν-th section key, but also that there are section keys which cover more master keys (and thus have a greater probability of being correct, compared to other keys which may be valid ν-th section keys). We can thus look for the keys for which $|P_K^\nu|$ is larger, or enjoy the fact that by applying the ACPKM transform many times, they appear with a larger probability. Such keys are of interest, as testing them costs only a single trial

Table 2: H_0 and H_1-entropy for 16-bit truncated Simon (key size is 32 bits).

Section	H_0	H_1	$\log_2(\kappa) - H_0$	$\log_2(\kappa) - H_1$	$H_1 - H_0$
1	31.338258	31.172739	0.661742	0.827261	-0.165519
2	30.906216	30.654282	1.093784	1.345718	-0.251934
4	30.319954	29.974645	1.680046	2.025355	-0.345309
8	29.596808	29.167133	2.403192	2.832867	-0.429675
16	28.769431	28.274346	3.230569	3.725654	-0.495085
32	27.871809	27.331440	4.128191	4.668560	-0.540369
64	26.930347	26.360492	5.069653	5.639508	-0.569855
128	25.962123	25.375141	6.037877	6.624859	-0.586982
256	24.978940	24.382827	7.021060	7.617173	-0.596113

encryption, yet offers better success rate for the attack.

We designed Algorithm 2 based on this idea. We start with a random key, test it for the ν-th section. If the trial encryption succeeded, we are done. Otherwise, we apply the ACPKM transform repeatedly (like in Algorithm 1), and each time test the new computed value. Once a cycle is detected, we abort the experiment and pick a new random master key to start the process from. We continue to do so until either the correct ν-th section key is recovered, or if some number of experiments is reached.

Algorithm 2 Algorithm to find the ν-th section master key

1: **for** cnt $= 1$ to threshold$_{\kappa,\nu}$ **do**
2:     Pick a key K at random.
3:     Set $cycledetected \leftarrow 0$
4:     **while** $cycledetected == 0$ **do**
5:         Test the key K using trial encryption
6:         **if** The key K is the ν-th section key **then**
7:             Output K and terminate
8:         Compute K = ACPKM(K)
9:         **if** Cycle Detected **then**
10:            Set $cycledetected \leftarrow 1$
11: Declare "Failure"

First, we note that this could be viewed as the application of Algorithm 1 to various random keys, and testing the keys suggested by Algorithm 1. One could avoid testing the same key twice by storing a large bitmap of 2^κ bits suggesting which keys were already covered before. In this case, one could also pick the next key by taking one of the keys that were not encountered during the attack.

Second, we note that unless such a large table is stored, one could simply run Algorithm 2 until all keys were covered. However, we note that the first iteration of the loop is expected to cover only keys that were not encountered before, whereas starting from the second iteration, we expect some of the found keys to be already explored (this follows the fact that the functional graph of the ACPKM, if modeled as a random graph, is expected to have one large component with a large cycle of length $2^{\kappa/2}$). Hence, it is easy to see that each new iteration of Algorithm 2 offers diminishing returns.

At some point the cost of covering the list of proposed keys (which can be done in a memory-less fashion using any of the memory-less cycle finding algorithms), would be greater than the number of new keys offered by the new iteration. While we tried to find an exact formula to estimate the actual point in which the gain is negative (or close to zero), we could not. Hence, we suggest to perform tests or trails to determine the point,

Table 3: Results on Algorithm 2 ($\kappa = 32, \nu = 256$).

Iteration	Avg. recovered key	Avg. computation	Effectiveness	Total recovered key
1	$2^{24.40}$	$2^{16.42}$	$2^{7.98}$	$2^{24.40}$
2	$2^{23.71}$	$2^{16.46}$	$2^{7.34}$	$2^{25.09}$
3	$2^{23.12}$	$2^{16.38}$	$2^{6.74}$	$2^{25.42}$
4	$2^{22.64}$	$2^{16.46}$	$2^{6.18}$	$2^{25.61}$
8	$2^{21.98}$	$2^{16.53}$	$2^{5.44}$	$2^{25.99}$
16	$2^{21.19}$	$2^{16.38}$	$2^{4.80}$	$2^{26.50}$
32	$2^{20.78}$	$2^{16.53}$	$2^{4.24}$	$2^{26.99}$
64	$2^{20.35}$	$2^{16.41}$	$2^{3.93}$	$2^{27.49}$
128	$2^{19.76}$	$2^{16.38}$	$2^{3.37}$	$2^{27.89}$
256	$2^{19.44}$	$2^{16.51}$	$2^{2.93}$	$2^{28.33}$
512	$2^{16.69}$	$2^{16.33}$	$2^{0.35}$	$2^{28.82}$

Table 4: Experimental verification of running Algorithm 2 with a single iteration ($\kappa = 32$).

Section(ν)	Avg. recovered key	Avg. computation	Effectiveness
16	$2^{20.49}$	$2^{16.49}$	$2^{3.99}$
32	$2^{21.49}$	$2^{16.46}$	$2^{5.03}$
64	$2^{22.51}$	$2^{16.53}$	$2^{5.97}$
128	$2^{23.39}$	$2^{16.41}$	$2^{6.99}$
256	$2^{24.37}$	$2^{16.39}$	$2^{7.99}$
512	$2^{25.40}$	$2^{16.42}$	$2^{8.99}$

threshold$_{\kappa,\nu}$ that for a κ-bit key in the ν-th section, the gain of this is approach (over a random guess) is positive.

We ran experiments to verify our attack on the 32-bit truncated version of AES-128 [DR02]. Table 3 gives for each iteration of Algorithm 2 what is the computational effort to do the iteration, how many new keys were encountered, what is the effectiveness (average number of recovered master key per computation of the ACPKM transformation), as well as on how many keys were covered so far. We note that these experiments were done for the case of $\kappa = 32, \nu = 256$. We performed 100 experiments and report the average of the results. As can be seen in Table 3, for $\kappa = 32, \nu = 256$, the gain becomes very close to 1 after 512 iterations. The results of one iteration in Algorithm 2 for various section numbers is given in Table 4.

We conclude this section with a discussion on the success probability of the attack. The size of the set of recovered master keys is $\sum_{\mathsf{K} \in \mathcal{K}^\nu} |P_{\mathsf{K}}^\nu|$ and consequently, the success probability of the attack is $\sum_{\mathsf{K} \in \mathcal{K}^\nu} |P_{\mathsf{K}}^\nu|/2^\kappa$. To give a lower bound on the success probability we need to bound the size of P_{K}^ν. To find expected number of ν-th preimage, we recall the following Theorem 2 from [BWGG17].

Theorem 2. *Let f be a random function from $\{0,1\}^\kappa$ to itself and ν be a positive integer. Let y be a random image of f^ν. Then the expected size of the set P_y^ν is*

$$E(|P_y^\nu|) \geq \nu.$$

Using Theorem 2 we can get the following lower bound

$$E(|\cup_{\mathsf{K}\in\mathcal{K}^\nu} P_\mathsf{K}^\nu|) = E(\sum_{\mathsf{K}\in\mathcal{K}^\nu} |P_\mathsf{K}^\nu|), \text{ since } P_\mathsf{K}^\nu \cap P_{\mathsf{K}'}^\nu = \emptyset \text{ for } \mathsf{K} \neq \mathsf{K}'$$

$$= \sum_{\mathsf{K}\in\mathcal{K}^\nu} E(|P_\mathsf{K}^\nu|)$$

$$\geq \sum_{\mathsf{K}\in\mathcal{K}^\nu} \nu = |\mathcal{K}^\nu|\nu.$$

As an example, consider the case that we choose the threshold to be 1, namely, Algorithm 2 performs only one iteration of the attack. One can see that of a section ν in the range $2^{\kappa/4} \leq \nu < 2^{\kappa/2}$, we expect to cover $2^{3\kappa/4}$ master-keys (this was also supported by our experiments, suggested in Table 3 and Table 4). In other words, the first iteration of Algorithm 2 alone, suggests an attack whose time complexity is about $2^{\kappa/2}$ and its success rate is $2^{-\kappa/4}$.

