
Key Control Security of Key Derivation Functions from NIST SP 800-108

Tetsu Iwata
Nagoya University

The 1st Workshop on
Generic Attacks and Proofs in Symmetric Cryptography, GAPS 2025
September 5, 2025, NTU, Singapore

Key Control Security of Key Derivation Functions from NIST SP 800-108



and PRF

Tetsu Iwata
Nagoya University

The 1st Workshop on
Generic Attacks and Proofs in Symmetric Cryptography, GAPS 2025
September 5, 2025, NTU, Singapore

This Talk

- based on:
 - Ritam Bhaumik, Avijit Dutta, Akiko Inoue, Tetsu Iwata, Ashwin Jha, Kazuhiko Minematsu, Mridul Nandi, Yu Sasaki, Meltem Sonmez Turan, and Stefano Tessaro, [Cryptographic Treatment of Key Control Security -- In Light of NIST SP 800-108](#)
 - CRYPTO 2025
 - ePrint 2025/1123
 - Ritam Bhaumik, Avijit Dutta, Tetsu Iwata, Ashwin Jha, Kazuhiko Minematsu, Mridul Nandi, Yu Sasaki, Meltem Sonmez Turan, and Stefano Tessaro, [A Note on Feedback-PRF Mode of KDF from NIST SP 800-108](#)
 - already sent to ePrint, will appear soon

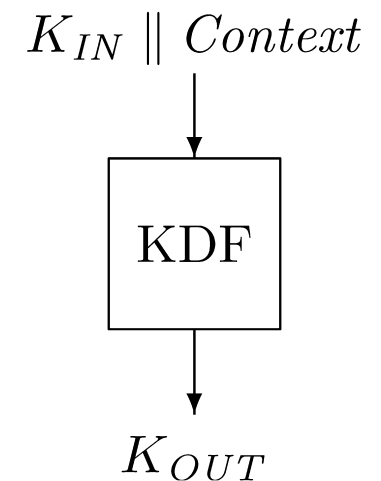


Outline

- Key Derivation Functions
- Security Requirements for KDFs
- Formalization of Key Control Security
- Proofs
- Attacks
- Summary

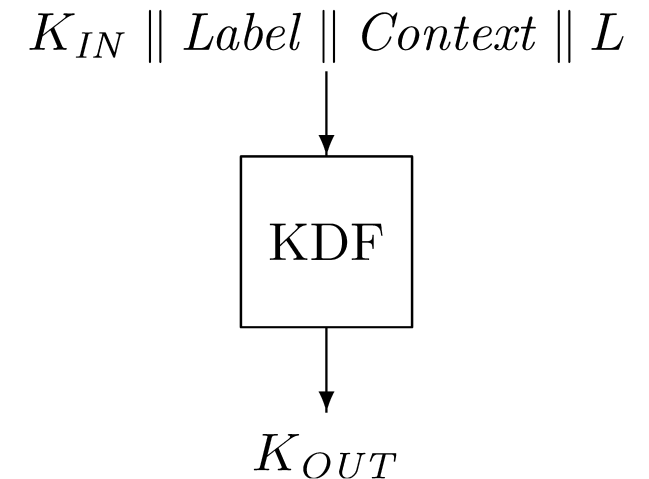
Key Derivation Functions, KDFs

- Key derivation function
 - outputs multiple session keys/user keys K_{OUT} from a single master key K_{IN}
 - takes other inputs
- Significantly used in practice
 - various OSs, HSM, cryptographic libraries, TEE, ...
- Various design approaches
 - HKDF [Kra10]
 - the extract-then-expand approach
 - Chuah et al. [WDNS12, WDS13]
 - Ones rely on passwords or biometrics [PJ16, KAA21, SPL+18]
 - low entropy secret



KDFs in NIST SP 800-108r1

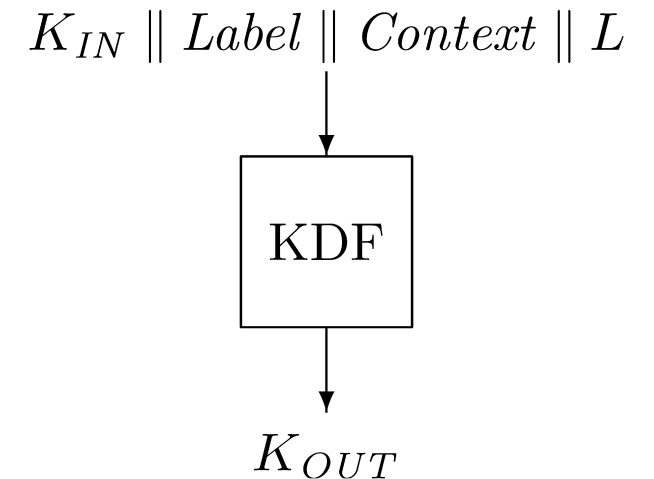
- We focus on KDFs in NIST SP 800-108r1 [Che22]
 - KDFs from a PRF
 - The input key K_{IN} is a cryptographic key
 - K_{IN} is a “cryptographically strong key,” no extraction step
- Other inputs:
 - *Label*: a bit string identifies the purpose of K_{OUT}
 - “encryption”, “authentication”,...
 - *Context*: a bit string containing the info. related K_{OUT}
 - identities of the users, nonces, ...
 - L : bit length of K_{OUT}



[Che22] Lily Chen. Recommendation for key derivation using pseudorandom functions. NIST Special Publication NIST SP 800-108r1, 2022.

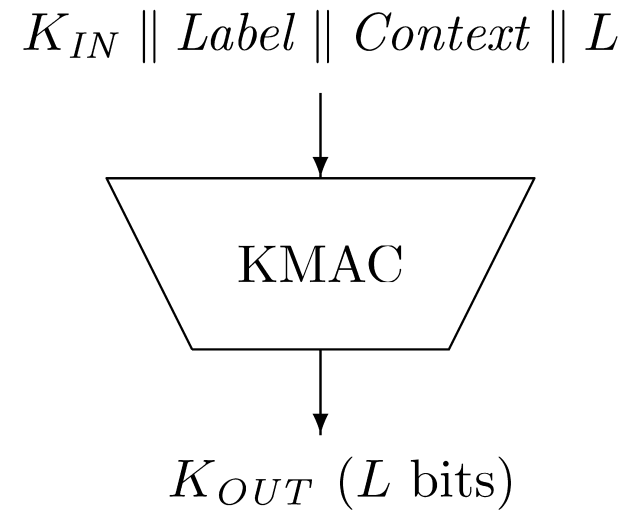
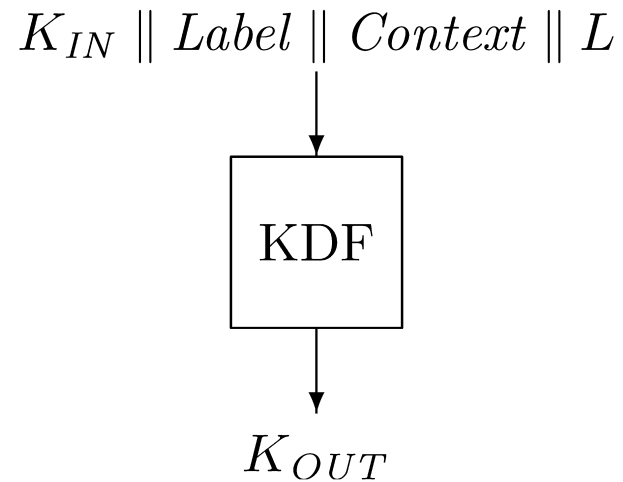
KDFs in NIST SP 800-108r1

- NIST SP 800-108r1 defines KDFs based on PRFs
 - PRFs: KMAC, CMAC, HMAC
- KDFs in NIST SP 800-108r1
 - KDF-KMAC
 - Three modes for CMAC and HMAC
 - Counter mode, CTR
 - Feedback mode, FB
 - Double-pipeline mode, DP
 - Three “strengthened” modes for CMAC
 - stCTR, stFB, stDP



