A Modular Approach to the Incompressibility of Block-Cipher-Based AEADs*

Akinori Hosoyamada[1], Takanori Isobe[2,3,4], Yosuke Todo[1][0000−0002−6839−4777], and Kan Yasuda[1]

[1] NTT Social Informatics Laboratories, Tokyo, Japan
{akinori.hosoyamada.bh,yosuke.todo.xt,kan.yasuda.hy}@hco.ntt.co.jp
[2] University of Hyogo, Hyogo, Japan takanori.isobe@ai.u-hyogo.ac.jp
[3] National Institute of Information and Communications Technology, Tokyo, Japan
[4] PRESTO, Japan Science and Technology Agency, Tokyo, Japan

Abstract. Incompressibility is one of the most fundamental security goals in white-box cryptography. Given recent advances in the design of efficient and incompressible block ciphers such as SPACE, SPNbox and WhiteBlock, we demonstrate the feasibility of reducing incompressible AEAD modes to incompressible block ciphers. We first observe that several existing AEAD modes of operation, including CCM, GCM(-SIV), and OCB, would be all insecure against white-box adversaries even when used with an incompressble block cipher. This motivates us to revisit and formalize incompressibility-based security definitions for AEAD schemes and for block ciphers, so that we become able to design modes and reduce their security to that of the underlying ciphers. Our new security notion for AEAD, which we name whPRI, is an extension of the pseudo-random injection security in the black-box setting. Similar security notions are also defined for other cryptosystems such as privacy-only encryption schemes. We emphasize that whPRI ensures quite strong authenticity against white-box adversaries: existential unforgeability beyond leakage. This contrasts sharply with previous notions which have ensured either no authenticity or only universal unforgeability. For the underlying ciphers we introduce a new notion of whPRP, which extends that of PRP in the black-box setting. Interestingly, our incompressibility reductions follow from a variant of public indifferentiability. In particular, we show that a practical whPRI-secure AEAD mode can be built from a whPRP-secure block cipher: We present a SIV-like composition of the sponge construction (utilizing a block cipher as its underlying primitive) with the counter mode and prove that such a construction is (in the variant sense) public indifferentiable from a random injection. To instantiate such an AEAD scheme, we propose a 256-bit variant of SPACE, based on our conjecture that SPACE should be a whPRP-secure cipher.

Keywords: symmetric-key cryptography · white-box cryptography · incompressibility · mode of operation · public indifferentiability

* This article is the full version of the paper with the same title accepted to Asiacrypt 2022,

1 Introduction

White-box cryptography, which has been introduced by Chow et al. for AES [29] and DES [30], is a technique to protect data in the presence of adversaries who have access to implementations of cryptographic algorithms. For two decades since Chow et al. published the seminal papers, target systems of white-box cryptography have spread out from digital rights management (DRM) to mobile payment and banking services [2, 50]. Today white-box cryptography is applied to a wide range of cryptographic algorithms [36], and in this paper we focus on symmetric-key encryption schemes.

Secure white-box implementations must resist key extraction and "code lifting" [36]. While the goal of key extraction is to retrieve a secret key from a white-box implementation, code lifting tries to isolate and copy (a part of) the functionality of the cryptographic algorithm. Security against code lifting is in general stronger than security against key extraction, as key extraction implies code lifting of the full functionality. Preventing code lifting is indispensable to realize secure white-box implementations because arbitrary message can be encrypted or decrypted once the program is copied.

Delerablée et al. [36] have introduced the notion of incompressibility to formalize resistance to code lifting. Roughly, a white-box program of an encryption scheme is incompressible if it is infeasible to compress the encryption program while keeping its functionality. Delerablée et al. have shown that incompressibility is achievable by an RSA-group-based construction. Follow-up work by Fouque et al. [43] has introduced variants of incompressibility regarding privacy (IND-COM) or limited authenticity of universal unforgeability (ENC-COM). They have presented randomized schemes ensuring each of the security notions but not both at the same time.[5] The more recent work by Bock et al. [21] has shown that an incompressible randomized encryption scheme can be built from one-way permutations. Closely related to incompressibility is the work by Bellare et al. on big-key symmetric encryption [9][6], which was later improved by Bellare and Dai [8]. They have provided efficient randomized encryption schemes with a high level of privacy (LIND) and without authenticity, in the setting where information of the key is partially leaked, by making the key big, say, 1GB.

While there exist other white-box security notions, we focus on incompressibility because it is achievable by relatively efficient schemes and without relying on special hardware. True that trusted execution environments are in common use today, but demands for software-only solutions are still high in various scenarios—e.g., cloud servers providing digital rights management based services, mobile phones running cloud-based payment services with host card emulations, and memory-leakage resilient software—as listed by Bogdanov et al. [25].

It should be noted that some pieces of previous work [24, 9, 43, 25] (and we also do) assume that a black-box adversary resides outside the target program

[5] A scheme in Section 2 of the paper [43] achieves authenticity but not privacy in the white-box setting, because its tag-generation part does not depend on keys.

[6] This work focuses on bounded retrieval model rather than white-box cryptography, but as Fouque et al. point out, its security notion almost matches IND-COM.

and tries to attack in the conventional sense. More precisely, the white-box adversary, which here we call a *lifter*, tries to isolate and copy the functionality of the encryption program. Then, the black-box adversary tries to break privacy and/or authenticity with the aid of leakage generated by the lifter. Here, the amount of leakage is properly restricted in agreement with the bounded retrieval model [32, 27] of leakage-resilient cryptography.

There is another line of research: Designing incompressible block ciphers. Bogdanov and Isobe [24] have introduced the concept of SPACE-hard white-box block ciphers and presented a concrete construction SPACE based on a dedicated design rather than on an obfuscated implementation of an existing cipher (such a direction of adopting dedicated designs for block ciphers was initiated with ASASA [17]). The notion of SPACE hardness is a variant of incompressibility and provides immunity against code lifting. Similar notions include weak white-box security [17] and ENC-TCOM [43]. Bogdanov and Isobe have shown that SPACE achieves SPACE hardness, assuming AES is secure. SPACE is reasonably efficient, running faster than a hundred cycles per byte on modern PCs. A number of follow-up SPACE-hard white-box block ciphers have been proposed, including SPNbox [25] and WhiteBlock [43].

Now our motivation behind this work becomes evident: There is a large gap between the two lines of research. Specifically, we would like to address the following issues:

1. There exist no modes of operation that turn incompressible block ciphers into incompressible authenticated encryption (AE) schemes. As described in Section 3 (and in Section B in the appendix), existing modes such as GCM [64], GCM-SIV [45], CCM [79], and OCB [57] would not yield incompressible AE even if combined with an incompressible block cipher. The state-of-the-art incompressible block ciphers mentioned above, though secure and reasonably efficient, are not utilized.
2. As mentioned above, there exist no AE schemes that simultaneously ensure both privacy and authenticity against white-box adversaries, unless one relies on special hardware. Moreover, the only type of authenticity that has been achieved in the context of incompressibility is universal unforgeability, which is much weaker than what has been done in the conventional setting. Similar discussions are provided in the previous work by Bock et al. [22] where the authors point out that "the definition of incompressibility does not capture any further security such as confidentiality and authenticity".
3. The lack of secure AE modes or schemes indicates the need for further investigation into the incompressibility notion. Specifically, we would like to come up with a usable definition of incompressible block ciphers as well as a new notion of incompressibility that captures more perfectly the privacy and authenticity requirements on AE schemes. Having done that, we should be able to design a mode that enjoys both privacy and authenticity in a strong sense, by relying on the underlying incompressible cipher.