5 ACPKM is not Misuse Resistant

While the previous attacks are inherent to the ACPKM mode (as well as the CPKM mode), these modes can also be wrongly instantiated in a way which further hinders the security. This increases the burden on the implementer, as a wrong implementation may have significant issues leading to a complete lack of security. As we show later, if the block size is not a multiple of 8 bits (as may happen in the case of Format Preserving Encryption), the use of the constants proposed in the RFC or the ISO standard may cause an immediate trivial attack on the scheme. Furthermore, when the underlining block cipher has a high probability differential or related-key differential (either as a design error or as a backdoor), the entropy of the new section keys significantly drops, and one should expect the previous attacks to aggravate. These issues suggest that ACPKM may be less suitable for deployment than other modes.

5.1 Special Block Sizes

Both ISO 10116 and RFC 8645 offer a concrete suggestion for the constants D_i, which are required to update the key in ACPKM. The constants should be pairwise different for any block size n and key size k and the most significant bit of every byte should be 1. The latter condition is important to prevent the trivial collision in the inputs to the block cipher in the cases of message processing and key updating: if somehow this collision occurs in some section, one can recover the next section key and consequently all the onward section keys. Specifically, consider the case with key size and block size of n.[3] If the input to the $s\kappa + \alpha$-th block (i.e., $\mathsf{IV}\|(s\kappa + \alpha)$) is equal with the constant D, then

$$E_k(\mathsf{IV}\|(s\kappa + \alpha)) = M[s\kappa + \alpha] \oplus C[s\kappa + \alpha]$$

is the next section key. This vulnerability can be prevented by choosing D such that the counter update can never be D.

We now turn our attention to the case where the block size is not a multiple of 8 (e.g., in format-preserving encryption scheme). Suppose that the block size is $n = 66$ and the initial counter is $\mathsf{IV}\|0^{33}$. If the adversary can set the IV then he can set the IV to the first 33 bits of D. For example, let us consider the example of D given in RFC 8645 [Smy19]. The 33-rd to 65-th bits of D are $00001001\ldots0111$. We can see that the 33-rd bit is not 1,

---

[3]When the key size is larger than n, the following reveals n-bit of information about the next section key.

so it is possible to reach this D within a valid number of sections. For the constant given in RFC 8645, this requires $303438365 \approx 2^{28}$ blocks to be encrypted (i.e., updates in the counter) which is less than 2^{32}. According to RFC 8645 the maximum message size can be 2^{c-1} blocks where $32 \leq c \leq (3/4)n$. So the above attack is practical. Once this key is recovered (in a known-plaintext attack), all the keys can also be recovered easily.

In other words, ACPKM (as well as CPKM) should not be instantiated with ciphers whose block sizes are not a multiple of 8 bits, if they are used with the constants suggested in RFC 8645 and in ISO 10116. Hence, we urge the standardization bodies to explicitly disallow this combination (though as we later argue, it may be better to remove these modes from the standards altogether).

5.2   A Related-key Distinguisher on CTR-ACPKM

This section proposes a related-key distinguisher on the CTR-ACPKM mode. The related-key attack model was independently suggested by Biham [Bih94] and Knudesn [Knu92]. In the common interpretation of the RK model, the adversary may choose any reasonable relation between the keys, where the exact definition of "reasonable" is left to the cryptanalyst (see for example the discussion of permissible relations in [BK03]). We now propose an attack based on the following key relation: let K and K' be two keys such that $K' = \mathsf{ACPKM}(K)$.[4]

Let us consider two CTR-ACPKM instances with section size s and s' such that $2s > s' > s$. To find a related-key distinguisher, we consider two master keys K and K' for the CTR-ACPKM with section size s and s', respectively, such that $K' = \mathsf{ACPKM}(K) = K^1$. Thus, the master key K' matches with the first section key of the master key K.

Now we choose a nonce IV and two random message $\mathsf{M}_1, \mathsf{M}_2$ for the two instances such that the number of blocks in the messages is at least s' blocks. Suppose that we query $(\mathsf{M}_1, \mathsf{IV})$ to the CTR-ACPKM instance with section size s and obtain the ciphertext C_1. Similarly, we query $(\mathsf{M}_2, \mathsf{IV})$ to the CTR-ACPKM instance with section size s' and obtain the ciphertext C_2. Now as we query with the same nonce, in this settings, we get the following relation

$$E_{\mathsf{K}'}(\mathsf{INC}_{\frac{n}{2}}^{s+1}(\mathsf{IV}\|0^{\frac{n}{2}})) = E_{\mathsf{K}_1}(\mathsf{INC}_{\frac{n}{2}}^{s+1}(\mathsf{IV}\|0^{\frac{n}{2}})).$$

This distinguishing property is easily detectable as we have the following relation on the $(s+1)$-th block:

$$C_2[s+1] \oplus M_2[s+1] = E_{\mathsf{K}'}(\mathsf{INC}_{\frac{n}{2}}^{s+1}(\mathsf{IV}\|0^{\frac{n}{2}})) = E_{\mathsf{K}_1}(\mathsf{INC}_{\frac{n}{2}}^{s+1}(\mathsf{IV}\|0^{\frac{n}{2}})) = C_1[s+1] \oplus M_1[s+1]$$

$$\implies C_1[s+1] \oplus C_2[s+1] = M_1[s+1] \oplus M_2[s+1].$$

5.3   Weak Block Ciphers

A high probability differential (whether it exists in the block cipher by accident or as a backdoor) is far from a good thing from a security perspective. However, for the case of ACPKM, the existence of a differential or even a related-key differential is significantly worse. In some cases such a (related-key) differential may significantly hinder the security of ACPKM in real life; especially when the constants D_i's are chosen to exploit the (related-key) differential property. We discuss a few attacks based on (related-key) differential and linear property of the underlying cipher in the following. The summary of our attacks is shown in Table 5.

---

[4]We note that the ACPKM transform is not invertible, as we suggested before. Most definitions of "permissible" relations assume that the key relation is invertible (to avoid trivial attacks or a huge entropy loss). Indeed, we need to consider a non-invertible transform, but to the best of our knowledge the ACPKM transformation does not allow a trivial key recovery attack (unlike the relations which were "disallowed") nor (in most of the cases) offer a huge entropy loss in the key after one application of ACPKM. Of course, if the block cipher is really weak or the entropy loss is huge, CTR-ACPKM has bigger issues than the proposed related-key attacks.

Table 5: Summary of attacks on ACPKM instantiated with a weak block cipher.