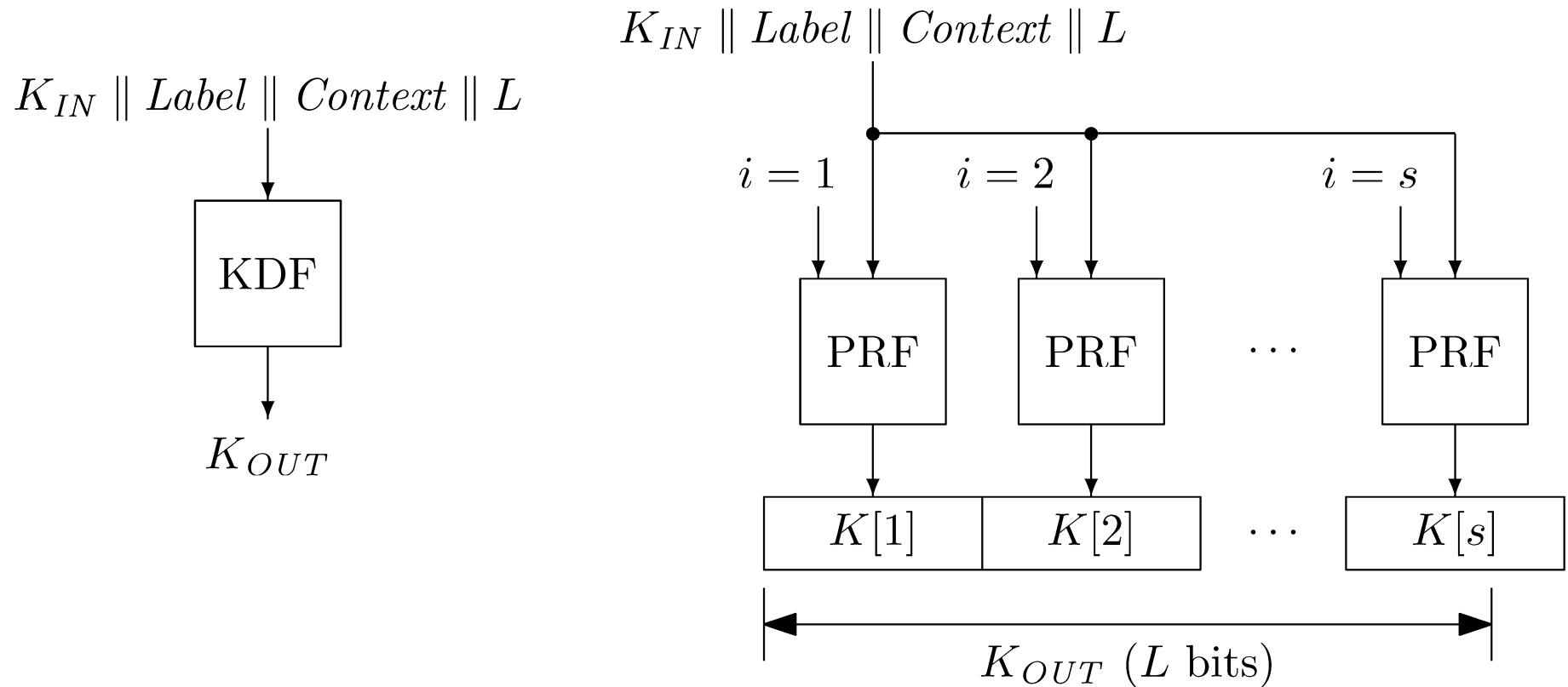
KDF-KMAC

- based on KMAC as a PRF
 - KMAC is a variable-output length PRF



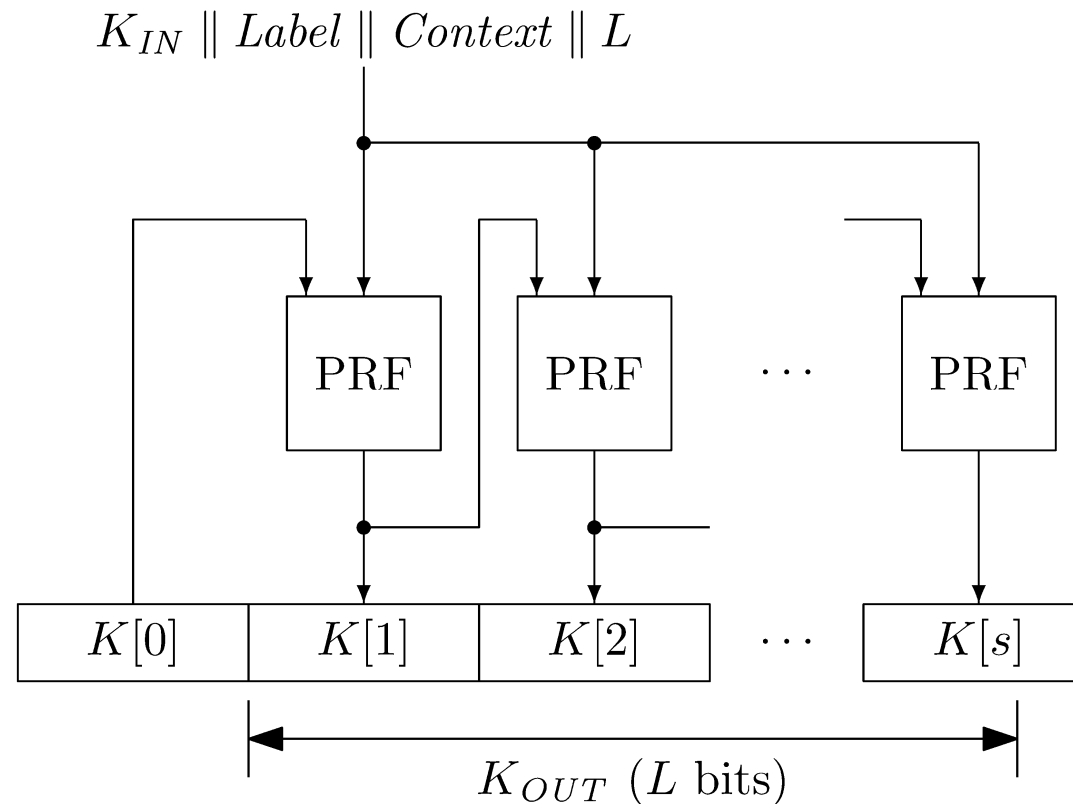
CTR-PRF

- PRF in Counter mode, PRF is CMAC or HMAC
- There is also a “strengthened version” for CMAC, called stCTR



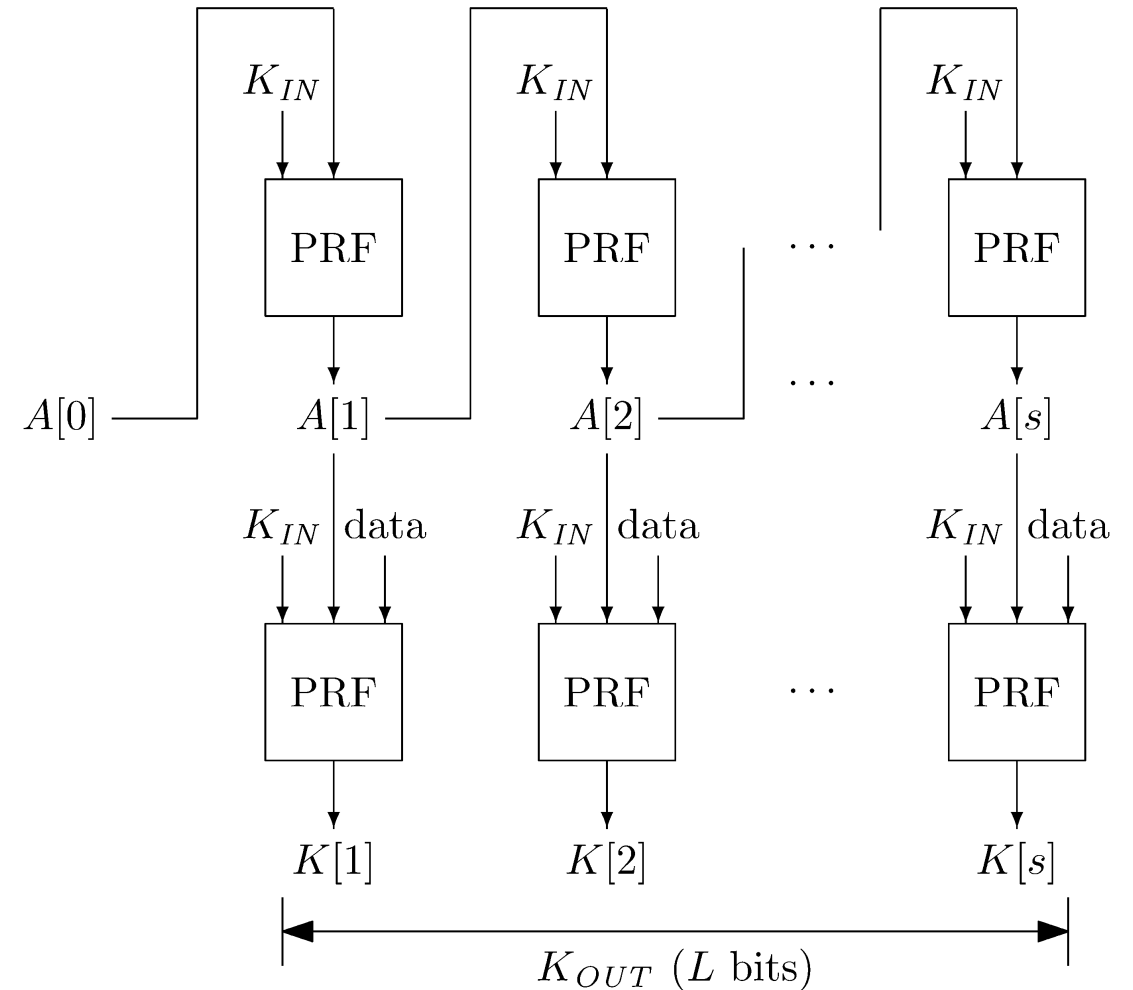
FB-PRF

- PRF in Feedback mode, PRF is CMAC or HMAC
- There is also a “strengthened version” for CMAC, called stFB



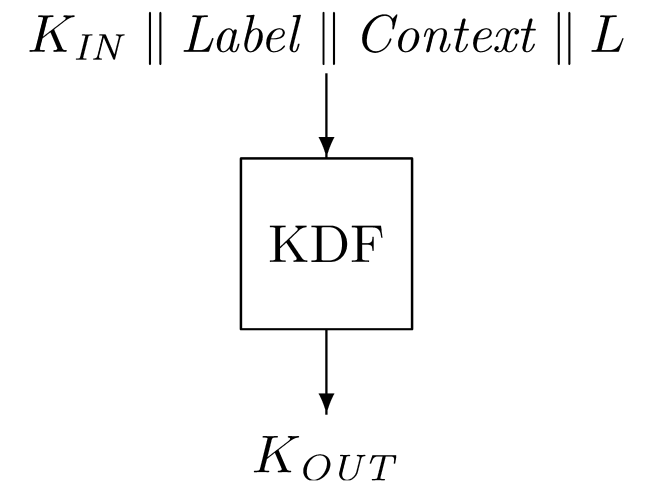
DP-PRF

- PRF in Double-Pipeline mode
- Combination of CTR and FB modes
- PRF is CMAC or HMAC
- $A[0] = \text{data} = \text{Label} \parallel \text{Context} \parallel L$
- There is also a “strengthened version” for CMAC, called stDP



KDFs in NIST SP 800-108r1

- Originally published in 2008
- Revised in 2022
 - KDF-KMAC was added
 - An issue of the **key control security** of CMAC was discussed



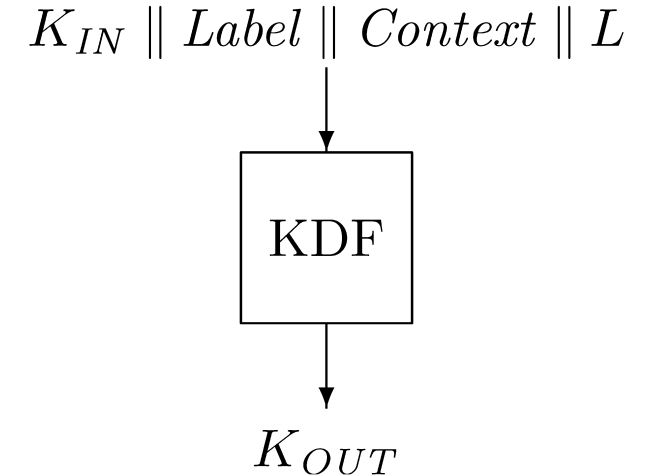


Outline

- Key Derivation Functions
- Security Requirements for KDFs
- Formalization of Key Control Security
- Proofs
- Attacks
- Summary