1.1 Our Contributions

We introduce new incompressibility-based white-box security notions for AEAD schemes and BCs which we name whPRI and whPRP, respectively. Intuitively, with the two notions we attempt to define the best possible security such that any λ-bit leakage from a lifter (e.g., malware) does not allow adversaries to break privacy and/or authenticity, or equivalently indistinguishability, except for λ-bit ciphertexts. In particular, the notions demand authenticity in quite a strong sense: existential unforgeability beyond leakage. Our definition, we believe, should be the first one to formalize this notion concretely. Obviously, this is a much stronger requirement than universal unforgeability. We remark that whPRI and whPRP are extensions of pseudo-random injections (PRI) [74] and pseudo-random permutations (PRP) in the black-box setting, respectively: they exactly match in the extreme case of $\lambda = 0$. The security games for our new definitions involve both of black-box adversaries and lifters. These games become inherently multi-stage.

We properly bound the computational resource t_{lif} of a lifter and the leakage size λ. Especially, no security is guaranteed after either t_{lif} or λ reaches a certain threshold, e.g., $t_{\mathrm{lif}} = 2^{50}$ or $\lambda = 2^{20}$. We expect that an attack (malware activity) should be detectable, before the threshold is reached, by some means, e.g., monitoring active processes and/or outgoing packets. We conjecture that SPACE should satisfy whPRP-security under some reasonable parameter settings.

For completeness we study theoretical possibilities of security reductions of various symmetric-key schemes; we introduce similar notions for keyed functions and conventional (privacy-only) encryption schemes. Our notion for keyed functions, which we call whPRF, is an extension of the standard pseudo-random function (PRF). For conventional encryption schemes, we define two security notions which we name whIND\$-CPA and wh$\widetilde{\mathrm{SPRP}}$. The former is an extension of IND\$-CPA security (for random-IV schemes) in the black-box setting. The latter is obtained as a special case of whPRI where ciphertext lengths are always equal to message lengths. Thus, wh$\widetilde{\mathrm{SPRP}}$ is an extension of the tweakable strong PRP ($\widetilde{\mathrm{SPRP}}$) security for tweakable enciphering schemes [47] in the black-box setting. We observe that meaningful counterparts of MAC security and nonce-based security notions seem unachievable in our context. Table 1 gives comparisons between various security notions for (authenticated) encryption schemes.

We prove that a reduction between the new security notions is possible if the construction in hand satisfies a variant of public indifferentiability [40, 82], which we name *weak public indifferentiability*. Then we demonstrate that all the new notions can be reduced to whPRP, by presenting corresponding constructions that are weak public indifferentiable.

Finally, as an example of practical AEAD modes of block ciphers, we show that a composition of the sponge construction [13] and the counter mode (CTR) via SIV [74] is whPRI-secure if the underlying block cipehr E_K is whPRP-secure. Here, the underlying primitive E_K is used both by the sponge and by the CTR.

Table 1: Comparison of incompressibility or related notions for symmetric-key (authenticated) encryption schemes. We assume that AEADs always take a nonce (or IV) as a part of input. Especially, a nonce is included into inputs of deterministic AEADs.

Security notion	Target scheme	Leakage	Adversarial goal	
(λ, δ)-incompressibility [36, 21]	deterministic or randomized encryption (RSA group or OWP-based schemes [36, 21])	whole implementation	δ-functionality with code size $< \lambda$	
			(Privacy)	(Authenticity)
LIND [9]	randomized encryption (Big-Key Encryption [9, 8])	via function with output size $= \ell$	distinguishing	—
IND-COM [43]	randomized AE (WhiteKey [43])	via function	distinguishing	—
ENC-COM [43]	randomized AE (WhiteKey+RO)	with entropy left $\geqslant \mu$	—	universal forgery
whPRI [Section 4.3]	deterministic AE (SIV+CTR (Section 7))	via lifter (malware) with output size $\leqslant \lambda$	distinguishing	existential forgery beyond leakage
whIND$-CPA [Section 5.3]	randomized encryption (CTR (Section 6.3))	via lifter (malware) with output size $\leqslant \lambda$	distinguishing	—
wh$\widetilde{\text{SPRP}}$ [Section 5.3]	tweakable enciphering scheme (6-round Feistel (Section 6.3))	via lifter (malware) with output size $\leqslant \lambda$	distinguishing	—

Roughly speaking, if E_K is secure up to λ-bit leakage, the resulting AEAD is secure as long as the amount of processed data is $\ll 2^{n/4}$ and leakage is less than λ. To instantiate E_K, we propose to use a 256-bit-block variant of SPACE which we name SPACE256. We conjecture that SPACE256 is secure up to 2^{20} bits of leakage. The resulting AEAD scheme is implemented on an Intel platform for experiments, and we confirm that the performance is practical. The size of the program is in an order of KB or MB, which is reasonably small for mobile applications. Unlike previous schemes achieving incompressibility, our scheme does not need random nonces. This is an advantage in the white-box setting because random number generators may be compromised by adversaries.

Note that our notions do not supersede previous ones but rather coexist with other white-box security approaches such as binding [23, 22]. Which security approaches, definitions or solutions one should choose changes depending on use cases and what one wants to achieve. Specifically, when trusted hardware is available or when lifters have much more limited access to programs, other security notions would be more suitable.

1.2 Related Work

Other Security Notions in White-Box Cryptography. The initial goal set by Chow et al. was to protect software implementations of existing block ciphers from key extraction when an attacker is given an unlimited access to a white-box implementation. Many pieces of previous work have proposed such implementations, but none of them remains unbroken [16, 59, 68, 80]. Some of the state-of-the-art work focus on limited white-box adversaries such as DCA and a certain class of algebraic attacks [26, 5, 19, 20].

Several solutions outside incompressibility have been suggested to mitigate code lifting. Chow et al. suggested external encoding [29], which yields a white-box implementation of $E'_K = G \circ E_K \circ F^{-1}$ for some functions F and G instead of E_K. The problem is that even an ordinary user needs a separate implementation of G^{-1} or F to compute E_K. Thus, white-box adversaries would also be able to peel off the external encoding, unless the encoding is stored in trusted hardware. Delerablée et al. suggested one-wayness [36], which formalizes the notion that one is unable to perform decryption even if an encryption program is given. They also suggested traceability [36], which allows a program distributor to trace malicious users who leak their encryption programs. Both are interesting, but they do not encompass resistance to copying encryption programs. Other works have discussed the possibility of binding [23, 22, 1], where the execution of encryption is bound by trusted hardware or applications. Unfortunately, cryptographically secure binding requires, together with secure hardware, primitives such as indistinguishability obfuscation (iO) or LWE, which are richer than usual symmetric-key primitives.

Symmetrically and Asymmetrically Hard Cryptography. Biryukov and Perrin [18] introduced the HSp mode (and its instantiation WHALE), which can be used to build an incompressible VIL/VOL hash function from a usual sponge hash (like SHA-3) and an FIL/FOL incompressible function. The mode is proven to achieve a universal-unforgeability-like security notion on incompressibility. Their result seems close to ours (in Section 6.3) that the sponge construction becomes a VIL/VOL whPRF if the underlying primitive is a whPRP (or FIL/FOL whPRF). Still, there are two differences between theirs and ours. First, they proved only universal-unforgeability-like security while we proved existential-unforgeability-like security (i.e., whPRF-security). Second, their proof is in the random oracle model while ours is in the standard model in that the existence of a whPRP (or a FIL/FOL whPRF) is a falsifiable assumption.

Leakage Resilient Cryptography. An important area related to white-box cryptography is *leakage resilient cryptography*, which aims to achieve provable security against side-channel attacks. Security models in leakage resilient cryptography are roughly classified into two types[7], depending on whether (1) an adversary is allowed to obtain arbitrary leakage from the secret key as long as the leakage length is bounded by a certain parameter, or (2) some form of security is assumed on memory or storage, and/or leakage is obtained only when some computation (e.g., encryption) is performed through a special class of functions such as the Hamming weight of internal states with some noise.