Differential Property			
Source	Issue	Complexity(Time)	Section
$\Delta_{\mathsf{X}} \xrightarrow{p} \Delta_{\mathsf{Y}}$	Section key-recovery	$2^n/p$	5.3.1
$\Delta_{\mathsf{X}_1} \xrightarrow{p_1} \Delta_{\mathsf{Y}_1}$ $\Delta_{\mathsf{X}_2} \xrightarrow{p_2} \Delta_{\mathsf{Y}_2}$	Section key-recovery	$2^n/\gamma, \gamma = \max\{p_1 p_2, p_1^2, p_2^2\}$	5.3.1
Related-key Differential Property			
Source	Issue	Complexity(Time)	Section
$0 \xrightarrow[\Delta_K]{p} 0$	Key entropy loss	–	5.3.2
Linear Property			
Source	Issue	Complexity(Time)	Section
$\lambda_I \xrightarrow{\epsilon} \lambda_O$	Key entropy loss	–	5.3.3

5.3.1 Differential Property

Suppose that the underlying block cipher E has a differential property such that the plaintext difference Δ_{X} propagates to the ciphertext difference Δ_{Y} with probability p. Consider the case of $k = 2n$, i.e., the key size is twice the block size. So if the ACPKM transformation is

$$\mathsf{K}_j = E_{\mathsf{K}_{j-1}}(D_1)||E_{\mathsf{K}_{j-1}}(D_2)$$

and $D_1 \oplus D_2 = \Delta_{\mathsf{X}}$ then with probability p, we get that the next section key is

$$\mathsf{K}_j = \mathsf{K}_{j,1}||\mathsf{K}_{j,2} = \mathsf{K}_{j,1}||\mathsf{K}_{j,1} \oplus \Delta_Y.$$

This property holds for any section key from the second section onwards. Using this differential, the attacker expects such an output difference by observing $O(1/p)$ sections and finding the instance for which the differential was satisfied in time $O(2^n/p)$. In other words it is trivial to try in all sections (starting from 2nd section) the keys $(\mathsf{K}, \mathsf{K} \oplus \Delta_Y)$. After one such a pair is found the adversary can determine all the future section keys. In fact, this attack offers gain for any key size κ with $\kappa > n + \log_2(1/p)$.

For the case of $\kappa \geq 4n$, we get a better attack. Suppose that the key update is

$$\mathsf{K}_j = E_{\mathsf{K}_{j-1}}(D_1)||E_{\mathsf{K}_{j-1}}(D_2)||E_{\mathsf{K}_{j-1}}(D_3)||E_{\mathsf{K}_{j-1}}(D_4).$$

In this case, one can observe difference between any of the $\binom{4}{2}$ pairs from $\{D_1, D_2, D_3, D_4\}$. Suppose that the underlying block cipher E has a differential property that the plaintext difference Δ_{X_1} propagates to the ciphertext difference Δ_{Y_1} with probability p_1 and the plaintext difference Δ_{X_2} propagates to the ciphertext difference Δ_{Y_2} with probability p_2. If $D_1 \oplus D_2 = \Delta_{\mathsf{X}_1}$ and $D_3 \oplus D_4 = \Delta_{\mathsf{X}_2}$ then with probability $p_1 p_2$, the section key $\mathsf{K}_j = \mathsf{K}_{j,1}||\mathsf{K}_{j,2}||\mathsf{K}_{j,3}||\mathsf{K}_{j,4} = \mathsf{K}_{j,1}||\mathsf{K}_{j,1} \oplus \Delta_{\mathsf{Y}_1}||\mathsf{K}_{j,3}||\mathsf{K}_{j,3} \oplus \Delta_{\mathsf{Y}_2}$. Furthermore, picking $D_1 \oplus D_2 = \Delta_{\mathsf{X}_1}$, $D_3 \oplus D_4 = \Delta_{\mathsf{X}_1}$ and $D_1 \oplus D_3 = \Delta_{\mathsf{X}_2}$ (which implies $D_2 \oplus D_4 = \Delta_{\mathsf{X}_2}$ as well) offers multiple "possibilities" for using the differentials. For example, with probability p_1^2 the resulting updated key is $\mathsf{K}_{j,1}||\mathsf{K}_{j,1} \oplus \Delta_{\mathsf{Y}_1}||\mathsf{K}_{j,3}||\mathsf{K}_{j,3} \oplus \Delta_{\mathsf{Y}_1}$, and with probability p_2^2 it is $\mathsf{K}_{j,1}||\mathsf{K}_{j,2}||\mathsf{K}_{j,1} \oplus \Delta_{\mathsf{Y}_2}||\mathsf{K}_{j,2} \oplus \Delta_{\mathsf{Y}_2}$. We note that, in RFC 8645 [Smy19] the suggested values for D_i have the following equality: $D_1 \oplus D_2 = D_3 \oplus D_4$, $D_1 \oplus D_3 = D_2 \oplus D_4$ and $D_1 \oplus D_4 = D_2 \oplus D_3$. Thus, if there is a differential property such that the difference of

one pair of the constants is equal to the input difference, then other pair also satisfies that. In that case, with probability p^2, the section key is $\mathsf{K}_j = \mathsf{K}_{j,1}||\mathsf{K}_{j,2}||\mathsf{K}_{j,3}||\mathsf{K}_{j,4} = \mathsf{K}_{j,1}||\mathsf{K}_{j,1} \oplus \Delta_\mathsf{Y}||\mathsf{K}_{j,3}||\mathsf{K}_{j,3} \oplus \Delta_\mathsf{Y}$. To reduce the chances of misuse, we suggest that the constants $D_i's$ should not have such relations.

A differential weakness of the underlying cipher can also be used differently. An adversary can choose a suitable nonce such that for some block, the counter becomes $D_1 \oplus \Delta_\mathsf{X}$. Suppose this counter occurs in the ν-th section, i.e., the $\nu s + \alpha$-th input to the block cipher is $D_1 \oplus \Delta_\mathsf{X}$ for some $\alpha \in [1, s]$. Then the $(\nu + 1)$-th section key is $\Delta_Y \oplus (\mathsf{M}[\nu s + \alpha] \oplus \mathsf{C}[\nu s + \alpha])$ with probability p. If one guesses this key with high probability, all the onward keys are known.

5.3.2 Related-key Differential Property

In addition to the above issues with regular differential, related-key differential may offer another attack strategy. Assume that the input difference 0 leads to an output difference 0 under some input key difference Δ_K with probability p. In this case, let us consider two different keys K and K' with $K \oplus K' = \Delta_K$. Observe that during updates, the ACPKM transformation uses the same constant input to the underlying block cipher, i.e., the input difference is 0. So both the updates produce the same keys for the next round with probability p. This definitely reduces the entropy of the keys for the next round.

We have experimented with our-made variants of SIMON [BSS$^+$15] and of TEA [WN94] block ciphers, where we have taken both block size of 16 bits and key size of 32 bits (see Appendix A for details of these ciphers). For a random function the key entropy drops by about 0.66 bits in the first update. However, due to the related-key differential properties of TEA,[5] we observe a drop of almost 2.34 bits in the key entropy already in the first update. For SIMON, we observed a drop of 0.67 bits in the H_0-entropy after the first update.

5.3.3 Linear Property

High probability linear characteristics also effectively reduces the key entropy. Consider the key-update of ACPKM

$$\mathsf{K}_j = E_{\mathsf{K}_{j-1}}(D_1)||E_{\mathsf{K}_{j-1}}(D_2).$$

Suppose that there are linear characteristics $(\lambda_I \to \lambda_O)$ with bias ϵ, i.e., we have

$$\lambda_I \cdot D_1 \oplus \lambda_O \cdot E_{\mathsf{K}_{j-1}}(D_1) = \lambda_K \cdot \mathsf{K}_{j-1} \tag{1}$$
$$\lambda_I \cdot D_2 \oplus \lambda_O \cdot E_{\mathsf{K}_{j-1}}(D_2) = \lambda_K \cdot \mathsf{K}_{j-1} \tag{2}$$

where λ_K is the key mask and both equations holds with probability $\epsilon + \frac{1}{2}$. Using the pilling-up lemma:

$$\lambda_I \cdot (D_1 \oplus D_2) \oplus \lambda_O \cdot (E_{\mathsf{K}_{j-1}}(D_1) \oplus \cdot E_{\mathsf{K}_{j-1}}(D_2)) = 0$$

holds with probability $2\epsilon^2 + \frac{1}{2}$. Here $(D_1 \oplus D_2)$ is known to the adversary, so information about the j-th section key is revealed. Moreover , the existence of such a relation reduces the key entropy.

