# Cryptographic Treatment of Key Control Security*

## In Light of NIST SP 800-108

Ritam Bhaumik[1] Avijit Dutta[2] Akiko Inoue[3] Tetsu Iwata[4]
Ashwin Jha[5] Kazuhiko Minematsu[3] Mridul Nandi[6] Yu Sasaki[7]
Meltem Sönmez Turan[8] Stefano Tessaro[9]

[1] TII, Abu Dhabi, UAE, bhaumik.ritam@gmail.com
[2] Institute for Advancing Intelligence, TCG CREST, Kolkata, India, avirocks.dutta13@gmail.com
[3] NEC Corporation, Kawasaki, Japan, a_inoue@nec.com, k-minematsu@nec.com
[4] Nagoya University, Nagoya, Japan, tetsu.iwata@nagoya-u.jp
[5] Ruhr-University Bochum, Bochum, Germany, ashwin.jha@ruhr-uni-bochum.de
[6] Indian Statistical Institute, Kolkata, Kolkata, India, mridul.nandi@gmail.com
[7] NTT Social Informatics Laboratories, Tokyo, Japan, yusk.sasaki@ntt.com
[8] National Institute of Standards and Technology, Gaithersburg, MD, USA, meltem.turan@nist.gov
[9] University of Washington, Seattle, WA, USA, tessaro@cs.washington.edu

**Abstract**

This paper studies the security of *key derivation functions* (KDFs), a central class of cryptographic algorithms used to derive *multiple* independent-looking keys (each associated with a particular context) from a *single* secret. The main security requirement is that these keys are pseudorandom (i.e., the KDF is a pseudorandom function). This paper initiates the study of an additional security property, called *key control* (KC) security, first informally put forward in a recent update to NIST Special Publication (SP) 800-108 standard for KDFs. Informally speaking, KC security demands that, given a *known* key, it is hard for an adversary to find a context that forces the KDF-derived key for that context to have a property that is specified a-priori and is hard to satisfy (e.g., that the derived key consists mostly of 0s, or that it is a weak key for a cryptographic algorithm using it). We provide a rigorous security definition for KC security, and then move on to the analysis of the KDF constructions specified in NIST SP 800-108. We show, via security proofs in the random oracle model, that the proposed constructions based on XOFs or hash functions can accommodate for reasonable security margins (i.e., 128-bit security) when instantiated from KMAC and HMAC. We also show, via attacks, that all proposed block-cipher based modes of operation (while implementing mitigation techniques to prevent KC security attacks affecting earlier version of the standard) only achieve *at best* 72-bit KC security for 128-bit blocks, as with AES.

**Keywords:** Key derivation function; NIST SP 800-108; Key control security; CMAC; HMAC; KMAC.

*An extended abstract of this article appears in the proceedings of CRYPTO 2025 [BDI+25] and is available via DOI ⟨TBD⟩ from the Springer website.

# Contents


**1 Introduction** **3**
- 1.1 Our Contributions . . . 4
- 1.2 Related Work . . . 5

**2 Preliminaries** **6**
- 2.1 Notation . . . 6
- 2.2 NIST SP 800-108 . . . 6
- 2.3 Syntax of NIST SP 800-108 KDF . . . 7

**3 Security Definitions** **8**
- 3.1 Key Control Security . . . 8
- 3.2 Rationale of the Definition . . . 12
- 3.3 Weak Keys in Standards . . . 13
- 3.4 Chosen-$K_{IN}$ Setting . . . 14

**4 Analysis of KMAC-based KDF** **14**

**5 Analysis of HMAC-based KDFs** **16**
- 5.1 Shared Proof Framework and Notations . . . 16
- 5.2 Analysis of CTR-HMAC[$H$] . . . 17
- 5.3 Analysis of FB-HMAC[$H$] . . . 19
- 5.4 Analysis of DP-HMAC[$H$] . . . 22

**6 Analysis of CMAC-based KDFs** **25**
- 6.1 Attacks on DP-CMAC[$E$] . . . 26
- 6.2 Attacks on stDP-CMAC[$E$] . . . 27
- 6.3 Note on stCTR-CMAC[$E$] and stFB-CMAC[$E$] . . . 28

**7 Conclusions** **28**

**References** **29**

**A Initial Observations on KC Security of** CTR-CMAC[$E$] **and** FB-CMAC[$E$] **by Arciszewski et al.** **35**

**B Real-World Applications of NIST SP 800-108** **35**

**C Switching from Without-Replacement to Uniform** **36**

**D Proof of Claim in Eq.** (20) **37**

**E Attack on** stCTR-CMAC[$E$] **38**

**F XTS Weak Keys with** FB-CMAC[$E$] **39**

**G Two-Block Attacks on** FB-CMAC[$E$] **and** CTR-CMAC[$E$] **40**

**H Note on Attacks against** HMAC**-based KDFs** **42**

# 1 Introduction

A *key derivation function* (KDF) is a fundamental and widely deployed cryptographic algorithm used to derive *multiple* key values, such as session keys or user keys, from a *single* secret value. There are several possible KDF candidates, e.g., HKDF [Kra10], Chuah et al. [WDNS12, WDS13], and ones relying on passwords or biometrics [PJ16, KAA21, SPK+18] or multi-factor authentication [NS23], involving different settings, but this paper will focus on constructions standardized in NIST SP 800-108 [Che22], which specifies several KDFs using a *pseudorandom function* (PRF). These KDFs are often called key-based key derivation functions (KBKDFs). Following the terminology used in the standard, we let a key derivation function be an algorithm KDF which takes two inputs, a *key derivation key* $K_{IN}$ and some auxiliary information *Context* that produces the derived key $K_{OUT} = \text{KDF}(K_{IN}, \mathit{Context})$. Note that KDF may also take additional input variables such as system parameters. As in the NIST standard, we assume the key derivation key to be a cryptographic key, i.e., it is cryptographically securely generated and shared among the parties. As in [Kra10], one could consider a more general setting where the key derivation key is merely guaranteed to be a high-entropy secret, but we will not do so in this paper.

**Key control security.** The standard security requirement of KDFs is the indistinguishability of its outputs from truly random bit strings, assuming that the key derivation key is unknown to the adversary. This is implied by the security of the KDF as a PRF, i.e., we require that $K_{OUT} = \text{KDF}(K_{IN}, \mathit{Context})$ behaves as a PRF, where $K_{IN}$ is a secret key and *Context* is under the control of the adversary.

The latest revision of NIST Special Publication (SP) 800-108 [Che22], which improves upon earlier versions from 2008 and 2022, describes three modes of operation: a *Counter* mode (CTR), a *Feedback* mode (FB), and a *Double-Pipeline* mode (DP), each of which defines the use of HMAC [Nat08] and CMAC [Dwo05] as the underlying PRF, and it also describes a KDF based on KMAC [KjCP16], which we call KDF-KMAC. Interestingly, a public comment by an Amazon team in [Nat22] triggered the inclusion in the standard of the discussion of a new security property, called *key control* (KC) security, which is informally defined as follows:

> **Definition 1** (Key Control Security [Che22])**.** *When multiple parties contribute to the input of a key-derivation process, key-control security (or key-control resistance) is attained when the parties have assurance that (even with knowledge of the key derivation key $K_{IN}$) no single party (or proper subset of the contributors) can manipulate the process in such a way as to force output keying material to a pre-selected value (regardless of the contributions of the others) to the detriment of any applications relying on that keying material.*

In other words, the concern is that for some applications a malicious party, after learning a legitimate $K_{IN}$ (e.g., as the result of a key-establishment protocol), may have control of the context value and may be able to force the corresponding derived key for that context to have some undesirable property–in particular, being a weak key for the cryptographic algorithm using the derived key.

Based on the informal definition of KC security [Nat22], the document [Che22] shows that the CTR KDF with CMAC as the underlying PRF is insecure with respect to KC security[1]. In response to this, SP 800-108 [Che22] suggests the use of KMAC or HMAC as the PRF,

[1] We review the attack in Appendix A, showing that even if the key derivation key $K_{IN}$ is securely generated, one of the communicating parties can set the value of $K_{OUT}$ to a pre-selected value by choosing *Context*. The observation directly applies to the KDF in the FB mode with CMAC.

Table 1: Provable key control security bounds. $q$ is the number of queries, $\delta$ is the success probability with a random guess, $\widetilde{k}$ denotes the average min-entropy of the key derivation key, $h$ is the output length of the PRF in bits, and $s = \lceil L/h \rceil$, where $L$ is the output length of the KDF.

| Scheme | Bound | Ref. |
|---|---|---|
| KDF-KMAC | $O\left(q2^{-\widetilde{k}} + q\delta\right)$ | Sect. 4 |
| CTR-HMAC | $O\left(q2^{-\widetilde{k}} + q\delta + s^2q^22^{-h}\right)$ | Sect. 5.2 |
| FB-HMAC | $O\left(q2^{-\widetilde{k}} + q\delta + s^2q^22^{-h}\right)$ | Sect. 5.3 |
| DP-HMAC | $O\left(q2^{-\widetilde{k}} + q\delta + s^2q^22^{-h}\right)$ | Sect. 5.4 |

and it also suggests strengthened versions of the CTR, FB, and DP modes when CMAC is used (See Sect. 2.2 and 2.3 for specifications). Although these KDFs do not seem to have trivial weaknesses, there is no supporting argument that they indeed attain KC security. More generally, to the best of our knowledge, there is no formal definition of what it means for a KDF to be secure in terms of KC security.

### 1.1 Our Contributions

We provide the first formal cryptographic treatment of key control security. We formalize it as a variant of preimage security of the KDF in the known-key setting, where the adversary can choose a set of target images (which can potentially be *exponentially* large), and after learning the key, needs to find a preimage, mapping to one of the target images, which has to follow the input format of the KDF. We formalize the security notion as a *key control (KC) security* game. This is a commit-then-find game that has two stages. In the first stage of the game, the adversary commits to a set of target images by outputting a predicate that decides whether an image is in the set, and in the second stage, the adversary learns the key derivation key $K_{IN}$ and attempts to find a preimage for any of the target images. Our formalization is inspired by the work of Coretti et al. [CDKT19] to define oracle-dependent adversarial sources of randomness for random number generators. Indeed, to maximize the generality of the security notion, we grant the choice of $K_{IN}$ and other parameters to the adversary, under suitable conditions, e.g., $K_{IN}$ has a sufficient average min-entropy[2], covering a wide range of attack scenarios.

We then provide a comprehensive key control security analysis of KDFs defined in NIST SP 800-108 [Che22] from provable security and cryptanalytic perspectives. We show that NIST's KMAC and HMAC-based KDFs are provably secure in the random oracle model. For CMAC-based KDFs, even with the strengthened versions, we show that they only achieve *at best* 72-bit key control security for 128-bit blocks, as with AES, by presenting concrete attack procedures[3]. See Table 1 for the summary of our provable security results and Table 2 for our attacks against AES-CMAC-based KDFs.

**Real-world implications.** Our analysis has direct implications for real-world use cases of NIST SP 800-108 KDFs, which are extensive. For example, components of major OSs including Linux [CAV, asp, jet, ora] and hardware security modules [HSM, tit, cce, arm, nxp, gp] adopt

[2]This is still in a known-key setting, as $K_{IN}$ is returned by the adversary but with sufficient randomness.

[3]The "attack" is with respect to the security notion of this paper. This does not mean to break some existing security claim, as no specific security notion was defined prior to this work.

Table 2: Attack complexity of AES-CMAC-based KDFs.

| Scheme | CTR-CMAC | | FB-CMAC | | DP-CMAC | |
|---|---|---|---|---|---|---|
| | Comp. | Ref. | Comp. | Ref. | Comp. | Ref. |
| Original | 1 | [Nat22] | 1 | [Nat22] | $2^{64}$ | Sect. 6.1 |
| Strengthened | $2^{72}$ | Sect. 6.3 | $2^{72}$ | Sect. 6.3 | $2^{72}$ | Sect. 6.2 |

NIST KDFs. Multiple Internet standards adopt them as well, including the widely deployed TCP Authentication Option (TCP-AO) [TBM10, RL10]. Among them, we found cases where KC security could be relevant since the input context has some freedom that could be exploited by a malicious party. Appendix B gives more details.

### 1.2 Related Work

**Treatment of KDFs.** The work of Adams et al. [AKMZ04] is, to the best of our knowledge, the first academic treatment of generic KDFs, albeit at a rather informal level. Krawczyk [Kra10] gives the first formal treatment of KDFs, however his main concern is the composition of a randomness extractor with a PRF to obtain a (salted) KDF where $K_{IN}$ is a non-necessarily uniform high-entropy *secret* value. Here, we are concerned with properties of KDFs in the *known-key setting*, thus our treatment is orthogonal to the work in [Kra10]. Of course, we also want KDFs to satisfy the standard property of producing pseudorandom keys, but we note here that NIST SP 800-108 does not deal with weak secrets, i.e., the secret is already assumed to be a uniform secret key (obtained e.g., from a key-establishment protocol), and the security target is that of being a PRF. Nonetheless, any of the KDFs put forward by NIST can be composed with a randomness extractor via Krawczyk's generic transformation.

See [SWG25] for the analysis of KDFs in NIST SP 800-108 in standard notions.

**Password-based key derivation functions.** The work of this paper is somewhat orthogonal to that on *password-based* KDFs, such as PBKDF (standardized in PKCS #5 [Kal00]), Scrypt [PJ16], and Argon2 [BDKJ21]. These KDFs explicitly target the problem of deriving a key from a secret with moderately high min-entropy, such as a password, and thus introduce additional measures (e.g., iteration, memory costs, bandwidth usage) to make the computation moderately hard.

**Correlation intractability and preimage resistance.** The security notion closest to KC security is that of *correlation intractability* [CGH98] for seeded hash functions, which asks that for a given known random seed, it is hard to find an input-output pair that satisfies an a-priori fixed "evasive" relation, i.e., a relation for which finding an input-output pair satisfying it is hard for a random oracle. One could think of KC security as a special case where $K_{IN}$ plays the role of the seed, but we allow the relation to be chosen, adaptively, in a *preprocessing* stage. Moreover, we only consider a subset of relations defined by an output predicate $p$ with sufficient sparsity, i.e., the relation consists of all pairs $(Context, Y)$ where $p(Y) = 1$. It is not hard to verify that such relations are evasive.

KC security also implies preimage resistance [RS04] by choosing a predicate that checks equality with a single output. It is also easy to see that it is strictly weaker–consider a function $G(X) = H(X) \,\|\, 0^h$, where $H : \{0,1\}^* \to \{0,1\}^h$ is a random oracle. It is preimage resistant, but also not KC secure with respect to the predicate that returns one on any string $Y \,\|\, 0^h$ for $Y \in \{0,1\}^h$. Multi-target preimage resistance [HRS16] is also related, where the adversary is

given multiple target images, and tries to find a preimage of one of them. It is easy to see that finding one of the preimages of $H(X_1) \| 0^h, H(X_2) \| 0^h, \ldots$ is infeasible.

**Secure coin flipping.** Different approaches could help ensure KC security. For example, secure coin flipping [Blu83] allows two or more parties to flip a coin over a network with the aim to ensure that neither party can bias the result in their favor, even in a malicious setting. Secure coin flipping, e.g., using commitment schemes, can be utilized to determine the input values such as the *context*, so that none of the parties can influence the output key.

# 2 Preliminaries

## 2.1 Notation

Let $\mathbb{N}$ be the set of non-negative integers. For integers $a \leq b$, let $[a..b] \coloneqq \{a, a+1, \ldots, b\}$. Let $\varepsilon$ be an empty string. For an integer $i \geq 0$, $\{0,1\}^i$ denotes the set of bit strings of $i$ bits, and $\{0,1\}^*$ denotes the set of all bit strings of finite lengths (including $\varepsilon$). For an integer $i \geq 0$, $0^i$ denotes the bit string of $i$ zero bits. For an integer $0 \leq i \leq 2^r - 1$, $\langle i \rangle_r$ denotes the encoding of $i$ as an $r$-bit string. For a bit string $X$, its length in bits is written as $|X|$, and for $t \leq |X|$, the most significant (least significant) $t$ bits of $X$ is written as $\mathtt{msb}_t(X)$ ($\mathtt{lsb}_t(X)$). For bit strings $X$ and $Y$, $X \| Y$ denotes their concatenation.

For a set $A$, $a \xleftarrow{\$} A$ indicates that the element $a \in A$ is uniformly chosen at random from $A$. For any $m > 0$ and $p : \{0,1\}^m \to \{0,1\}$, we write $U_m$ to denote a uniformly distributed string over $\{0,1\}^m$ and $\delta_{p,m} \coloneqq \Pr[p(U_m) = 1]$.

For a bit string $X$ and an integer $n$, let $(X[1], X[2], \ldots, X[m]) \xleftarrow{n} X$ denote the parsing of $X$ into $n$-bit blocks: when $X \neq \varepsilon$, we let $m = \lceil |X|/n \rceil$ and we have $|X[i]| = n$ for $i \in [1..m-1]$ and $|X[m]| \leq n$; when $X = \varepsilon$, we let $m = 1$ and $X[1] = \varepsilon$.

A block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ is a keyed permutation, where $k$ is the key length and $n$ is the block length. For any key $K \in \{0,1\}^k$, $E(K, \cdot) = E_K(\cdot)$ is a permutation over $\{0,1\}^n$. Let $\mathrm{Perm}(n)$ be the set of all permutations over $\{0,1\}^n$. The $(n,k)$ ideal cipher, IC, is the set of $2^k$ independent random permutations that models an ideally secure block cipher, where for each key $K \in \{0,1\}^k$, $\mathrm{IC}_K(\cdot) \xleftarrow{\$} \mathrm{Perm}(n)$.

Let $H : \{0,1\}^* \to \{0,1\}^h$ be a hash function, where $h$ is the output length. We model the hash function as a random oracle, which is an oracle that responds to every unique query with an element from $\{0,1\}^h$ chosen uniformly at random.

**Average min-entropy.** We formalize our security notion based on the average min-entropy notion from [DRS04]. For random variables $A$ and $B$, $\widetilde{\mathbf{H}}_\infty(A \mid B)$ denotes the *average min-entropy* of $A$ given $B$, which is defined as

$$\widetilde{\mathbf{H}}_\infty(A \mid B) = -\log \left( \mathop{\mathbb{E}}_{b \xleftarrow{\$} B} \left[ 2^{-\mathbf{H}_\infty(A|B=b)} \right] \right),$$

where $\mathbf{H}_\infty(A \mid B = b) = -\log(\max_a \Pr[A = a \mid B = b])$ is the min-entropy of $A$ given $B = b$, and $\mathbb{E}$ denotes the expectation.

## 2.2 NIST SP 800-108

NIST SP 800-108 [Che22] proposes KDFs using PRFs like HMAC [Nat08], CMAC [Dwo05], and KMAC [KjCP16]. If the desired length of the derived key is larger than the output of

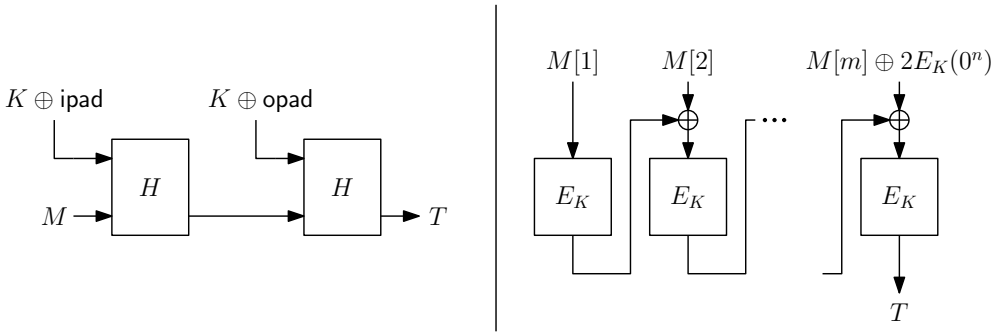


Figure 1: HMAC with a hash function $H$ (left) and CMAC with an $n$-bit block cipher $E$ (right) for the case $|M[m]| = n$. If $|M[m]| < n$, $(M[m] \,\|\, 100\cdots) \oplus 2^2 E_K(0^n)$ is used for the last block instead of $M[m] \oplus 2E_K(0^n)$.

the underlying PRF, a mode of iteration is used to call the PRF as many times as needed. The standard defines three iteration modes for CMAC and HMAC: CTR (Counter), FB (Feedback), and DP (Double-Pipeline). See Fig. 1 for CMAC and HMAC. The KMAC-based KDF, KDF-KMAC, has no iteration mode; hence there are no variations. Thus we have seven KDF schemes: KDF-KMAC and Mode-PRF, where Mode $\in \{\mathrm{CTR}, \mathrm{FB}, \mathrm{DP}\}$ and PRF $\in \{$CMAC, HMAC$\}$. In the latest revision of the standard [Che22], security issues were pointed out against CTR-CMAC and FB-CMAC (reviewed in Appendix A). In order to mitigate these weaknesses, NIST specified a "strengthened" version of each mode for CMAC [Che22]. We denote these modes as stCTR, stFB, and stDP. Including these three additional KDFs, in total, SP 800-108 specifies the following ten KDF schemes:

- KDF-KMAC,
- CTR-HMAC, FB-HMAC, DP-HMAC,
- CTR-CMAC, FB-CMAC, DP-CMAC,
- stCTR-CMAC, stFB-CMAC, and stDP-CMAC.