Models of the First Type. A typical model of the first type closely related to our results is the Bounded Retrieval Model (BRM) [32, 41], where large (e.g., 1GB) keys are used to prevent key exfiltration. The BRM and related notions

[7] This classification is based on (still not completely the same as) the one in [52, 53].

have been studied in a long line of research [32, 41, 27, 4, 3, 9, 8]. Among others, Bellare et al. [9] showed practical symmetric-key encryption schemes achieving confidentiality in the BRM, which was later improved by Bellare and Dai [8].

As pointed out by Fouque et al [43], the goals of Bellare et al. [8] and incompressibility are quite close. Still, each of the BRM and incompressibility has its own advantages. An advantage of the BRM is that, for well designed schemes such as the one by Bellare et al. [8], bounding the running time of a lifter (malware) is not mandatory (it is mandatory for incompressible ciphers because the secret key sizes are very small). Meanwhile, no previous works on symmetric encryption scheme in the BRM achieve both confidentiality and authenticity simultaneously[8], while we prove that SIV+CTR achieves whPRI.

Models of the Second Type. Major models of the second type include the "only computation leaks information" (OCL) model [65] and wire-probing leakage [51]. In models of the second type, lots of previous works have shown various leakage resilient schemes including AEADs [65, 42, 70, 60, 76, 12, 38, 7, 35, 39, 11, 46, 56]. Especially, Krämer and Struck [56] showed that the security of a leakage-resilient AEAD can be reduced to the security of leakage-resilient PRFs in the "only comoputation leaks information" model [65]. However, these results are incomparable to ours because they essentially assume that attackers do not have a direct full access to memory or storage that stores the secret key.

A clear advantage of the second type is that the size of implementations can be small, compared to incompressibility and the first type. When we can assume that adversaries do not have a full direct access to memory or storage (e.g., leakage can be obtained only by measuring power consumption of a circuit), models of the second type will be more suitable than incompressibility and the first type. When we cannot, incompressibility or the first type will be suitable.

1.3 Paper Organization

Section 2 introduces basic notations and definitions, and review basics on (public) indifferentiability. Section 3 shows an observation that GCM is unlikely to achieve incompressibility. In Section 4, we introduce whPRI, a new security notions for AEADs. New security notions for other schemes are introduced in Section 5. Section 6 introduces weak public indifferentiability and shows that weak public indifferentiability implies white-box security reductions. The section also demonstrates that our new notions on various schemes can be reduced to whPRP, by showing (weak) public indifferentiable constructions. In Section 7 we show that a practical whPRI-secure AEAD mode of whPRP can be realized as a composition by SIV of the sponge construction and the counter mode.

[8] A scheme by Bellare et al. [9] also achieves authenticity, but only in the absence of leakage (See also Table 1).

2 Preliminaries

Throughout the paper, $\mathrm{len}(M)$ denotes the bit length for a bit string M. Given a positive integer $m < \mathrm{len}(X)$, we write $(A, B) \xleftarrow{m,*} X$ to mean assignment of bit strings, the leftmost m bits of X to A and the remaining bits to B. The variable A or B may be omitted with the symbol "$\cdot$" in which case the corresponding bits are not assigned to any variable. When we write like $(X_1, X_2, \ldots, X_\ell) \xleftarrow{n} X$ we mean partitioning X into n-bit blocks and assigning them to $X_1, X_2, \ldots, X_\ell$ where $\ell = \lceil \mathrm{len}(X)/n \rceil$ and the last X_ℓ is possibly fractional, i.e., $\mathrm{len}(X_\ell) < n$. The symbol $\|$ stands for concatenation of bit strings and the symbol $\oplus$ exclusive OR of two bit stings of the same length. By block length of M we denote $\lceil \mathrm{len}(M)/n \rceil$ when the parameter n is clear from the context. For an invertible function F by $F^{\pm}$ we denote the oracles of F and F^{-1}. We denote the empty bit string by ε and define $\{0,1\}^0 := \{\varepsilon\}$. $\{0,1\}^*$ denotes the set of all bit strings of arbitrary length. For positive integers x and n, by $x \bmod n$ we denote the minimum positive integer i such that $i \equiv x \bmod n$. We say an m-input function $f : (\mathbb{Z}_{\geq 0})^{\times m} \to \mathbb{R}_{\geq 0}$ is non-decreasing if $f(x_1, \ldots, x_i + z, \ldots, x_m) \geq f(x_1, \ldots, x_m)$ holds for arbitrary $1 \leq i \leq m$, $(x_1, \ldots, x_m) \in (\mathbb{Z}_{\geq 0})^{\times m}$, and $z \in \mathbb{Z}_{\geq 0}$.

Definition 1 (Variable-key and fixed-key random injection). *Let $\tau \geq 0$ be an integer and $\mathrm{Inj}_\tau(\mathbf{K} \times \mathbf{N} \times \mathbf{A} \times \mathbf{M}, \mathbf{C})$ denote the set of functions $F : \mathbf{K} \times \mathbf{N} \times \mathbf{A} \times \mathbf{M} \to \mathbf{C}$ such that $F_{K,N,A} := F(K, N, A, \cdot)$ is an injection for each (K, N, A) and $\mathrm{len}(F(K, N, A, M)) = \mathrm{len}(M) + \tau$. A* variable-key random injection *F is an injection chosen uniformly at random from $\mathrm{Inj}_\tau(\mathbf{K} \times \mathbf{N} \times \mathbf{A} \times \mathbf{M}, \mathbf{C})$. The inverse $F^{-1} : \mathbf{K} \times \mathbf{N} \times \mathbf{A} \times \mathbf{C} \to \mathbf{M} \cup \{\perp\}$ is defined so that $F^{-1}(K, N, A, F(N, A, M)) = M$ for each (K, N, A, M) and $F^{-1}(K, N, A, C) = \perp$ for all $C \notin F_{K,N,A}(\mathbf{M})$. If $\mathbf{K}$ is a set that contains exactly a single element, we say F is a* fixed-key *random injection and omit to write $\mathbf{K}$ and K.*

Syntax of Symmetric-Key Cryptosystems and Basic Constructions.

Keyed Functions. A keyed function is a function $f : \{0,1\}^\kappa \times \mathbf{X} \to \mathbf{Y}$. Here, κ is a positive integer and $\{0,1\}^\kappa$ is called the key space. We write $f_K(M)$ and $f(K, M)$ interchangeably.

Block Ciphers. A block cipher is a keyed function $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ such that $E(K, \cdot)$ is a permutation for each K. The inverse function $(E_K)^{-1}$ is denoted by D_K, and we write $D_K(C)$ and $D(K, C)$ interchangeably. E and D are called the encryption and decryption functions.