---

[5]We remind the reader that TEA has 3 related key differential properties of the form $0 \xrightarrow{\frac{1}{\Delta_K}} 0$ These related-key properties are discussed in Appendix A.2.

6 ACPKM in the Multi-User Setting

We now discuss the multi-user security of the ACPKM contstruction, taking into consideration the H_0-entropy loss. One of the most critical implications of the H_0-entropy loss is that the probability of a collision between two or more user keys increases. We take u to be the number of users and we let $\nu \leq 2^{\frac{\kappa}{2}}$ be the section number, then, due to the H_0-entropy loss and by the birthday paradox, we can observe that if

$$u > 2^{\frac{\kappa+1-\log_2(\nu)}{2}} = \sqrt{\frac{2^{\kappa+1}}{\nu}},$$

then, with high probability, two or more user keys collide after ν iterations of ACPKM. This vulnerability in the ACPKM construction highlights the impact of the entropy loss in real life. For example, if two different user keys collide in section ν, then all the section keys of those two users after the ν-th section are the same. Thus, if two or more user keys collide in some section ν, then the counter mode in CTR-ACPKM provides the same keystream for all the subsequent sections, provided that the users use the same IV.

6.1 Distinguishing Attack Using Key Collision

We propose a distinguishing attack on CTR-ACPKM, using the above key collision. Assume that there are u users, where the adversary makes a single query to each user asking for the encryptions of w_i sections, with $i \in \{1, 2, ..., u\}$. We denote the set of all encryptions by $w = \min\{w_1, ..., w_u\}$. In this setting, we construct a deterministic distinguisher D for a distinguishing game to distinguish CTR-ACPKM construction from the *ideal world* as follows:

ATTACK ALGORITHM. First, the adversary picks a random nonce, say IV and messages M^i for the i-th user with $\|\mathsf{M}^i\|_n = 2^{sw}$ for all $i \in \{1, 2, ..., u\}$. So each M^i has w sections numbered $0, ..., w-1$. Recall that s is the length of each section in n-bit blocks and κ is the key size. Then the adversary advances as follows:

1. Choose a suitable section ν with $\nu \leq 2^{\frac{\kappa}{2}}$ and $u^2\nu \geq 2^{\kappa+2}$.

2. For $i = 1, ..., u$, asks for the encryption of $(i, \mathsf{IV}, \mathsf{M}^i)$ form the i-th user under key K^i and obtained C^i.

3. If there are i and j such that $\mathsf{C}^i[\beta s + \alpha] + \mathsf{M}^i[\beta s + \alpha] = \mathsf{C}^j[\beta s + \alpha] + \mathsf{M}^j[\beta s + \alpha]$ for all $\beta \in [\nu, w-1]$ and $\alpha \in [1, s]$ then output 1, else output 0.

The distinguishing advantage of D is more than $1/2$ when the number of users is at least $2^{\kappa/2+1}/\sqrt{\nu}$ and D makes at least one query through each users. The detailed analysis of the algorithm is given in Appendix B.

   We now look at the practical security implications of the key collision between multiple users. Consider a system with a key size of $\kappa = 128$ bits, a block size of $n = 128$ bits, and a section size of $s = 2^3$ blocks. Now consider a message M consisting of 2^m blocks. From this we can see that the maximum number of sections is 2^{m-3}. Note that due to Theorem 1 we cannot ask for more than $\nu \leq 2^{64}$ sections per user. However, since we are in the multi-user setting we can still generate a collision. As we have seen before we can generate a key collision at the ν-th section using u users where

$$u > 2^{\frac{128+1-\log_2(\nu)}{2}} = 2^{\frac{129-\log_2(\nu)}{2}}$$

with $\nu \leq \min\{2^{64}, 2^{m-3}\}$. Thus, in this example we need 2^{32} users to get a collision in the 2^{64}-th section key.

   Naturally we can increase the number of users to reduce the number of sections each user needs to encrypt. If in the above example we use 2^{36} users, the attacker only needs to request for 2^{56} sections to generate a collision in the section key.

6.2   Multi-user Key Recovery Attack

Next we discuss a section-key recovery attack in the multi-user setting. We use a similar idea to the H_0-entropy loss and birthday paradox as we discussed in the previous distinguishing attack. This idea is motivated by [Bih96]. Again, we consider a case with u users. We construct an adversary that makes a single query to each user. Let w_i be the number of sections in the i-th user query for all $i \in \{1, 2, ..., u\}$ and $w = \min\{w_1, ..., w_u\}$. We show that the adversary can recover the section key of a targeted section for at least one user. We present the result in the following Lemma 1.

Lemma 1. *Let $\nu \in [1, w]$ be a fixed section number and s be the section size in the n-bit blocks. There exists an adversary* A *that makes a single query corresponding to each of the u users, can recover the ν-th section key of at least one user on the* CTR-ACPKM *in the multi-user setting. The success probability of the attack is more than $1/2$ when the number of users is at least $2^{\kappa+1}/\nu\ell$ and* A *makes at least one query through each users, where κ is the key size and each query is with at least νs blocks.*

Proof. We construct a deterministic adversary A. Let us consider there are u users and A makes queries to the i-th user of the form $(i, \mathsf{IV}^i, \mathsf{M}^i)$.

ATTACK ALGORITHM. A randomly chooses a nonce IV and chooses message M^i for the i-th user for all $i \in \{1, 2, ..., u\}$. In that case, the adversary does the following:

1. Choose a suitable section ν with $\nu \leq 2^{\frac{\kappa}{2}}$.

2. For $i = 1, ..., u$, queries encryption of $(i, \mathsf{IV}, \mathsf{M}^i)$ for i-th user under key K_i and obtained C^i.

3. Compute the following set $\{\mathsf{X}_1, ..., \mathsf{X}_s\}$, where $\mathsf{X}_j = \mathsf{INC}_{\frac{n}{2}}^{s\kappa+j}(\mathsf{IV}\|0^{\frac{n}{2}})$ for $j \in [1, s]$.

4. Guess a set of ℓ keys $\{\mathsf{K}_1^g, \mathsf{K}_2^g, ..., \mathsf{K}_\ell^g\}$ which are valid for the ν-th section (to find valid keys, one can use Algorithm 1 as described in Section 4.1).

5. For all $j \in [1, \kappa]$, encrypt X_j with each guessed key and obtain $\mathsf{Y}_{p,j} = E_{\mathsf{K}_p^g}(\mathsf{X}_j)$ for all $p \in \{1, 2, ..., \ell\}$.

6. If there exists i and p such that

$$\mathsf{C}^i[\nu s + j] \oplus \mathsf{M}^i[\nu s + j] = \mathsf{Y}_{p,j} \forall j = 1, 2, ..., s$$

then output K_p^g as the i-th user key.