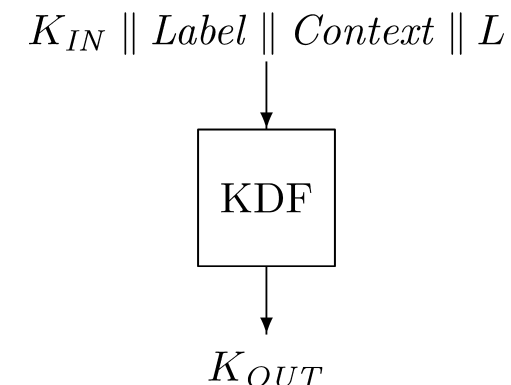
Security Requirements for KDFs

- A KDF itself is a PRF (for random and secret K_{IN})
 - See [SWG25], covering the PRF proofs of {CTR, FB, DP}-{CMAC, HMAC}
 - [SWG25] also covers analyses of collision resistance and preimage resistance



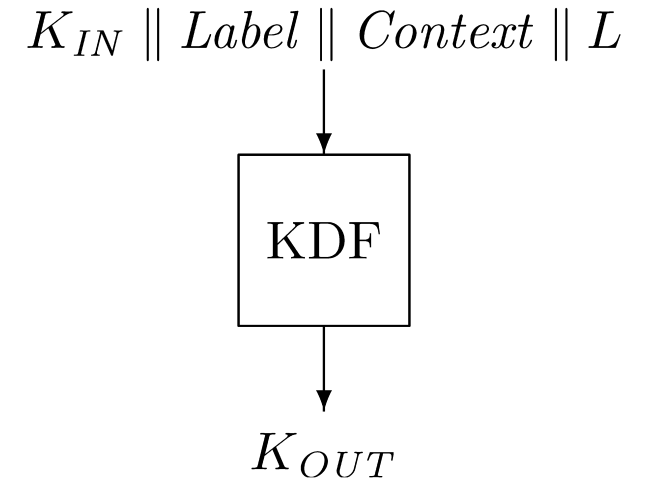
Security Requirements for KDFs

- 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 input 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 preselected value (regardless of the contributions of the others) to the detriment of any applications relying on that keying material.
- Added in the revision in 2022 based on the public comments from Amazon team



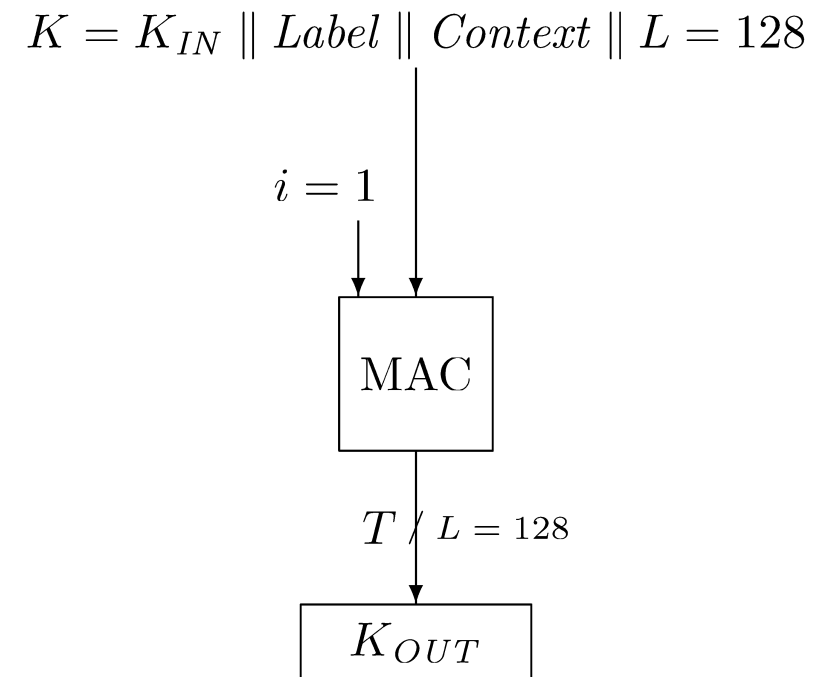
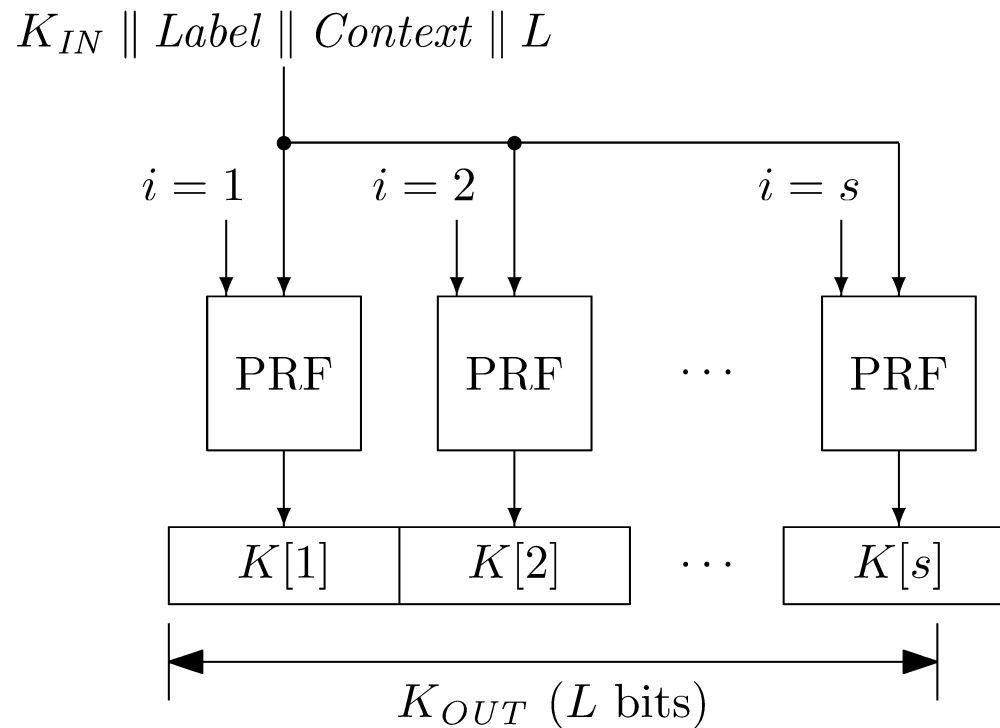
Key Control Security [Che22]

- The adversary knows K_{IN}
 - KCS is in a **known-key** setting
- The goal is to force K_{OUT} being a preselected value
 - e.g., a weak key for some cipher
- by controlling *Context*
- *Label* and L are usually determined by a higher protocol



Example: KCS of CTR-PRF

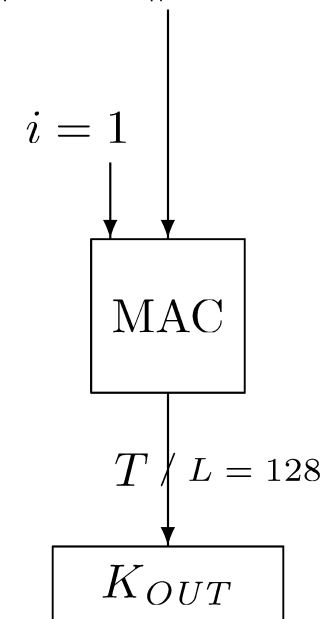
- KDF with MAC as a PRF, MAC built on a block cipher of $n = 128$ bits
- Consider the case $L = 128$; the counter is fixed to $i = 1$
- In what follows, we write K for K_{IN} , and T for the output of MAC



Example: KCS of CTR-PRF

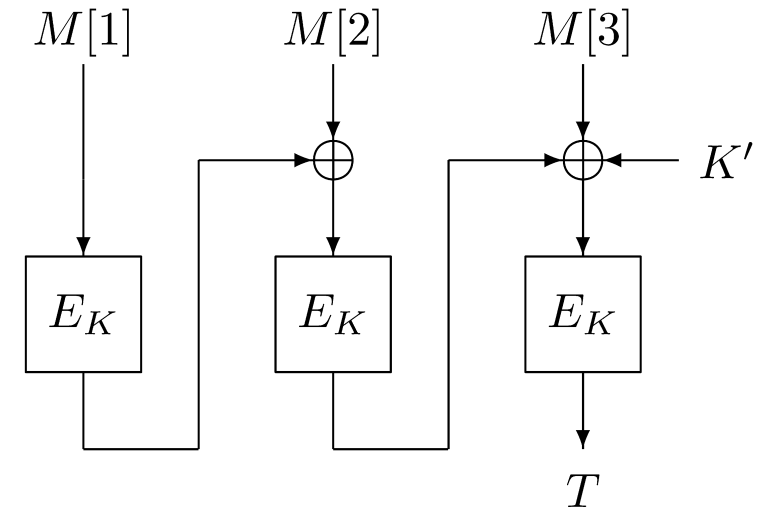
- K is given
 - Assume that *Label* and L are fixed and given, and are not under the control of the adversary
- If the adversary outputs *Context* such that the output is a preselected value $T = K_{OUT}$, then the attack succeeds
 - The KDF is insecure in terms of KCS
- Similar to a preimage attack w/ known key
 - T can be preselected, and a part of the input is fixed

$$K = K_{IN} \parallel Label \parallel Context \parallel L = 128$$

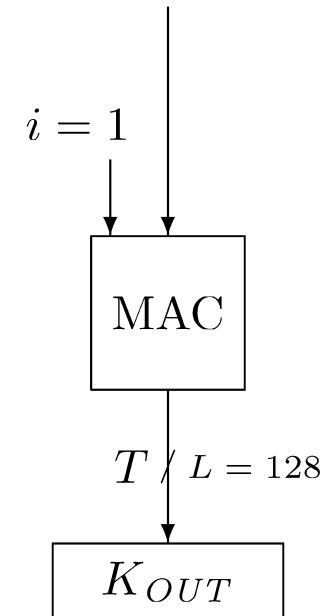


Example: KCS of CTR-PRF

- $K' = 2 \cdot E_K(0^n)$
- Format specified in NIST SP 800-108r1
- $[1]_2 \parallel \text{Label} \parallel 0x00 \parallel \text{Context} \parallel [L]_2$
- $M[1] = [1]_2 \parallel \text{Label} \parallel 0x00 \parallel \text{Context}[1]$
- $M[2] = \text{Context}[2]$
- $M[3] = \text{Context}[3] \parallel [128]_2$
- T is fixed to some preselected value
 - to a weak key of cipher, by the adversary