AEADs. An AEAD scheme is a tuple $\Pi = (\mathcal{E}, \mathcal{D})$. The first element of Π is an encryption function $\mathcal{E} : \mathbf{K} \times \mathbf{N} \times \mathbf{A} \times \mathbf{M} \to \mathbf{C}$. Here, $\mathbf{K}$ is the key space from which the secret key is chosen uniformly at random. The set $\mathbf{N} = \{0,1\}^\nu$ is a nonce space with the nonce length ν being a non-negative integer. The sets $\mathbf{A}, \mathbf{M}, \mathbf{C}$ correspond to the spaces of associated data, plaintext and ciphertext, respectively, where $\mathbf{M} = \mathbf{C} = \{0,1\}^*$. We write interchangeably

Algorithm 1: $\mathrm{CTR}(K, IV, M)$

$(M_1, M_2, \ldots, M_\ell) \xleftarrow{n} M$
for $i = 1$ **to** ℓ **do**
  $y \leftarrow f_K(IV + i - 1)$,
  $(y', \cdot) \xleftarrow{\mathrm{len}(M_i),*} y$,
  $C_i \leftarrow M_i \oplus y'$
return $C_1 \,\|\, C_2 \,\|\, \cdots \,\|\, C_\ell$

Algorithm 2: $\mathcal{E}_{K_1,K_2}(N, A, M)$

$IV \leftarrow f_{K_1}(N, A, M)$
$C' \leftarrow \mathcal{E}'_{K_2}(IV, M)$
return $C \leftarrow IV \,\|\, C'$

Algorithm 3: $\mathcal{D}_{K_1,K_2}(N, A, C)$

$(IV, C') \xleftarrow{\tau,*} C$
$M \leftarrow \mathcal{D}_{K_2}(IV, C')$
$T \leftarrow f_{K_1}(N, A, M)$
if $IV = T$ **then**
  return M
else
  return $\perp$

$\mathcal{E}(K, N, A, M) = \mathcal{E}_K(N, A, M) = \mathcal{E}_{K,N,A}(M)$. For each $(K, N, A) \in \mathbf{K} \times \mathbf{N} \times \mathbf{A}$ we demand that $\mathrm{len}\big(\mathcal{E}_{K,N,A}(M)\big) = \mathrm{len}(M) + \tau$ should hold for all $M \in \mathbf{M}$, where τ a fixed non-negative integer. The second element of Π is a decryption function $\mathcal{D} : \mathbf{K} \times \mathbf{N} \times \mathbf{A} \times \mathbf{C} \to \mathbf{M} \cup \{\perp\}$. Here, the symbol $\perp$ signifies rejection. We write interchangeably $\mathcal{D}(K, N, A, M) = \mathcal{D}_K(N, A, M) = \mathcal{D}_{K,N,A}(M)$. For each (K, N, A, M) we demand that $\mathcal{D}_{K,N,A}(\mathcal{E}_{K,N,A}(M)) = M$ should hold.

Conventional Encryption Schemes. The syntax for a conventional (privacy-only) encryption scheme is essentially the same as that of AEAD except that it does not take any associated data, i.e., $\mathbf{A} = \{\varepsilon\}$, and $\tau = 0$. In addition, nonce N and nonce space $\mathbf{N}$ are renamed as IV and $\mathbf{IV}$. We assume IV is chosen uniformly at random for every encryption query or arbitrarily chosen by adversary depending on security notions we focus on.

Counter Mode. Counter mode (CTR) is the construction to convert a keyed function into a conventional encryption scheme. Let $f : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^n$ be a keyed function. The encryption function of CTR based on f, which we denote by $\mathrm{CTR}(K, IV, M)$, is computed as in Algorithm 1. The key, IV, and message spaces are $\{0,1\}^\kappa$, $\{0,1\}^m$, and $\{0,1\}^*$, respectively. The decryption function is identical to the encryption function.

SIV. SIV is the construction introduced by Rogaway and Shrimpton to realize a deterministic AEAD [74]. Let $\mathbf{N}$ and $\mathbf{A}$ be arbitrarily chosen space of nonces and associated data. (We assume $\mathbf{N} = \{0,1\}^\nu$ for some $\nu \in \mathbb{Z}_{>0}$ and $\mathbf{A} = \{0,1\}^*$ unless otherwise noted.) Let $f : \{0,1\}^{\kappa_1} \times (\mathbf{N} \times \mathbf{A} \times \{0,1\}^*) \to \{0,1\}^\tau$ be a keyed function and $\Pi' = (\mathcal{E}', \mathcal{D}')$ be a conventional encryption scheme with the key space $\{0,1\}^{\kappa_2}$, IV space $\{0,1\}^\tau$, and message space $\{0,1\}^*$. The SIV construction based on f and Π is an AEAD with key space $\{0,1\}^{\kappa_1} \times \{0,1\}^{\kappa_2}$, nonce space $\mathbf{N}$, associated data space $\mathbf{A}$, and message space $\{0,1\}^*$. The encryption function $\mathcal{E}$ and decryption function $\mathcal{D}$ are defined as in Algorithm 2 and Algorithm 3, respectively. We call an output of f a *tag* and f a tag-generation part.

Programs and White-Box Compilers. We follow the abstraction and notation used by Delerablée et al. [36] for dealing with programs and compilers. A

program implements an algorithm, specific to some explicit language and execution model. A program can be read, copied and modified at will. A program can be viewed as a bit string, and its binary code can be executed locally. A program is inherently stateless. A program may, via APIs including system calls, make use of external resources such as random coins and additional functionalities. A white-box compiler $\mathcal{C}_E$ of a block cipher E is an algorithm that takes $K \in \{0,1\}^\kappa$ as an input and outputs a program that implements E_K. We use the notation $[\![E_k]\!]$ to denote a white-box implementation of E_K in a context where explicitly indicating the compiler is unnecessary. Moreover, we call $[\![E_K]\!]$ *white-box block cipher* simply. A white-box compiler may be probabilistic, outputting different programs for the same key[9]. White-box compilers of other cryptosystems are defined in the same way.

Indifferentiability. Let T^{P} be an algorithm (a cryptographic scheme, e.g., a VIL hash function) making queries to P, where P is an ideally random primitive (e.g., a FIL random oracle). In addition, let R be an ideally random scheme corresponding to T^{P} with the same input-output interface (e.g., a VIL random oracle). Then, the indifferentiability advantage of $\mathcal{A}$ against $(\mathsf{T}^{\mathsf{P}}, \mathsf{R})$ with respect to a simulator $\mathcal{S}$ is defined as $\mathsf{Adv}^{\mathrm{indiff}}_{\mathsf{T},\mathsf{R},\mathcal{S}}(\mathcal{A}) := \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{T}^{\mathsf{P}},\mathsf{P}}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{R},\mathcal{S}^{\mathsf{R}}}\right]$. Informally, we say T^{P} is indifferentiable from $\mathcal{R}$ if there is an efficient simulator $\mathcal{S}$ such that the above advantage becomes negligibly small for any efficient $\mathcal{A}$. We call T^{P} (resp., $\mathcal{R}$) a construction oracle and $\mathcal{P}$ a primitive oracle. We call queries to T^{P} or R (resp., P or $\mathcal{S}^{\mathsf{R}}$) construction queries (resp., primitive queries).

The most important feature of indifferentiability is the general "composition theorem" [63, 71]: Suppose the following (1)-(3) hold: (1) A scheme (or protocol) Π^{R} depending on the ideal object R is proven secure. (2) T^{P} is indifferentiable from R. (3) The security of Π is defined by single-stage games. Then the composition theorem guarantees that $\Sigma^{\mathsf{T}^{\mathsf{P}}}$ is secure [63]. Note that not only (1) and (2) but also (3) is crucial; the composition theorem does not necessarily hold for schemes of which security is defined by multi-stage games [71]. We do not get into further details because it is not directly related to our results.

Indifferentiability of Sponge. Let $r, c > 0$ and $f : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ be a function. Let $\mathsf{pad} : \{0,1\}^* \to (\{0,1\}^r)^+$ be an injective padding function such that the last (r-bit) block of $\mathsf{pad}(X)$ is not 0^r for every X.[10] The sponge construction Sponge^f maps bit strings of arbitrary length to bit strings of any requested length as in Algorithm 4 (i.e., Sponge^f can be regarded as a function from $\{0,1\}^* \times \mathbb{N}$ to $\{0,1\}^\infty$). The parameters r and c are called rate and capacity.

[9] In practice, many white-box implementations of AES are the output of the probabilistic compiler. On the other hand, the dedicated white-box block cipher such as SPACE uses the deterministic compiler in general.