ANALYSIS OF THE ATTACK. The condition in step 6 holds if one user's ν-th section-key matches with one of the guessed key. We consider the indicator random variables

$$\mathcal{R}_{i,p} = \begin{cases} 1, & \text{if } \mathsf{K}_p^g = \mathsf{K}_\nu^i \\ 0, & \text{otherwise} \end{cases}$$

and $\mathcal{R} = \sum_{i,p} \Pr[\mathcal{R}_{j,k}]$. From Theorem 1 we can get the H_0-entropy of the ν-th section key is $H = 2^{\kappa+1-\log_2(\nu)}$. Thus we have $\Pr[\mathcal{R}_{i,p} = 1] = \frac{1}{2^H}$. From the pairwise independence of the variables $\mathcal{R}_{j,k}$ and using Chebyshev's inequality, we have

$$\Pr[\mathcal{R} \geq 1] \geq \left(1 - \frac{\ell u}{2^H}\right) \geq \left(1 - \frac{\ell u \nu}{2^{\kappa+1}}\right).$$

Finally, as we have the condition $u\nu\ell \geq 2^{\kappa+2}$ from the statement of the lemma,

$$\Pr[\mathcal{R} \geq 1] \geq \frac{1}{2}.$$

Table 6: These tables are showing the relation between the number of user and number of offline block cipher calls required to get a key-recovery advantage near $\frac{1}{2}$. We consider a key size of $\kappa = 128$ bits, a block size of $n = 128$ bits and section size $s = 1024$.

(a) $\log_2(\nu) = 64$		(b) $\log_2(\nu) = 48$		(c) $\log_2(\nu) = 32$	
$\log_2(u)$	$\log_2(\ell)$	$\log_2(u)$	$\log_2(\ell)$	$\log_2(u)$	$\log_2(\ell)$
20	45	20	61	20	77
25	40	25	56	25	72
30	35	30	51	30	67
35	30	35	46	35	62
40	25	40	41	40	57
45	20	45	36	45	52
50	15	50	31	50	47

Again, random collision can occurs that satisfies the condition on the step 6 of the attack without being the key match. Such cases are *false positive* cases and the attack fails in such cases. Let us consider the following indicator random variables

$$\mathcal{S}_{i,p} = \begin{cases} 1, & \text{if step 6 holds} \\ 0, & \text{otherwise} \end{cases}$$

and $\mathcal{S} = \sum_{i,p} \Pr[\mathcal{S}_{j,k}]$. Thus we get the expected number of *false positive* cases

$$\mathsf{E}(\mathcal{S}|\mathcal{R} = 0) = \sum_{i,p} \Pr[\mathcal{S}_{j,k} = 1|\mathcal{R} = 0] = \frac{\ell u}{((2^n)_s)} \leq \frac{\ell u}{2^{\kappa+d}}, \tag{3}$$

for some sufficiently large d such that $s = \left\lceil \frac{\kappa+d}{n-1} \right\rceil$. Thus, for sufficiently large d, the expected number of *false positive* cases are small enough.

Again, if one section key is known to the adversary, all the onward section keys are revealed.

We can observe that the total complexity of the above attack critically depends on the adversary's target section. For a fixed number of users, if the adversary targets a section towards the end, it needs less computation. In Table 6 we can see the number of primitive queries (ℓ) and the number of users (u) that are required to get a key-recovery advantage of $\approx \frac{1}{2}$ in a fixed section (ν).

The same idea can also be implemented in the single-user setting. In that case the adversary also needs to guess the IV and the required condition is

$$u\nu\ell \geq 2^{\kappa+m+2},$$

where m is the size of the nonce IV.

7 Discussion and Future Work

We have presented attacks based on the entropy reduction of the key-space in the ACPKM internal re-keying technique. Based on the H_0-entropy loss, we proposed an improved exhaustive search for the section keys. We also discussed a key collision attack, and a key-recovery attack in the multi-user setting due to the H_0-entropy loss. Finally, we discussed that the H_1-entropy loss is much more effective than H_0-entropy loss and based on this, we propose a novel master-key recovery attack.

In addition to the attacks based on entropy loss, we proposed attacks based on faulty or backdoored implementations of CTR-ACPKM. Furthermore, we show that a malicious designer may further harm the mode if a specific related-key differential property exists in the underlying block cipher. Finally, we propose a related-key distinguisher on the CTR-ACPKM mode which is independent of the underlying primitive.

One open direction of research is to consider the entropy reduction into the security proofs of [AAS18]. Note that using ACPKM without any change is acceptable if the key size is large (to begin with), the number of sections encrypted under one master key is not too large, and the delicate implementation issues are addressed. However, given our findings, we suggest that ISO and IETF will reconsider the use of ACPKM. Even if the use of ACPKM is not discouraged, appropriate warnings about the use cases should be clearly added to the respective standards.

Finally, we note that ACPKM (and CPKM) is suggested to use together with Russian standards such as GOST 28147-89 (Magam) and Kuznyechik. The design rationale of these ciphers was never revealed, and in the case of Kuznyechik multiple works suggested hidden design rationale [BPU16, Per19]. Given that these modes are extremely sensitive to differential-based backdoors in the design, we further suggest using these modes only with block ciphers whose design criteria are completely trustworthy, and security against differential cryptanalysis can be guaranteed (as much as possible).

References

[AAO+16] Liliya R. Ahmetzyanova, Evgeny K. Alekseev, Igor B. Oshkin, Stanislav V. Smyshlyaev, and Lolita A. Sonina. On the properties of the CTR encryption mode of the Magma and Kuznyechik block ciphers with re-keying method based on cryptopro key meshing. *IACR Cryptol. ePrint Arch.*, page 628, 2016.

[AAOS17] Liliya R. Ahmetzyanova, Evgeny K. Alekseev, Igor B. Oshkin, and Stanislav V. Smyshlyaev. Increasing the lifetime of symmetric keys for the GCM mode by internal re-keying. *IACR Cryptol. ePrint Arch.*, page 697, 2017.

[AAS18] Liliya R. Akhmetzyanova, Evgeny K. Alekseev, and Stanislav V. Smyshlyaev. Security bound for CTR-ACPKM internally re-keyed encryption mode. *IACR Cryptol. ePrint Arch.*, page 950, 2018.

[AASO20] Liliya R. Akhmetzyanova, Evgeny K. Alekseev, Stanislav Smyshlyaev, and Igor B. Oshkin. On internal re-keying. In *SSR*, volume 12529 of *Lecture Notes in Computer Science*, pages 23–45. Springer, 2020.

[AB00] Michel Abdalla and Mihir Bellare. Increasing the lifetime of a key: A comparative analysis of the security of re-keying techniques. In *Advances in Cryptology – Proceedings of ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 546–559. Springer, 2000.

[ANI+08] Alexandr Afanasiev, Nikolaj Nikishin, Boleslav Izotov, Elena Minaeva, Serguei Murugov, Igor Ustinov, Anatolij Erkin, Grigorij Chudov, and Serguei Leontiev. GOST 28147-89 Cipher Suites for Transport Layer Security (TLS). Internet-Draft draft-chudov-cryptopro-cptls-04, Internet Engineering Task Force, December 2008. Work in Progress.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology – Proceedings of EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000.

[BBT16]   Mihir Bellare, Daniel J. Bernstein, and Stefano Tessaro. Hash-function based prfs: AMAC and its multi-user security. In *Advances in Cryptology – Proceedings of EUROCRYPT (1)*, volume 9665 of *Lecture Notes in Computer Science*, pages 566–595. Springer, 2016.

[BGW18]   Zhenzhen Bao, Jian Guo, and Lei Wang. Functional graphs and their applications in generic attacks on iterated hash constructions. *IACR Trans. Symmetric Cryptol.*, 2018(1):201–253, 2018.

[BHT18]   Priyanka Bose, Viet Tung Hoang, and Stefano Tessaro. Revisiting AES-GCM-SIV: multi-user security, faster key derivation, and better bounds. In *Advances in Cryptology – Proceedings of EUROCRYPT (1)*, volume 10820 of *Lecture Notes in Computer Science*, pages 468–499. Springer, 2018.

[Bih94]   Eli Biham. New types of cryptanalytic attacks using related keys. *J. Cryptol.*, 7(4):229–246, 1994.