$$K = K_{IN} \parallel \text{Label} \parallel \text{Context} \parallel L = 128$$

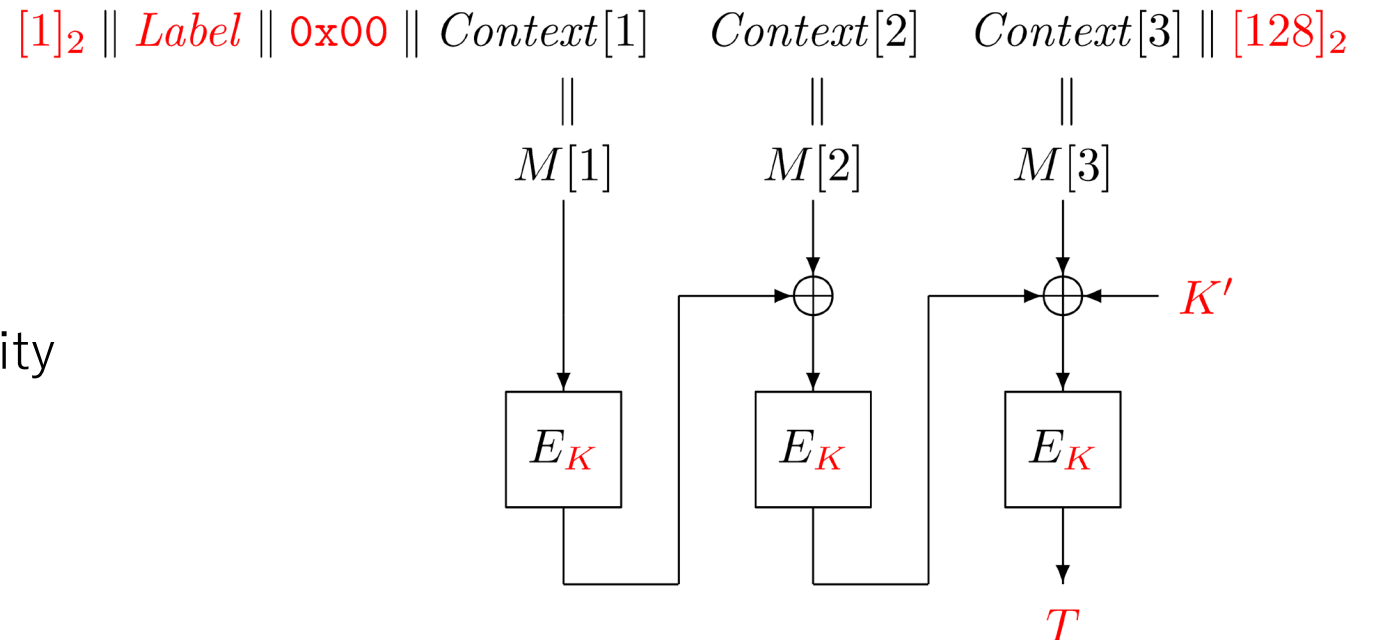


Example: KCS of CTR-PRF

- 2 KCS attacks in [Che22] (by Amazon team) that use 2 blocks
- 3-block attack

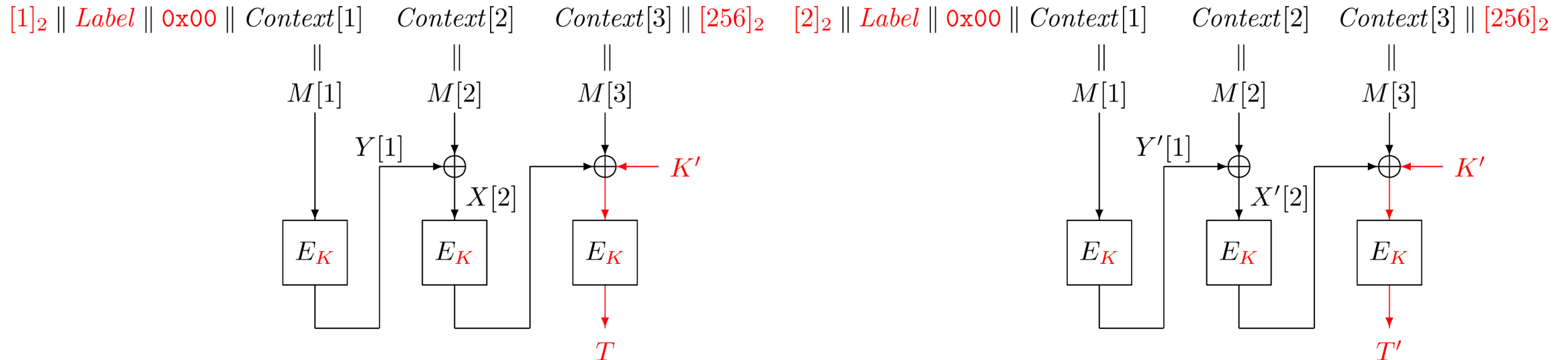
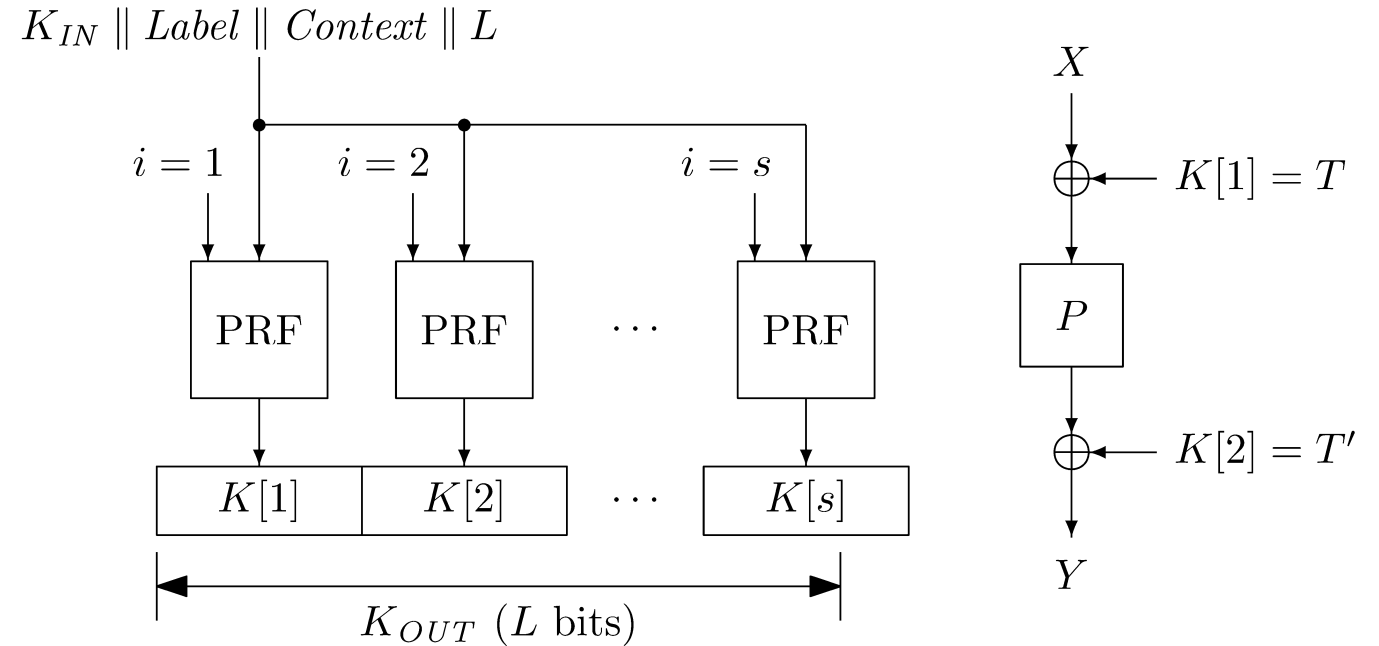
1. Fix $Context[1]$
2. Fix $Context[3]$
3. Compute $Context[2]$

- A KCS attack with $O(1)$ complexity
- NIST: Use KMAC or HMAC, or “strengthened modes”