KMAC uses an extendable-output function (XOF) [Nat15, BDPA08], HMAC uses a hash function, and CMAC uses a block cipher. NIST further specifies the concrete instantiations of the internal PRF for each KDF, which will be described in the relevant sections. Also, we may write Mode[PRF] to mean Mode-PRF.

### 2.3 Syntax of NIST SP 800-108 KDF

We describe the details of NIST SP 800-108 KDFs. Formally, let $\mathcal{K}_{IN} \subseteq \{0,1\}^*$ be the set of key derivation keys, $\mathcal{K}_{OUT} \subseteq \{0,1\}^*$ be the set of keying materials, $\mathcal{L} \subseteq \{0,1\}^*$ be the set of labels, and $\mathcal{C} \subseteq \{0,1\}^*$ be the set of contexts. Typically, a label *Label* is a bit string which is used to identify the purpose of the derived keying material, and a context *Context* is a bit string which is used to denote the information on the derived keying material, such as identities of the parties who are deriving and/or using the derived keying material and, optionally, a nonce known by the parties who derive the keys. For example, TCP-AO (RFC 5295, see Appendix B) sets *Label* as ASCII string "TCP-AO" and *Context* as a sequence of the source/destination addresses, the port numbers, and the random numbers generated by the sender and the receiver. Then $\mathrm{KDF} : \mathcal{K}_{IN} \times \mathcal{L} \times \mathcal{C} \times \mathbb{N} \to \mathcal{K}_{OUT}$ takes $(K_{IN}, \mathit{Label}, \mathit{Context}, L) \in \mathcal{K}_{IN} \times \mathcal{L} \times \mathcal{C} \times \mathbb{N}$ as input

to return $K_{OUT} \in \mathcal{K}_{OUT}$, where $|K_{OUT}| = L$. We write

$$\mathrm{KDF}(K_{IN}, \mathit{Label}, \mathit{Context}, L) = K_{OUT}\,.$$

Optionally, a KDF may take an initial value $IV \in \mathcal{IV} \subseteq \{0,1\}^*$ as additional input. This value can be a public or secret value. For simplicity, we assume that $IV$ is a public constant determined by $(\mathit{Label}, L)$. We remark that $L$ is the length of $K_{OUT}$ in bits and we interchangeably use it as an encoded bit string of the pre-specified length of $L$.

**Overview of KDFs.** We give a brief overview of the KDFs listed in [Che22]. Let $F$ be a PRF, where $F : \mathcal{K}_{IN} \times \{0,1\}^* \to \{0,1\}^h$. Figure 2 shows the pseudocode of KDFs based on $F$ defined in [Che22]. The first construction is KDF-KMAC which uses KMAC as the underlying PRF. The next one is CTR$[F]$, where the CTR mode is instantiated with a PRF $F$ that takes the key $K_{IN}$, $r$-bit counter where, following [Che22], $r \in [1..32]$, and fixed data consisting of *Label*, *Context*, and $L$ (see Fig. 3). Another proposed design of KDF is FB$[F]$ which is similar to CTR$[F]$, except that the $i$-th call of $F$ takes the previous output of $F$, $K[i-1]$, as a part of input in addition to $K_{IN}$, *Label*, *Context*, and $L$, and the first call of $F$ takes $K[0]$ which is $IV$ (see Fig. 4). The final mode of operation is DP$[F]$. It consists of two layers, where the first layer is a plain feedback mode of $F$ taking $K_{IN}$ to produce the sequence $A[1], \ldots, A[s]$ for $s = \lceil L/h \rceil$, and the second one is similar to CTR$[F]$ but its $i$-th call takes $A[i]$ in addition to other input variables (see Fig. 5). Their strengthened counterparts, i.e., stCTR$[F]$, stFB$[F]$, and stDP$[F]$ have either additional input to $F$ or a different derivation of $K[0]$ from the base scheme. In FB, DP, stFB, and stDP, the use of an $r$-bit counter $\langle i \rangle_r$ at the input of $F$ is left to users as an option. In our analysis, we assume that the base versions of FB and DP do not have a counter, and those of stFB and stDP have a counter.

# 3 Security Definitions

## 3.1 Key Control Security

Let $\mathrm{KDF} : \mathcal{K}_{IN} \times \mathcal{L} \times \mathcal{C} \times \mathbb{N} \to \mathcal{K}_{OUT}$ be a key derivation function. We consider a key derivation function KDF that internally uses an ideal primitive $P$, denoted as $\mathrm{KDF}^P$, i.e., KDF makes oracle calls to $P$ to compute its output. Typically, $P$ is a random oracle or an ideal cipher. We formalize the security notion of a KDF as a key control (KC) security game, which is defined in Fig. 6. The game is a commit-then-find game and has two stages.

At a high level, this is a preimage game in the known-key setting. In the commit stage, the adversary specifies an instance of the KDF by choosing a label and the output length. It also outputs a predicate over the key space to specify a set of target images. In addition to this, the adversary returns the key derivation key $K^*_{IN}$ as a result of, e.g., running a key-establishment protocol. The transcript and any auxiliary information are also returned. In the find stage, the goal is to find a preimage of any of the target images. We start by defining a key control (KC) adversary.

**Definition 2** (KC Adversary)**.** *Let* $\Gamma$ *denote the tuple* $(q_{\mathrm{cmt}}, q_{\mathrm{fnd}}, b_{\min}, b_{\max}, L_{\min}, L_{\max}, \widetilde{k}, \delta)$*. Then a* $\Gamma$*-KC adversary* $\mathcal{A}$ *is defined as a pair of algorithms* $(\mathcal{A}_{\mathrm{cmt}}, \mathcal{A}_{\mathrm{fnd}})$*, which act as follows:*

**Commit stage:** $\mathcal{A}_{\mathrm{cmt}}$ *makes* $q_{\mathrm{cmt}}$ *queries to its oracle* $P$ *and outputs* $(K^*_{IN}, S^*)$ *with* $S^* = (\mathit{Label}^*, L^*, p^*, \tau^*_{\mathrm{cmt}}, \mathrm{aux}^*)$*, where*

- $\mathit{Label}^* \in \mathcal{L}$ *denotes the label with* $|\mathit{Label}^*| \in [b_{\min}..b_{\max}]$*;*
- $L^* \in [L_{\min}..L_{\max}]$ *denotes the bit-length of the desired* KDF *output;*

**Algorithm** KDF-KMAC$[H](K_{IN}, \mathit{Label}, \mathit{Context}, L)$
1. **if** $L > 2^{1040} - 1$ **then return** $\perp$
2. $K_{OUT} \leftarrow \mathrm{KMAC}[H](K_{IN}, \mathit{Context}, L, \mathit{Label})$
3. **return** $K_{OUT}$

**Algorithm** CTR$[F](K_{IN}, \mathit{Label}, \mathit{Context}, L)$
1. $s \leftarrow \lceil L/h \rceil$
2. $K[0] \leftarrow F(K_{IN}, \mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L)$
3. **for** $i = 1$ **to** $s$ **do**
4. $\quad K[i] \leftarrow F(K_{IN}, \langle i \rangle_r \,\|\, \mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L \,\|\, K[0])$
5. $K_{OUT} \leftarrow \mathtt{msb}_L(K[1] \,\|\, \cdots \,\|\, K[s])$
6. **return** $K_{OUT}$

**Algorithm** FB$[F](K_{IN}, \mathit{Label}, \mathit{Context}, L)$
1. $s \leftarrow \lceil L/h \rceil$; $K[0] \leftarrow IV$ // $IV$ is a constant defined by $(\mathit{Label}, L)$
2. **if** $s > 2^{32} - 1$ **then return** $\perp$
3. $K[0] \leftarrow F(K_{IN}, \mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L)$
4. **for** $i = 1$ **to** $s$ **do**
5. $\quad K[i] \leftarrow F(K_{IN}, K[i-1] \,\|\langle i \rangle_r\, \|\, \mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L)$
6. $K_{OUT} \leftarrow \mathtt{msb}_L(K[1] \,\|\, \cdots \,\|\, K[s])$
7. **return** $K_{OUT}$

**Algorithm** DP$[F](K_{IN}, \mathit{Label}, \mathit{Context}, L)$
1. $s \leftarrow \lceil L/h \rceil$
2. **if** $s > 2^{32} - 1$ **then return** $\perp$
3. $A[0] \leftarrow \mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L$
4. **for** $i = 1$ **to** $s$ **do**
5. $\quad A[i] \leftarrow F(K_{IN}, A[i-1])$
6. $\quad K[i] \leftarrow F(K_{IN}, A[i] \,\|\langle i \rangle_r\, \|\, \mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L)$
7. $K_{OUT} \leftarrow \mathtt{msb}_L(K[1] \,\|\, \cdots \,\|\, K[s])$
8. **return** $K_{OUT}$

Figure 2: Algorithms for NIST SP 800-108 KDFs using PRF $F$, where $h$ denotes the output bit length of $F$. CTR$[F]$, FB$[F]$, and DP$[F]$ are defined without the gray boxes, and stCTR$[F]$, stFB$[F]$, and stDP$[F]$ include the gray boxes, with $F$ fixed to CMAC.

- $p^*$ *denotes a predicate on* $\{0,1\}^{L^*}$*;*
- $\tau^*_{\mathrm{cmt}}$ *denotes the transcript of interaction between* $\mathcal{A}_{\mathrm{cmt}}$ *and* $P$*;*
- $\mathrm{aux}^*$ *denotes auxiliary information output by* $\mathcal{A}_{\mathrm{cmt}}$*;*
- $K^*_{IN}$ *is a key derivation key obtained as a result of, e.g., running a key-establishment protocol.*

$\mathcal{A}_{\mathrm{cmt}}$ *is subject to the restrictions that* $\widetilde{\mathbf{H}}_\infty(K^*_{IN} \mid S^*) \geq \widetilde{k}$*, and that*

$$\mathop{\mathbb{E}}_{S^* \xleftarrow{\$} \mathcal{A}^P_{\mathrm{cmt}}} [\delta_{p^*, L^*}] \leq \delta\,.$$

*Recall that* $\delta_{p^*,L^*} := \Pr[p^*(U_{L^*}) = 1]$ *with* $U_{L^*} \xleftarrow{\$} \{0,1\}^{L^*}$.

**Find stage:** $\mathcal{A}_{\mathrm{fnd}}$ *receives* $(K^*_{IN}, S^*)$*, makes* $q_{\mathrm{fnd}}$ *queries to* $P$*, and outputs a context* $\mathit{Context}^* \in \mathcal{C}$.

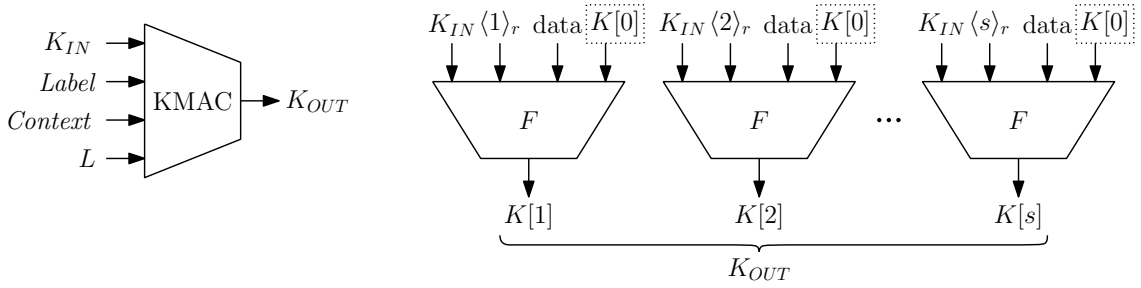


Figure 3: KDF-KMAC (left) and CTR and stCTR modes with PRF $F$ (right), where data = $(\mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L)$, $s = \lceil L/h \rceil$ (in this figure, $h$ divides $L$). stCTR includes $K[0]$ in the boxes with dotted lines.

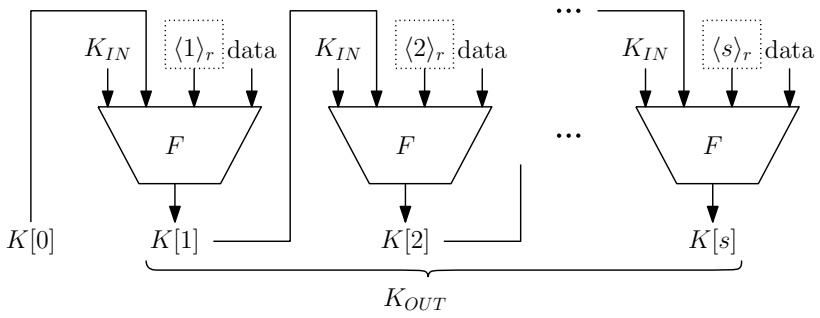


Figure 4: FB and stFB modes with PRF $F$, where data = $(\mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L)$, $s = \lceil L/h \rceil$ (in this figure, $h$ divides $L$). $K[0]$ is $IV$ with original FB, but it is derived via $F$ for stFB. stFB includes $\langle i \rangle_r$ for $i \in [1..s]$ in the boxes with dotted lines.

The queries of the former type will be referred to as *commit* queries, and those of the latter type will be referred to as *find* queries.

We next define the key control security (KCS) advantage. See Fig. 6 for the definition of $\mathrm{KCS}_{\mathrm{KDF}^P}(\mathcal{A}^P)$. If the game returns 1, it means $\mathcal{A}$ wins.

**Definition 3** (Key Control Security Advantage)**.** *We say that* $\mathrm{KDF}^P$ *is* $\epsilon$*-KC secure if for all* $\Gamma$*-KC adversaries* $\mathcal{A}$*, we have*

$$\Pr[\mathrm{KCS}_{\mathrm{KDF}^P}(\mathcal{A}^P) = 1] \leq \epsilon(\Gamma)\,.$$

We may omit the subscript in $\mathrm{KCS}_{\mathrm{KDF}^P}(\mathcal{A}^P)$ and write $\mathrm{KCS}(\mathcal{A}^P)$ if it is clear from the context.

In the commit stage, the adversary specifies $(K^*_{IN}, S^*)$ with $S^* = (\mathit{Label}^*, L^*, p^*, \tau^*_{\mathrm{cmt}}, \mathrm{aux}^*)$. The transcript $\tau^*_{\mathrm{cmt}}$ is output implicitly during the commit stage, since the queries and responses of $\mathcal{A}_{\mathrm{cmt}}$ are always visible to the security experiment; the rest are output explicitly by $\mathcal{A}_{\mathrm{cmt}}$.[4] The auxiliary information $\mathrm{aux}^*$ is any information associated with the attack produced by $\mathcal{A}_{\mathrm{cmt}}$, as long as it does not reduce the min-entropy of $K^*_{IN}$.

Our definition allows the adversary to determine the distribution of $\mathit{Label}^*$ and $L^*$, subject to certain length restrictions. In real-world applications, these values are typically selected by the protocol running the KDF. Consequently, there is a significant possibility that the resulting distribution may reduce the adversary's KCS advantage. However, we grant this additional

[4]Note that even if $\mathcal{A}_{\mathrm{cmt}}$ were to explicitly output $\tau^*_{\mathrm{cmt}}$, it would not stand to gain any advantage by outputting a tampered transcript, so we can always assume without loss of generality that $\tau^*_{\mathrm{cmt}}$ is the honest transcript of the commit stage.

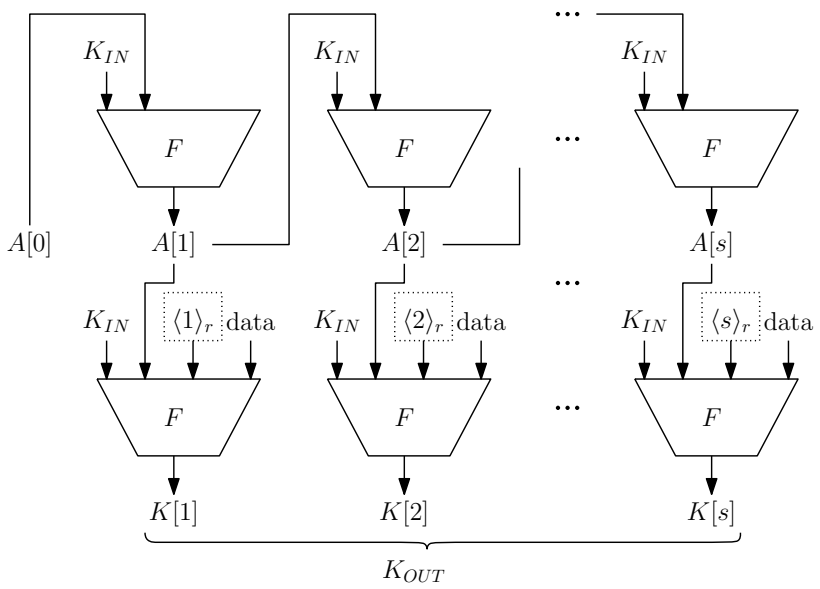


Figure 5: DP and stDP modes with PRF $F$, where $A[0]$ is defined as $A[0] = \text{data} = (\mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L)$, $s = \lceil L/h \rceil$ (in this figure, $h$ divides $L$). stDP includes $\langle i \rangle_r$ for $i \in [1..s]$ in the boxes with dotted lines.

**Game** $\mathrm{KCS}_{\mathrm{KDF}^P}(\mathcal{A}^P)$

1. $(K^*_{IN}, S^*) \xleftarrow{\$} \mathcal{A}^P_{\mathrm{cmt}}$, where $S^* = (\mathit{Label}^*, L^*, p^*, \tau^*_{\mathrm{cmt}}, \mathrm{aux}^*)$ // commit stage
2. $\mathit{Context}^* \xleftarrow{\$} \mathcal{A}^P_{\mathrm{fnd}}(K^*_{IN}, S^*)$ // find stage
3. $K^*_{OUT} \leftarrow \mathrm{KDF}^P(K^*_{IN}, \mathit{Label}^*, \mathit{Context}^*, L^*)$ // $K^*_{OUT}$ is computed
4. **if** $p^*(K^*_{OUT}) = 1$ **then return** 1
5. **else return** 0

Figure 6: The key control security game.

power to the adversary in order to account for a wide range of scenarios, thereby strengthening our provable security results. Simultaneously, our attacks on CTR-CMAC work regardless of the choice of the label, provided the length restrictions are respected (see Sect. 6 for details). This goes on to demonstrate the effectiveness of our attacks. The predicate $p^*$ is used to specify the set of target images $\mathcal{T} \subseteq \{0,1\}^{L^*}$, i.e., $p^*(T) = 1$ iff $T \in \mathcal{T}$. We require that $|\mathit{Label}^*| \in [b_{\min}..b_{\max}]$ and $L^* \in [L_{\min}..L_{\max}]$. In particular, we require that $L^*$ is not too small, since such a keying material cannot be cryptographically strong for a key of a PRF.

We require $\widetilde{\mathbf{H}}_\infty(K^*_{IN} \mid S^*) \geq \widetilde{k}$, i.e., $K^*_{IN}$ has sufficient average min-entropy, conditioned on $S^* = (\mathit{Label}^*, L^*, p^*, \tau^*_{\mathrm{cmt}}, \mathrm{aux}^*)$. As long as the min-entropy bound is satisfied, we do not impose any temporal order on the sampling of $S^*$. For instance, $\mathcal{A}_{\mathrm{cmt}}$ can query $P$ *after* $K^*_{IN}$ is sampled, but not *with* $K^*_{IN}$, as this would violate the min-entropy bound. In real use cases, $K^*_{IN}$ could be computed through a key-establishment protocol, while our formalization lets the adversary output the value with the restriction that $K^*_{IN}$ has sufficient randomness given $S^*$ (in particular, $p^*$ and $\tau^*_{\mathrm{cmt}}$). This captures scenarios where the randomness is introduced in the key-establishment protocol, e.g., $K^*_{IN}$ is computed with Diffie-Hellman key exchange with the adversary interacting with an honest third party. We remark that we do not make any assumption on the running of the key-establishment protocol beyond the restriction that $K^*_{IN}$ has sufficient randomness given $S^*$, i.e., the adversary can behave maliciously during the protocol execution or can deviate from the protocol. Our approach to modeling an oracle-

dependent sampling process follows [CDKT19] to define adversarial sources of randomness for random number generators. We note that although Definition 2 mentions a key-establishment protocol, we even allow this to be absent, i.e., we allow the adversary to output *any* $K^*_{IN}$ with $\widetilde{\mathbf{H}}_\infty(K^*_{IN} \mid S^*) \geq \widetilde{k}$. This captures use cases of KDFs to, e.g., derive subkeys from an embedded entropy of a master keying material, which is given to a KDF as input. See Appendix B for examples of such use cases, including [asp, arm].

We emphasize that the requirement that $K^*_{IN}$ has entropy conditioned on $S^*$ is crucial, e.g., to prevent a trivial attack against the definition, where $\mathcal{A}_{\mathrm{cmt}}$ computes a KDF on any input, gets some $K^*_{OUT}$, then outputs a predicate $p^*$ that is satisfied for this particular $K^*_{OUT}$. We do not consider the case where $K^*_{IN}$ has low randomness, and we discuss this in Sect. 3.4.

An adversary wins the game if a preimage of one of the target images specified by $p^*$ is found in the find stage. We use a predicate to describe $\mathcal{T}$, however, this is not the only way to represent the set of target images. The advantage of using a predicate is that a predicate $p^*$ with a compact description can describe a large set, and be output by an efficient adversary, which otherwise cannot output a set description of the same large set.