[Bih96]   E. Biham. How to forge DES-encrypted messages in 2^{28} steps. 1996.

[Bih02]   Eli Biham. How to decrypt or even substitute des-encrypted messages in 2^{28} steps. *Inf. Process. Lett.*, 84(3):117–124, 2002.

[BK03]   Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 491–506. Springer, 2003.

[BL16]   Karthikeyan Bhargavan and Gaëtan Leurent. On the practical (in-)security of 64-bit block ciphers: Collision attacks on http over tls and OpenVPN. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 456–467. Association for Computing Machinery, 2016.

[BMS05]   Alex Biryukov, Sourav Mukhopadhyay, and Palash Sarkar. Improved time-memory trade-offs with multiple data. In *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 110–127. Springer, 2005.

[BPU16]   Alex Biryukov, Léo Perrin, and Aleksei Udovenko. Reverse-engineering the s-box of streebog, kuznyechik and stribobr1. In *Advances in Cryptology – Proceedings of EUROCRYPT (1)*, volume 9665 of *Lecture Notes in Computer Science*, pages 372–402. Springer, 2016.

[BSS+15]   Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference*, DAC '15, New York, NY, USA, 2015. Association for Computing Machinery.

[BT16]   Mihir Bellare and Björn Tackmann. The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In *Advances in Cryptology – Proceedings of CRYPTO (1)*, volume 9814 of *Lecture Notes in Computer Science*, pages 247–276. Springer, 2016.

[BWGG17]   Zhenzhen Bao, Lei Wang, Jian Guo, and Dawu Gu. Functional graph revisited: Updates on (second) preimage attacks on hash combiners. In *Advances in Cryptology – Proceedings of CRYPTO (2)*, volume 10402 of *Lecture Notes in Computer Science*, pages 404–427. Springer, 2017.

[CMS11]   Sanjit Chatterjee, Alfred Menezes, and Palash Sarkar. Another look at tightness. In *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 293–319. Springer, 2011.

[DFH+16]   Stefan Dziembowski, Sebastian Faust, Gottfried Herold, Anthony Journault, Daniel Masny, and François-Xavier Standaert. Towards sound fresh re-keying with hard (physical) learning problems. In *Advances in Cryptology – Proceedings of CRYPTO (2)*, volume 9815 of *Lecture Notes in Computer Science*, pages 272–301. Springer, 2016.

[DL14]   Itai Dinur and Gaëtan Leurent. Improved generic attacks against hash-based MACs and HAIFA. In *Advances in Cryptology – Proceedings of CRYPTO (1)*, volume 8616 of *Lecture Notes in Computer Science*, pages 149–168. Springer, 2014.

[DR02]   Joan Daemen and Vincent Rijmen. AES and the wide trail design strategy. In *Advances in Cryptology – Proceedings of EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 108–109. Springer, 2002.

[FO89]   Philippe Flajolet and Andrew M. Odlyzko. Random mapping statistics. In *Advances in Cryptology – Proceedings of EUROCRYPT*, volume 434 of *Lecture Notes in Computer Science*, pages 329–354. Springer, 1989.

[FS09]   Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[Hel80]   Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory*, 26(4):401–406, 1980.

[HK05]   Jin Hong and Woo-Hwan Kim. Tmd-tradeoff and state entropy loss considerations of streamcipher MICKEY. In *Progress in Cryptology – INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 2005.

[HT17]   Viet Tung Hoang and Stefano Tessaro. The multi-user security of double encryption. In *Advances in Cryptology – Proceedings of EUROCRYPT (2)*, volume 10211 of *Lecture Notes in Computer Science*, pages 381–411, 2017.

[HTT18]   Viet Tung Hoang, Stefano Tessaro, and Aishwarya Thiruvengadam. The multi-user security of GCM, revisited: Tight bounds for nonce randomization. In *CCS*, pages 1429–1440. ACM, 2018.

[ISO]   ISO/IEC 10116:2017/Amd 1:2021, Information technology — Security techniques — Modes of operation for an n-bit block cipher, CTR-ACPKM mode of operation, 2021. Standard, International Organization for Standardization, Geneva, CH.

[Knu92]   Lars R. Knudsen. Cryptanalysis of LOKI91. In *AUSCRYPT*, volume 718 of *Lecture Notes in Computer Science*, pages 196–208. Springer, 1992.

[LC06]   Serguei Leontiev and Grigorij Chudov. Using the GOST 28147-89, GOST R 34.11-94, GOST R 34.10-94, and GOST R 34.10-2001 Algorithms with Cryptographic Message Syntax (CMS). RFC 4490, May 2006.

[LMP17]   Atul Luykx, Bart Mennink, and Kenneth G. Paterson. Analyzing multi-key security degradation. In *Advances in Cryptology – Proceedings of ASIACRYPT (2)*, volume 10625 of *Lecture Notes in Computer Science*, pages 575–605. Springer, 2017.

[LPW13] Gaëtan Leurent, Thomas Peyrin, and Lei Wang. New generic attacks against hash-based MACs. In *Advances in Cryptology – Proceedings of ASIACRYPT (2)*, volume 8270 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2013.

[ML15] Nicky Mouha and Atul Luykx. Multi-key security: The even-mansour construction revisited. In *Advances in Cryptology – Proceedings of CRYPTO (1)*, volume 9215 of *Lecture Notes in Computer Science*, pages 209–223. Springer, 2015.

[MPS20] Andrew Morgan, Rafael Pass, and Elaine Shi. On the adaptive security of MACs and PRFs. In *Advances in Cryptology – Proceedings of ASIACRYPT (1)*, volume 12491 of *Lecture Notes in Computer Science*, pages 724–753. Springer, 2020.

[Per19] Léo Perrin. Partitions in the s-box of streebog and kuznyechik. *IACR Trans. Symmetric Cryptol.*, 2019(1):302–329, 2019.

[PLK06] Vladimir Popov, Serguei Leontiev, and Igor Kurepkin. Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms. RFC 4357, January 2006.

[PvO96] Bart Preneel and Paul C. van Oorschot. On the security of two MAC algorithms. In *Advances in Cryptology – Proceedings of EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 19–32. Springer, 1996.

[PW20] Thomas Peyrin and Haoyang Wang. The MALICIOUS framework: Embedding backdoors into tweakable block ciphers. In *Advances in Cryptology – Proceedings of CRYPTO (3)*, volume 12172 of *Lecture Notes in Computer Science*, pages 249–278. Springer, 2020.

[Smy19] S. Smyshlyaev. Re-keying mechanisms for symmetric keys. RFC 8645, RFC Editor, August 2019.

[WN94] David J. Wheeler and Roger M. Needham. Tea, a tiny encryption algorithm. In *FSE*, volume 1008 of *Lecture Notes in Computer Science*, pages 363–366. Springer, 1994.

A Reduced Versions of the Ciphers Used in the Experiment

To provide experimental supports to our analysis in this paper, we instantiated CTR-ACPKM with toy versions of Simon [BSS+15], TEA [WN94] and AES [DR02]. We give a detailed configuration of our-made version the ciphers.