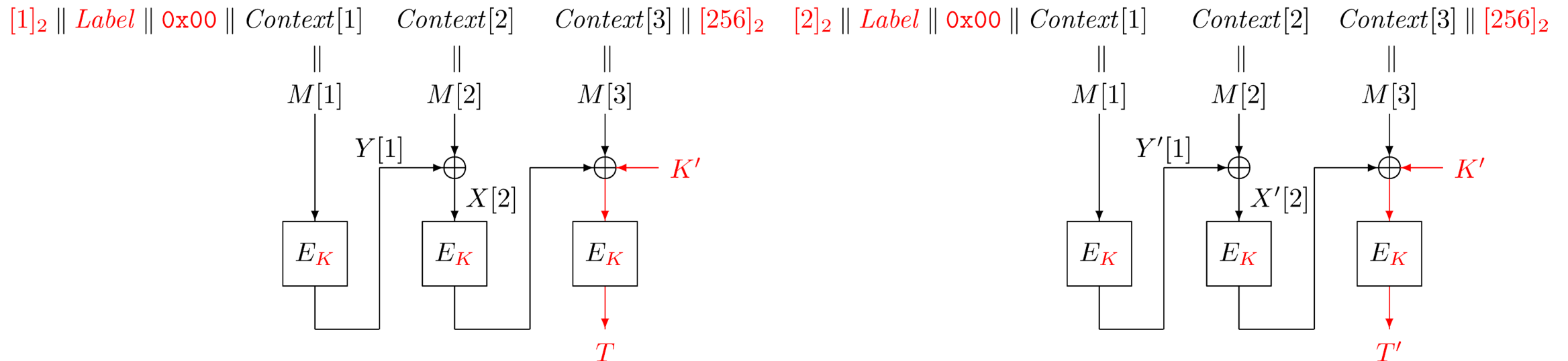
Case $L = 2n$

- Case $L = 2n$
- e.g., Even-Mansour cipher



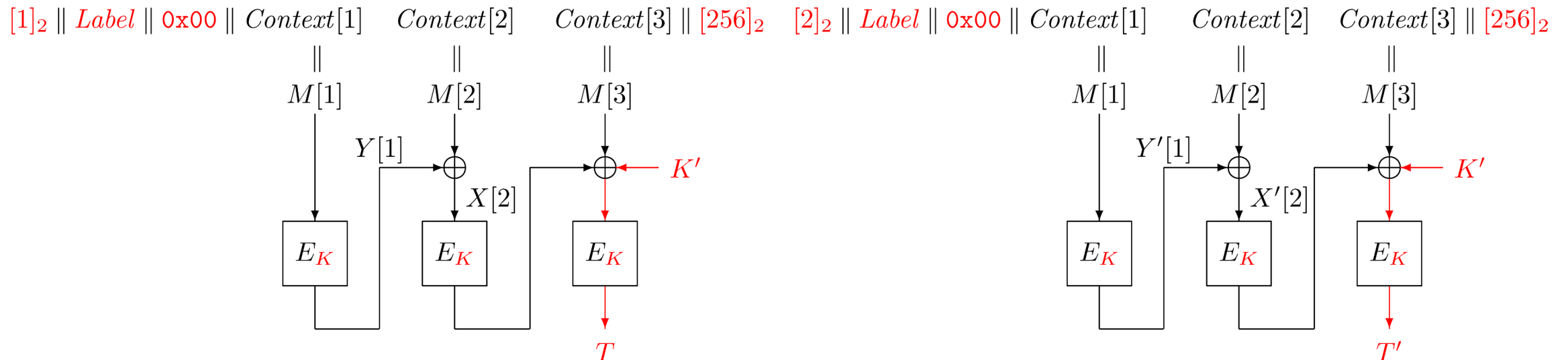
Case $L = 2n$

- The complexity of a generic attack is 2^{2n}
- Can you do it better?
 - $2^n, 2^{n/2}$, or a constant time?



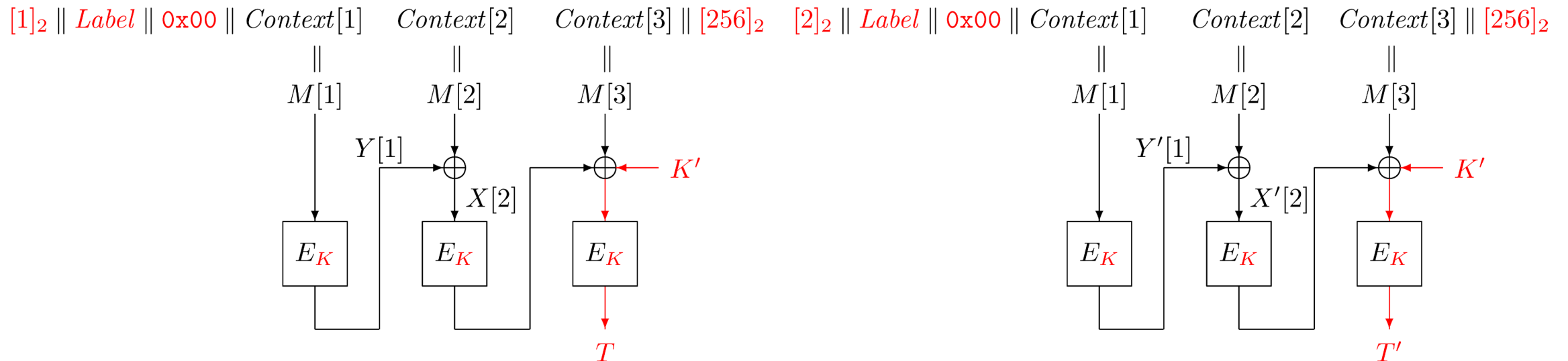
Case $L = 2n$

1. Store 2^{64} values of $Y[1] \oplus Y'[1]$ for 2^{64} values of $Context[1]$
2. Store 2^{64} values of $X[2] \oplus X'[2]$ for 2^{64} values of $Context[3]$
3. Find $Context[1]$ and $Context[3]$ such that $Y[1] \oplus Y'[1] = X[2] \oplus X'[2]$
4. Compute $Context[2] = Y[1] \oplus X[2]$ ($= Y'[1] \oplus X'[2]$)



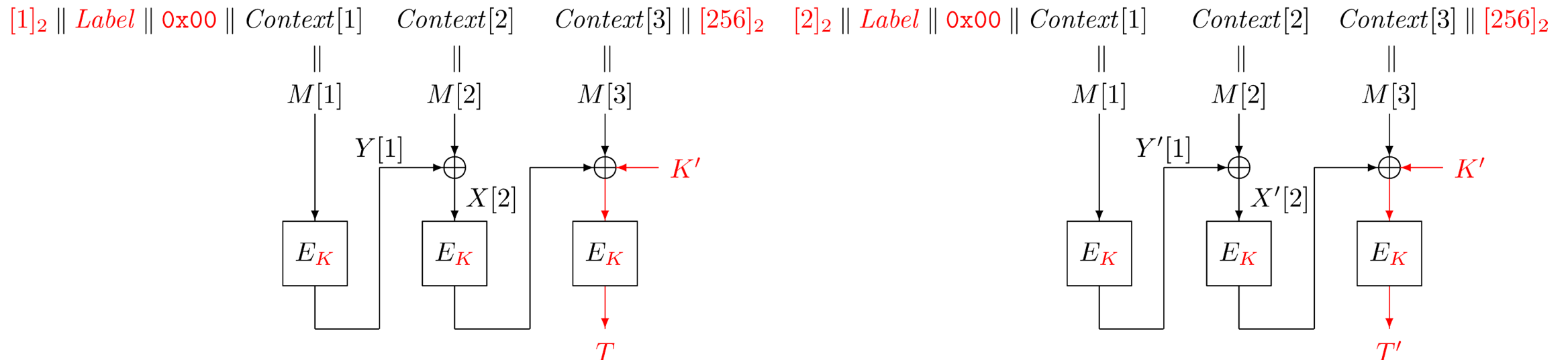
Case $L = 2n$

- MitM attack, a KCS attack with $2^{n/2} = 2^{64}$ complexity, possibly the best attack
- Assuming that 2^{64} is big, is this a secure usage of KDF-CMAC?



Case $L = 2n$

- MitM attack, a KCS attack with $2^{n/2} = 2^{64}$ complexity, possibly the best attack
- Assuming that 2^{64} is big, is this a secure usage of KDF-CMAC?
- No, it is easy to compute *Context* such that, e.g., $(T, T') = (0^n, T')$ efficiently
 - by first ignoring T'
- A “weak-key” of, e.g., Even-Mansour cipher



Key Control Security

- Key control security is close to the preimage security in a known-key setting, but **the target image can be selected by the adversary** and **the targets can be exponentially large**
 - Also close to multi-target preimage security
- A formal security definition of KCS is missing
- Our contributions
 - Formalize a security definition for KCS
 - Analysis of KDFs in NIST SP 800-108
 - proofs that KDF-KMAC and {CTR, FB, DP}-HMAC are secure
 - birthday bound attacks on {DP, stCTR, stFB, stDP}-CMAC
 - constant time attacks on {CTR, FB}-CMAC by Amazon team

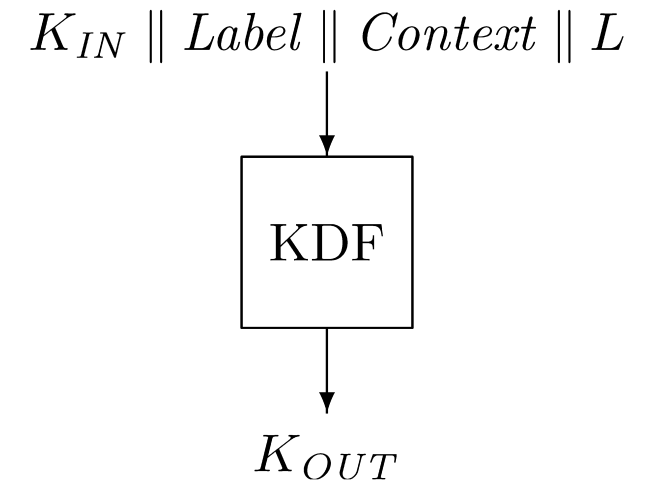


Outline

- Key Derivation Functions
- Security Requirements for KDFs
- Formalization of Key Control Security
- Proofs
- Attacks
- Summary

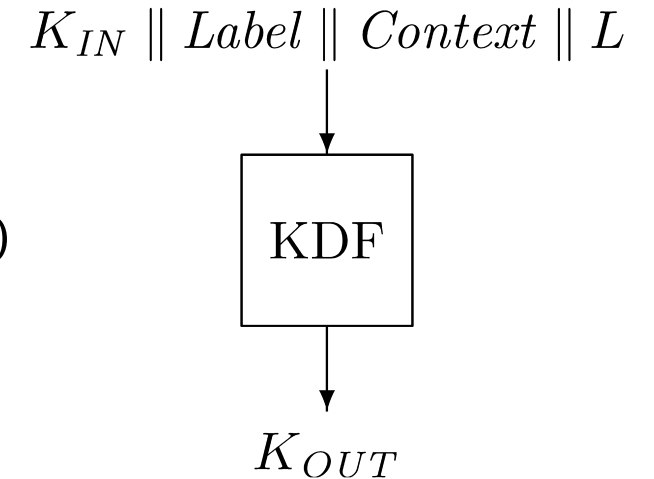
Our Formalization

- Assume the use of an ideal primitive P , could be a random oracle, or an ideal cipher
- The first approach
 - A challenger selects $(Label, L)$ and gives it to A : $(Label, L) \rightarrow A$
 - A^P chooses K_{OUT}
 - A challenger selects K_{IN} and gives it to A : $K_{IN} \rightarrow A$
 - A^P chooses $Context$
 - A wins if $K_{OUT} = \text{KDF}(K_{IN}, Label, Context, L)$
- A naïve approach following [Che22]
- Does not cover a large set of target images