[10] In what follows, we assume the padding function pads "1" and the minimum number of zeroes so that the total length of the padded string becomes multiple of r, i.e., $\mathsf{pad}(X) := X||1||0^{\mathrm{len}(X) \bmod r-1}$.

Algorithm 4: $\mathsf{Sponge}^f(X)$ with requested output length m

1: $(X_1, \ldots, X_\ell) \xleftarrow{r} \mathsf{pad}(X)$, $s \leftarrow 0^{r+c}$, $y \leftarrow \varepsilon$
2: **for** $i = 0$ **to** $\ell - 1$ **do**
3: $\quad s \leftarrow f(s \oplus (X_{i+1}||0^c))$
4: **for** $i = \ell$ **to** $\ell + \lceil \frac{m}{r} \rceil - 1$ **do**
5: $\quad y \leftarrow y||$(the upper r bits of s), $s \leftarrow f(s)$
6: **return** y

Bertoni et al. [13] proved that the sponge construction is indifferentiable from a VIL/VOL random oracle $\mathsf{RO} : \{0,1\}^* \to \{0,1\}^\infty$ if f is an ideally random function. More precisely, they showed the following theorem[11].

Theorem 1 (Theorem 1 of [13]). *Let* $\epsilon(q) := 1 - \prod_{i=1}^{q} \left(1 - \frac{1}{2^c}\right)$. *There exists a simulator* $\mathcal{S}$ *making queries of total length at most* $\lceil \frac{r+c}{r} \rceil q^2$ *such that* $\mathbf{Adv}^{\mathrm{indiff}}_{\mathsf{Sponge},\mathsf{RO},\mathcal{S}}(\mathcal{A}) \leq \epsilon(q)$ *holds for any adversary* $\mathcal{A}$ *that calls* f *at most* $q(< 2^c)$ *times in the real world, either directly or indirectly through* Sponge^f.

Indifferentiable AEAD Schemes. Barbosa and Farshim studied indifferentiable AEAD schemes [6], where the ideal oracle is a *variable-key* random injection F and its inverse F^{-1} (see Definition 1)[12]. Note that a variable random injection takes not only nonce, associated data, and message (or ciphertext) but also a key as an input. They especially showed that indifferentiable AEADs cannot be achieved by some generic compositions such as SIV, and that indifferentiable constructions can be built by Encode-then-Encipher (EtE) or 3-round Feistel-based scheme. In particular, by using the sponge construction for round functions of the Feistel-based scheme, an indifferentiable AEAD can be built from a FIL/FOL random function. See Section C in the appendix for details.

Public Indifferentiability. Again, let T^{P} be a construction calling an ideal primitive P, and R be an ideal object of which interface is compatible with T^{P}. Public indifferentiability [40, 82] is defined in the same way as the original indifferentiability, except that a simulator $\mathcal{S}$ is allowed to observe all the queries by

[11] The theorem roughly says Sponge^f is secure up to $2^{c/2}$ queries because $\epsilon(q) \approx 1 - e^{-\frac{q(q+1)}{2^{c+1}}} < \frac{q(q+1)}{2^{c+1}}$ holds for $q \ll 2^c$. The original theorem in [13] did not mention the exact number of queries by $\mathcal{S}$ but we can deduce it is at most $\lceil \frac{r+c}{r} \rceil q$ by checking the details of the proof.

[12] The parameter τ (the length of ciphertext-stretch) is also considered as an input to AEADs and random injections in [6], but this paper considers the special case where is τ fixed to a constant.

adversaries to R and the responses. (Public-indifferentiability is actually a special case of indifferentiability rather than a variant. However, we regard it as a variant for readability.) More precisely, in the ideal world, there is an additional oracle-query interface to reveal the list of all the queries made so far to R and the responses, and an access to this interface is given to $\mathcal{S}$ (but not to $\mathcal{A}$). We call this interface the *revealing interface*, and denote by $\mathsf{Rev}[\mathsf{R}]$. This models the condition that every input to R (and the output) is visible to all the parties involved in a security game, and the general "composition theorem" on public-indifferentiability holds only for schemes of which security games satisfy such a condition. The restriction that the "composition theorem" does not necessarily hold for multi-stage games also applies to public-indifferentiability, but the theorem holds for single-stage games as long as this condition holds. The public indifferentiability advantage is defined as $\mathsf{Adv}^{\text{pub-indiff}}_{\mathsf{T},\mathsf{R},\mathcal{S}}(\mathcal{A}) := \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{T}^{\mathsf{P}},\mathsf{P}}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{R},\mathcal{S}^{\mathsf{R},\mathsf{Rev}[\mathsf{R}]}}\right]$. Informally, we say T^{P} is public indifferentiable from $\mathcal{R}$ if there exists an efficient simulator $\mathcal{S}$ such that the above advantage becomes negligibly small for any efficient $\mathcal{A}$. The Merkle-Damgård construction is proven public indifferentiable [40].

Remark 1. While it is straightforward to show the composition of two indifferentiable constructions becomes again indifferentiable[13], its seems quite hard (or even impossible) to prove that the composition of two public indifferentiabile constructions becomes again public indifferentiable. (See Section 6.1 for details).

3 Code Lifting on GCM

This section briefly explains that GCM [64] is unlikely to achieve incompressibility in the presence of a lifter given an unlimited access to an implementation, even when used with an incompressible block cipher. Recall that GCM is an AEAD mode of 128-bit block cipher composed of CTR and a universal hash function called GHASH (see Fig. 4 in the appendix for details). As an input, the encryption function of GCM takes a tuple of a nonce N, associated data A, and a message M. Given an input (N, A, M), CTR first encrypts M into a ciphertext C' with an IV derived from N. Then, a tag value T is computed as $T := \mathrm{GHASH}_{E_K(0^{128})}(A, C') \oplus E_K(N||1)$. The output of the encryption function is $T||C'$. GCM is proven secure in the nonce-respecting scenario where each nonce is never repeated for encryption queries[14]. When a nonce is repeated, GCM is broken even in the black-box setting.

An important feature of GCM is that the authenticity heavily relies on the value $E_K(0^{128})$: Suppose we know $E_K(0^{128})$ in addition to the tag T and the ciphertext C' for an input (N, A, M). Then, for arbitrary $\tilde{A}$ and $\tilde{M}$ with $\mathrm{len}(\tilde{M}) \leq \mathrm{len}(M)$, we can produce the tag $\tilde{T}$ and the ciphertext $\tilde{C}'$ corresponding to

[13] If $\mathcal{S}$ (resp., $\mathcal{S}'$) is a simulator for a construction T^{P} (resp., U^{Q}) making the indifferentiability advantage small (and if the interfaces are compatible), then $\mathcal{S}'^{\mathcal{S}}$ makes the advantage for $\mathsf{T}^{\mathsf{U}^{\mathsf{Q}}}$ small.

[14] Note that nonce reuse for *decryption* is allowed.

$(N, \tilde{A}, \tilde{M})$ *without knowing the secret key* K as $\tilde{C}' = \tilde{M} \oplus$(the upper len$(\tilde{M})$ bits of $M \oplus C$) and $\tilde{T} = \mathrm{GHASH}_{E_K(0^{128})}(A, C') \oplus T \oplus \mathrm{GHASH}_{E_K(0^{128})}(\tilde{A}, \tilde{C}')$. This means the universal forgery attack is possible and the authenticity of GCM is completely broken once an adversary retrieves $E_K(0^{128})$.

In the black-box setting, the value $E_K(0^{128})$ is hidden from adversaries and GCM achieves authenticity. However, in the white-box setting where a lifter has an unlimited access to a white-box implementation of GCM, the lifter could copy and leak the value $E_K(0^{128})$ to an attacker to break authenticity[15]. This attack works even if the underlying block cipher E_K is incompressible. Just copying a single 128-bit string $E_K(0^{128})$ would not be difficult no matter how hard copying the full functionality of E_K is.