A.1 TEA16

TEA16 is the reduced version of the TEA [WN94] block cipher, reduced to a 16-bit block cipher with a 32-bit key master key, where the word size is 8 bits. TEA16 is a 32-rounds balanced Feistel cipher with a 32-bit master key K. TEA16's key schedule follows that of TEA [WN94]. At first the 32-bit master key parsed into four keys of size 8 bits as $K[3]||K[2]||K[1]||K[0] \overset{8}{\leftarrow} K$. Then, in the even rounds we use the 16-bit part $K[1]||K[0]$ and in the odd rounds we use the other 16-bit part $K[3]||K[2]$. The round function of TEA16 is applied to the block (L_r, R_r) as follows:

$$\begin{cases} R_{r+1} = L_r + (((R_r \lll_4) + K_r^2) \oplus (R_r + c) \oplus ((R_r \ggg_5) + K_r^1)) \\ L_{r+1} = R_r \end{cases}$$

for $r = 0 \cdots 32$ and $K_r^1 \| K_r^2$ is the r-th round key. $X \lll_i$ and $X \ggg_i$ denotes the i bit left shift and i bit right shift of X, respectively. The value c is initially 0 and incremented by a fixed constant `0x9E` at every two rounds. A schematic diagram of one cycle (two rounds) is given in Figure 1.
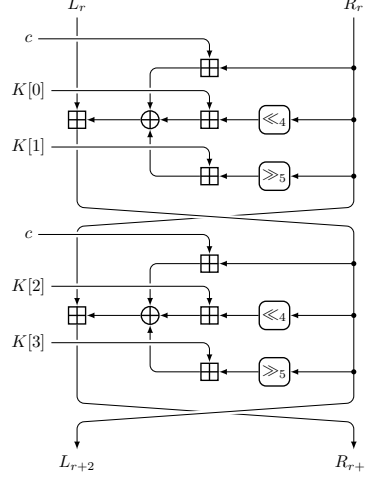
Figure 1: Round function of the TEA16.

A.2 Related Key Properties of TEA16

Here we recall three related-key differential property of the block cipher TEA16, that holds for zero input difference. Let us denote one cycle of TEA16 by F. The following iterative differential characteristics holds with probability 1:

$$F(X \| Y, K_1[3] \| K_1[2] \| K_1[1] \| K_1[0], c) \oplus F(X \| Y, K_2[3] \| K_2[2] \| K_2[1] \| K_2[0], c) = 0$$

where two keys $K_1[3] \| K_1[2] \| K_1[1] \| K_1[0]$ and $K_2[3] \| K_2[2] \| K_2[1] \| K_2[0]$ hold any of the following relations:

1. Simultaneously flipping the most significant bit of $K_1[2]$ and $K_1[3]$, i.e.,

$$\begin{cases} K_2[0] = K_1[0] \\ K_2[1] = K_1[1] \\ K_2[2] = K_1[2] + 2^7 \\ K_2[3] = K_1[3] + 2^7 \end{cases}$$

2. Simultaneously flipping the most significant bit of $K_1[0]$ and $K_1[1]$, i.e.,

$$\begin{cases} K_2[0] = K_1[0] + 2^7 \\ K_2[1] = K_1[1] + 2^7 \\ K_2[2] = K_1[2] \\ K_2[3] = K_1[3] \end{cases}$$

3. Simultaneously flipping the most significant bit of $K_1[0], K_1[1], K_1[2]$ and $K_1[3]$, i.e.,

$$\begin{cases} K_2[0] = K_1[0] + 2^7 \\ K_2[1] = K_1[1] + 2^7 \\ K_2[2] = K_1[2] + 2^7 \\ K_2[3] = K_1[3] + 2^7 \end{cases}$$

Thus, TEA has 3 related-key differential property that takes zero input difference to zero output difference under 3 possible key difference.

A.3 Simon16

Simon16 is a reduced version of Simon [BSS+15] with a 16-bit block and 32-bit master key. It is an Feistel structure where the round function follows ARX construction and has 32 Feistel rounds. One round of Simon16 applied to the block $L_r || R_r$ and produce $L_{r+1} || R_{r+1}$ as follows:

$$\begin{cases} L_{r+1} = R_r \oplus ((S^1(L_r) \wedge S^2(L_r)) \oplus S^3(L_r) \oplus K_r) \\ R_{r+1} = R_r \end{cases}$$

where $S^j(X)$ denotes the left circular shift of X by j positions and K_i denotes the round key. Thus, we have changes the left circular shift amounts to 1,2, and 3 instead of 8, 1 and 2 used in the original Simon [BSS+15]. A schematic diagram of one round is given in Figure 2.
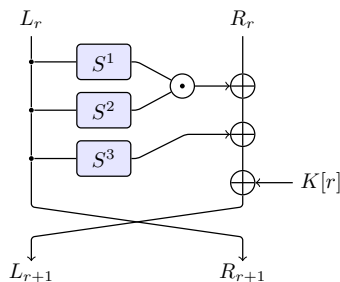
Figure 2: Round function of the Simon16.

The key schedule of Simon16 follows exactly the same key schedule as used in the original Simon [BSS+15]. At first the 32-bit master key K is parsed into four keys of size 8 bits to generate subkeys for the first 4 rounds, i.e., $K[3]\|K[2]\|K[1]\|K[0] \overset{8}{\leftarrow} K$. The rest of the 28 keys are generated as follows: for $i = 0, ..., 27$ we compute

$$K[i+4] = K[i] + (S^{-3}(K[i+3]) \oplus K[i+1]) \oplus S^{-1}((S^{-3}(K[i+3]) \oplus K[i+1])) \oplus Z_i \oplus 0xF3,$$

where $S^{-j}(X)$ denotes the right circular shift of X by j positions. Here Z_i is the i-th bit of a fixed bit sequence $Z = 00011001110000110101001000101111$.

We are not aware of any high probability related-key differential in Simon [BSS+15] and we expect none exist in Simon16.

A.4 AES16/AES32

AES16 (AES32) is the reduced version of AES [DR02] reduced to a 16-bit (32-bit) block cipher with 32-bit key. We take 32-bit key and plaintext, and add zero to the 12 most significant bytes to make it 128-bit key. Finally, the output of AES-128 is chopped to 16 bits (32 bits).

B Analysis of the Distinguishing Attack from Subsection 6.1

Assume that there are u users, where the adversary makes a single query to each user. Let w_i be the number of sections in the i-th user-query for all $i \in \{1, 2, ..., u\}$ and

$w = \min\{w_1, ..., w_u\}$. We show that this adversary can distinguish CTR-ACPKM from a random function. We present our main result of this section in the following Lemma 2.

Lemma 2. *Let $\nu \in [1, w]$ be a fixed section number and s is the section size in n-bit blocks. There exists an adversary* D *that makes a single query to each of the u users, can distinguish* CTR-ACPKM *construction from a random function in the multi-user setting. The distinguishing advantage of* D *is more than $1/2$ when the number of users is at least $2^{\kappa/2+1}/\sqrt{\nu}$ and* D *makes at least one query through each users, where κ is the key size and each query is with at least νs blocks.*

Proof. We construct a deterministic distingusher D on a distinguishing game to distinguish CTR-ACPKM construction from the *ideal world*. Let us consider there are u users and D makes queries to the i-th user of the form $(i, \mathsf{IV}^i, \mathsf{M}^i)$.

ATTACK ALGORITHM. First, the adversary picks a random nonce, say IV and messages M^i for the i-th user with $\|\mathsf{M}^i\|_n = 2^{sw}$ for all $i \in \{1, 2, ..., u\}$. Recall that s is the length of each section in n-bit blocks. So each M^i has w sections numbered $0, ..., w - 1$. Then the adversary does the following:

1. Choose a suitable section ν with $\nu \le 2^{\frac{\kappa}{2}}$ and $u^2\nu \ge 2^{\kappa+2}$.

2. For $i = 1, ..., u$, asks for the encryption of $(i, \mathsf{IV}, \mathsf{M}^i)$ form the i-th user under key K^i and obtained C^i.