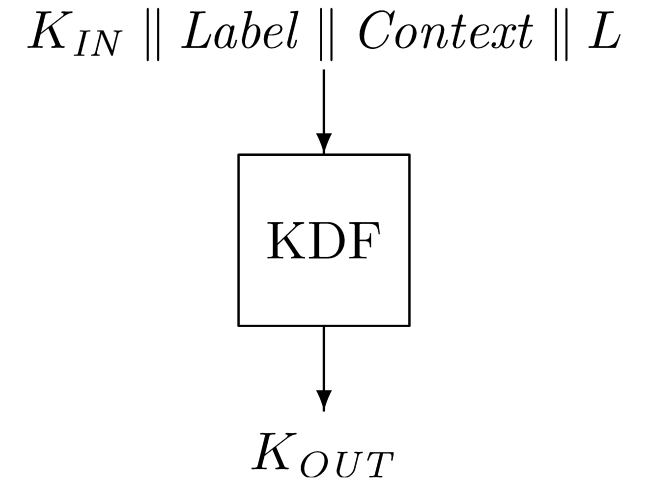
Our Formalization

- Assume the use of an ideal primitive P , could be a random oracle, or an ideal cipher
- The second approach
 - A challenger selects $(Label, L)$ and gives it to A : $(Label, L) \rightarrow A$
 - A^P chooses a predicate p
 - A challenger selects K_{IN} and gives it to A : $K_{IN} \rightarrow A$
 - A^P chooses $Context$
 - A wins if $p(K_{OUT}) = 1$ for $K_{OUT} = \text{KDF}(K_{IN}, Label, Context, L)$
- A predicate $p: \{0,1\}^L \rightarrow \{0,1\}$ specifies a set of target images
 - K_{OUT} is a target image iff $p(K_{OUT}) = 1$
- can handle a large set of target images



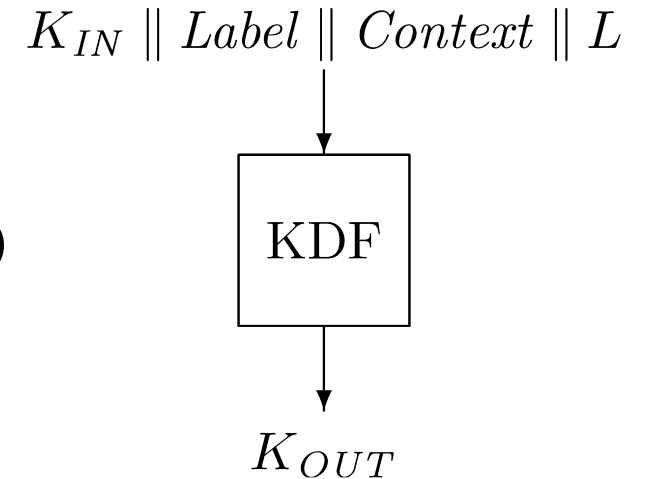
Our Formalization

- Stronger adversaries could choose $(Label, L)$, and could even affect the generation process of K_{IN}
- Treat K_{IN} as an oracle-dependent adversarial source of randomness
 - follows [CDKT19] analyzing random number generators
- We let the adversary choose $(Label, L)$ and K_{IN}
 - with a suitable restriction;
 - $|Label|$ and L must be in a suitable range
 - K_{IN} has a sufficient average min-entropy



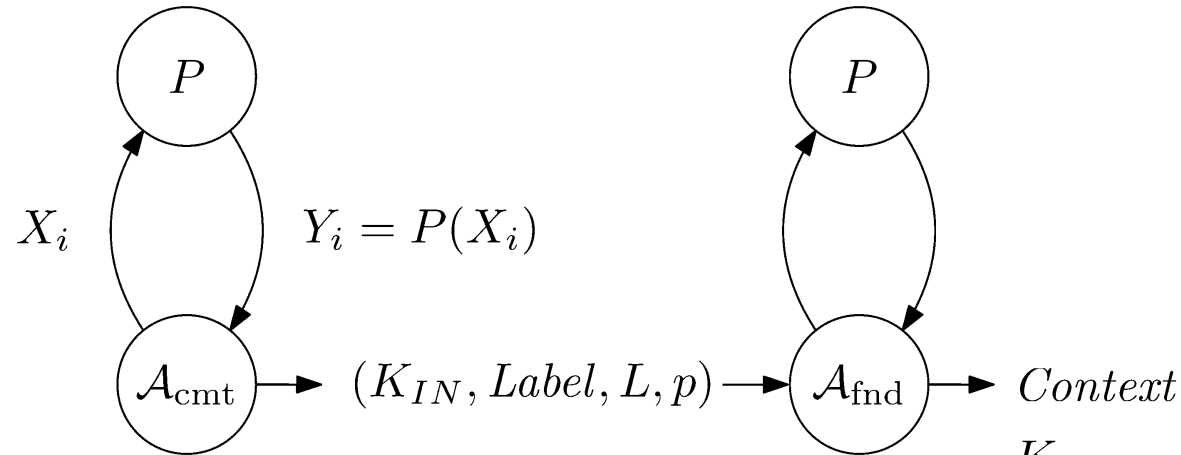
Our Formalization: KCS Game

- Assume the use of an ideal primitive P , could be a random oracle, or an ideal cipher
- Our formalization: $A = (A_{\text{cmt}}, A_{\text{fnd}})$ (commit-then-find game)
 - A_{cmt}^P chooses (Label, L, p)
 - A_{cmt}^P chooses K_{IN}
 - with a restriction $\tilde{H}_{\infty}(K_{IN} \mid \text{Label}, L, p) \geq k$
 - A_{fnd}^P chooses Context
 - A wins if $p(K_{OUT}) = 1$ for $K_{OUT} = \text{KDF}(K_{IN}, \text{Label}, \text{Context}, L)$
- $\tilde{H}_{\infty}(K_{IN} \mid \text{Label}, L, p) \geq k$: K_{IN} has a sufficient average min-entropy, given (Label, L, p)
- still in a known-key setting, since K_{IN} has randomness

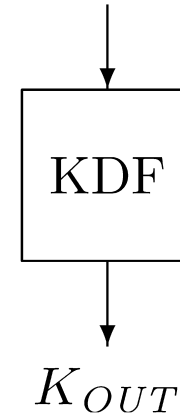


Our Formalization: KCS Game

RO or Ideal Cipher



$K_{IN} \parallel Label \parallel Context \parallel L$



$K_{OUT} = \text{KDF}(K_{IN}, Label, Context, L)$

$p(K_{OUT}) \stackrel{?}{=} 1$

- $\tilde{H}_{\infty}(K_{IN} \parallel Label, L, p) \geq k$
- Please see the paper for a more precise definition

Results on KDFs in NIST SP 800-108

Table 1: Provable key control security bounds. q is the number of queries, δ is the success probability with a random guess, \tilde{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	$O\left(q2^{-\tilde{k}} + q\delta\right)$	Sect. 4
CTR-HMAC	$O\left(q2^{-\tilde{k}} + q\delta + s^2q^22^{-h}\right)$	Sect. 5.2
FB-HMAC	$O\left(q2^{-\tilde{k}} + q\delta + s^2q^22^{-h}\right)$	Sect. 5.3
DP-HMAC	$O\left(q2^{-\tilde{k}} + q\delta + s^2q^22^{-h}\right)$	Sect. 5.4

- KDF-KMAC is secure
- {CTR, FB, DP}-HMAC are secure up to the birthday bound (w.r.t. the output len. of the hash function used)

Results on KDFs in NIST SP 800-108

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

- {CTR, FB}-CMAC are known to be insecure [Che22]
- {DP, stCTR, stFB, stDP}-CMAC admit attacks with the birthday bound complexity, plus a small cost




Outline

- Key Derivation Functions
- Security Requirements for KDFs
- Formalization of Key Control Security
- Proofs
- Attacks
- Summary

Results on KDFs in NIST SP 800-108

Table 1: Provable key control security bounds. q is the number of queries, δ is the success probability with a random guess, \tilde{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.



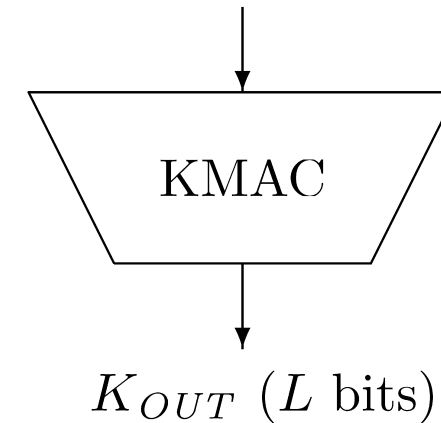
Scheme	Bound	Ref.
KDF-KMAC	$O\left(q2^{-\tilde{k}} + q\delta\right)$	Sect. 4
CTR-HMAC	$O\left(q2^{-\tilde{k}} + q\delta + s^2q^22^{-h}\right)$	Sect. 5.2
FB-HMAC	$O\left(q2^{-\tilde{k}} + q\delta + s^2q^22^{-h}\right)$	Sect. 5.3
DP-HMAC	$O\left(q2^{-\tilde{k}} + q\delta + s^2q^22^{-h}\right)$	Sect. 5.4

- KDF-KMAC is secure
- {CTR, FB, DP}-HMAC are secure up to the birthday bound (w.r.t. the output len. of the hash function used)