The above attack shows that GCM fails to inherit incompressibility from E_K: A relatively small amount of data $E_K(0^{128})$ leaks information on an exponentially many input-output pairs of GCM. Similar attacks exist for other AE modes such as CCM, GCM-SIV, and OCB. See Section B in the appendix for details.

4 New AEAD Security Notion

This section gives us a formal definition of incompressibility-based white-box security of an AEAD implementation. Security notions for other cryptosystems are given later based on the definition for AEADs.

The attack in the previous section (and the ones in Section B) shows that, with raw implementation of AEAD modes such as GCM, a small amount of leakage from the underlying white-box cipher could lead to giving the adversary a great deal of information concerning valid ciphertext values of the overlying AEAD scheme. Clearly this is an undesirable situation.

Basically, we want that a small amount of leakage would only lead to a small amount of valid ciphertext information, but there is a subtlety. A white-box attacker, or *lifter* (e.g., malware) could locally encrypt a large number of messages and then compute leakage of a small size from the obtained ciphertexts. As a result, the leakage, as a function, may depend on a large number of ciphertext values. Intuitively, we want that:

1. The leakage should not contain information yielding ciphertext values that have not been computed by the lifter, so that the ciphertexts that the adversary can compute from the leakage are limited to those that have been already computed by the lifter, and
2. The number of ciphertexts that the adversary can compute from the leakage should be small likewise the leakage size.

We establish a security notion that formalizes these requirements.

[15] The value $E_K(0^{128})$ could be protected from some white-box attacks with software or hardware countermeasures. Still, the effectiveness of such countermeasures would be limited, given that existing white-box implementations of AES ensure security only when adversaries have limited access to implemented algorithms. In addition, our aim is to achieve white-box security without assuming trusted hardware.

4.1 White-Box AEAD Attack Model

This section shows our attack model on white-box AEADs. We discuss on the security *after* the code lifting because no security can be guaranteed before and during the code lifting.

First, we provide an intuitive observation on what kind of attackers we have to take into account. Assume a white-box AEAD scheme is running on a target device, e.g., a remote server or a smartphone. Real-world attackers will behave as follows: First, an attacker performs advance preparation on the target scheme, making black-box queries to the encryption and decryption functions if possible. Then the attacker creates a lifter, e.g., a malware or an analysis tool, and give the lifter access to the white-box implementation by any means[16]. After analyzing the implementation, the lifter leaks some information on the scheme to the attacker. Finally, the attacker tries to break the privacy or authenticity of the scheme by using the leakage.

Based on the above observation, we reach the following attack model. Formally, let $\Pi = (\mathcal{E}, \mathcal{D})$ be an AEAD and $\mathcal{C}_\Pi$ its compiler. A *white-box adversary* $\mathcal{A} = (\mathcal{A}_{\mathrm{create}}, \mathcal{A}_{\mathrm{dist}})$ is a pair of oracle-aided, probabilistic random-access machines (RAMs.) The adversary $\mathcal{A}$ attacks $\mathcal{C}_\Pi$, running in two stages, as follows:

Initialization. A key K is chosen uniformly at random. Then using this key we put $\mathcal{P} \leftarrow \mathcal{C}_\Pi(K)$.

1st stage: creating a lifter. The first-stage is run by the sub-adversary $\mathcal{A}_{\mathrm{create}}$ which has only black-box access to $\mathcal{P}$, making queries to oracles $\mathcal{E}_K$ and $\mathcal{D}_K$. The goal of $\mathcal{A}_{\mathrm{create}}$ is to output a deterministic RAM $\mathcal{L}$ which we call a *lifter*.

Lifter execution. Once created, the lifter $\mathcal{L}$ gets full access to the AEAD program $\mathcal{P}$. The lifter $\mathcal{L}$ tries to extract some useful information out of the implementation $\mathcal{P}$, for example key material or compressed codes, and sends leakage data L to the adversary. The size of L is restricted to λ bits, which are properly smaller than the description of $\mathcal{P}$.

2nd stage: distinguishing. Upon receiving leakage L from the lifter, the second-stage sub-adversary $\mathcal{A}_{\mathrm{dist}}$ resumes querying to $\mathcal{E}_k$ and $\mathcal{D}_k$, and finally outputs a bit string.

We could consider various sorts of adversarial goals, such as key recovery, plaintext recovery and ciphertext forgery. Of these, we choose the distinguishing attack, extending the "gold-standard" IND-CCA in the black-box setting: We assume $\mathcal{A}_{\mathrm{dist}}$ finally outputs a bit b. The final goal of the adversary $\mathcal{A}$ is to distinguish between the real world ($b = 1$) and the ideal world ($b = 0$), i.e., whether the oracles $\mathcal{E}_K$ and $\mathcal{D}_K$ and the leakage have been real, or they have been some random and simulated ones.

Of course, white-box implementation is not present in the black-box security definitions, so we shall define how the leakage is computed in the ideal world.

[16] For instance, if the target device is a remote server, the lifter would be a malware that sneaks into the server. If the device is a smartphone, the lifter would be an analysis tool and the attacker may take the advantage of a slight opportunity to analyze the smartphone while the owner does not pay attention to it.

Consequently, we shall later introduce a simulator that imitates the behavior of the lifter $\mathcal{L}$.

Even if it is impossible to prevent lifters from getting access to the white-box implementation, we expect it is still possible to notice an attack is being mounted when non-negligible amount of data is sent to a strange and suspicious direction, by monitoring outgoing packets. Hence we define security notions when the leakage size λ is limited, e.g., up to 2^{20} bits. No security is guaranteed after λ reaches the limitation. In addition, basically we assume the running time of the lifter t_{lif} is much smaller than that of the adversary t (e.g., $t_{\mathrm{lif}} = 2^{50}$ while $t = 2^{112}$) and the intrusion of the lifter can be detected after t_{lif} time has passed.

We do not formalize the attack model in such a way $\mathcal{L}$ communicates with $\mathcal{A}$ since $\mathcal{L}$ can do everything $\mathcal{A}$ can do, and thus communications do not help much to break the scheme.

4.2 Ideal Oracles and Simulators

It remains to describe the ideal world in order to give a formal definition of white-box AEAD security.

When *black-box* security of nonce-based AEAD is studied, typically the ideal encryption (resp., decryption) oracle is set to be the one that always returns a random ciphertext (resp., the reject symbol). The adversary is prohibited to forward outputs from the encryption oracle to the decryption oracle to exclude trivial attacks.

On the other hand, in our white-box setting we cannot set the ideal oracles like above because the adversary can distinguish the ideal decryption oracle from the real one if the lifter leaks a valid ciphertext C and the adversary queries C to the decryption oracle.

Thus we set *a fixed-key random injection* F and its inverse F^{-1} as the ideal oracles (see Definition 1), following previous works on pseudorandom injection (PRI) security of AEADs [74, 48]. In particular, our security notion will completely match the black-box PRI security when $\lambda = 0$.

Note that the black-box PRI security matches the misuse-resistant AE (MRAE) security [74, 48] if the tag length τ is sufficiently long: Roughly speaking, the difference between the PRI advantage and the MRAE advantage of an AEAD scheme is upper-bounded by $O(q^2/2^\tau)$, where q is the number of black-box oracle queries [48, Thorem 1]. Thus our white-box security notion will require a secure scheme to be at least MRAE-secure in the black-box model.

Simulators. Now what remains of the real world is the program $\mathcal{P}$ and the lifter $\mathcal{L}$. $\mathcal{P}$ does not exist in the ideal world and it is non-trivial how we should define the behavior of $\mathcal{L}$. To remedy this, we introduce a simulator that imitates the behavior of $\mathcal{L}$.