3. If there are i and j such that $\mathsf{C}^i[\beta s + \alpha] + \mathsf{M}^i[\beta s + \alpha] = \mathsf{C}^j[\beta s + \alpha] + \mathsf{M}^j[\beta s + \alpha]$ for all $\beta \in [\nu, w - 1]$ and $\alpha \in [1, s]$ then output 1, else output 0.

ANALYSIS OF THE ATTACK. We show that in the above attack D outputs 1 with high probability while interacting with the CTR-ACPKM construction and with very low probability in the *ideal world*. Note that $\mathsf{C}^i[\beta s + \alpha] + \mathsf{M}^i[\beta s + \alpha] = \mathsf{C}^j[\beta s + \alpha] + \mathsf{M}^j[\beta s + \alpha]$ implies $E_{\mathsf{K}^i_\beta}(\mathsf{INC}^{\beta s+\alpha}_{\frac{n}{2}}(\mathsf{IV}\|0^{\frac{n}{2}})) = E_{\mathsf{K}^j_\beta}(\mathsf{INC}^{\beta s+\alpha}_{\frac{n}{2}}(\mathsf{IV}\|0^{\frac{n}{2}}))$ where K^*_β denotes the β-th section key of the $*$-th user. Thus, in the *real world*, if the i-th user key and j-th user key collide at the ν-th section, then $E_{\mathsf{K}^i_\beta}(\mathsf{INC}^{\beta s+\alpha}_{\frac{n}{2}}(\mathsf{IV}\|0^{\frac{n}{2}})) = E_{\mathsf{K}^j_\beta}(\mathsf{INC}^{\beta s+\alpha}_{\frac{n}{2}}(\mathsf{IV}\|0^{\frac{n}{2}}))$ for all $\beta \in [\nu, w - 1]$ and $\alpha \in [1, s]$ and the D outputs 1 at step 3 of the algorithm.

We analyze the attack in two steps. In the first step we show that such key collision occurs with high probability. To find this collision probability of two users key at the ν-th section, we recall a result from [DL14] in the following Lemma 3.

Lemma 3 ([DL14]). *Let $\nu \le 2^{\frac{\kappa}{2}}$ be a non-negative integer and f is a random function from $\{0,1\}^\kappa$ to itself. Then, the image of two arbitrary inputs to the ν-th iterate f^ν collide with probability about $\frac{\nu}{2^\kappa}$.*

In the second step we show that in the *ideal world*, D outputs 1 at step 3 of the algorithm with low probability. Thus D can distinguish the CTR-ACPKM construction from the *ideal world*.

Step-I: Let us consider the following indicator random variable

$$U_{i,j} = \begin{cases} 1, & \text{if } \mathsf{K}^i_\nu = \mathsf{K}^j_\nu \\ 0, & \text{otherwise.} \end{cases}$$

Now from Lemma 3, any two user keys collide at the ν-th section with probability $\frac{\nu}{2^k}$ where $\nu \le 2^{\frac{\kappa}{2}}$. Thus, $\Pr[U_{j,k} = 1] = \frac{\nu}{2^\kappa}$. Also, let $U = \sum_{i,j} U_{i,j}$ and we have to show that

$U \geq 1$ with high probability. As all the $U_{i,j}$'s are pairwise independent random variables, by using Chebyshev's inequality we have

$$\Pr[\|U - \mathsf{E}(U)\| \geq \mathsf{E}(U)] \leq \frac{\mathsf{V}(U)}{\mathsf{E}(U)^2}$$

$$\Rightarrow \Pr[U = 0] \leq \Pr[\|U - \mathsf{E}(U)\| \geq \mathsf{E}(U)] \leq \frac{\mathsf{V}(U)}{\mathsf{E}(U)^2}$$

$$\Rightarrow \Pr[U \geq 1] \geq 1 - \frac{\mathsf{V}(U)}{\mathsf{E}(U)^2}. \tag{4}$$

We compute the expectation as follows:

$$\mathsf{E}(U) = \mathsf{E}\Big[\sum_{j,k} U_{j,k}\Big] = \sum_{j,k} \Pr[U_{j,k} = 1] = \frac{\binom{u}{2}\nu}{2^\kappa}. \tag{5}$$

Similarly, from the pairwise independence of the variables $U_{j,k}$, we can compute the variance as follows:

$$\sum_{j,k} \mathsf{V}[U_{j,k}] = \sum_{j,k} \mathsf{E}[U_{j,k}^2] - (\mathsf{E}[U_{j,k}])^2 = \sum_{j,k} \frac{\nu}{2^\kappa} - \Big(\frac{\nu}{2^\kappa}\Big)^2 \leq \sum_{j,k} \frac{\nu}{2^\kappa} = \frac{\binom{u}{2}\nu}{2^\kappa}. \tag{6}$$

Thus from Equation 4, we get

$$\Pr[U \geq 1] \geq \Big(1 - \frac{2^\kappa}{\binom{u}{2}\nu}\Big) \geq \Big(1 - \frac{2^{\kappa+1}}{u^2\nu}\Big).$$

Finally, from the statement of the lemma, as D makes at least $2^{\kappa/2+1}/\sqrt{\nu}$ queries, we have the relation that $u^2\nu \geq 2^{\kappa+2}$. Thus, we get

$$\Pr[U \geq 1] \geq \frac{1}{2}.$$

Step-II: In this step, we bound the probability that D outputs 1 in the step 3 of the above algorithm while interacting with the *ideal world*. Note that, D outputs 1 if two user keys collides. However, the condition $\mathsf{C}^i[\beta s + \alpha] + \mathsf{M}^i[\beta s + \alpha] = \mathsf{C}^j[\beta s + \alpha] + \mathsf{M}^j[\beta s + \alpha]$ can occur for all $\beta \in [\nu, w - 1]$ and $\alpha \in [1, s]$ without being a key collision. Such cases are due to random collisions and can occur in *ideal world*. Let us consider the indicator random variable

$$Z_{i,j} = \begin{cases} 1, & \text{if } \mathsf{C}^i[\beta s + \alpha] + \mathsf{M}^i[\beta s + \alpha] = \mathsf{C}^j[\beta s + \alpha] + \mathsf{M}^j[\beta s + \alpha] \ \forall \beta \in [\nu, w - 1], \alpha \in [1, s] \\ 0, & \text{otherwise} \end{cases}$$

and let $Z = \sum_{i,j} Z_{j,k}$. It is easy to observe that a random collision occurs if there exist i, j such that $Z_{i,j} = 1$. As the outputs are chosen at uniformly random in the *ideal world*, such a random collision occurs with the probability at most $\frac{1}{((2^n)_s)^{w-\nu}}$. Thus the expected number of such cases:

$$\mathsf{E}(Z|U = 0) = \sum_{j,k} \Pr[Z_{j,k} = 1|U = 0] = \frac{\binom{u}{2}}{((2^n)_s)^{w-\nu}} \leq \frac{u^2}{((2^n)_s)^{w-\nu}}. \tag{7}$$

To simplify the above inequality, we recall an idea from [HT17]. Let $s(w - \nu) = \left\lceil \frac{\kappa+d}{n-1} \right\rceil$ for some suitable choice of d. In that case we have

$$(2^n)_s = 2^n(2^n - 1)...(2^n - s + 1) \geq 2^{(n-1)s}.$$

Thus from the above Equation 7 and using Markov's inequality we can bound the probability that D outputs 1 in the *ideal world* as follows:

$$\Pr[Z \geq 1] \leq \mathsf{E}(Z|U = 0) \leq \frac{u^2}{((2^n)_s)^{w-\nu}} \leq \frac{u^2}{2^{\kappa+d}}.$$

Thus, for sufficiently large d (consequently, for sufficiently large $w - \nu$), the expected number of random collisions is small enough.