We remark that the adversary has full control over the context, while this is supposed to include specific information on the derived keying material. This follows the attacks against CTR-CMAC in [Nat22, Che22] (reviewed in Appendix A), and this treatment makes our provable security results strong.

We also remark the multi-user nature of Definition 1. In real use cases, multiple parties can be involved in a key-establishment protocol to share $K^*_{IN}$, and a subset of the parties may attempt to manipulate the output of the KDF. Assume that $a$ parties $\mathcal{A}_1, \ldots, \mathcal{A}_a$ are malicious. Our KC adversary $\mathcal{A}$ in Definition 2 can be a collection of all the corrupted parties $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_a)$, and if $\mathcal{A}_i$ has its own predicate $p_i$, we define $p^*$ as $p^* = p_1 \vee \cdots \vee p_a$, and hence Definition 3 naturally captures such cases.

Finally, the above definition captures the information-theoretic setting. In the computational setting, we consider $(\Gamma, t)$-KC adversary $\mathcal{A}$ that runs in time $t$. In this setting, $\mathcal{A}$ is given the description of the KDF, and $\mathcal{A}$ returns $(K^*_{IN}, S^*)$ as the output of the commit stage. The adversary then returns a context $\mathit{Context}^*$ as the output of the find stage, where the running time of the adversary is at most $t$, and this includes the time complexity of describing the predicate.

### 3.2 Rationale of the Definition

In SP 800-108 [Che22], KC security is informally defined as in Definition 1. Intuitively, this requires that, with our notation, given the key derivation key $K^*_{IN}$ and some pre-selected value $K^*_{OUT}$, it is infeasible for the adversary to find $\mathit{Context}^*$ such that $K^*_{OUT} = \mathrm{KDF}(K^*_{IN}, \mathit{Label}, \mathit{Context}^*, L)$, where *Label* and $L$ are typically defined in the protocol that runs KDF. This is the preimage security of KDF under the known-key setting (as $K^*_{IN}$ is given), where a part of the preimage is fixed by the protocol (as *Label* and $L$ are fixed), and the adversary can pre-select *one* target image $K_{OUT}$. We point out that the above intuition is insufficient to capture the security requirement of KC security. Assume that the output keying material $K_{OUT}$ consists of $2n$ bits, where $n$ denotes a block length (e.g., $n = 128$). The above intuition requires the infeasibility of finding a preimage for a $2n$-bit target image $K_{OUT}$. However, it does not rule out a possibility that the adversary can still find a preimage for $K_{OUT}$ where the first $n$ bits of $K_{OUT}$ is $0^n$, i.e., $K_{OUT} = 0^n \,\|\, K'$ for some $n$-bit $K'$. This forms a class of *weak keys* of size $2^n$ out of $2^{2n}$ space of the keying material for Even-Mansour cipher [EM97], allowing a third party to find the remaining $n$-bit key $K'$ in a simple and efficient way. Concretely, consider CTR[CMAC], and let $K^*_{OUT} = (0^n \,\|\, K') \in \{0,1\}^{2n}$. Given $K^*_{IN}$ and *Label*, the adversary can

compute $Context^*$ such that $0^n = \mathtt{msb}_n(\mathrm{CTR[CMAC]}(K^*_{IN}, Label, Context^*, 2n))$ following the attack in [Nat22, Che22] and Appendix A. Then $K'$ is fixed from the computed $Context^*$ and is not under the control of the adversary. As a result, the adversary can efficiently compute $Context^*$ that is consistent with $K^*_{OUT} = 0^n \,\|\, K'$.

It turns out that the formalization of KC security requires one to rule out such adversaries that can potentially specify *a set of target images*, which can be an exponentially large set, and we require that the adversary is unable to find a preimage for any of these target images.

As we will see in Sect. 3.3, there are various types of weak key classes, and we use a predicate in order to allow the adversary to flexibly select a set of target images. Note that the value of $\delta$ in Definition 2 is the expectation of the probability of finding a keying material $K^*_{OUT}$ such that $p^*(K^*_{OUT}) = 1$ with a random guess of $K^*_{OUT}$, i.e., $\delta$ is the expectation of $|\mathcal{T}|/2^{L^*}$. Our definition allows $|\mathcal{T}|$ to be exponentially large, however, this still has to be small compared to $2^{L^*}$. We do not cover the case where $\delta$ is non-negligible, since such a predicate holds with a non-negligible probability even with an ideally secure KDF with a random guess of the input. In other words, all key derivation functions can output a weak keying material if the set of weak keys is non-negligibly large.

### 3.3 Weak Keys in Standards

In addition to the aforementioned Even-Mansour cipher, we identify some practical cryptographic schemes with large weak key classes. The first example is XTS-AES, a widely-deployed standard for storage encryption (IEEE 1619.2 [IEE] and NIST SP 800-38E [Dwo10]), and is also used as a protection mechanism inside modern CPUs such as AMD EPYC[5]. It uses two AES keys, either 128 or 256 bits, and if they collide, XTS reduces to XEX [Rog04]. XTS with colliding keys is broken since the original XTS uses a block index starting from zero, while XEX must avoid block index being zero. Refer to Minematsu [Min06] for the attack. This means that $K_{OUT} = K[1] \,\|\, K[2] \in \{0,1\}^k \times \{0,1\}^k$ with $K[1] = K[2]$ is a weak key for XTS using AES-$k$ for $k \in \{128, 256\}$, forming a large set of weak keys. See Appendix F for the analysis of this type of weak keys.

Moreover, several MAC modes defined in ISO/IEC 9797-1:2011 [ISOb] need multiple block cipher keys and have large weak key classes based on key collision. Concretely, MAC Algorithms 3 and 6 have two keys, $K$ and $K'$, and a collision $K = K'$ reduces them to the plain CBC-MAC hence are broken by length extension attack (see e.g. Iwata and Wang [IW14] for the algorithms). In general, cryptographic schemes using multiple independent keys potentially have weak keys based on a key collision. Generic compositions for authenticated encryption [BN00, NRS14], including one in ISO/IEC 19772:2020 [ISOa], are typical examples. Triple DES is also known to have large weak key classes defined by key collisions that reduces TDES to a single DES. Either the first and second or the second and the third key collisions yield a weak key, resulting in about $2^{112}$ weak keys among 168-bit key space (although these weak keys are not allowed by NIST [BM17]). Even modern block ciphers can have large weak keys, e.g., Midori-64 [BBI+15] which has $2^{32}$ weak keys [GJN+16] or $2^{64}$ weak keys [TLS16] in 128-bit key space.

Given the examples above, instead of focusing on *one* target image $K_{OUT}$, we require the infeasibility of computing $Context^*$ for a set of target images that can potentially be exponentially large. That is, KC security is the preimage security of KDF under the known-key setting, where a part of the preimage is fixed, and the adversary can pre-select *a set of target images*, and our definition substantiates this interpretation.

[5] `https://www.amd.com/system/files/documents/4th-gen-epyc-processor-architecture-white-paper.pdf`

### 3.4 Chosen-$K_{IN}$ Setting

Our Definition 2 is defined with the goals to minimize the dependencies to a key-establishment protocol and to give the adversary strong abilities (choice of $p^*$, and full control over *Context*$^*$ and other parameters). The assumption, sufficient min-entropy in $K_{IN}$, does not capture the setting where $K_{IN}$ is chosen by the adversary. The key transport [BCR+19] is an example where the chosen-$K_{IN}$ setting is relevant. In this two-party key exchanage protocol, one party, a sender, chooses $K_{IN}$ and sends it to the receiver with a public key encryption scheme. In a sender malicious setting, this gives the adversary to *choose* $K_{IN}$.

As an example, the most basic key transport [BCR+19, Sect. 9] works as follows:

1. A sender chooses $K_{IN}$, and obtains the public key of a receiver.
2. The sender computes the ciphertext $C$ of $K_{IN}$ with the public key, and sends $C$ to the receiver.
3. The receiver decrypts $C$ to obtain $K_{IN}$.

Then a KDF is run on both sides to have $K_{OUT} = \mathrm{KDF}(K_{IN}, \mathit{Label}, \mathit{Context}, L)$. Now [BCR+19, Sect. 5.5] outlines the content of *Context*, and this includes, e.g., a salt, initialization value, information related to the sender and receiver, a certificate of the public key, nonces, counters, pre-shared secret data, and so on. It is conceivable that all these values are known to the sender when choosing $K_{IN}$, in which case the sender can compute $K_{OUT} = \mathrm{KDF}(K_{IN}, \mathit{Label}, \mathit{Context}, L)$ locally at the time of sending $C$ to the receiver, and declare that $K_{OUT}$ is the pre-selected value to win the key control game[6]. That is, in a basic key transport, if the sender is malicious, the key control security cannot be expected.

Technically, Definition 2 covers chosen-$K_{IN}$ setting by letting $\widetilde{k} = 0$. However, no meaningful security is achievable in this case, as the adversary can trivially win the game by selecting a predicate with $p^*(K_{OUT}) = 1$ with locally computed $K_{OUT}$. This is consistent with the example of the basic key transport above.

However, in practice, a key transport process includes various randomness in the transcript. For instance, TCP Authentication Option (TCP-AO, RFC 5925 and 5926 [TBM10, RL10]) includes random nonces of both sender and receiver in the input of a KDF, and GlobalPlatform's Card Technology Secure Channel Protocol [gp] includes the random challenges of the host and recipient, i.e., with the notation of Definition 3, randomness is expected in $\tau^*_{\mathrm{cmt}}$, aux$^*$, or *Context*$^*$, and in practice, they could be used as a source of security even in the chosen-$K_{IN}$ setting. We leave the formalization and analysis of this direction as a future work, since such formalization depends on the details of the key transport, and the result of the analysis depends on the various subtleties how the transcript is incorporated into a KDF.

Finally, we remark that none of the results of the paper implies the insecurity of key transport protocols used in practice.

## 4 Analysis of KMAC-based KDF

KDF-KMAC employs KMAC as a PRF, where Keccak[256] or Keccak[512] [BDPA13] is used as the underlying extendable-output function (XOF) [Nat15, BDPA08].

[6] We remark that whether $K_{OUT}$ forms a weak key (class) of some cryptographic algorithm is not relevant here. If the winning condition is not selected by the adversary, then it can be analyzed within the context of preimage resistance [RS04] or multi-target preimage resistance [HRS16] (or close variants of them), outside of KC security.

More generally, for a XOF $H : \{0,1\}^* \times \mathbb{N} \to \{0,1\}^*$, the KMAC$[H] : \mathcal{K}_{IN} \times \mathcal{C} \times \mathbb{N} \times \mathcal{L} \to \mathcal{K}_{OUT}$ function takes $(K, \mathit{Context}, L, \mathit{Label})$ as input and returns $T = \mathtt{msb}_L(H(M, L))$, where $(M, L) = \mathsf{enc}(K, \mathit{Label}, \mathit{Context}, L)$ for an injective encoding $\mathsf{enc}$. We omit the details of the encoding (which can be found in [Che22, Nat15]) as it is irrelevant for the following analysis. The KDF-KMAC$[H] : \mathcal{K}_{IN} \times \mathcal{L} \times \mathcal{C} \times \mathbb{N} \to \mathcal{K}_{OUT}$ key derivation function takes $(K, \mathit{Label}, \mathit{Context}, L)$ as input and returns KMAC$[H](K, \mathit{Context}, L, \mathit{Label})$ as output.

We prove the KC security of KDF-KMAC$[H]$ in the random oracle model, i.e., $H$ is modeled as a random oracle. Additionally, we assume that the length of the KDF key is fixed.

**Theorem 1.** *Fix some* $\Gamma = (q_{\mathrm{cmt}}, q_{\mathrm{fnd}}, b_{\min}, b_{\max}, L_{\min}, L_{\max}, \widetilde{k}, \delta)$. *Let* KDF $=$ KMAC$[H]$, *where* $H : \{0,1\}^* \times \mathbb{N} \to \{0,1\}^*$ *is a random oracle. Then, for any* $\Gamma$*-KC adversary* $\mathcal{A}$, KDF *is* $\epsilon$*-KC secure for* $\epsilon(\Gamma) = q_{\mathrm{cmt}} 2^{-\widetilde{k}} + \overline{q_{\mathrm{fnd}}}\delta$, *where* $\overline{q_{\mathrm{fnd}}} = q_{\mathrm{fnd}} + 1$.

*Proof.* Without loss of generality, we assume that $\mathcal{A}_{\mathrm{cmt}}$ makes exactly $q_{\mathrm{cmt}}$ queries to $H$, and $\mathcal{A}_{\mathrm{fnd}}$ makes exactly $q_{\mathrm{fnd}}$ queries to $H$. Let $(K^*, S^*)$ denote the output of $\mathcal{A}_{\mathrm{cmt}}$, where $S^* = (\mathit{Label}^*, L^*, p^*, \tau^*_{\mathrm{cmt}}, \mathrm{aux}^*)$. Let $\mathit{Context}^*$ denote the output of $\mathcal{A}_{\mathrm{fnd}}$.

Let $\tau^*$ denote the combined history of commit and find queries. Let $q_{\mathrm{tot}} = q_{\mathrm{cmt}} + q_{\mathrm{fnd}}$. Since find queries always follow commit queries, we can write $\tau^* = (\tau^*_1, \ldots, \tau^*_{q_{\mathrm{tot}}})$, where the first $q_{\mathrm{cmt}}$ indices correspond to the commit queries, and the remaining $q_{\mathrm{fnd}}$ indices correspond to the find queries.

For any $i \in [1..q_{\mathrm{tot}}]$, $\tau^*_i$ is said to be *useful* if it is of the form $(M^*_{C_i}, L^*, T^*_{C_i})$ where $(M^*_{C_i}, L^*) = \mathsf{enc}(K^*, \mathit{Label}^*, C_i, L^*)$ and $T^*_{C_i} = H(M^*_{C_i}, L^*)$ for some $C_i \in \mathcal{C}$. Without loss of generality, we assume that all the find queries are useful, as any other form of find queries would be pointless in $\mathcal{A}_{\mathrm{fnd}}$'s pursuit. In particular, we assume that $\mathcal{A}_{\mathrm{fnd}}$ makes a (possibly repeated) query $(M^*_{\mathit{Context}^*}, L^*)$ at the end of the find stage, but before it outputs $\mathit{Context}^*$. Let $\mathcal{J} \subseteq [1..q_{\mathrm{tot}}]$ denote the set of indices of all the useful queries. Define the events:

$$\begin{aligned} \mathtt{bad}_1 &: \text{ there exists } i \in [1..q_{\mathrm{cmt}}] \text{ such that } \tau^*_i \text{ is useful.} \\ \mathtt{bad}_2 &: \text{ there exists } i \in \mathcal{J} \text{ such that } p^*(T^*_{C_i}) = 1. \end{aligned}$$

Clearly, $\mathtt{bad}_2$ occurs whenever KCS$(\mathcal{A}^H)$ occurs. Thus, we have

$$\begin{aligned} \Pr\left[\mathrm{KCS}(\mathcal{A}^H) = 1\right] &= \Pr\left[\mathtt{bad}_2\right] \\ &= \sum_{S \in \mathsf{Supp}(S^*)} \Pr\left[\mathtt{bad}_2 \mid S^* = S\right] \times \Pr\left[S^* = S\right], \end{aligned} \tag{1}$$

where $\mathsf{Supp}(S^*)$ denotes the support of $S^*$. Next, we have

$$\Pr\left[\mathtt{bad}_2 \mid S^* = S\right] \le \Pr\left[\mathtt{bad}_1 \mid S^* = S\right] + \Pr\left[\mathtt{bad}_2 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right]. \tag{2}$$

Let $S = (\mathit{Label}, L, p, \tau_{\mathrm{cmt}}, \mathrm{aux})$. Then, we have:

1. $\underline{\Pr\left[\mathtt{bad}_1 \mid S^* = S\right]}$: For any $i \in [1..q_{\mathrm{cmt}}]$, the probability of $(M^*_{C_i}, L) = \mathsf{enc}(K^*, \mathit{Label}, C_i, L)$ is upper bounded by $2^{-\mathbf{H}_\infty(K^* \mid S^* = S)}$. Then, the union bound gives

$$\Pr\left[\mathtt{bad}_1 \mid S^* = S\right] \le q_{\mathrm{cmt}} 2^{-\mathbf{H}_\infty(K^* \mid S^* = S)}. \tag{3}$$

2. $\underline{\Pr\left[\mathtt{bad}_2 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right]}$: On account of $\neg\mathtt{bad}_1$, we know that $\mathcal{J} \subseteq [q_{\mathrm{cmt}} + 1..q_{\mathrm{tot}}]$, i.e., we only have to consider the find queries. For any $i \in \mathcal{J}$, $T^*_{C_i} = H(M^*_{C_i}, L^*)$ is uniformly distributed over $\{0,1\}^{L^*}$ given the history up until this point. Thus, conditioned on $\neg\mathtt{bad}_1$, $p^*(T^*_{C_i})$ is identically distributed to $p^*(U_{L^*})$ given all the previous queries, and the latter

coincides with $p(U_L)$ when we condition further on $S^* = S$. A union bound over all $i \in \mathcal{J}$ yields

$$\begin{aligned}\Pr\left[\mathtt{bad}_2 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] &\leq \sum_{i\in\mathcal{J}} \Pr\left[p^*(T^*_{C_i}) = 1 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] \\ &= \sum_{i\in\mathcal{J}} \Pr\left[p(U_L) = 1 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] \\ &= \sum_{i\in\mathcal{J}} \Pr\left[p(U_L) = 1\right] \leq \overline{q_{\mathrm{fnd}}}\delta_{p,L}\,, \end{aligned} \tag{4}$$

since $|\mathcal{J}| \leq \overline{q_{\mathrm{fnd}}}$ and $U_L$ is independent of $(K^*, S^*)$.

Finally, Eqs. (1)–(4) and $\widetilde{\mathbf{H}}_\infty(K^* \mid S^*) \geq \widetilde{k}$ gives

$$\begin{aligned}\Pr\left[\mathrm{KCS}(\mathcal{A}^H) = 1\right] &\leq \sum_{S\in\mathsf{Supp}(S^*)} \left(\frac{q_{\mathrm{cmt}}}{2^{\mathbf{H}_\infty(K^*|S^*=S)}} + \overline{q_{\mathrm{fnd}}}\delta_{p,L}\right) \times \Pr\left[S^* = S\right] \\ &\leq \mathop{\mathbb{E}}_{S^*\xleftarrow{\$}\mathcal{A}^H_{\mathrm{cmt}}} \left[\frac{q_{\mathrm{cmt}}}{2^{\mathbf{H}_\infty(K^*|S^*=S)}}\right] + \mathop{\mathbb{E}}_{S^*\xleftarrow{\$}\mathcal{A}^H_{\mathrm{cmt}}} \left[\overline{q_{\mathrm{fnd}}}\delta_{p^*,L^*}\right] \\ &\leq \frac{q_{\mathrm{cmt}}}{2^{\widetilde{k}}} + \overline{q_{\mathrm{fnd}}}\delta\,. \end{aligned}$$

□

# 5 Analysis of HMAC-based KDFs

Fix some $hlen_H, B_H \in \mathbb{N}$. Let $H : \{0,1\}^* \to \{0,1\}^h$ be a hash function, where $h = 8hlen_H$. Fix some distinct $\mathsf{ipad}, \mathsf{opad} \in \{0,1\}^{8B_H}$. For any $K \in \{0,1\}^{8B_H}$, we define $K_\top := K \oplus \mathsf{ipad}$ and $K_\bot := K \oplus \mathsf{opad}$. The HMAC$[H] : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^h$ function takes $(K, M)$ as input and returns

$$T := H(K'_\bot \,\|\, H(K'_\top \,\|\, M))\,,$$

where $K' = K$ when $|K| = 8B_H$, $K' = H(K) \,\|\, 0^{8B_H - h}$ when $|K| > 8B_H$, and $K' = K \,\|\, 0^{8B_H - |K|}$ when $|K| < 8B_H$. See Fig. 1. In practice, $M$ is restricted to satisfy $0 \leq |M| < 2^{B_H} - 8B_H$, and $H$ is instantiated with either SHA-256 or SHA-512, with $B_H = 64$ in the former case and $B_H = 128$ in the latter case, covering a wide range of use cases.

## 5.1 Shared Proof Framework and Notations

To avoid redundancy in the upcoming proofs, we outline the high-level approach, along with the common notations and definitions shared by all three proofs.

We prove the KC security of CTR-HMAC$[H]$, FB-HMAC$[H]$, and DP-HMAC$[H]$ in the random oracle model, i.e., $H : \{0,1\}^* \to \{0,1\}^h$ is modeled as a random oracle. Additionally, we assume that $K_{IN}$ is fixed to $8B_H$ bits. Without loss of generality, we assume that $\mathcal{A}_{\mathrm{cmt}}$ and $\mathcal{A}_{\mathrm{fnd}}$ make exactly $q_{\mathrm{cmt}}$ and $q_{\mathrm{fnd}}$ queries to $H$, respectively. We denote the output of $\mathcal{A}_{\mathrm{cmt}}$ by $(K^*, S^*)$ and the output of $\mathcal{A}_{\mathrm{fnd}}$ by $\mathit{Context}^*$, where $S^* = (\mathit{Label}^*, L^*, p^*, \tau^*_{\mathrm{cmt}}, \mathrm{aux}^*)$. Additionally, we define $s^* = \lceil L^*/h \rceil$. For any $C \in \mathcal{C}$, it would be convenient to define $M^*_C := \mathit{Label}^* \,\|\, 0^8 \,\|\, C \,\|\, L^*$. A query to $H$ is considered *useful* if it is of the form $K \,\|\, X$, where $K \in \{K^*_\top, K^*_\bot\}$ and $|X| \geq 0$. Without loss of generality, we assume that all of the find queries are useful, as any other form of find queries would be irrelevant to $\mathcal{A}_{\mathrm{fnd}}$'s objective.