Recall our intuition on the property that a secure white-box scheme must meet: Any leakage on a secure scheme does not contain information enabling an adversary to compute ciphertext values that have not been computed by a lifter. In other words, information that a lifter $\mathcal{L}$ can send to an adversary $\mathcal{A}_{\mathrm{dist}}$ (with

reasonable computational resources) is only those computable or *simulatable* from some input-output pairs of $\mathcal{E}_K$ and $\mathcal{D}_K$.

We model this situation by existence of a simulator $\mathcal{S}$ working as follows. Given the description of a lifter $\mathcal{L}$[17] and oracle access to F and F^{-1} in the ideal world, $\mathcal{S}$ produces a bit string L_{ideal} which is, to $\mathcal{A}_{\text{dist}}$, indistinguishable from leakage L_{real} by $\mathcal{L}$ in the real world.

Since F is an ideally random object, $\mathcal{S}$ in the ideal world cannot leak more than λ bits of information on $F^{\pm}$ via λ-bit leakage L_{ideal}. Hence, intuitively, if $\mathcal{A}_{\text{dist}}$ cannot L_{real} and L_{ideal}, then $\mathcal{L}$ in the real world cannot leak more than λ bits of information on $\mathcal{E}_K$ and $\mathcal{D}_K$ via λ-bit leakage L_{real}.

More specifically, a simulator $\mathcal{S}$ is an oracle-aided RAM. We give $\mathcal{S}$ the ability to do its job as follows:

1. We give $\mathcal{S}$ as its input the lifter $\mathcal{L}$ just as it is. Then $\mathcal{S}$ can perform static and dynamic analyses on $\mathcal{L}$. The code of $\mathcal{L}$ can be read, dissected and studied, so that $\mathcal{S}$ can determine the functionality of $\mathcal{L}$.
2. Needless to say, we let $\mathcal{S}$ have oracle access to $F^{\pm 1}$.
3. We give $\mathcal{S}$ sufficient computational power and do not explicitly bound its running time. We only demand that the algorithm $\mathcal{S}$ be a finite sequence of well-defined instructions and operations. By doing so, we believe that our security notion should become achievable by a sound portion of AEAD programs while dismissing the rest.

In addition, we assume that $\mathcal{S}$ can observe all the queries to $F^{\pm}$ by $\mathcal{A}_{\text{create}}$ and the responses. The reasons that we assume this is as follows. First, if we define a security notion for conventional encryption schemes similarly *without this assumption*, then a conventional encryption scheme (random-IV CTR) which intuitively seems white-box-secure is deemed insecure (see Section D in the appendix for details). However, if the assumption is included in the definition, random-IV CTR can be proven secure (Section 6.3). Thus it seems reasonable to include the assumption into the definition for conventional encryption schemes. Second, We would like to make security definitions for various cryptosystems consistent as much as possible. Thus we include this assumption not only for conventional encryption schemes but also for AEADs.

4.3 Formal Security Notion: whPRI

Now we are ready to define new security notion of AEAD programs. We call our notion *white-box pseudo-random injection security (whPRI)*.

We consider a white-box adversary $\mathcal{A} = (\mathcal{A}_{\text{create}}, \mathcal{A}_{\text{dist}})$ running in two different experiments called games. The real white-box PRI game ($\boxed{\text{PRI}}$-real) is an experiment in the real world as described in Sect. 4.1. We assume that the white-box program $\mathcal{P}$ contains an implementation of not only encryption but also decryption. The ideal white-box PRI game ($\boxed{\text{PRI}}$-ideal) is an experiment

[17] Note that a lifter is also made by a first-stage adversary $\mathcal{A}_{\text{create}}$ in the ideal world, but the black-box oracles given to $\mathcal{A}_{\text{create}}$ are (F, F^{-1}) instead of $(\mathcal{E}_K, \mathcal{D}_K)$.

Experiment 5: $\mathbf{Exp}^{\boxed{\text{PRI}}\text{-real}}_{\Pi,\mathcal{C}_\Pi,\mathcal{A}}$	**Experiment 6:** $\mathbf{Exp}^{\boxed{\text{PRI}}\text{-ideal}}_{\mathcal{S},\mathcal{A}}$
1: $K \xleftarrow{\$} \mathbf{K}$, $\mathcal{P} \leftarrow \mathcal{C}_\Pi(K)$	1: $F \xleftarrow{\$} \mathrm{Inj}_\tau(\mathbf{N} \times \mathbf{A} \times \mathbf{M}, \mathbf{C})$
2: $(\mathcal{L}, S) \leftarrow {\mathcal{A}_{\text{create}}}^{\mathcal{E}_K, \mathcal{D}_K}()$	2: $(\mathcal{L}, S) \leftarrow {\mathcal{A}_{\text{create}}}^{F, F^{-1}}()$
3: $L \leftarrow \mathcal{L}(\mathcal{P})$	3: $L \leftarrow \mathcal{S}^{F,F^{-1}}(\mathcal{L}, \mathsf{List}_{\text{create}})$
4: $\beta \leftarrow {\mathcal{A}_{\text{dist}}}^{\mathcal{E}_K, \mathcal{D}_K}(S, L)$	4: $\beta \leftarrow {\mathcal{A}_{\text{dist}}}^{F, F^{-1}}(S, L)$
5: **return** β	5: **return** β

Fig. 1: Experiments for whPRI. In the ideal experiment, $\mathsf{List}_{\text{create}}$ denotes the list of queries by $\mathcal{A}_{\text{create}}$ to $F^{\pm}$ and the responses.

in the ideal world, where the oracles and the lifter are replaced with a random injection and a simulator, respectively. These two games are formally defined in Exp. 5 and Exp. 6.

Now, given an AEAD scheme Π and its compiler $\mathcal{C}_\Pi$, let us define the *wh-PRI advantage* of a white-box adversary $\mathcal{A} = (\mathcal{A}_{\text{create}}, \mathcal{A}_{\text{dist}})$ with respect to a simulator $\mathcal{S}$ as $\mathbf{Adv}^{\text{whPRI}}_{\Pi,\mathcal{C}_\Pi,\mathcal{S}}(\mathcal{A}) := \Pr\left[\mathbf{Exp}^{\boxed{\text{PRI}}\text{-real}}_{\Pi,\mathcal{C}_\Pi,\mathcal{A}} = 1\right] - \Pr\left[\mathbf{Exp}^{\boxed{\text{PRI}}\text{-ideal}}_{\mathcal{S},\mathcal{A}} = 1\right]$.

Definition 2 (whPRI). *The pair of an AEAD scheme Π and a compiler $\mathcal{C}_\Pi$ is $(\lambda, t, q, \sigma, t_{\text{lif}}, q_{\text{sim}}, \sigma_{\text{sim}}, \epsilon)$-whPRI-secure white-box AEAD if the following condition is satisfied: Let $\mathcal{A}$ be an arbitrary adversary running in time t and making queries at most q times. The lengths of the queries are at most σ in total*[18]*. In addition, $\mathcal{A}$ creates a lifter that runs in time t_{lif} and outputs at most λ-bit leakage. For arbitrary such $\mathcal{A}$, there exists a simulator $\mathcal{S}$ that makes at most q_{sim} queries of which lengths are at most σ_{sim} in total, and satisfies an inequality $\mathbf{Adv}^{\text{whPRI}}_{\Pi,\mathcal{C}_\Pi,\mathcal{S}}(\mathcal{A}) < \epsilon$.*

Informally, suppose the following claim holds: For any "efficient" $\mathcal{A}$, there exists a simulator $\mathcal{S}$ that makes a "reasonable amount of" queries and making the whPRI-advantage small[19]. Then we say that Π is whPRI-secure.