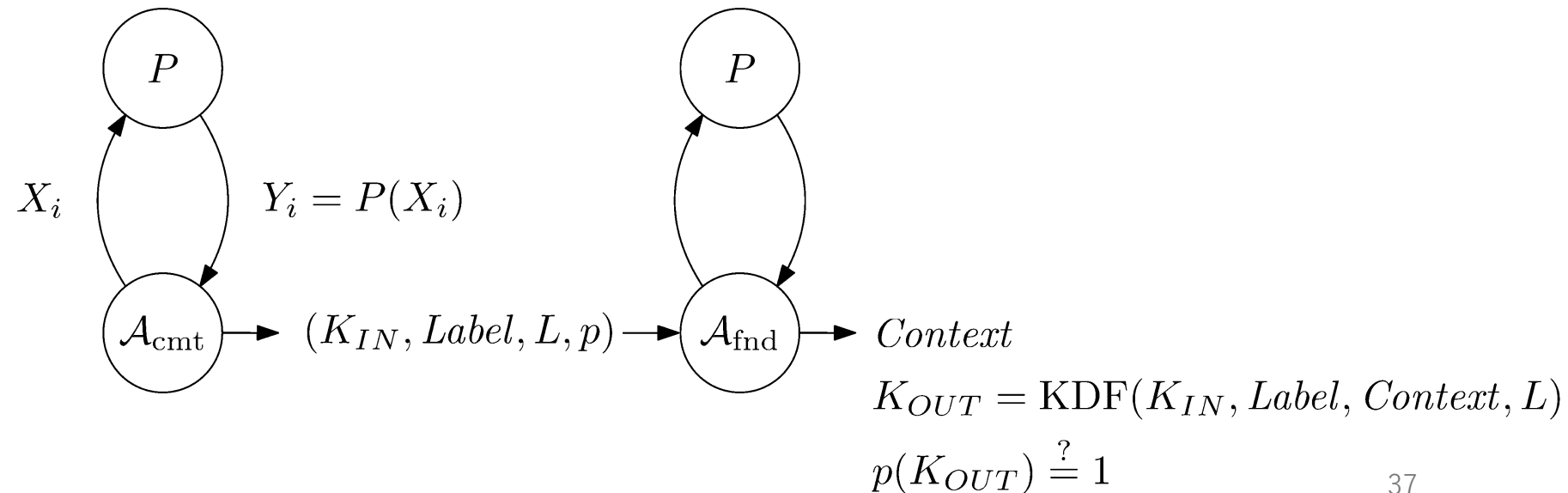
KDF-HMAC Is KCS-Secure

- bad1: K_{IN} is one of K_i 's in the commit stage
- bad2: $p(Y_i) = 1$ for some Y_i in the find state
- $\Pr[\text{bad1} \mid \text{Label}, L, p]$ is small from the randomness of K_{IN}
- $\Pr[\text{bad2} \mid \neg \text{bad1} \ \& \ \text{Label}, L, p]$ is small since $Y_i = \text{uniform}$

$K_{IN} \parallel \text{Label} \parallel \text{Context} \parallel L$



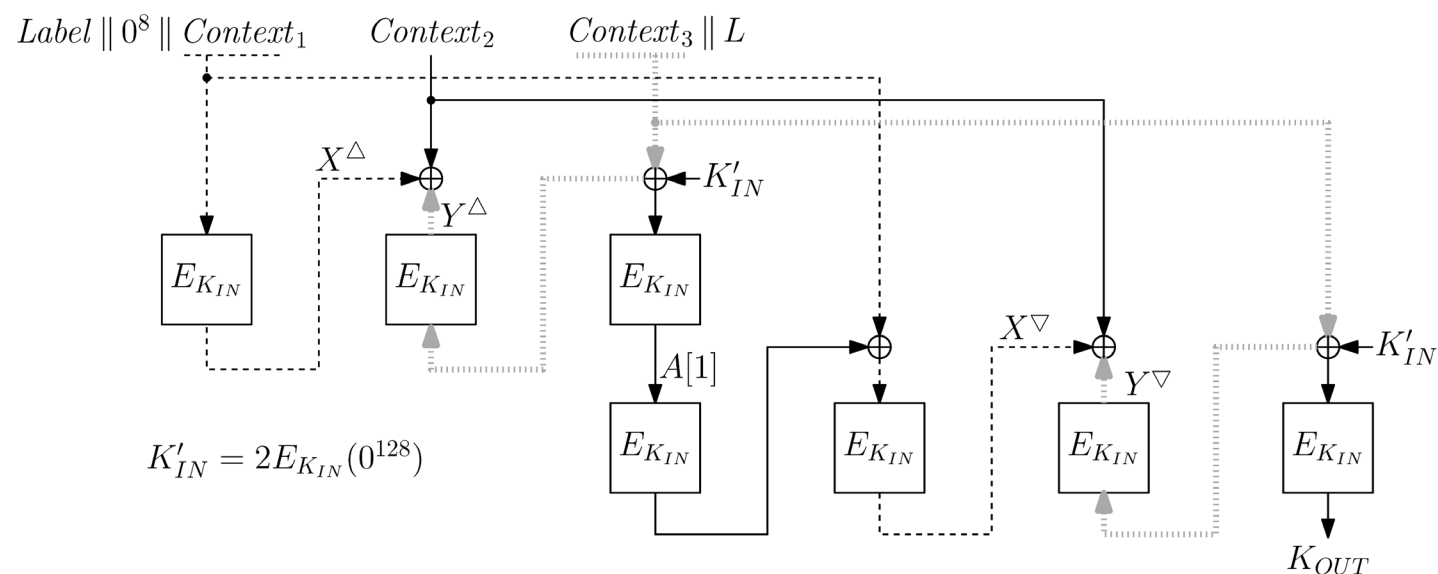
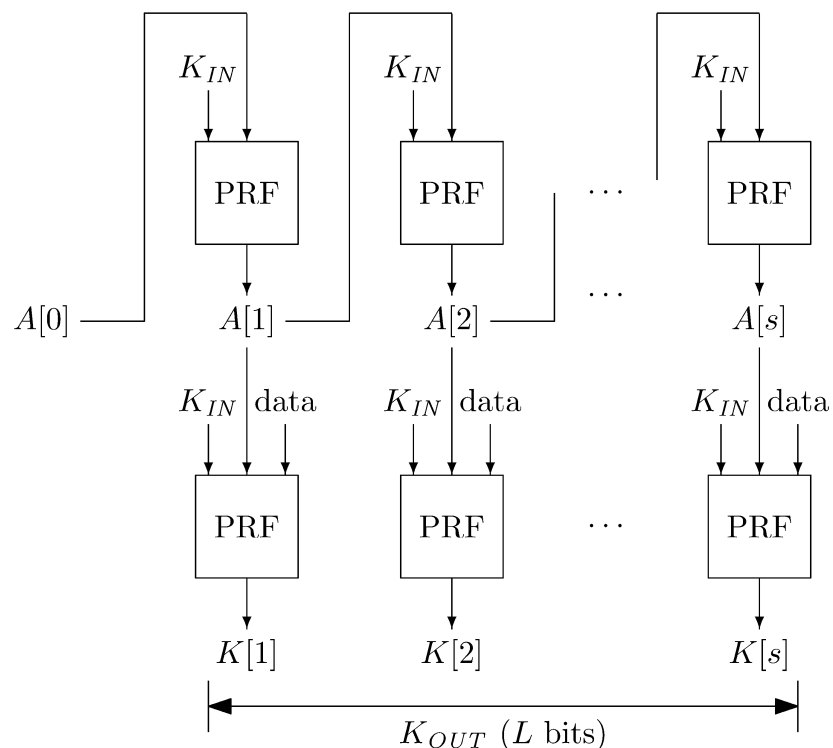
RO or Ideal Cipher



Outline

- Key Derivation Functions
- Security Requirements for KDFs
- Formalization of Key Control Security
- Proofs
- Attacks
 - KCS Attack against DP-CMAC
 - Distinguishing Attack on FB-PRF
- Summary

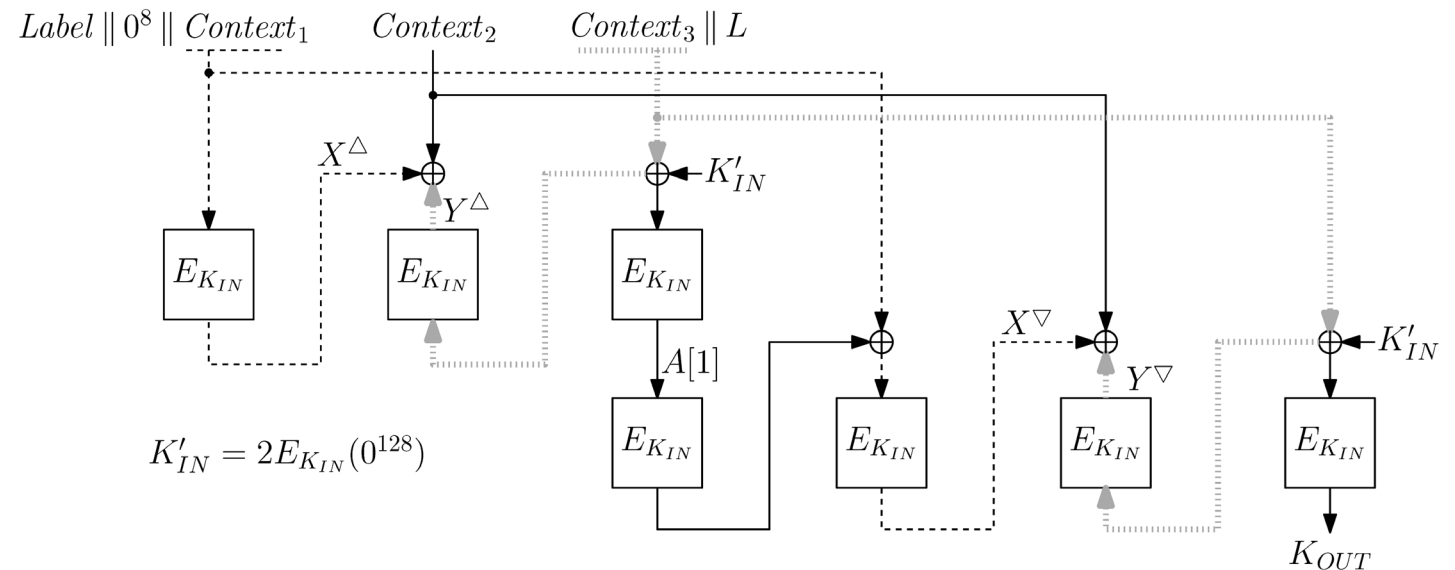
KCS Attack against DP-CMAC (Birthday)



$$A[0] = \text{data} = Label \parallel Context \parallel L$$

KCS Attack against DP-CMAC (Birthday)

1. Fix $A[1]$
2. Store 2^{64} values of $X^\Delta \oplus X^\nabla$ for 2^{64} values of $Context_1$
3. Store 2^{64} values of $Y^\Delta \oplus Y^\nabla$ for 2^{64} values of $Context_3$
4. Find $Context_1$ and $Context_3$ such that $X^\Delta \oplus X^\nabla = Y^\Delta \oplus Y^\nabla$
5. Compute $Context_2 = X^\Delta \oplus Y^\Delta (= X^\nabla \oplus Y^\nabla)$



Distinguishing Attack on FB-PRF (ePrint)

- Flexibility in the specification
 - IV can be public input, secret input, or even empty, IV len. is also flexible
 - the use of block counter is optional