**High-level proof overview.** For convenience, across all three security proofs, we augment the oracle to maintain a *history*, denoted by $\tau^*$, and preemptively compute the KDF output for some $C \in \mathcal{C}$ whenever $\mathcal{A}_{\text{fnd}}$ makes an appropriate *context-revealing* find query (which will be specified individually for each proof).

In particular, the augmented oracle can make $O(s^*)$ queries to $H$ before responding to a useful find query. Therefore, the augmented oracle makes at most $O(q_{\text{fnd}} s^*)$ queries to $H$. As before, we write $\tau^* = (\tau_1^*, \ldots, \tau_{\text{tot}}^*)$, where the first $q_{\text{cmt}}$ indices correspond to the commit queries, and the remaining $q_{\text{tot}} - q_{\text{cmt}}$ indices correspond to the queries made to $H$ in the find stage. For any $i \in [1..q_{\text{tot}}]$, if $\tau_i^*$ is useful, we represent it generically as $(K_i^* \,\|\, X_i^*, Y_i^*)$.

We also define certain bad events on the history $\tau^*$ and demonstrate that, provided these bad events do not occur, the probability that the KDF output satisfies $p^*$ remains within a $q_{\text{fnd}}$-factor of the ideal probability.

### 5.2 Analysis of CTR-HMAC$[H]$

**Theorem 2.** *Fix $k = |K_{IN}| = 8B_H$ and some $\Gamma = (q_{\text{cmt}}, q_{\text{fnd}}, b_{\min}, b_{\max}, L_{\min}, L_{\max}, \widetilde{k}, \delta)$. Let* KDF $=$ CTR-HMAC$[H]$ *where $H : \{0,1\}^* \to \{0,1\}^h$ is a random oracle. Then, for any $\Gamma$-KC adversary $\mathcal{A}$,* KDF *is $\epsilon$-KC secure for*

$$\epsilon(\Gamma) = \frac{q_{\text{cmt}}}{2^{\widetilde{k}}} + \frac{\overline{q_{\text{fnd}}} s_{\max}^2 (q_{\text{fnd}} + 2)}{2^h} + \overline{q_{\text{fnd}}}\delta\,,$$

*where $\overline{q_{\text{fnd}}} = q_{\text{fnd}} + 1$ and $s_{\max} = \lceil L_{\max}/h \rceil$.*

*Proof.* For any $C \in \mathcal{C}$ and any $j \in [1..s^*]$, let $X_C^*[j] \coloneqq \langle j \rangle_r \,\|\, M_C^*$. Any useful query $K \,\|\, X$ to $H$ is said to be:

- A $(C, j)$-*upper query* if $K \,\|\, X = K_\top^* \,\|\, X_C^*[j]$, for some $(C, j) \in \mathcal{C} \times [1..s^*]$.
- A *lower query* if $K = K_\perp^*$.

For any $C \in \mathcal{C}$, the $(C, j)$-upper query is said to be *context-sharing* with a $(C', j')$-upper query if $C' = C$ and $j \neq j'$. An upper query will be referred to as a *context-revealing* query if the history does not contain a corresponding context-sharing find query. The augmented oracle works as follows:

1. All the commit queries are relayed directly to the random oracle $H$, and its output is relayed back to $\mathcal{A}_{\text{cmt}}$ after they are recorded in $\tau^*$.
2. When the $i$-th find query is a context-revealing query, denoted $K_\top^* \,\|\, X_{C_i}^*[j]$, the KDF is evaluated as follows (also see Fig. 7):
   (a) For all $j' \in [1..s^*]$: whenever an entry is not in $\tau^*$, $H$ is queried with $K_\top^* \,\|\, X_{C_i}^*[j']$ and $(K_\top^* \,\|\, X_{C_i}^*[j'], R_{C_i}^*[j'])$ is stored in $\tau^*$, where $R_{C_i}^*[j']$ denotes the corresponding outputs.
   (b) For all $j' \in [1..s^*]$: whenever an entry is not in $\tau^*$, $H$ is queried with $K_\perp^* \,\|\, R_{C_i}^*[j']$ and $(K_\perp^* \,\|\, R_{C_i}^*[j'], T_{C_i}^*[j'])$ is stored in $\tau^*$, where $T_{C_i}^*[j']$ denotes the corresponding outputs. Let $T_{C_i}^* \coloneqq \mathtt{msb}_L(T_{C_i}^*[1] \,\|\, \ldots \,\|\, T_{C_i}^*[s^*])$.
   (c) Finally, $R_{C_i}^*[j]$ is returned to $\mathcal{A}_{\text{fnd}}$.
3. All other find queries are relayed directly to the random oracle $H$, and the outputs are relayed back to $\mathcal{A}_{\text{fnd}}$ after they are recorded in $\tau^*$.

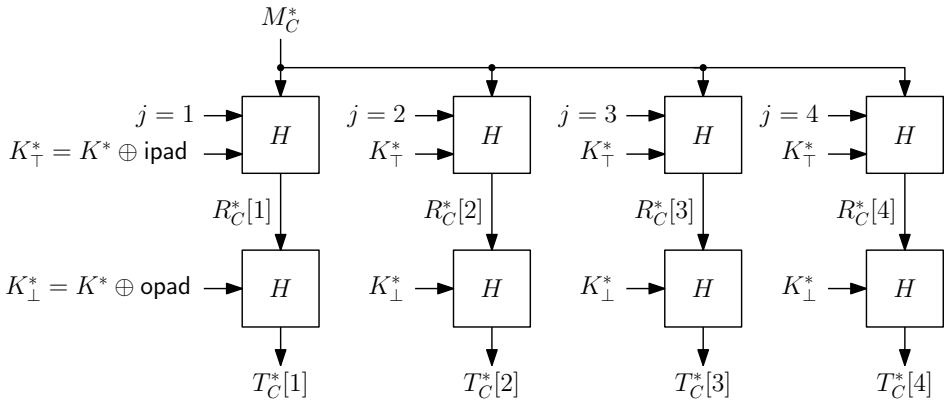


Figure 7: CTR-HMAC with $L^* = 4h$ and $s^* = 4$. $M^* = (\mathit{Label}^* \,\|\, 0^8 \,\|\, C \,\|\, L^*)$ and $K_{OUT} = T_C^* = (T_C^*[1] \,\|\, \cdots \,\|\, T_C^*[4])$.

It is clear that $q_{\mathrm{tot}} \le q_{\mathrm{cmt}} + 2s^* q_{\mathrm{fnd}}$. Let $\mathcal{I}_\top, \mathcal{I}_\bot \subseteq [1..q_{\mathrm{tot}}]$ denote the sets of all upper and lower find queries, respectively. Let $\mathcal{J} \subseteq [1..q_{\mathrm{tot}}]$ denote the set of all context-revealing queries.

Any pair $(i, i') \in \mathcal{I}_\top \times \mathcal{I}_\bot$ is said to be a $(C, j)$-*chain* (or simply a *chain*) if $X_i^* = X_C^*[j]$ and $X_{i'}^* = Y_i^*$. A context $C$ is said to be *complete* if the $(C, j)$-chain exists in $\tau^*$, for each $j \in [1..s^*]$. By definition of HMAC[$H$], if $\mathcal{A}_{\mathrm{fnd}}$ can compute $\mathrm{KDF}(K^*, \mathit{Label}^*, C, L^*)$ for some $C$, then $C$ is complete. In particular, all context-revealing queries give a complete context. See Fig. 7 for an example. We assume that $\mathit{Context}^*$ is complete by providing $\mathcal{A}_{\mathrm{fnd}}$ an additional context-revealing query at the end. We define the events:

$\mathtt{bad}_1$ : there exists $i \in [1..q_{\mathrm{cmt}}]$ such that $\tau_i^*$ is useful.
$\mathtt{bad}_2$ : there exists context-sharing $i, i' \in \mathcal{I}_\top$ such that $Y_i^* = Y_{i'}^*$.
$\mathtt{bad}_3$ : there exists $i \in \mathcal{I}_\top$ and $i' \in \mathcal{I}_\bot$ such that $i' < i$ and $Y_i^* = X_{i'}^*$.
$\mathtt{bad}_4$ : there exists $i \in \mathcal{J}$ such that $p^*(T_{C_i}^*) = 1$.

If the $(\mathit{Context}^*, j)$-chain contains a commit query for some $j \in [1..s^*]$, then $\mathtt{bad}_1$ occurs. On the other hand, when $\neg\mathtt{bad}_1$ holds, the commit phase history becomes irrelevant for $\mathcal{A}_{\mathrm{fnd}}$. Moreover, $\neg\mathtt{bad}_1$ also ensures that $C_i$ is completed during the find phase for each $i \in \mathcal{I}_\top$. Finally, $\mathrm{KCS}(\mathcal{A}^H) \wedge \neg\mathtt{bad}_1$ implies that $\mathtt{bad}_4$ takes place. The event $\neg(\mathtt{bad}_1 \vee \mathtt{bad}_2 \vee \mathtt{bad}_3)$ ensures the freshness of $R_{C_i}^*[j]$ for all $i \in \mathcal{J}$ and $j \in [1..s^*]$. The aforementioned discussion implies that $(\mathtt{bad}_1 \vee \mathtt{bad}_4)$ occurs whenever $\mathrm{KCS}(\mathcal{A}^H)$ does. Thus, by definition, we have

$$\Pr\left[\mathrm{KCS}(\mathcal{A}^H) = 1\right] \le \Pr\left[\mathtt{bad}_1 \vee \mathtt{bad}_4\right] \le \Pr\left[\mathtt{bad}_1\right] + \Pr\left[\mathtt{bad}_4 \mid \neg\mathtt{bad}_1\right]. \tag{5}$$

Using similar arguments as used in the proof of Theorem 1 gives

$$\Pr\left[\mathtt{bad}_1\right] \le \frac{q_{\mathrm{cmt}}}{2^{\widetilde{k}}}. \tag{6}$$

Next, we have

$$\Pr\left[\mathtt{bad}_4 \mid \neg\mathtt{bad}_1\right] = \sum_{S \in \mathsf{Supp}(S^*)} \Pr\left[\mathtt{bad}_4 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] \times \Pr\left[S^* = S\right]. \tag{7}$$

We are interested in the conditional probability of $\mathtt{bad}_4$ given $\neg\mathtt{bad}_1 \wedge S^* = S$. Let $\mathtt{bad}_{23} \coloneqq \mathtt{bad}_2 \vee \mathtt{bad}_3$ and $\mathtt{bad}_{123} \coloneqq \mathtt{bad}_1 \vee \mathtt{bad}_{23}$. First, the law of total probability gives

$$\Pr\left[\mathtt{bad}_4 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] \le \Pr\left[\mathtt{bad}_{23} \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] + \Pr\left[\mathtt{bad}_4 \mid \neg\mathtt{bad}_{123} \wedge S^* = S\right]$$

$$\leq \Pr\left[\mathtt{bad}_2 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] + \Pr\left[\mathtt{bad}_3 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] + \Pr\left[\mathtt{bad}_4 \mid \neg\mathtt{bad}_{123} \wedge S^* = S\right]. \quad (8)$$

Let $S = (\mathit{Label}, L, p, \tau_{\mathrm{cmt}}, \mathrm{aux})$, and $s = \lceil L/h \rceil$. We can bound the three terms on the RHS as follows:

1. $\Pr\left[\mathtt{bad}_2 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right]$: Since $\neg\mathtt{bad}_1$ holds, it is sufficient to concentrate on the find queries. Fix some context-sharing $i, i' \in \mathcal{I}_\top$. Since $X_i^* \neq X_{i'}^*$, we note that $Y_i^*$ is independent of $Y_{i'}^*$ and uniformly distributed over $\{0,1\}^h$, whence by taking a union bound over all such pairs of context-sharing upper find queries (at most $\overline{q_{\mathrm{fnd}}} s^2$), we have
$$\Pr\left[\mathtt{bad}_2 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] \leq \frac{\overline{q_{\mathrm{fnd}}} s^2}{2^h} \leq \frac{\overline{q_{\mathrm{fnd}}} s_{\max}^2}{2^h}. \quad (9)$$

2. $\Pr\left[\mathtt{bad}_3 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right]$: In light of $\neg\mathtt{bad}_1$, we only consider the find queries. Fix any $i \in \mathcal{I}_\top$ and $i' \in \mathcal{I}_\bot$. Since $K_\top^* \neq K_\bot^*$, conditioned on all the previous queries, $Y_i^*$ is uniformly distributed over $\{0,1\}^h$. Thus, the probability that $Y_i^* = X_{i'}^*$ is upper bounded by $2^{-h}$. Since $|\mathcal{I}_\top|, |\mathcal{I}_\bot| \leq \overline{q_{\mathrm{fnd}}} s$, the union bound gives
$$\Pr\left[\mathtt{bad}_3 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] \leq \frac{\overline{q_{\mathrm{fnd}}}^2 s^2}{2^h} \leq \frac{\overline{q_{\mathrm{fnd}}}^2 s_{\max}^2}{2^h}. \quad (10)$$

3. $\Pr\left[\mathtt{bad}_4 \mid \neg\mathtt{bad}_{123} \wedge S^* = S\right]$: On account of $\neg\mathtt{bad}_1$, we must have $\mathcal{J} \subseteq [q_{\mathrm{cmt}}+1..q_{\mathrm{tot}}]$. Fix some $i \in \mathcal{J}$. $\neg\mathtt{bad}_1$ implies that $C_i$ is complete, $\neg\mathtt{bad}_3$ implies that the lower queries only appear in $\tau^*$ after the corresponding context-sharing upper queries, and $\neg\mathtt{bad}_2$ implies that the said lower queries are all distinct. Thus, $T_{C_i}^*$ is uniformly distributed over $\{0,1\}^L$, given the previous history. Then, conditioned on $\neg\mathtt{bad}_1$ and $S^* = S$, $p^*(T_{C_i}^*)$ is identical in distribution to $p(U_L)$ given all previous queries. Finally, a union bound over all $i \in \mathcal{J}$ (at most $\overline{q_{\mathrm{fnd}}}$) yields
$$\begin{aligned}\Pr\left[\mathtt{bad}_4 \mid \neg\mathtt{bad}_{123} \wedge S^* = S\right] &\leq \sum_{i\in\mathcal{J}} \Pr\left[p^*(T_{C_i}^*) = 1 \mid \neg\mathtt{bad}_{123} \wedge S^* = S\right] \\ &\leq \sum_{i\in\mathcal{J}} \Pr\left[p(U_L) = 1 \mid \neg\mathtt{bad}_{123} \wedge S^* = S\right] \\ &\leq \sum_{i\in\mathcal{J}} \Pr\left[p(U_L) = 1\right] = \overline{q_{\mathrm{fnd}}}\delta_{p,L}. \end{aligned} \quad (11)$$

To conclude the proof, using Eqs. (7)–(11), we have
$$\begin{aligned}\Pr\left[\mathrm{KCS}(\mathcal{A}^H) = 1\right] &\leq \frac{q_{\mathrm{cmt}}}{2^{\widetilde{k}}} + \mathop{\mathbb{E}}_{S^* \xleftarrow{\$} \mathcal{A}^H_{\mathrm{cmt}}}\left[\frac{\overline{q_{\mathrm{fnd}}} s_{\max}^2 (q_{\mathrm{fnd}}+2)}{2^h} + \overline{q_{\mathrm{fnd}}}\delta_{p^*,L^*}\right] \\ &\leq \frac{q_{\mathrm{cmt}}}{2^{\widetilde{k}}} + \frac{\overline{q_{\mathrm{fnd}}} s_{\max}^2 (q_{\mathrm{fnd}}+2)}{2^h} + \overline{q_{\mathrm{fnd}}}\delta, \end{aligned} \quad (12)$$

where the final inequality follows from the hypothesis. The result now follows by Eqs. (5), (6), and (12). □

### 5.3 Analysis of FB-HMAC[$H$]

Recall that FB-HMAC[$H$] has an initial value $IV \in \mathcal{IV}$ as part of the input. We focus on the case that the value of $IV$ is pre-fixed at the construction level and that $|IV| = h$.

**Theorem 3.** *Fix $k = |K_{IN}| = 8B_H$, some $IV \in \{0,1\}^h$, and some $\Gamma = (q_{\mathrm{cmt}}, q_{\mathrm{fnd}}, b_{\min}, b_{\max}, L_{\min}, L_{\max}, \widetilde{k}, \delta)$. Let* KDF = FB-HMAC[$H$] *where $H : \{0,1\}^* \to \{0,1\}^h$ is a random oracle. Then, for any $\Gamma$-KC adversary $\mathcal{A}$,* KDF *is $\epsilon$-KC secure for*

$$\epsilon(\Gamma) = \frac{3\overline{q_{\mathrm{fnd}}}^2 s_{\max}^2}{2^h} + \frac{\overline{q_{\mathrm{fnd}}} s_{\max}^2}{2^h} + \frac{q_{\mathrm{cmt}}}{2^{\widetilde{k}}} + \overline{q_{\mathrm{fnd}}}\delta\,,$$

*where $\overline{q_{\mathrm{fnd}}} = q_{\mathrm{fnd}} + 1$ and $s_{\max} = \lceil L_{\max}/h \rceil$.*

*Proof.* Any useful query $K \,\|\, X$ to $H$ is said to be:

- A $(C, I)$-*upper query* if $K \,\|\, X = K^*_\top \,\|\, (I \,\|\, M^*_C)$, for some $(C, I) \in \mathcal{C} \times \{0,1\}^*$.
- A *lower query* if $K = K^*_\bot$.

For any $C \in \mathcal{C}$, the $(C, I)$-upper query is said to be *context-sharing* with a $(C', I')$-upper query if $C' = C$ and $I \neq I'$. An upper query will be referred to as a *context-revealing* query if the history does not contain a corresponding context-sharing find query. The augmented oracle works as follows:

1. All the commit queries are relayed directly to the random oracle $H$, and its output is relayed back to $\mathcal{A}_{\mathrm{cmt}}$ after they are recorded in $\tau^*$.
2. When the $i$-th find query is a context-revealing query, denoted $K^*_\top \,\|\, T^*_i \,\|\, M^*_{C_i}$, the KDF is evaluated as follows (also see Fig. 8):
   (a) Let $T^*_{C_i}[0] = IV$.
   (b) For all $j \in [1..s^*]$:
      i. Whenever an entry is not in $\tau^*$, $H$ is queried with $K^*_\top \,\|\, T^*_{C_i}[j-1] \,\|\, M^*_{C_i}$ and $(K^*_\top \,\|\, T^*_{C_i}[j-1] \,\|\, M^*_{C_i}, R^*_{C_i}[j])$ is stored in $\tau^*$, where $R^*_{C_i}[j]$ denotes the corresponding output.
      ii. Whenever an entry is not in $\tau^*$, $H$ is queried with $K^*_\bot \,\|\, R^*_{C_i}[j]$ and $(K^*_\bot \,\|\, R^*_{C_i}[j], T^*_{C_i}[j])$ is stored in $\tau^*$, where $T^*_{C_i}[j]$ denotes the corresponding output.
   (c) Let $T^*_{C_i} \coloneqq \mathtt{msb}_L(T^*_{C_i}[1] \,\|\, \ldots \,\|\, T^*_{C_i}[s^*])$.
   (d) Whenever an entry is not in $\tau^*$, $H$ is queried with $K^*_\top \,\|\, T^*_i \,\|\, M^*_{C_i}$ and $(K^*_\top \,\|\, T^*_i \,\|\, M^*_{C_i}, Y^*_i)$ is stored in $\tau^*$, where $Y^*_i$ denotes the corresponding output.
   (e) Finally, $Y^*_i$ is returned to $\mathcal{A}_{\mathrm{fnd}}$.
3. All other find queries are relayed directly to the random oracle $H$, and the outputs are relayed back to $\mathcal{A}_{\mathrm{fnd}}$ after they are recorded in $\tau^*$.

It is clear that $q_{\mathrm{tot}} \leq q_{\mathrm{cmt}} + (2s^* + 1)q_{\mathrm{fnd}}$. Let $\mathcal{I}_\top, \mathcal{I}_\bot \subseteq [1..q_{\mathrm{tot}}]$ denote the sets of all upper and lower find queries, respectively. Let $\mathcal{J} \subseteq [1..q_{\mathrm{tot}}]$ denote the set of all context-revealing queries.

Any pair $(i, i') \in \mathcal{I}_\top \times \mathcal{I}_\bot$ is said to be a $(C, I, T)$-*chain* (or simply a *chain*) if $X^*_i = I \,\|\, M^*_C$, $X^*_{i'} = Y^*_i$, and $Y^*_{i'} = T$. A context $C$ is said to be *complete* if there exists a sequence of $T[0], T[1], \ldots, T[s^*] \in \{0,1\}^h$ such that $T[0] = IV$ and $(C, T[j-1], T[j])$-chain exists for all $j \in [1..s^*]$. By definition of HMAC[$H$], if $\mathcal{A}_{\mathrm{fnd}}$ can compute KDF$(K^*, \mathit{Label}^*, C, L^*)$ for some $C$, then $C$ is complete. In particular, all context-revealing queries give a complete context. See Fig. 8 for an example. In particular, we assume that $\mathit{Context}^*$ is complete by providing $\mathcal{A}_{\mathrm{fnd}}$ an additional context-revealing query at the end.

We define the following events:

$\mathtt{bad}_1$ : there exists $i \in [1..q_{\mathrm{cmt}}]$ such that $\tau^*_i$ is useful.

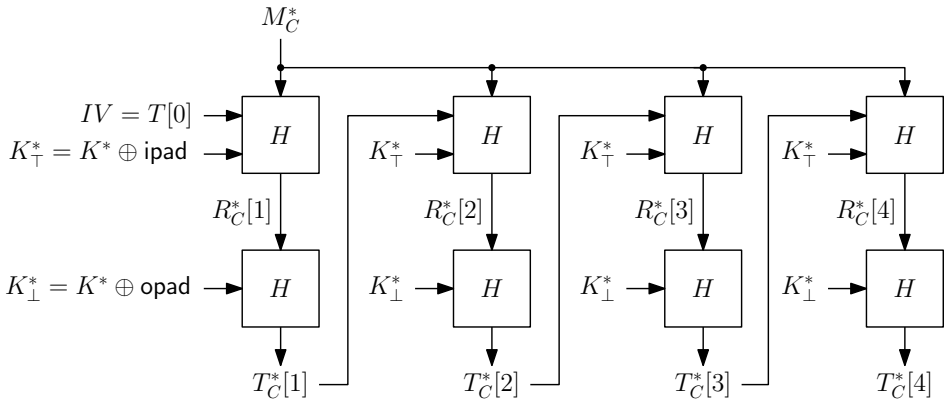


Figure 8: FB-HMAC with $L^* = 4h$ and $s^* = 4$. $M_C^* = (\mathit{Label}^* \,\|\, 0^8 \,\|\, C \,\|\, L^*)$ and $K_{OUT} = T_C^* = (T_C^*[1] \,\|\, \cdots \,\|\, T_C^*[4])$.

$\mathtt{bad}_2$ : there exists context-sharing $i, i' \in \mathcal{I}_\top$ such that $Y_i^* = Y_{i'}^*$.
$\mathtt{bad}_3$ : there exists $i \in \mathcal{I}_\top$ and an $i' \in \mathcal{I}_\bot$ such that $i' < i$ and $Y_i^* = X_{i'}^*$.
$\mathtt{bad}_4$ : there exists $i' \in \mathcal{I}_\top$ and an $i \in \mathcal{I}_\bot$ such that $i' < i$ and $Y_i^* \| \star = X_{i'}^*$.
$\mathtt{bad}_5$ : there exists $i \in \mathcal{J}$ such that $p^*(T_{C_i}^*) = 1$.

If the $(\mathit{Context}^*, I, T)$-chain contains a commit query for some $(I, T) \in \{0,1\}^* \times \{0,1\}^h$, then $\mathtt{bad}_1$ occurs. On the other hand, when $\neg\mathtt{bad}_1$ holds, the commit phase history becomes irrelevant for $\mathcal{A}_{\text{fnd}}$. Moreover, $\neg\mathtt{bad}_1$ also ensures that $C_i$ is completed during the find phase for each $i \in \mathcal{I}_\top$. The event $\text{KCS}(\mathcal{A}^H) \wedge \neg\mathtt{bad}_1$ implies that $\mathtt{bad}_5$ takes place. The event $\neg(\mathtt{bad}_2 \vee \mathtt{bad}_3 \vee \mathtt{bad}_4)$ ensures the freshness of $R_{C_i}^*[j]$ for all $i \in \mathcal{J}$ and $j \in [1..s^*]$. The aforementioned discussion implies that $(\mathtt{bad}_1 \vee \mathtt{bad}_5)$ occurs whenever $\text{KCS}(\mathcal{A}^H)$ does. Thus we have

$$\begin{aligned}\Pr\left[\text{KCS}(\mathcal{A}^H) = 1\right] &\leq \Pr\left[\mathtt{bad}_1 \vee \mathtt{bad}_5\right] \leq \Pr\left[\mathtt{bad}_1\right] + \Pr\left[\mathtt{bad}_5 \mid \neg\mathtt{bad}_1\right] \\ &\leq \frac{q_{\text{cmt}}}{2^{\widetilde{k}}} + \Pr\left[\mathtt{bad}_5 \mid \neg\mathtt{bad}_1\right] . \end{aligned} \tag{13}$$

Next, we have

$$\Pr\left[\mathtt{bad}_5 \mid \neg\mathtt{bad}_1\right] = \sum_{S \in \mathsf{Supp}(S^*)} \Pr\left[\mathtt{bad}_5 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] \times \Pr\left[S^* = S\right] . \tag{14}$$

We are interested in the conditional probability of $\mathtt{bad}_5$ given $\neg\mathtt{bad}_1 \wedge S^* = S$. Let $\mathtt{bad}_{1234} \coloneqq \mathtt{bad}_1 \vee \mathtt{bad}_2 \vee \mathtt{bad}_3 \vee \mathtt{bad}_4$. First, the law of total probability gives

$$\begin{aligned}\Pr\left[\mathtt{bad}_5 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] &\leq \Pr\left[\mathtt{bad}_2 \vee \mathtt{bad}_3 \vee \mathtt{bad}_4 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] \\ &\quad + \Pr\left[\mathtt{bad}_5 \mid \neg\mathtt{bad}_{1234} \wedge S^* = S\right] \\ &\leq \Pr\left[\mathtt{bad}_2 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] + \Pr\left[\mathtt{bad}_3 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] \\ &\quad + \Pr\left[\mathtt{bad}_4 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] + \Pr\left[\mathtt{bad}_5 \mid \neg\mathtt{bad}_{1234} \wedge S^* = S\right] . \end{aligned} \tag{15}$$

Let $S = (\mathit{Label}, L, p, \tau_{\text{cmt}}, \text{aux})$, and $s = \lceil L/h \rceil$. We can bound the first three terms on the RHS using similar arguments as employed in the proof of Theorem 2 yielding

$$\Pr\left[\mathtt{bad}_i \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] \leq \frac{\overline{q_{\text{fnd}}}^2 s_{\max}^2}{2^h} \tag{16}$$

for all $i \in \{2, 3, 4\}$. As for the last term, firstly $\neg\mathtt{bad}_1$ implies we only have to consider the find queries. Fix some $i \in \mathcal{J}$. $\neg\mathtt{bad}_1$ implies that $C_i$ is complete. Next we have:

- $\neg\mathtt{bad}_2$ implies the corresponding upper queries are all fresh. The definition then dictates $T^*_{C_i}[j] \neq T^*_{C_i}[j']$ for all $j \neq j' \in [1..s]$.
- $\neg\mathtt{bad}_3$ implies that in any $(C_i, T^*_{C_i}[j-1], T^*_{C_i}[j])$-chain, the lower query $(K^*_\top \,\|\, R^*_{C_i}[j])$ appears only after the corresponding upper query $(K^*_\bot \,\|\, T^*_{C_i}[j-1] \,\|\, R^*_{C_i}[j])$, for all $j \in [1..s]$.
- $\neg\mathtt{bad}_3 \wedge \neg\mathtt{bad}_4$ implies that the $(C_i, T^*_{C_i}[j-1], T^*_{C_i}[j])$-chain appears before $(C_i, T^*_{C_i}[j], T^*_{C_i}[j+1])$-chain, for all $j \in [1..s-1]$.

Thus, for all $j \in [1..s]$, $T^*_{C_i}[j]$ is uniformly distributed over $\{0,1\}^L \setminus \{T^*_{C_i}[j'] : 0 \leq j' \leq j-1\}$, given the previous history. Thus, conditioned on $\neg\mathtt{bad}_{1234}$ and $S^* = S$, $(T^*_{C_i}[1], \ldots, T^*_{C_i}[s])$ is identical in distribution to a without-replacement uniform sequence of size $s$, denoted $W_s$, over $\{0,1\}^L \setminus \{IV\}$. Then we have

$$\begin{aligned}
\Pr\left[\mathtt{bad}_5 \mid \neg\mathtt{bad}_{1234} \wedge S^* = S\right] &\leq \sum_{i\in\mathcal{J}} \Pr\left[p^*(T^*_{C_i}) = 1 \mid \neg\mathtt{bad}_{1234} \wedge S^* = S\right] \\
&= \sum_{i\in\mathcal{J}} \Pr\left[p(\mathtt{msb}_L(W_s)) = 1\right] \\
&\leq \sum_{i\in\mathcal{J}} \left(\Pr\left[p(\mathtt{msb}_L(U_{sh})) = 1\right] + \frac{s^2+s}{2^{h+1}}\right) \\
&\leq \sum_{i\in\mathcal{J}} \left(\Pr\left[p(U_L) = 1\right] + \frac{s^2_{\max}}{2^h}\right) \\
&\leq \overline{q_{\mathrm{fnd}}}\delta_{p,L} + \frac{\overline{q_{\mathrm{fnd}}}s^2_{\max}}{2^h}\,, \qquad (17)
\end{aligned}$$

where the third inequality follows by switching $W_s$ with $U_{sh}$ leading to the additional $s(s+1)/2^{h+1}$ term. (For a derivation of the switching result, see Appendix C; we use this on the predicate $p \circ \mathtt{msb}_L$.) Using Eqs. (13)–(17), we have

$$\Pr\left[\mathrm{KCS}(\mathcal{A}^H) = 1\right] \leq \frac{q_{\mathrm{cmt}}}{2^{\widetilde{k}}} + \frac{3\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}}{2^h} + \frac{\overline{q_{\mathrm{fnd}}}s^2_{\max}}{2^h} + \overline{q_{\mathrm{fnd}}}\delta\,. \qquad \square$$

### 5.4 Analysis of DP-HMAC[$H$]

**Theorem 4.** *Fix $k = |K_{IN}| = 8B_H$ and some $\Gamma = (q_{\mathrm{cmt}}, q_{\mathrm{fnd}}, b_{\min}, b_{\max}, L_{\min}, L_{\max}, \widetilde{k}, \delta)$. Let* KDF = DP-HMAC[$H$] *where $H : \{0,1\}^* \to \{0,1\}^h$ is a random oracle. Then, for any $\Gamma$-KC adversary $\mathcal{A}$,* KDF *is $\epsilon$-KC secure for*

$$\epsilon(\Gamma) = \frac{q_{\mathrm{cmt}}}{2^{\widetilde{k}}} + \frac{80\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}}{2^h} + \frac{\overline{q_{\mathrm{fnd}}}s^2_{\max}}{2^h} + \overline{q_{\mathrm{fnd}}}\delta\,,$$

*where $\overline{q_{\mathrm{fnd}}} = q_{\mathrm{fnd}} + 1$ and $s_{\max} = \lceil L_{\max}/h \rceil$.*

*Proof.* Any useful query $K \,\|\, X$ to $H$ is said to be an *upper query* if $K = K^*_\top$, and a *lower query* if $K = K^*_\bot$. An upper query is said to be:

- A *$C$-initialization* query if it is of the form $K^*_\top \,\|\, M^*_C$ for some $C \in \mathcal{C}$.
- A *chaining* upper query if it is of the form $K^*_\top \,\|\, X$, where $|X| = h$.
- A *$(C, I)$-outputting* upper query if it is of the form $K^*_\top \,\|\, I \,\|\, M^*_C$ for some $(C, I) \in \mathcal{C} \times \{0,1\}^h$.

An initialization query is said to be *context-revealing* query if it is a find query. The augmented oracle works as follows:

1. All the commit queries are relayed directly to the random oracle $H$, and its output is relayed back to $\mathcal{A}_{\mathrm{cmt}}$ after they are recorded in $\tau^*$.
2. When the $i$-th find query is a context-revealing query, denoted $K^*_\top \,\|\, M^*_{C_i}$, the KDF is evaluated as follows (also see Fig. 9):
   (a) Let $U^*_{C_i}[0] = M^*_{C_i}$.
   (b) For all $j \in [1..s^*]$:
      i. Whenever an entry is not in $\tau^*$, $H$ is queried with $K^*_\top \,\|\, U^*_{C_i}[j-1]$ and $(K^*_\top \,\|\, U^*_{C_i}[j-1], V^*_{C_i}[j])$ is stored in $\tau^*$, where $V^*_{C_i}[j]$ denotes the corresponding output.
      ii. Whenever an entry is not in $\tau^*$, $H$ is queried with $K^*_\bot \,\|\, V^*_{C_i}[j]$ and $(K^*_\bot \,\|\, V^*_{C_i}[j], U^*_{C_i}[j])$ is stored in $\tau^*$, where $U^*_{C_i}[j]$ denotes the corresponding output.
      iii. Whenever an entry is not in $\tau^*$, $H$ is queried with $K^*_\top \,\|\, U^*_{C_i}[j] \,\|\, M^*_{C_i}$ and $(K^*_\top \,\|\, U^*_{C_i}[j] \,\|\, M^*_{C_i}, R^*_{C_i}[j])$ is stored in $\tau^*$, where $R^*_{C_i}[j]$ denotes the corresponding output.
      iv. Whenever an entry is not in $\tau^*$, $H$ is queried with $K^*_\bot \,\|\, R^*_{C_i}[j]$ and $(K^*_\bot \,\|\, R^*_{C_i}[j], T^*_{C_i}[j])$ is stored in $\tau^*$, where $T^*_{C_i}[j]$ denotes the corresponding output.
   (c) Let $T^*_{C_i} \coloneqq \mathtt{msb}_L(T^*_{C_i}[1] \,\|\, \ldots \,\|\, T^*_{C_i}[s^*])$.
   (d) Finally, $V^*_{C_i}[1]$ is returned to $\mathcal{A}_{\mathrm{fnd}}$.
3. All other find queries are relayed directly to the random oracle $H$, and the outputs are relayed back to $\mathcal{A}_{\mathrm{fnd}}$ after they are recorded in $\tau^*$.

It is clear that $q_{\mathrm{tot}} \le q_{\mathrm{cmt}} + 4s^* q_{\mathrm{fnd}}$. Let $\mathcal{I}_\top, \mathcal{I}_\bot \subseteq [1..q_{\mathrm{tot}}]$ denote the sets of all upper and lower find queries, respectively. Let $\mathcal{J} \subseteq [1..q_{\mathrm{tot}}]$ denote the set of all context-revealing queries. Any pair $(i, i') \in \mathcal{I}_\top \times \mathcal{I}_\bot$ is said to be:

- A *$C$-initialization-chain* if $X^*_i = M^*_C$ and $X^*_{i'} = Y^*_i$ for some $C \in \mathcal{C}$.
- An *upper-chain* if $|X^*_i| = |X^*_{i'}| = h$ and $X^*_{i'} = Y^*_i$.
- A *$(C,U)$-lower-chain* if $X^*_i = U \,\|\, M^*_C$ and $X^*_{i'} = Y^*_i$ for some $C \in \mathcal{C}$ and $U \in \{0,1\}^h$.

Here, $Y^*_i$ is the output of $H$ corresponding to the query $X^*_i$. A context $C \in \mathcal{C}$ is said to be *complete* if there exists:

- A $C$-initialization-chain $(i_0, i_1)$.
- A sequence of $(s^* - 1)$ upper-chains $(i_2, i_3), \ldots, (i_{2(s^*-1)}, i_{2s^*-1})$ such that:
  - $X^*_{i_{2j}} = Y^*_{i_{2j-1}}$ for all $j \in [1..s^* - 1]$.
  - There exists $s^*$ lower-chain $(C, Y^*_{i_1}), (C, Y^*_{i_3}), \ldots, (C, Y^*_{i_{2s^*-1}})$.

By definition of HMAC[$H$], if $\mathcal{A}_{\mathrm{fnd}}$ can compute $\mathrm{KDF}(K^*, \mathit{Label}^*, C, L^*)$ for some $C$, then $C$ is complete. In particular, all context-revealing queries give a complete context. See Fig. 9 for an example. We assume that $\mathit{Context}^*$ is complete by providing $\mathcal{A}_{\mathrm{fnd}}$ an additional context-revealing query at the end.
Define the events:

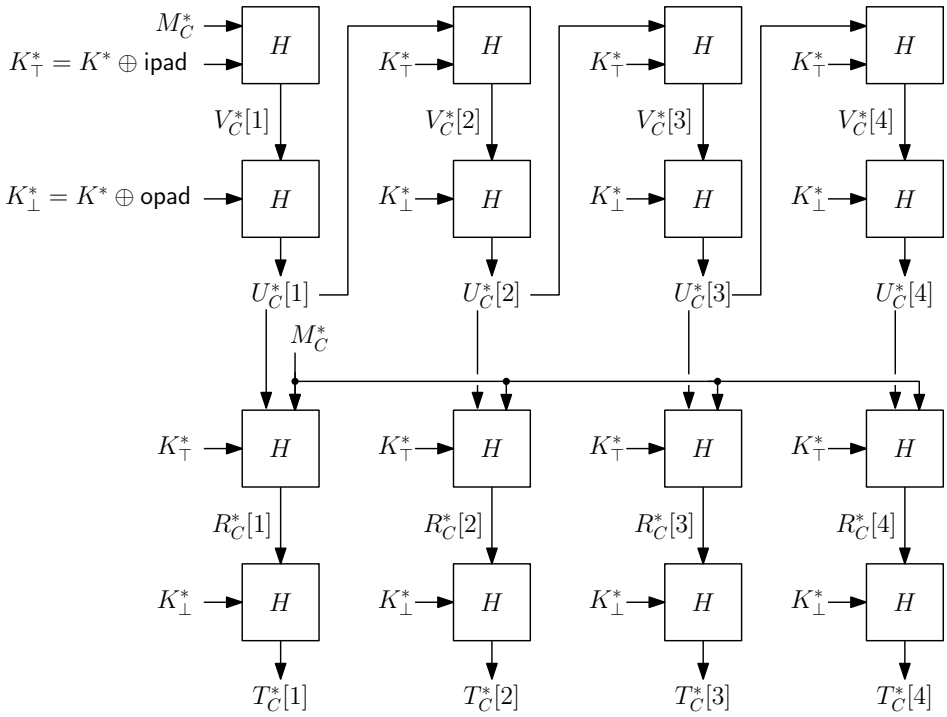


Figure 9: DP-HMAC with $L = 4h$ and $s = 4$. $M_C^* = (\mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L)$ and $K_{OUT} = T_C^* = (T_C^*[1] \,\|\, \cdots \,\|\, T_C^*[4])$.

$\mathtt{bad}_1$: there exists $i \in [1..q_{\mathrm{cmt}}]$ such that $\tau_i^*$ is useful.

$\mathtt{bad}_2$: there exists $i \in \mathcal{J}$, $i' < i \in [1..q_{\mathrm{tot}}]$, $j \in [1..s^* - 1]$, $C \in \mathcal{C}$ such that:

$$U_{C_i}^*[j] = M_C^* \text{ and } V_{C_i}^*[j+1] = X_{i'}^*.$$

$\mathtt{bad}_3$: there exists $i \in \mathcal{J}$, $i' < i \in [1..q_{\mathrm{tot}}]$, $j \in [1..s^* - 1]$, $C \in \mathcal{C}$ such that:

$$U_{C_i}^*[j] = M_C^* \text{ and } U_{C_i}^*[j+1] = X_{i'}^*.$$

$\mathtt{bad}_4$: there exists $i \in \mathcal{J}$, $i' < i \in [1..q_{\mathrm{tot}}]$, $j \in [1..s^*]$, $C \in \mathcal{C}$ such that:

$$U_{C_i}^*[j] \,\|\, U_{C_i}^*[0] = M_C^* \text{ and } R_{C_i}^*[j] = X_{i'}^*.$$

$\mathtt{bad}_5$: there exists $i \in \mathcal{J}$, $i' < i \in [1..q_{\mathrm{tot}}]$, $j \in [1..s^*]$, $C \in \mathcal{C}$ such that:

$$U_{C_i}^*[j] \,\|\, U_{C_i}^*[0] = M_C^* \text{ and } T_{C_i}^*[j] = X_{i'}^*.$$

$\mathtt{bad}_6$: there exists $i \in \mathcal{J}$, $i' < i \in [1..q_{\mathrm{tot}}]$ and $j \in [1..s^*]$ such that:

$$R_{C_i}^*[j] = X_{i'}^*.$$

$\mathtt{bad}_7$: there exists $i \in \mathcal{J}$, and distinct $j, j' \in [1..s^*]$ such that:

$$R_{C_i}^*[j] = R_{C_i}^*[j'].$$

$\mathtt{bad}_8$: there exists $i \in \mathcal{J}$ such that $p^*(T_{C_i}^*) = 1$.

**Algorithm** CMAC$[E](K, M)$
1. $(M[1], M[2], \ldots, M[m]) \xleftarrow{n} M$, $T \leftarrow 0^n$
2. **if** $|M[m]| = n$ **then** $M[m] \leftarrow M[m] \oplus 2E_K(0^n)$
3. **else** $M[m] \leftarrow (M[m] \,\|\, 10^{n-|M[m]|-1}) \oplus 2^2 E_K(0^n)$
4. **for** $i = 1$ **to** $m$ **do** $T \leftarrow E_K(T \oplus M[i])$
5. **return** $T$

Figure 10: Algorithm of CMAC for $n$-bit tag.