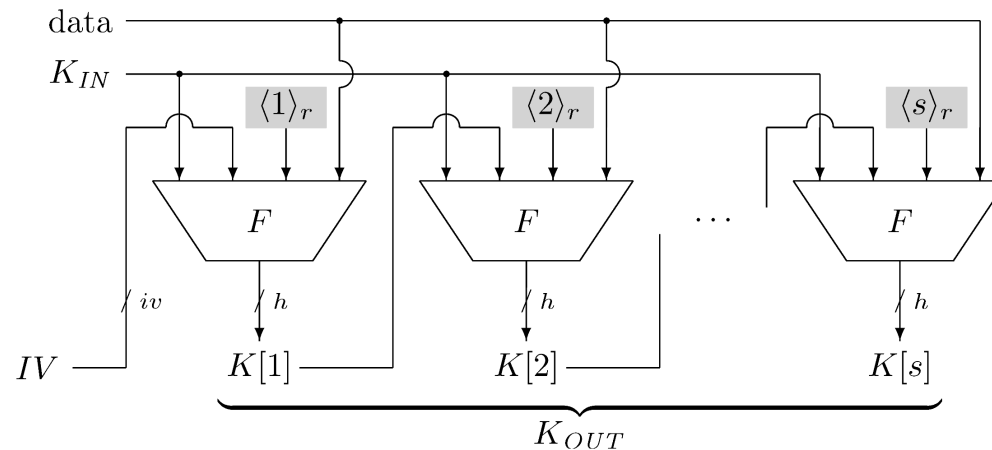
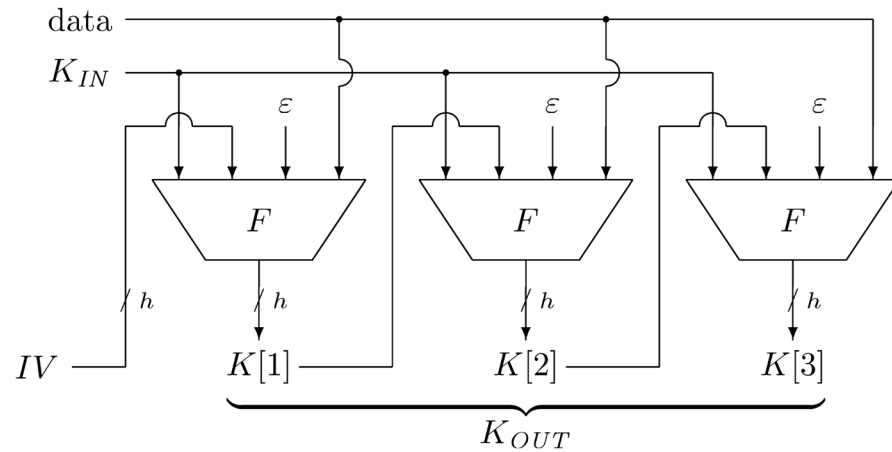


Figure 2: Illustration of $FB[F]$, where $data = Label \parallel 0^8 \parallel Context \parallel L$.

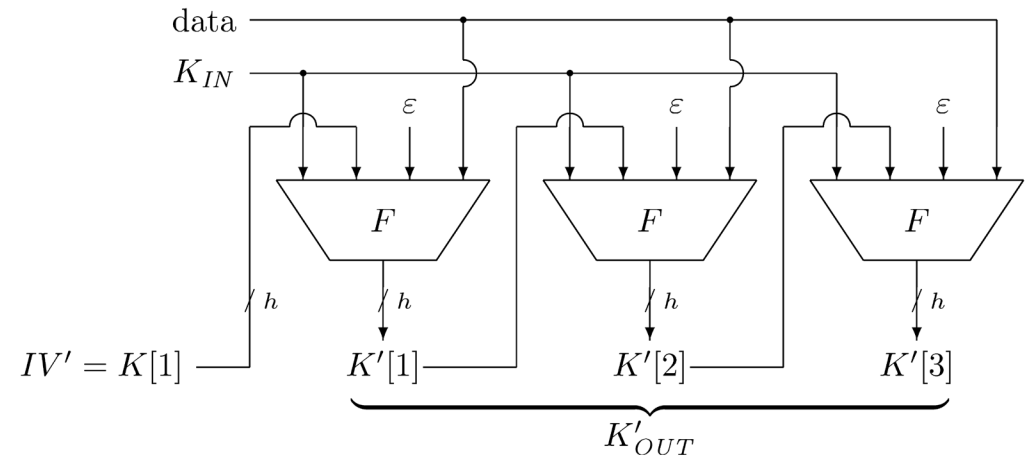
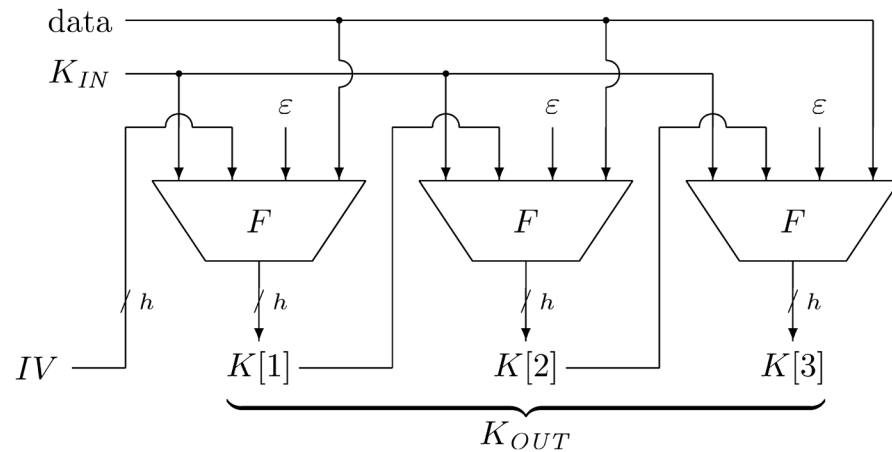
Distinguishing Attack on FB-PRF (ePrint)

- Consider the case
 - IV is a public input and IV len = output len of PRF
 - block counter is not used



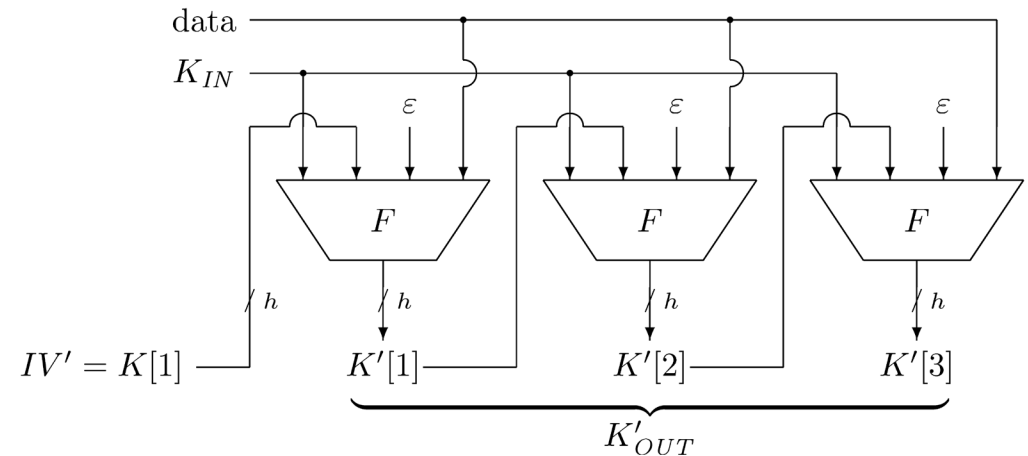
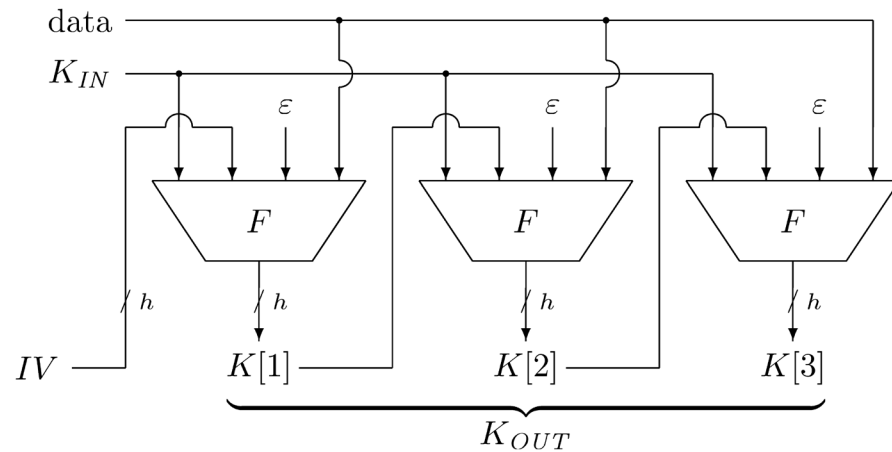
Distinguishing Attack on FB-PRF (ePrint)

- Assume the adversary has K_{OUT} for some $(IV, Label, Context)$
- Then the adversary immediately knows the first $2h$ bits of K'_{OUT} for $(IV', Label, Context)$ with $IV' = K[1]$ are $K'[1] \parallel K'[2] = K[2] \parallel K[3]$



Distinguishing Attack on FB-PRF (ePrint)

- Works for any PRF, many ways to avoid the vulnerability. The attack does not work if:
 - $IV \text{ len} \neq \text{output len of PRF}$
 - the protocol restricts the selection of IV to a small set of possible values
 - the block counter is used
 - IV is derived from Context as in the strengthened mode



Summary 1

- Formalization of key control security
 - close to (multi-target) preimage security
 - the targets can be preselected by the adversary
 - Our formalization covers strong adversaries that can choose various inputs, with suitable restrictions
- Analyzed the security of KDFs in NIST SP 800-108
 - Proofs for KMAC and HMAC-based KDFs, up to the birthday bound
 - Attacks for CMAC-based KDFs, birthday complexity (plus a small cost)
 - Proofs are missing
- KCS has just been formalized, lots of open problems
 - Stronger notion, proofs of KDFs based on various PRFs, attacks of KDFs based on various PRFs



Summary 2

- A particular instance of FB-PRF is PRF-insecure
- There are PRF-secure instances [SWG25], but not all
 - So, be careful when you use it