As before, in the event of $\neg\mathtt{bad}_1$, the commit queries become irrelevant for $\mathcal{A}_{\mathrm{fnd}}$. Let $\mathtt{bad}_{23} \coloneqq \mathtt{bad}_2 \vee \mathtt{bad}_3$ and $\mathtt{bad}_{45} \coloneqq \mathtt{bad}_4 \vee \mathtt{bad}_5$. In the event of $\neg\mathtt{bad}_1 \wedge \neg\mathtt{bad}_{23} \wedge \neg\mathtt{bad}_{45}$, $C \in \mathcal{C}$ is complete only if $C = C_i$ for some $i \in \mathcal{J}$. Furthermore, $\neg\mathtt{bad}_1 \wedge \neg\mathtt{bad}_{23} \wedge \neg\mathtt{bad}_{45} \wedge \neg\mathtt{bad}_{67}$ ensures that $R^*_{C_i}[j]$ is always fresh. Finally, $\mathrm{KCS}(\mathcal{A}^H) \wedge \neg\mathtt{bad}_1$ implies that $\mathtt{bad}_8$ takes place. Therefore, the aforementioned discussion implies that $(\mathtt{bad}_1 \vee \mathtt{bad}_8)$ occurs whenever $\mathrm{KCS}(\mathcal{A}^H)$ does. Thus we have

$$\begin{aligned}\Pr\left[\mathrm{KCS}(\mathcal{A}^H) = 1\right] &\le \Pr\left[\mathtt{bad}_1 \vee \mathtt{bad}_8\right] \\ &\le \Pr\left[\mathtt{bad}_1\right] + \Pr\left[\mathtt{bad}_8 \mid \neg\mathtt{bad}_1\right] \\ &\le \frac{q_{\mathrm{cmt}}}{2^{\widetilde{k}}} + \Pr\left[\mathtt{bad}_8 \mid \neg\mathtt{bad}_1\right] . \end{aligned} \tag{18}$$

Next, we have

$$\Pr\left[\mathtt{bad}_8 \mid \neg\mathtt{bad}_1\right] = \sum_{S \in \mathsf{Supp}(S^*)} \Pr\left[\mathtt{bad}_8 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] \times \Pr\left[S^* = S\right] . \tag{19}$$

For any $S = (\mathit{Label}, L, p, \tau_{\mathrm{cmt}}, \mathrm{aux}) \in \mathsf{Supp}(S^*)$, we claim

$$\Pr\left[\mathtt{bad}_8 \mid \neg\mathtt{bad}_1 \wedge S^* = S\right] \le \frac{80\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}}{2^h} + \frac{\overline{q_{\mathrm{fnd}}} s^2_{\max}}{2^h} + \overline{q_{\mathrm{fnd}}}\delta_{p,L} . \tag{20}$$

A proof of this claim is available in Appendix D. To conclude the proof, using Eqs. (18)–(20), we have

$$\begin{aligned}\Pr\left[\mathrm{KCS}(\mathcal{A}^H) = 1\right] &\le \frac{q_{\mathrm{cmt}}}{2^{\widetilde{k}}} + \mathop{\mathbb{E}}_{S^* \xleftarrow{\$} \mathcal{A}^H_{\mathrm{cmt}}} \left[\frac{80\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}}{2^h} + \frac{\overline{q_{\mathrm{fnd}}} s^2_{\max}}{2^h} + \overline{q_{\mathrm{fnd}}}\delta_{p^*,L^*}\right] \\ &\le \frac{q_{\mathrm{cmt}}}{2^{\widetilde{k}}} + \frac{80\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}}{2^h} + \frac{\overline{q_{\mathrm{fnd}}} s^2_{\max}}{2^h} + \overline{q_{\mathrm{fnd}}}\delta . \end{aligned}$$

□

# 6 Analysis of CMAC-based KDFs

This section describes attacks on CMAC-based KDFs[7]. The specification of CMAC is given in Fig. 10. See also Fig. 1. We may write $\mathrm{CMAC}(K, M)$ to mean $\mathrm{CMAC}[E](K, M)$ if $E$ is clear from the context. We also write $K' := 2E_K(0^n)$ and $K'' := 2^2 E_K(0^n)$. The attacks depend on the choice of KDF schemes, and whether or not the strengthening mechanism specified in [Che22] is taken into account. The attack complexity when $E$ is AES is summarized in Table 2. Note that the attack complexity is independent of the key size.

[7] As noted in Footnote 3, "attack" does not mean to break some existing security claim. HMAC- and KMAC-based KDFs cannot be attacked, while for HMAC, there is still room for attacks if the premise are relaxed. Details are explained in Appendix H.

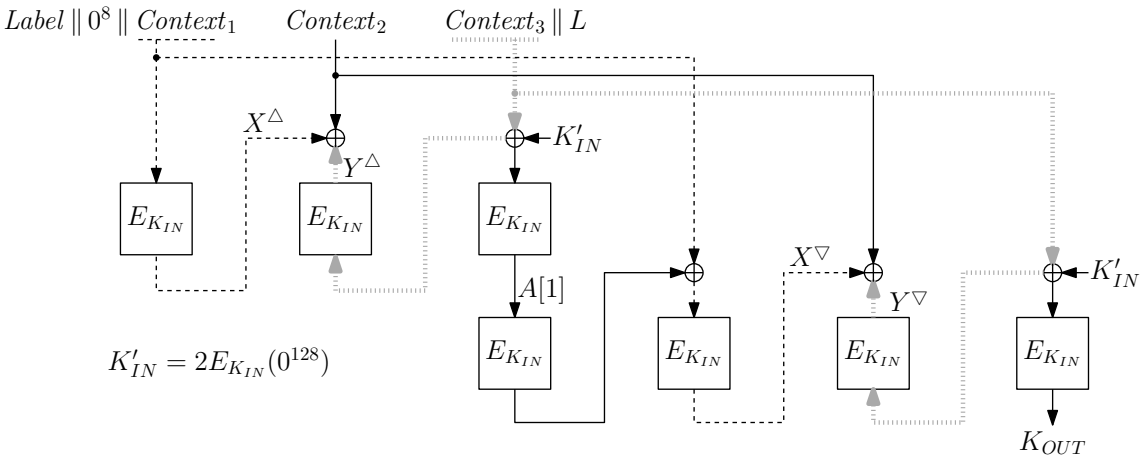


Figure 11: Birthday Attacks on DP-CMAC[$E$].

Our formalization in Definition 2 captures the most generalized adversaries that are able to control $K_{OUT}$ though the predicate, making provable security results strong. For attacks, we aim to present strong attacks on practical settings, i.e., we aim at attacks that work for *any given* $K_{IN}$, $K_{OUT}$, and *Label*, without interacting with the oracle in advance. Namely, $\mathcal{A}_{\mathrm{cmt}}$ is omitted, and the values of $K_{IN}$, $K_{OUT}$, and *Label* are randomly given. The goal is to run $\mathcal{A}_{\mathrm{fnd}}$ to find a value of *Context* generating $K_{OUT}$, i.e., we have one target image $K_{OUT}$, and the predicate is 1 for this value of $K_{OUT}$ only. For other parameters, we follow a practical scenario, where the length of the binary encoding of the counter $r$ is 16 bits, the lengths of *Label* and the encoding of $L$ are 16 bits, and the block size $n$ is 128 bits. Also, we basically assume $L$ is 1 block, i.e., $n$ bits in general[8].

### 6.1 Attacks on DP-CMAC[$E$]

We present an attack on DP-CMAC[$E$] that breaks KC security with $O(2^{n/2})$ complexity as depicted in Fig. 11.

The attacker chooses *Context* so that the message length becomes three blocks, denoted by $M[1]$, $M[2]$, and $M[3]$. $M[1]$ contains 16 bits of *Label*, 8 bits of zeros, followed by *Context*. $M[2]$ is fully occupied by *Context*. $M[3]$ is filled with *Context* and the 16-bit encoding of $L$. We denote the *Context* value positioned in $M[1]$, $M[2]$, and $M[3]$ by $Context_1$, $Context_2$, and $Context_3$, respectively. In this attack, after $K_{IN}$ and $K_{OUT}$ are given to the attacker, $A[1]$ is fixed to an arbitrary value, which fixes $E_{K_{IN}}(A[1])$ and $E^{-1}_{K_{IN}}(A[1])$. Then, the attacker mounts a meet-in-the-middle attack to find a valid *Context* as follows:

1. For $2^{n/2}$ choices of $Context_1$, compute two $n$-bit values $(X^{\triangle}, X^{\triangledown})$ by $X^{\triangle} = E_{K_{IN}}(M[1])$ and $X^{\triangledown} = E_{K_{IN}}(E_{K_{IN}}(A[1]) \oplus M[1])$, and compute $X^{\triangle} \oplus X^{\triangledown}$ to store it in a hash table. (broken lines in Fig. 11.)

2. For $2^{n/2}$ choices of $Context_3$, compute two $n$-bit values $(Y^{\triangle}, Y^{\triangledown})$ by $Y^{\triangle} = E^{-1}_{K_{IN}}(E^{-1}_{K_{IN}}(A[1]) \oplus M[3] \oplus K'_{IN})$ and $Y^{\triangledown} = E^{-1}_{K_{IN}}(E^{-1}_{K_{IN}}(K_{OUT}) \oplus M[3] \oplus K'_{IN})$, and compute $Y^{\triangle} \oplus Y^{\triangledown}$ to store it in a hash table. (dotted gray lines in Fig. 11.)

3. Find a match between $X^{\triangle} \oplus X^{\triangledown}$ and $Y^{\triangle} \oplus Y^{\triangledown}$. Then, return $X^{\triangle} \oplus Y^{\triangle}$ of the matched pair as $Context_2$.

[8] A few specific applications may warrant attacks with $L > n$. We discuss attacks for such cases in Appendices F and G.

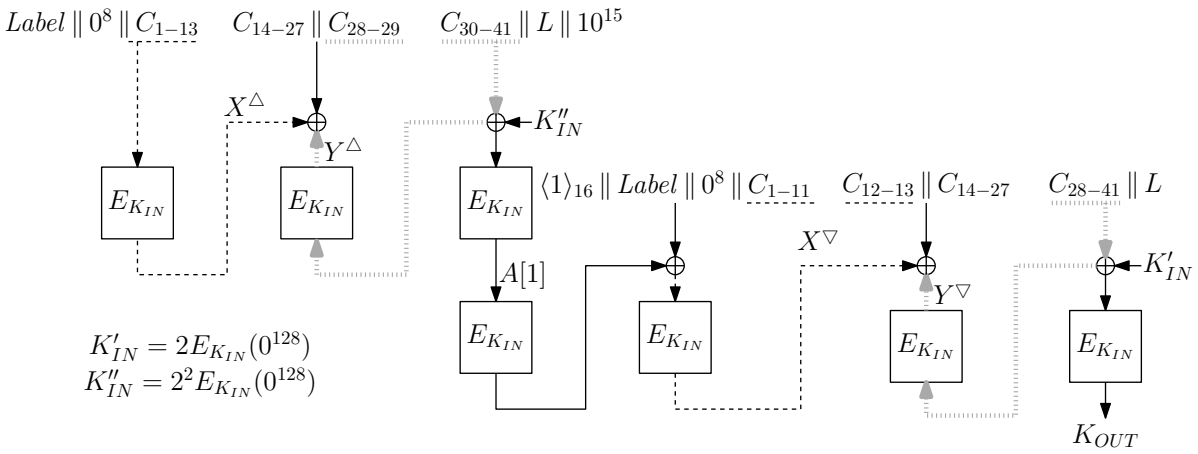


Figure 12: Birthday Attacks on stDP-CMAC[$E$].

At a first glance, the attack requires a collision of $2n$-bit values, but the necessary conditions are $X^{\triangle} = Y^{\triangle}$ and $X^{\triangledown} = Y^{\triangledown}$, which is written as $X^{\triangle} \oplus X^{\triangledown} = Y^{\triangle} \oplus Y^{\triangledown}$, so by finding collisions of $n$-bit differences, the *Context* value that outputs $K_{OUT}$ is found. The attack complexity is $O(2^{n/2})$.

### 6.2 Attacks on stDP-CMAC[$E$]

NIST proposed countermeasures to mitigate attacks against KC security, which we call stCTR-CMAC[$E$], stFB-CMAC[$E$], and stDP-CMAC[$E$]. We found that all of them can be attacked about the birthday-bound complexity of the block size of $E$, but with a slightly higher complexity due to the unexpected effect of the counter value. Because the attacks are similar, we only demonstrate the attack against stDP-CMAC[$E$] in the main body. An interesting outcome is that the attack complexity for DP-CMAC[$E$] and stDP-CMAC[$E$] are almost the same, indicating that the effect of stDP-CMAC[$E$] is limited.

When $L$ is $n$ bits, i.e., 1 block, stDP-CMAC[$E$] first computes $A[1] \leftarrow \mathrm{CMAC}(K_{IN}, Label \| 0^8 \| Context \| L)$, and then computes $K_{OUT}$ as $K_{OUT} \leftarrow \mathrm{CMAC}(K_{IN}, A[1] \| \langle 1 \rangle_r \| Label \| 0^8 \| Context \| L)$. Note that the computation of $K_{OUT}$ has $A[1]$ and the counter value $\langle 1 \rangle_r$ at the beginning of the message. We again assume $r = 16$ bits. Then, the bit positions of the *Context* for computing $A[1]$ and $K_{OUT}$ are desynchronized by (1 block and) 16 bits. Indeed, *Context* starts from the 4th byte inside the first block for computing $A[1]$ while it starts from the 6th byte inside the second block for computing $K_{OUT}$. This desyncronization of the bit positions makes the attack complexity slightly higher. Because of this shift in 2 bytes, describing the attack for a general block cipher with $n$-bit block causes unnecessary complex evaluation. For simplicity, here we assume that the underlying $E$ is AES, having $n = 128$ bits; 16 bytes.

The basic attack approach, which is depicted in Fig. 12, is the same as one for DP-CMAC[$E$]. The attacker uses *Context* of 41 bytes. $C_i, 1 \leq i \leq 41$ denotes the $i$-th byte of *Context*, and similarly sub-indices of a variable denote byte positions of that variable. In the computation of $A[1]$, $M[1]$ consists of 2 bytes of *Label*, 8 bits of zeros, followed by 13 bytes of *Context*. $M[2]$ fully consists of *Context*. $M[3]$ consists of 12 bytes of *Context* and 2 bytes of the encoding of $L$, and the one-zero padding is applied to fill the remaining 2 bytes. In this attack, $K_{IN}$ and $K_{OUT}$ are given, and the goal is to find *Context*. The attacker uses $C_1$ to $C_{13}$ as degrees of freedom for one part of the MitM attack and uses $C_{28}$ to $C_{41}$ for the other part. The full attack is described as follows:

1. $A[1]$ is fixed to an arbitrary value, which also fixes $E_{K_{IN}}(A[1])$ and $E^{-1}_{K_{IN}}(A[1])$.

2. For $2^{72}$ choices of $C_1$ to $C_{13}$, compute two $n$-bit values $(X^{\triangle}, X^{\triangledown})$ by $X^{\triangle} = E_{K_{IN}}(M[1])$ and $X^{\triangledown} = E_{K_{IN}}(E_{K_{IN}}(A[1]) \oplus (\langle 1 \rangle_{16} \,\|\, M[1] \gg 16))$, and store $X^{\triangledown}_{1-2} \oplus C_{12-13} \| X^{\triangle}_{1-14} \oplus X^{\triangledown}_{3-16} \| X^{\triangle}_{15-16}$ in a hash table. (broken lines in Fig. 12.)

3. For $2^{72}$ choices of $C_{28}$ to $C_{41}$, compute two $n$-bit values $(Y^{\triangle}, Y^{\triangledown})$ by $Y^{\triangle} = E^{-1}_{K_{IN}}(E^{-1}_{K_{IN}}(A[1]) \oplus M[3] \oplus K''_{IN})$ and $Y^{\triangledown} = E^{-1}_{K_{IN}}(E^{-1}_{K_{IN}}(K_{OUT}) \oplus (C_{28-41} \,\|\, L) \oplus K'_{IN})$, and store $Y^{\triangledown}_{1-2} \| Y^{\triangle}_{1-14} \oplus Y^{\triangledown}_{3-16} \| Y^{\triangle}_{15-16} \oplus C_{28-29}$ in a hash table. (dotted gray lines in Fig. 12.)

4. Find a pair meeting the following equality of total 18 bytes: $X^{\triangledown}_{1-2} \oplus C_{12-13} = Y^{\triangledown}_{1-2}$, $X^{\triangle}_{15-16} = Y^{\triangle}_{15-16} \oplus C_{28-29}$, and $X^{\triangle}_{1-14} \oplus X^{\triangledown}_{3-16} = Y^{\triangle}_{1-14} \oplus Y^{\triangledown}_{3-16}$.

The left-hand side of the equations only depends on $C_1$ to $C_{13}$ and the right-hand side only depends on $C_{28}$ to $C_{41}$, hence the meet-in-the-middle attack is applied. The size of the match is 18 bytes and the attack complexity is $2^{72}$.

The attack is trivially generalized for an underlying block cipher $E$ with a block length of $n$ bits and the counter value with its encoded length of $r$ bits. The attack complexity is $2^{n/2+r/2}$ in general, and the above attack is the case with $n = 128$ bits and $r = 16$.

### 6.3 Note on stCTR-CMAC[$E$] and stFB-CMAC[$E$]

stCTR-CMAC[$E$] is essentially the same as stDP-CMAC[$E$] except for the way they process the result of the first layer ($A[1]$ in stDP-CMAC[$E$]). It is prefixed before the message in stDP-CMAC[$E$] while it is suffixed after the message in stCTR-CMAC[$E$]. From the structural similarity, the same attack with the same complexity is applied to stCTR-CMAC[$E$]. Details are given in Appendix E. stFB-CMAC[$E$] is exactly the same as stCTR-CMAC[$E$] when the output size is 1 block. Hence, stFB-CMAC[$E$] admits an attack with the same complexity.

## 7 Conclusions

This paper has provided the first formal treatment of key control security for KDFs, and studied to what extent the KDF constructions from NIST SP 800-108 achieve this notion. Our analysis suggests that instantiations of KDFs based on KMAC and HMAC can achieve better concrete security than instantiations of CMAC based KDFs with AES (or more generally, block ciphers with 128-bit blocks), even when considering mitigation techniques introduced to prevent attacks against key control security.

We stress that our bounds for HMAC KDFs are affected by the birthday bound (with respect to the hash output length $h$), and our attacks for CMAC KDFs do not beat the birthday barrier (with respect to the block length). The only distinction is that $h$ is generally much larger than 128, and therefore a possible mitigation would be to replace AES in CMAC KDFs with a block cipher with a larger block size. This would however still require *proofs* of good security bounds for CMAC KDFs, and our preliminary investigation suggests that this is a technically involved question which we leave as an open problem.

Our proofs for HMAC-based and KMAC-based KDFs rely on the random oracle model, and if we want to translate these bounds to concrete constructions based on a simpler primitive (e.g., a permutation or a compression function), it is natural to attempt to resort to indifferentiability [MRH04, CDMP05], and it is left as an interesting research direction to investigate the connection of KC security to the indifferentiability notion. The main technical challenge is

that our adversaries are two-stage adversaries if we interpret $K^*_{IN}$ as a challenge rather than $\mathcal{A}_{\text{cmt}}$'s output [9]. which *in general* limits the application of indifferentiability [RSS11]. However, it appears that our definition interacts well with indifferentiability if we relax the min-entropy requirement to use a smooth version, or if we consider stronger notions of indifferentiability. We leave this for future work. We also remark that there are good reasons to give direct proofs. First, these could give better bounds, and second, this could allow us to consider hash functions which are not indifferentiable (such as MD without prefix-free encoding). We also leave this as an open problem for future work.

Finally, as discussed in Sect. 3.4, we leave the formalization and analysis of KC security where the randomness is assumed in *Context* or other input of a KDF instead of a key derivation key as future work.

### Acknowledgements

The authors thank the organizers of Dagstuhl Seminar 24041, Symmetric Cryptography, for inviting them to the seminar, where the collaboration has initiated. We thank Aishwarya Thiruvengadam for the early phase of the discussion. Part of this research was conducted when Ritam Bhaumik was affiliated with EPFL, Switzerland. Tetsu Iwata was supported in part by JSPS KAKENHI Grant Number JP24K07489. Ashwin Jha was supported by the German Research Foundation (DFG) within the framework of the Excellence Strategy of the Federal Government and the States — EXC 2092 CaSa — 39078197. Stefano Tessaro was partially supported by NSF grants CNS-2026774, CNS-2154174, CNS-2426905, and a Stellar Development Foundation Academic Research Award.

## References

[AKMZ04] Carlisle M. Adams, Guenther Kramer, Serge Mister, and Robert J. Zuccherato. On The Security of Key Derivation Functions. In Kan Zhang and Yuliang Zheng, editors, *ISC 2004*, volume 3225 of *Lecture Notes in Computer Science*, pages 134–145. Springer, 2004. *(Cited on p. 5.)*

[arm] ARM Trustzone CryptoCell-712. `https://developer.arm.com/documentation/102897/latest/`. *(Cited on pp. 4, 12, and 36.)*

[asp] Subkey derivation and authenticated encryption in ASP.NET Core. `https://learn.microsoft.com/en-us/aspnet/core/security/data-protection/implementation/subkeyderivation?view=aspnetcore-8.0`. *(Cited on pp. 4, 12, and 35.)*

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A Block Cipher for Low Energy. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436. Springer, 2015. *(Cited on p. 13.)*

[BCR+19] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, Richard Davis, and Scott Simon. Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-56B, Rvision 2, 3 2019. *(Cited on p. 14.)*

[9] However, whether this is a multi-stage game depends on how one defines it.

[BDI+25] Ritam Bhaumik, Avijit Dutta, Akiko Inoue, Tetsu Iwata, Ashwin Jha, Kazuhiko Minematsu, Mridul Nandi, Yu Sasaki, Meltem Sönmez Turan, and Stefano Tessaro. Cryptographic Treatment of Key Control Security – In Light of NIST SP 800-108. In Seny Kamara and Yael Tauman Kalai, editors, *CRYPTO 2025*, Lecture Notes in Computer Science. Springer, 2025. *(Cited on p. 1.)*

[BDKJ21] Alex Biryukov, Daniel Dinu, Dmitry Khovratovich, and Simon Josefsson. Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications. RFC 9106, September 2021. *(Cited on p. 5.)*

[BDPA08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the Indifferentiability of the Sponge Construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008. *(Cited on pp. 7 and 14.)*

[BDPA13] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 313–314. Springer, 2013. *(Cited on p. 14.)*

[Blu83] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News*, 15(1):23–27, jan 1983. *(Cited on p. 6.)*

[BM17] Elaine Barker and Nicky Mouha. Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-67r2, 2017. *(Cited on p. 13.)*

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000. *(Cited on p. 13.)*

[CAV] Cryptographic Algorithm Validation Program. `https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program`. *(Cited on pp. 4 and 35.)*

[cce] Huawei Mobile Devices with EMUI 11.0 Security Target. `https://www.ocsi.gov.it/documenti/certificazioni/huawei/st_huawei_p40_emui11_v1.11.pdf`. *(Cited on pp. 4 and 36.)*

[CDKT19] Sandro Coretti, Yevgeniy Dodis, Harish Karthikeyan, and Stefano Tessaro. Seedless Fruit Is the Sweetest: Random Number Generation, Revisited. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 205–234. Springer, 2019. *(Cited on pp. 4 and 12.)*

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005. *(Cited on p. 28.)*

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited (Preliminary Version). In Jeffrey Scott Vitter, editor, *ACM Symposium on the Theory of Computing, 1998*, pages 209–218. ACM, 1998. *(Cited on p. 5.)*

[Che22] Lily Chen. Recommendation for Key Derivation Using Pseudorandom Functions. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-108 Rev. 1, 8 2022. *(Cited on pp. 3, 4, 6, 7, 8, 12, 13, 15, 25, 39, and 42.)*

[CJZ11] Jon Callas, Alan Johnston, and Philip Zimmermann. ZRTP: Media Path Key Agreement for Unicast Secure RTP. *RFC*, 6189, 2011. *(Cited on p. 36.)*

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam D. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540. Springer, 2004. *(Cited on p. 6.)*

[Dwo05] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-38B, 5 2005. *(Cited on pp. 3 and 6.)*

[Dwo10] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-38E, 2010. *(Cited on p. 13.)*

[EM97] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *J. Cryptol.*, 10(3):151–162, 1997. *(Cited on p. 12.)*

[GJN+16] Jian Guo, Jérémy Jean, Ivica Nikolic, Kexin Qiao, Yu Sasaki, and Siang Meng Sim. Invariant Subspace Attack Against Midori64 and The Resistance Criteria for S-box Designs. *IACR Trans. Symmetric Cryptol.*, 2016(1):33–56, 2016. *(Cited on p. 13.)*

[gp] Secure Channel Protocol 03 Card Specification v 2.2. `https://globalplatform.org/wp-content/uploads/2019/03/GPC_2.2_D_SCP03_v1.0.pdf`. *(Cited on pp. 4, 14, and 36.)*

[HRS16] Andreas Hülsing, Joost Rijneveld, and Fang Song. Mitigating Multi-target Attacks in Hash-Based Signatures. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 387–416. Springer, 2016. *(Cited on pp. 5 and 14.)*

[HSM] Thales Luna Cloud HSM SDK Reference guide. `https://thalesdocs.com/dpod/services/luna_cloud_hsm/extern/client_guides/Content/sdk/extensions/counter_mode_kdf_mechanisms.htm`. *(Cited on pp. 4 and 35.)*

[Hud12] Greg Hudson. Camellia Encryption for Kerberos 5. *RFC*, 6803, 2012. *(Cited on p. 36.)*

[IEE] Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices. Standard, IEEE Security in Storage Working Group. *(Cited on p. 13.)*

[ISOa] ISO/IEC 19772:2020 Authenticated encryption. Standard, International Standard Organization. *(Cited on p. 13.)*

[ISOb] ISO/IEC 9797-1:2011 Message Authentication Codes (MACs), Part 1: Mechanisms using a block cipher. Standard, International Standard Organization. *(Cited on p. 13.)*

[IW14] Tetsu Iwata and Lei Wang. Impact of ANSI X9.24-1: 2009 Key Check Value on ISO/IEC 9797-1: 2011 MACs. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 303–322. Springer, 2014. *(Cited on p. 13.)*

[jet] Jetson Linux API Reference Documentation. `https://docs.nvidia.com/jetson/l4t-multimedia/index.html`. *(Cited on pp. 4 and 35.)*

[KAA21] Gaurav Kodwani, Shashank Arora, and Pradeep K. Atrey. On Security of Key Derivation Functions in Password-based Cryptography. In *CSR 2021*, pages 109–114. IEEE, 2021. *(Cited on p. 3.)*

[Kal00] Burt Kaliski. PKCS #5: Password-Based Cryptography Specification Version 2.0. RFC 2898, September 2000. *(Cited on p. 5.)*

[KjCP16] John Kelsey, Shu jen Chang, and Ray Perlner. SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-185, 12 2016. *(Cited on pp. 3 and 6.)*

[Kra10] Hugo Krawczyk. Cryptographic Extraction and Key Derivation: The HKDF Scheme. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 631–648. Springer, 2010. *(Cited on pp. 3 and 5.)*

[KS05] John Kelsey and Bruce Schneier. Second Preimages on n-Bit Hash Functions for Much Less than $2^n$ Work. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 474–490. Springer, 2005. *(Cited on p. 42.)*

[Min06] Kazuhiko Minematsu. Improved Security Analysis of XEX and LRW Modes. In Eli Biham and Amr M. Youssef, editors, *SAC 2006*, volume 4356 of *Lecture Notes in Computer Science*, pages 96–113. Springer, 2006. *(Cited on p. 13.)*

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004. *(Cited on p. 28.)*

[Nat08] National Institute of Standards and Technology. The Keyed-Hash Message Authentication Code (HMAC). Federal Information Processing Standards (FIPS) Publication 198-1, 7 2008. *(Cited on pp. 3 and 6.)*

[Nat15] National Institute of Standards and Technology. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Information Processing Standards (FIPS) Publication 202, 8 2015. *(Cited on pp. 7, 14, and 15.)*

[Nat22] National Institute of Standards and Technology. Public Comments and Resolutions on Draft NIST SP 800-108 Revision 1, Recommendation for Key Derivation Using Pseudorandom Functions. Available at `https://csrc.nist.gov/pubs/sp/800/108/r1/upd1/final`, 1 2022. *(Cited on pp. 3, 5, 12, 13, 35, and 36.)*

[NRS14] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering Generic Composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 257–274. Springer, 2014. *(Cited on p. 13.)*

[NS23] Vivek Nair and Dawn Song. Multi-factor key derivation function (MFKDF) for fast, flexible, secure, & practical key management. In *USENIX Security Symposium*, pages 2097–2114. USENIX Association, 2023. *(Cited on p. 3.)*

[nxp] Huawei Mobile Devices with EMUI 11.0 Security Target. `https://www.commoncriteriaportal.org/files/epfiles/1136a_pdf.pdf`. *(Cited on pp. 4 and 36.)*

[ora] Oracle Linux 8 NSS Cryptographic Module. `https://www.oracle.com/a/ocom/docs/corporate/140sp4226.pdf`. *(Cited on pp. 4 and 35.)*

[PJ16] Colin Percival and Simon Josefsson. The scrypt Password-Based Key Derivation Function. *RFC*, 7914:1–16, 2016. *(Cited on pp. 3 and 5.)*

[RL10] Eric Rescorla and Gregory M. Lebovitz. Cryptographic Algorithms for the TCP Authentication Option (TCP-AO). *RFC*, 5926, 2010. *(Cited on pp. 5, 14, and 36.)*

[Rog04] Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004. *(Cited on p. 13.)*

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004. *(Cited on pp. 5 and 14.)*

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with Composition: Limitations of the Indifferentiability Framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, 2011. *(Cited on p. 29.)*

[SPK+18] Minhye Seo, Jong Hwan Park, Youngsam Kim, Sangrae Cho, Dong Hoon Lee, and Jung Yeon Hwang. Construction of a New Biometric-Based Key Derivation Function and Its Application. *Secur. Commun. Networks*, 2018:6107912:1–6107912:14, 2018. *(Cited on p. 3.)*

[SWG25] Yaobin Shen, Lei Wang, and Dawu Gu. Security Analysis of NIST Key Derivation Using Pseudorandom Functions. *IACR Cryptol. ePrint Arch.*, 2025/815, 2025. *(Cited on p. 5.)*

[TBM10] Joseph D. Touch, Ron Bonica, and Allison J. Mankin. The TCP Authentication Option. *RFC*, 5925, 2010. *(Cited on pp. 5, 14, and 36.)*

[tit] Google Pixel Phones on Android 12.0 Security Target. `https://www.commoncriteriaportal.org/files/epfiles/st_vid11239-st.pdf`. *(Cited on pp. 4 and 36.)*

[TLS16] Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear Invariant Attack - Practical Attack on Full SCREAM, iSCREAM, and Midori64. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 3–33, 2016. *(Cited on p. 13.)*

[WDNS12] Chuah Chai Wen, Ed Dawson, Juan Manuel González Nieto, and Leonie Simpson. A Framework for Security Analysis of Key Derivation Functions. In Mark Dermot Ryan, Ben Smyth, and Guilin Wang, editors, *ISPEC 2012*, volume 7232 of *Lecture Notes in Computer Science*, pages 199–216. Springer, 2012. *(Cited on p. 3.)*

[WDS13] Chuah Chai Wen, Ed Dawson, and Leonie Simpson. Key Derivation Function: The SCKDF Scheme. In Lech J. Janczewski, Henry B. Wolfe, and Sujeet Shenoi, editors, *SEC 2013*, volume 405 of *IFIP Advances in Information and Communication Technology*, pages 125–138. Springer, 2013. *(Cited on p. 3.)*

# A Initial Observations on KC Security of CTR-CMAC[$E$] and FB-CMAC[$E$] by Arciszewski et al.

Arciszewski et al. found a constant-time attack on the key control security of CTR-CMAC[$E$], where $E$ is the block cipher AES [Nat22]. Although Arciszewski et al. demonstrated the attack with 2 *Context* blocks with a higher complexity, we explain the attack with 3 *Context* blocks for the sake of ease. Note that the possibility of such generalization has already been mentioned in [Nat22].

In CTR-CMAC[$E$] with a secret key $K_{IN}$ and a user-provided context *Context*, an output keying material $K_{OUT}$ is computed as $K_{OUT} = \text{CTR-CMAC}[E](K_{IN}, \mathit{Context})$. To be more precise, CTR-CMAC[$E$] also takes a system parameter (defined by an upper-layer protocol) *Label* and the bit-length of the derived keying material $L$ as input, and computes the following by increasing the counter value $i$, until the necessary keying materials are generated.

$$K[i] \leftarrow \text{CMAC}[E](K_{IN}, \langle i \rangle_{16} \,\|\, \mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L)\,,$$

where $\langle i \rangle_{16}$ is the 16-bit encoding of the counter value, $0^8$ is eight bits of zeros.

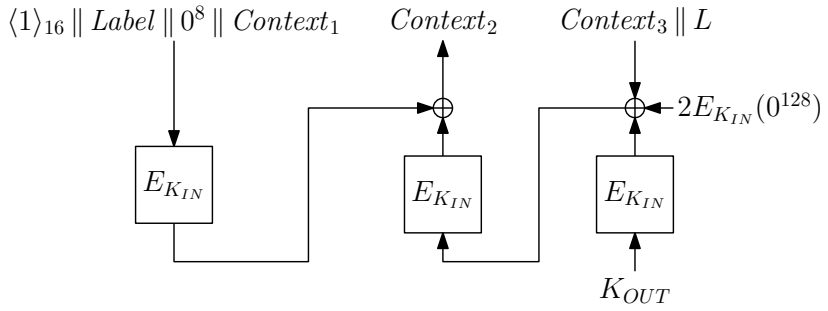


Figure 13: Attacks on CTR-CMAC[$E$] and FB-CMAC[$E$] [Nat22].

The attack shown in Fig. 13 demonstrates that, for any $K_{IN}$ and 1 block (128 bits for AES) of $K_{OUT}$, the corresponding *Context* can be computed in a constant time. The attack only generates 1 block of $K_{OUT}$, hence $\langle i \rangle_{16}$ takes a fixed value of $\langle 1 \rangle_{16}$ and $L$ is fixed to 128. *Context* is chosen s.t. the length of $\langle 1 \rangle_{16} \,\|\, \mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L$ is exactly 3 blocks. It is assumed that $|\mathit{Label}|$ is sufficiently short such that $\langle 1 \rangle_{16} \,\|\, \mathit{Label}$ does not occupy the whole block, then the first, second, and third message blocks are $M[1] = \langle 1 \rangle_{16} \,\|\, \mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context}_1$, $M[2] = \mathit{Context}_2$, and $M[3] = \mathit{Context}_3 \,\|\, L$, respectively, where $\mathit{Context} = \mathit{Context}_1 \,\|\, \mathit{Context}_2 \,\|\, \mathit{Context}_3$.

After the target $K_{OUT}$ is fixed, the attacker chooses arbitrary values for $\mathit{Context}_1$ and $\mathit{Context}_3$, and computes $E_{K_{IN}}(M[1])$ and $E^{-1}_{K_{IN}}(E^{-1}_{K_{IN}}(K_{OUT}) \oplus M[3] \oplus 2E_{K_{IN}}(0^{128}))$. By setting $\mathit{Context}_2$ to the XOR of those two values, $K_{OUT}$ is ensured to be produced from this *Context*. This is a constant time attack.

Because FB-CMAC[$E$] is exactly the same as CTR-CMAC[$E$] when $L$ is 1 block, FB-CMAC[$E$] can be attacked in the same way.

# B Real-World Applications of NIST SP 800-108

We present a list of real-world applications of NIST SP 800-108 (not exhaustive). Current Cryptographic Algorithm Validation Program (CAVP) list shows that Ubuntu Linux or VMware Photon OS for certain Intel processors use these KDFs [CAV]. Other examples are ASP.NET [asp], Thales's HSM [HSM], Nvidia's Jetson Linux [jet], and Oracle Linux 8 NSS Cryptographic Module [ora]. Major cryptographic libraries, such as OpenSSL, BouncyCastles, and Botan support

SP 800-108 KDFs. Moreover, several major smartphones use (a subset of) SP 800-108 KDFs in their Trusted Execution Environment (TEE), such as Google Pixel's Titan M [tit], and Huawei's CC engine [cce]. ARM Trustzone [arm] and several smart card specifications [nxp, gp] adopt them as well.

Multiple Internet standards adopt SP 800-108 KDFs. Notably, TCP Authentication Option (TCP-AO, RFC 5925 and 5926 [TBM10, RL10]), which is widely deployed for protecting long-lived TCP connections, employs SP 800-108 KDFs based on HMAC-SHA1 and CMAC-AES-128. More examples exist, such as ZRTP (RFC 6189 [CJZ11]) and Kerberos v5 (RFC 6803 [Hud12]).

Key control security is relevant when the KDF input could be maliciously chosen. The public comment by Amazon [Nat22] mentioned Milenage ETSI 3GPP as an example of protocols containing such KDFs. Indeed, we identified some real applications where an input parameter to the SP800-108 KDF (context) has a certain freedom that could be exploited by a malicious party. For example, in the aforementioned TCP-AO, the context contains 4-byte Initial Sequence Numbers (ISNs) that are recommended to be pseudorandomly chosen (See [TBM10, Sect. 5.2]). GlobalPlatform's Card Technology Secure Channel Protocol [gp] specifies CMAC-KDF and its context contains the host and the recipient (card)'s 8-byte random challenges ([gp, Sect. 6.2.2.2]). On the other extreme, we also found cases where KC security is irrelevant. For instance, the KDFs in RFC 6803 do not use a context. These cases suggest potential application-dependent risks that need further investigations.

# C Switching from Without-Replacement to Uniform

Fix some finite set $\mathcal{B}$, some subset $\mathcal{B}' \subseteq \mathcal{B}$, and some $s \leq |\mathcal{B}'|$. Let $U$ be uniformly distributed over $\mathcal{B}^s$. Let $V = (V_1 \| \dots \| V_s)$ be a uniform-without-replacement sample $(V_1, \dots, V_s)$ of size $s$ from $\mathcal{B}'$. Let $(\mathcal{B}')_s$ denote the support of $(V_1, \dots, V_s)$. Then for any predicate $f\colon \mathcal{B}^s \to \{0,1\}$, we have

$$\begin{aligned}
\Pr[f(V)=1] &= \sum_{f(v_1\|\dots\|v_s)=1} \Pr[V_1 = v_1, \dots, V_s = v_s] \\
&= \sum_{\substack{(v_1,\dots,v_s)\in(\mathcal{B}')_s \\ f(v_1\|\dots\|v_s)=1}} \frac{1}{(|\mathcal{B}'|)_s} \\
&= \frac{|f^{-1}(1) \cap (\mathcal{B}')_s|}{(|\mathcal{B}'|)_s} \\
&\leq \frac{|f^{-1}(1)|}{(|\mathcal{B}'|)_s} \\
&= \frac{|f^{-1}(1)|}{|\mathcal{B}|^s} \cdot \prod_{j=0}^{s-1} \left(1 - \frac{j}{|\mathcal{B}|} - \frac{|\mathcal{B}| - |\mathcal{B}'|}{|\mathcal{B}|}\right)^{-1} \\
&\leq \Pr[f(U)=1] \cdot \left(1 - \sum_{j=0}^{s-1} \frac{j}{|\mathcal{B}|} - \frac{s(|\mathcal{B}| - |\mathcal{B}'|)}{|\mathcal{B}|}\right)^{-1} \\
&\leq \Pr[f(U)=1] \cdot \left(1 - \frac{s^2}{2|\mathcal{B}|} - \frac{s(|\mathcal{B}| - |\mathcal{B}'| - 0.5)}{|\mathcal{B}|}\right)^{-1}.
\end{aligned}$$

Thus, in general, we have

$$\Pr[f(U)=1] \geq \Pr[f(V)=1] - \frac{s^2}{2|\mathcal{B}|} - \frac{s(|\mathcal{B}| - |\mathcal{B}'| - 0.5)}{|\mathcal{B}|},$$

and, in particular, for $|\mathcal{B}| - |\mathcal{B}'| = 1$, we have

$$\Pr[f(U) = 1] \geq \Pr[f(V) = 1] - \frac{s^2 + s}{2|\mathcal{B}|}\,.$$

# D Proof of Claim in Eq. (20)

For convenience, let $E_i = (\neg \mathtt{bad}_{1..i} \wedge S^* = S)$ for all $i \in [1..7]$, where $\mathtt{bad}_{1..i} = \mathtt{bad}_1 \vee \ldots \vee \mathtt{bad}_i$. Next, using the law of total probability, we have

$$\begin{aligned}\Pr\left[\mathtt{bad}_8 \mid E_1\right] &\leq \Pr\left[\mathtt{bad}_{1..7} \mid E_1\right] + \Pr\left[\mathtt{bad}_8 \mid E_7\right] \\ &\leq \sum_{i=2}^{8} \Pr\left[\mathtt{bad}_i \mid E_{i-1}\right]\,. \end{aligned} \tag{21}$$

We handle the eight terms on the RHS one by one as follows:

- $\underline{\Pr\left[\mathtt{bad}_2 \mid E_1\right]}$: Fix some $i \in \mathcal{J}$ and $j \in [1..s^*]$ and some $C \in \mathcal{C}$. There are at most $q_{\mathrm{fnd}} s$ $\underline{i'' < i, \text{ such that } C = C_{i''}}$. For any such choice the probability that $U^*_{C_i}[j] = U^*_{C_{i''}}[0]$ occurs is bounded by $2^{-h}$, in which case, the union bound yields $16\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}/2^h$.

  If no such $i''$ exists, then conditioned on $U^*_{C_i}[j] = \mathit{Label} \,\|\, 0^8 \,\|\, \star \,\|\, L$, we have that $V^*_{C_i}[j{+}1] = X^*_{i'}$ for some $i'$ with at most $2^{-h}$ probability. Again the union bound gives $16\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}/2^h$.

  Since the two cases are mutually exclusive, we can take the maximum of the two, i.e., we have

$$\Pr\left[\mathtt{bad}_2 \mid E_1\right] \leq \frac{16\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}}{2^h}\,. \tag{22}$$

- $\underline{\Pr\left[\mathtt{bad}_3 \mid E_2\right]}$: Using a similar argument and under the assumption that $\neg\mathtt{bad}_2$ holds, we have

$$\Pr\left[\mathtt{bad}_3 \mid E_2\right] \leq \frac{16\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}}{2^h}\,. \tag{23}$$

- $\underline{\Pr\left[\mathtt{bad}_4 \mid E_3\right]}$: Again, fix some $i \in \mathcal{J}$ and $j \in [1..s^*]$ and some $C \in \mathcal{C}$. There are at most $q_{\mathrm{fnd}} s$ $i'' < i$, such that $C = C_{i''}$. For any such choice the probability that $U^*_{C_i}[j] \,\|\, U^*_{C_i}[0] = U^*_{C_{i''}}[0]$ occurs is bounded by $2^{-h}$, in which case the union bound yields $16\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}/2^h$.

  If no such $i''$ exists then conditioned on $\mathtt{msb}_{|\mathit{Label}|+8}(U^*_{C_i}[j]) = \mathit{Label} \,\|\, 0^8$, we have that $R^*_{C_i}[j] = X^*_{i'}$ for some $i'$ with at most $2^{-h}$ probability. Again the union bound gives $16\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}/2^h$.

  Since the two cases are mutually exclusive, we can take the maximum of the two, i.e., we have

$$\Pr\left[\mathtt{bad}_4 \mid E_3\right] \leq \frac{16\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}}{2^h}\,. \tag{24}$$

- $\underline{\Pr\left[\mathtt{bad}_5 \mid E_4\right]}$: Using a similar argument and under the assumption that $\neg\mathtt{bad}_4$ holds, we have

$$\Pr\left[\mathtt{bad}_5 \mid E_4\right] \leq \frac{16\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}}{2^h}\,. \tag{25}$$

- $\underline{\Pr\left[\mathtt{bad}_6 \mid E_5\right]}$: Under the assumption that $E_5$ holds, we can use similar arguments as before to get

$$\Pr\left[\mathtt{bad}_6 \mid E_5\right] \leq \frac{16\overline{q_{\mathrm{fnd}}}^2 s^2_{\max}}{2^h}\,. \tag{26}$$

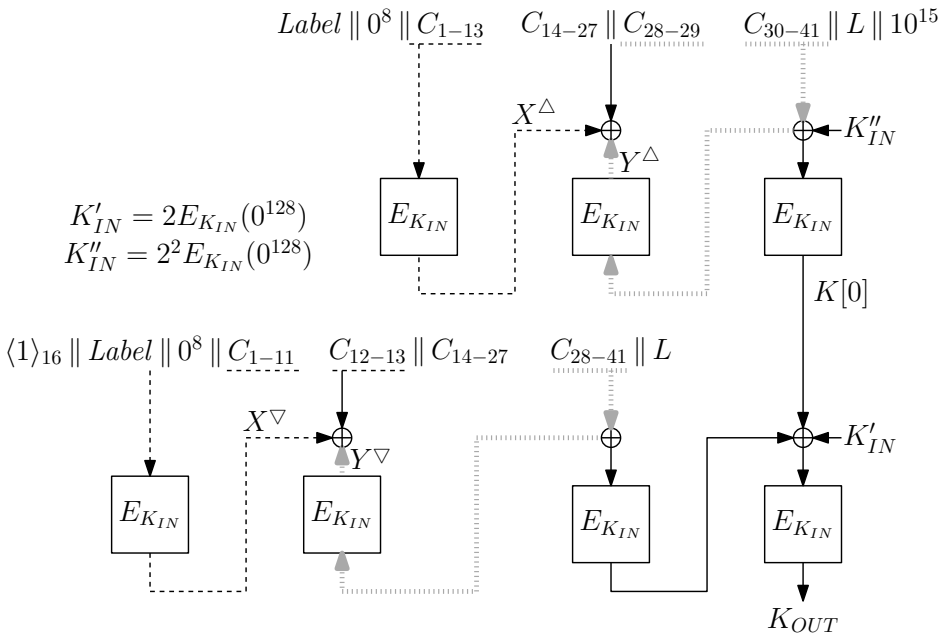


Figure 14: Birthday Attack on stCTR-CMAC[$E$].

- $\underline{\Pr[\mathtt{bad}_7 \mid E_6]}$: Using similar arguments as used in the previous proofs, we have

$$\Pr[\mathtt{bad}_7 \mid E_6] \leq \frac{\overline{q_{\mathrm{fnd}}} s_{\max}^2}{2^h}\,. \tag{27}$$

- $\underline{\Pr[\mathtt{bad}_8 \mid E_7]}$: Fix any $i \in \mathcal{J}$. Under the assumption that $E_7$ holds, we know that $\overline{K^*_{\perp} \,\|\, R^*_{C_i}[j]}$ is a fresh query for all $j \in [1..s]$ given the previous history. Thus, $T^*_{C_i}$ is uniformly distributed on $\{0,1\}^L$. Then, using similar arguments as used in the proof of Theorem 2, we have

$$\Pr[\mathtt{bad}_8 \mid E_7] \leq \delta_{p,L}\,. \tag{28}$$

The result follows by summing Eqs. (21)–(28). □

# E Attack on stCTR-CMAC[$E$]

When $L$ is 1 block, stCTR-CMAC[$E$] first computes $K[0] \leftarrow$ CMAC($K_{IN}$, $Label \,\|\, 0^8 \,\|\, Context \,\|\, L$), and then computes $K_{OUT}$ as $K_{OUT} \leftarrow$ CMAC($K_{IN}$, $\langle 1\rangle_r \,\|\, Label \,\|\, 0^8 \,\|\, Context \,\|\, L \,\|\, K[0]$). As usual, we again assume $r = 16$ bits. As in the case of stDP-CMAC, the bit positions of the *Context* for computing $K[0]$ and $K_{OUT}$ are yet again desynchronized by 16 bits, resulting in a sightly higher attack complexity.

The attack works almost similar to the one for stDP-CMAC in Sect. 6.2 and uses *Context* of 41 bytes. Let $C_i, 1 \leq i \leq 41$ denote the $i$-th byte of *Context*, and similarly sub-indices of a variable denote byte positions of that variable. In the computation of $K[0]$, $M[1]$ consists of 2 bytes of *Label*, 8 bits of zeros, followed by 13 bytes of *Context*. $M[2]$ fully consists of *Context*. $M[3]$ consists of 12 bytes of *Context* and 2 bytes of the encoding of $L$, and the one-zero padding is applied to fill the remaining 2 bytes. In this attack, $K_{IN}$ and $K_{OUT}$ are given, and the goal is to find *Context*. The attacker uses $C_1$ to $C_{13}$ as degrees of freedom for one part of the MitM attack and uses $C_{28}$ to $C_{41}$ for the other part. The full attack is described as follows:

1. $K[0]$ is fixed to an arbitrary value, which also fixes $E^{-1}_{K_{IN}}(K[0])$ and $E^{-1}_{K_{IN}}(K[0] \oplus K'_{IN} \oplus E^{-1}_{K_{IN}}(K_{OUT}))$.

2. For $2^{72}$ choices of $C_1$ to $C_{13}$, compute two $n$-bit values $(X^{\triangle}, X^{\triangledown})$ by $X^{\triangle} = E_{K_{IN}}(M[1])$ and $X^{\triangledown} = E_{K_{IN}}(\langle 1 \rangle_{16} \,\|\, M[1] \gg 16)$, and store

$$X^{\triangledown}_{1-2} \oplus C_{12-13} \,\|\, X^{\triangle}_{1-14} \oplus X^{\triangledown}_{3-16} \,\|\, X^{\triangle}_{15-16}$$

   in a hash table. (broken lines in Fig. 14.)

3. For $2^{72}$ choices of $C_{28}$ to $C_{41}$, compute two $n$-bit values $(Y^{\triangle}, Y^{\triangledown})$ by $Y^{\triangle} = E^{-1}_{K_{IN}}(E^{-1}_{K_{IN}}(K[0]) \oplus M[3] \oplus K^{"}_{IN})$ and $Y^{\triangledown} = E^{-1}_{K_{IN}}(E^{-1}_{K_{IN}}(E^{-1}_{K_{IN}}(K_{OUT}) \oplus K[0] \oplus K'_{IN}) \oplus (M[3] \gg 16)$, and store

$$Y^{\triangledown}_{1-2} \,\|\, Y^{\triangle}_{1-14} \oplus Y^{\triangledown}_{3-16} \,\|\, Y^{\triangle}_{15-16} \oplus C_{28-29}$$

   in a hash table. (dotted gray lines in Fig. 14.)

4. Find a pair meeting the following equality of total 18 bytes.

$$\begin{aligned} X^{\triangledown}_{1-2} \oplus C_{12-13} &= Y^{\triangledown}_{1-2}\,, \\ X^{\triangle}_{15-16} &= Y^{\triangle}_{15-16} \oplus C_{28-29}\,, \\ X^{\triangle}_{1-14} \oplus X^{\triangledown}_{3-16} &= Y^{\triangle}_{1-14} \oplus Y^{\triangledown}_{3-16}\,. \end{aligned}$$

The left-hand side of the equality can be computed depending on $C_1$ to $C_{13}$ and the right-hand side can be computed depending on $C_{28}$ to $C_{41}$, hence the meet-in-the-middle attack is applied. The size of the match is 18 bytes and the attack complexity is $2^{72}$.

# F XTS Weak Keys with FB-CMAC[$E$]

The attacks in Sect. 6 targeted 1 block of $K_{OUT}$. Meanwhile, there exist applications where attackers' capability of generating more than 1 block of $K_{OUT}$ is practically relevant. For instance $K_{OUT} = K[1] \,\|\, K[2]$ with $K[1] = K[2]$ (also referred to as a duplicated-key) is a weak key for XTS. (See Sect. 3.3 for more examples of weak keys.) Here, we write the 2-block targets of such a form.

We first observe that, CTR-CMAC[$E$] cannot produce such $K_{OUT}$, because for any common suffix Suffix, CMAC's outputs for $\langle i \rangle \,\|\, \text{Suffix}$ and $\langle j \rangle \,\|\, \text{Suffix}$ with $i \neq j$ are distinct with probability of 1, namely $K[1] \neq K[2]$ with certainty.[10]

Our second observation is that FB-CMAC[$E$] can produce duplicated-key with $O(2^{n/2})$ complexity. Note that the counter is not used in FB-CMAC[$E$]. Then, 2 blocks of $K_{OUT}$ are computed as follows. First, $K[0] \leftarrow IV$. Then,

$$\begin{aligned} K[1] &\leftarrow \mathrm{CMAC}(K_{IN}, K[0] \,\|\, \mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L)\,, \\ K[2] &\leftarrow \mathrm{CMAC}(K_{IN}, K[1] \,\|\, \mathit{Label} \,\|\, 0^8 \,\|\, \mathit{Context} \,\|\, L)\,. \end{aligned}$$

According to SP 800-108 [Che22], "... *the length for an IV should be specified by the application or protocol* ..." When the length of $IV$ is not a multiple of $n$, attackers can adjust the length of

[10] In Appendix G we show that beyond this obvious limitation, CTR-CMAC is still amenable to birthday-bound attack, producing $K_{OUT} = K[1] \,\|\, K[2]$, where $(K[1], K[2])$ satisfies some efficiently verifiable relation, other than equality.

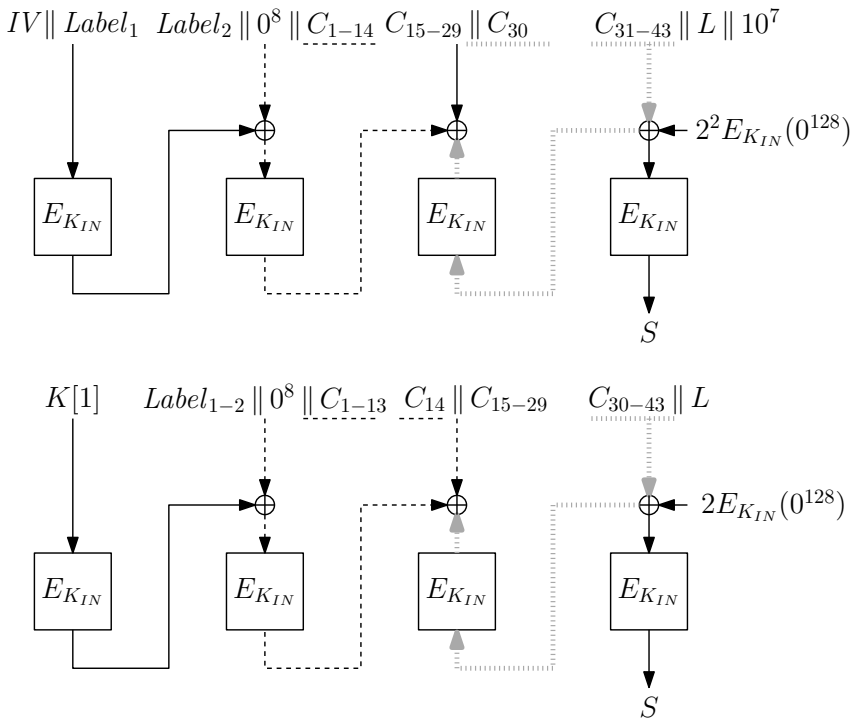


Figure 15: Generating XTS weak key with FB-CMAC[$E$].

*Context* such that the one-zero padding is applied for computing $K[1]$, while it is not for $K[2]$. This also differentiates the mask value when the last message block is XORed; $K'_{IN}$ and $K''_{IN}$. Those enable the attacker to generate a *Context* yielding $K[1] = K[2]$.

The attack is illustrated in Fig. 15. Here we assume that, the length of $IV$ is 15 bytes and the length of the context is 43 bytes. With this configuration, we expect that the $n$-bit state values after the encryption of first blocks ($IV$ and $K[1]$) are different, and similarly the $n$-bit state values before the XOR of the last message blocks are different. The meet-in-the-middle attack can be mounted by using $C_1 - C_{14}$ and $C_{30} - C_{43}$ for degrees of freedom. Due to a similarity with the earlier attacks, we omit the details here. The attack complexity is $2^{n/2+r/2}$ as mentioned in Sect. 6.3, and with $n = 128$ bits and $r = 8$ bits, this becomes $2^{68}$. In Appendix G, we provide a generalization of the above attack that can produce a two block $K_{OUT}$ satisfying some arbitrary efficiently-verifiable relation.

# G Two-Block Attacks on FB-CMAC[$E$] and CTR-CMAC[$E$]

In Appendix F, we gave an attack where the adversary can produce duplicated-key, i.e. $K_{OUT} = K[1] \| K[2]$ with $K[1] = K[2]$, for FB-CMAC[$E$]. As it turns out, the adversary can generalize without extra effort. Specifically, we now present an attack that can produce a 2-block $K_{OUT}$ such that $K[1] \| K[2]$ satisfies any adversary-chosen (efficiently) verifiable relation $R$. Indeed, the attack in Appendix F is a special case satisfying the equality relation $K[1] = K[2]$.

The attack assumes $IV$ length of 15 bytes and produces context of length 43 bytes. $C_i$, $1 \le i \le 43$ denotes the $i$-th byte of *Context*, and similarly sub-indices of a variable denote byte positions of that variable. Let $K[0] = IV$. In the attack, $(K[1], K[2])$ is fixed to a pair satisfying the relation $R$. Then, by definition, we have

$$K[1] = \mathrm{CMAC}(K_{IN}, M[1] \| M[2] \| M[3] \| M[4])$$

$$K[2] = \mathrm{CMAC}(K_{IN}, M'[1] \,\|\, M'[2] \,\|\, M'[3] \,\|\, M'[4])$$

where

$$\begin{aligned} M[1] &= K[0] \,\|\, \mathit{Label}_1 & M'[1] &= K[1] \\ M[2] &= \mathit{Label}_2 \,\|\, 0^8 \,\|\, C_{1-14} & M'[2] &= \mathit{Label} \,\|\, 0^8 \,\|\, C_{1-13} \\ M[3] &= C_{15-30} & M'[3] &= C_{14-29} \\ M[4] &= C_{31-43} \,\|\, L \,\|\, 10^7 & M'[4] &= C_{30-43} \,\|\, L \end{aligned}$$

The adversary knows $K_{IN}$ and it finds a valid context as follows:

1. For $2^{n/2}$ choices of $C_1$ to $C_{14}$, compute two $n$-bit values $(X^{\triangle}, X^{\triangledown})$ by $X^{\triangle} = E_{K_{IN}}(E_{K_{IN}}(M[1]) \oplus M[2])$ and $X^{\triangledown} = E_{K_{IN}}(E_{K_{IN}}(M'[1]) \oplus M'[2])$, and store
$$X_1^{\triangledown} \oplus C_{14} \,\|\, X_{1-15}^{\triangle} \oplus X_{2-16}^{\triangledown} \,\|\, X_{16}^{\triangle}$$
in a hash table.

2. For $2^{n/2}$ choices of $C_{30}$ to $C_{43}$, compute two $n$-bit values $(Y^{\triangle}, Y^{\triangledown})$ by $Y^{\triangle} = E_{K_{IN}}^{-1}(E_{K_{IN}}^{-1}(K[1]) \oplus M[4] \oplus K_{IN}^{"})$ and $Y^{\triangledown} = E_{K_{IN}}^{-1}(E_{K_{IN}}^{-1}(K[2]) \oplus M'[4] \oplus K'_{IN})$, and store
$$Y_1^{\triangledown} \,\|\, X_{1-15}^{\triangle} \oplus X_{2-16}^{\triangledown} \,\|\, Y_{16}^{\triangle} \oplus C_{30}$$
in a hash table.

3. Find a pair meeting the following equality of total 17 bytes.
$$\begin{aligned} X_1^{\triangledown} \oplus C_{14} &= Y_1^{\triangledown}\,, \\ X_{16}^{\triangle} &= Y_{16}^{\triangle} \oplus C_{30}\,, \\ X_{1-15}^{\triangle} \oplus X_{2-16}^{\triangledown} &= Y_{1-15}^{\triangle} \oplus Y_{2-16}^{\triangledown}\,. \end{aligned}$$

The left-hand side of the equality can be computed depending on $C_1$ to $C_{14}$ and the right-hand side can be computed depending on $C_{30}$ to $C_{43}$, hence the meet-in-the-middle attack is applied. The size of the match is 17 bytes and the attack complexity is $2^{68}$.

We give a simple modification of the above attack that can be applied on CTR-CMAC[$E$] for all relations admitting at least one pair $(K[1], K[2])$ satisfying $K[1] \neq K[2]$. (This automatically excludes the impossible equality relation.)

The attack assumes counter length $r = 16$ bits and produces context of length 41 bytes. By definition, we have

$$\begin{aligned} K[1] &= \mathrm{CMAC}(K_{IN}, M[1] \,\|\, M[2] \,\|\, M[3]) \\ K[2] &= \mathrm{CMAC}(K_{IN}, M'[1] \,\|\, M'[2] \,\|\, M'[3]) \end{aligned}$$

where

$$\begin{aligned} M[1] &= \langle 1 \rangle_{16} \,\|\, \mathit{Label} \,\|\, 0^8 \,\|\, C_{1-11} & M'[1] &= \langle 2 \rangle_{16} \,\|\, \mathit{Label} \,\|\, 0^8 \,\|\, C_{1-11} \\ M[2] &= C_{12-27} & M'[2] &= C_{12-27} \\ M[3] &= C_{28-41} \,\|\, L & M'[3] &= C_{28-41} \,\|\, L \end{aligned}$$

Now, the attack follows the same approach as above, creating a meet-in-the-middle collision to fix $C_{12-27}$ by utilizing the degree of freedoms at $C_{1-11}$ (using $E_{K_{IN}}(\cdot)$ calls) and $C_{28-41}$ (using $E_{K_{IN}}^{-1}(\cdot)$ calls).

## H Note on Attacks against HMAC-based KDFs

The attacks on CMAC-based KDFs cannot be applied to HMAC- and KMAC-based KDFs. This is because, for CMAC, the knowledge of the key helps adversaries to invert the last block cipher from $K_{OUT}$, while hash functions in HMAC and KMAC are not invertible even with the knowledge of the key. That is, preimage resistance of hash functions helps preventing the attack. However, for HMAC, there is still room for attacks if the premise are relaxed to allow second-preimage like attacks. Namely, not only the target $K_{OUT}$, the corresponding input including $K_{IN}$, *Label*, *Context* are given to the adversary, and the goal is now to find another *Context*$'$ producing the same $K_{OUT}$. Note that in many real use cases discussed in Sect. 1, *Label* and *Context* are public, thus the adversary can observe them.

The attack needs to focus on a single call of HMAC in the KDF, hence the attack targets are CTR-HMAC and FB-HMAC with a single output block. The applicability of the second-preimage attack depends on the underling primitive. Standard choices often used in practice are SHA-1 and SHA-256. Those are known to allow long message attacks, namely, for a first-preimage of $2^\ell$ blocks, a second-preimage can be generated at a computational cost of $2^{160-\ell}$ and $2^{256-\ell}$, respectively [KS05]. When $\ell$ is exceptionally long, second-preimages can be generated with the same cost as the birthday paradox for the output size, which matches the bounds in Table 1. Note that, strictly speaking, the maximum value of $\ell$ for SHA-1 and SHA-256 are 64, thus the attack efficiency cannot reach the birthday-bound. Moreover, using such a long input as *Context* for the KDF is unlikely in practice. This attack is of theoretical interest only. Nevertheless, we want to mention it because it may give an intuition of the birthday bound proven for CTR-HMAC and FB-HMAC.

An interesting observation is that NIST SP 800-108 [Che22, Appendix D] refers to the preimage resistance of hash functions as a security level for the KDF usage.

> *This Recommendation uses the strength of the preimage resistance of a hash function as a contributing factor when determining the security strength provided by a key-derivation function.*

The attack in this section indicates that the second-preimage resistance could also be relevant to the security of KDFs in the context of key control security. We hope that this paper will lead to a deeper understanding of KDF and enable more accurate management of standardization techniques.