The attacks on GCM, GCM-SIV, CCM, OCB in Section 3 (or in Section B) show that, for each of those schemes, there exists a lifter $\mathcal{L}$ that leaks the information on exponentially many number of input-output pairs by only a small amount of leakage. In the ideal world, the information of input-output pairs of the black-box oracle $F^{\pm}$ that a simulator can output by a λ-bit leakage is at

[18] The unit of length can be set arbitrarily (e.g., bit or block) depending on the context.

[19] We set quantifiers as $\forall\mathcal{A}\exists\mathcal{S}$ rather than $\exists\mathcal{S}\forall\mathcal{A}$ so that the possibility of existence of primitives will increase, and the order of the quantifiers seems to have little impact on whether a practical scheme is judged secure or not. Indeed, our proofs in later sections, in addition to the discussions about the attacks on GCM, GCM-SIV, CCM, OCB mentioned below, work regardless of the order of the quantifiers.

Experiment 7: $\mathbf{Exp}^{\boxed{\text{PRP}}\text{-real}}_{E,\mathcal{C}_E,\mathcal{A}}$

1: $K \xleftarrow{\$} \{0,1\}^\kappa$, $\mathcal{P} \leftarrow \mathcal{C}_E(K)$
2: $(\mathcal{L}, S) \leftarrow {\mathcal{A}_{\text{create}}}^{E_K}()$
3: $L \leftarrow \mathcal{L}(\mathcal{P})$
4: $\beta \leftarrow {\mathcal{A}_{\text{dist}}}^{E_K}(S, L)$
5: **return** β

Experiment 8: $\mathbf{Exp}^{\boxed{\text{PRP}}\text{-ideal}}_{\mathcal{S},\mathcal{A}}$

1: $P \xleftarrow{\$} \text{Perm}(n)$
2: $(\mathcal{L}, S) \leftarrow {\mathcal{A}_{\text{create}}}^{P}()$
3: $L \leftarrow \mathcal{S}^{P,P^{-1}}(\mathcal{L}, \mathsf{List}_{\text{create}})$
4: $\beta \leftarrow {\mathcal{A}_{\text{dist}}}^{P}(S, L)$
5: **return** β

Fig. 2: Experiments for whPRP. $\mathsf{List}_{\text{create}}$ in Experiment 2 denotes the list of queries to P by $\mathcal{A}_{\text{create}}$ and the responses.

most λ-bit. Hence no simulator will be able to mimic the behavior of such $\mathcal{L}$. Therefore those modes are unlikely to achieve whPRI-security.

5 New White-Box Security Notions for Other Schemes

This section introduces white-box security notions for block ciphers, keyed functions, and conventional encryption schemes.

5.1 whPRP: Secure White-Box Block Ciphers

We call the new security notion for white-box block ciphers *white-box pseudorandom permutation security* (*whPRP*). The definition of whPRP is similar to that of whPRI; the oracles $\mathcal{E}_K$ and $\mathcal{D}_K$ are now just E_K, and its counterpart in the ideal game is a random permutation $P \in \text{Perm}(n)$, where $\text{Perm}(n)$ denotes the set of permutations on $\{0,1\}^n$.

We again consider a white-box adversary $\mathcal{A} = (\mathcal{A}_{\text{create}}, \mathcal{A}_{\text{dist}})$ running in two games: the real white-box PRP game ($\boxed{\text{PRP}}$-real) which is formally defined in Exp. 7 and the ideal white-box PRP game ($\boxed{\text{PRP}}$-ideal) in Exp. 8. We assume that the white-box program given to a lifter contains an implementation of not only encryption but also decryption. Then, given a block cipher E and its compiler $\mathcal{C}_E$, let us define the *whPRP advantage* of a white-box adversary $\mathcal{A} = (\mathcal{A}_{\text{create}}, \mathcal{A}_{\text{dist}})$ with respect to a simulator $\mathcal{S}$ as $\mathbf{Adv}^{\text{whPRP}}_{E,\mathcal{C}_E,\mathcal{S}}(\mathcal{A}) := \Pr\left[\mathbf{Exp}^{\boxed{\text{PRP}}\text{-real}}_{E,\mathcal{C}_E,\mathcal{A}} = 1\right] - \Pr\left[\mathbf{Exp}^{\boxed{\text{PRP}}\text{-ideal}}_{\mathcal{S},\mathcal{A}} = 1\right]$.

Definition 3 (whPRP). *The pair of a block cipher E and a compiler $\mathcal{C}_E$ is a $(\lambda, t, q, t_{\text{lif}}, q_{\text{sim}}, \epsilon)$-secure whPRP if the following condition is satisfied: Let $\mathcal{A}$ be an arbitrary adversary running in time t and making at most q queries. $\mathcal{A}$ makes a lifter that runs in time t_{lif} and outputs at most λ-bit leakage. For arbitrary such $\mathcal{A}$, there exists a simulator $\mathcal{S}$ that runs in time t_{sim}, makes at most q_{sim} queries, and satisfies an inequality $\mathbf{Adv}^{\text{whPRP}}_{E,\mathcal{C}_E,\mathcal{S}}(\mathcal{A}) < \epsilon$.*

Informally, suppose the following claim holds: For any "efficient" $\mathcal{A}$, there exists a simulator $\mathcal{S}$ that makes a "reasonable amount of" queries and making the whPRP-advantage small. Then we say that Π is whPRP-secure.

The definition of whPRP is a strengthening of the conventional black-box PRP, as the latter corresponds to the case $\lambda = 0$. It should be noted that we allow the simulator $\mathcal{S}$ to make queries to P^{-1}.

We can also consider the strong PRP version, whSPRP, where $\mathcal{A}$ is given oracle access to not only E_K but also E_K^{-1}. It is strictly stronger than whPRP.

As a candidate of whPRP, we conjecture[20] that SPACE-n_a ($n_a \in \{8, 16, 24, 32\}$) is a $(\lambda, t, q, t_{\mathrm{lif}}, q_{\mathrm{sim}}, \epsilon)$-secure whPRP with $t \approx 2^{\kappa}$, $q \approx 2^n$, $\lambda \approx (n - n_a) \cdot 2^{n_a - 2}$, and $\epsilon \ll 1$, as long as $t_{\mathrm{lif}} \ll q_{\mathrm{sim}} (< 2^n)$. Here, n and κ denote the block and key length, which are 128. More details on why we expect that this conjecture holds can be found in Section M in the appendix.

5.2 whPRF: Secure White-Box Keyed Functions

We call the new security notion for white-box keyed functions *white-box pseudo-random function security* (*whPRF*)[21], which is defined in the same way as whPRP except that the black-box oracle given to the adversary is a random function RF instead of a random permutation P, and that simulators have access to RF instead of P and P^{-1}. Real and ideal experiments in addition to a distinguishing advantage are defined in the same way as those for whPRP.

5.3 White-Box Security of Conventional Encryption Schemes

We define two security notions on conventional IV-based encryption schemes, which we name *tweakable strong PRP security* (whSPRP) and *white-box IND$-CPA security* (whIND$-CPA).

whSPRP. The most natural way to obtain a definition of conventional IV-based encryption schemes is to consider the special case of whPRI where $\mathbf{A} = \{\varepsilon\}$ and $\tau = 0$. This is an extension of (VIL) tweakable strong PRP (SPRP) security for enciphering schemes in the black-box setting [47], and thus we call it whSPRP.

[20] Note that it is unrealistic to "prove" whPRP-security of SPACE-256-16 in the same sense as proving PRP security of AES is unrealistic. Generally, the only realistic way to be confident with security of a block cipher is to see whether it withstands various attempts of cryptanalysis by experts. Recently, the security of some space-hard block ciphers was reviewed against a similar adversary to whPRP in [78].

[21] We define a white-box version of PRF security but does not for MAC security such as existential unforgeability. This is because a lifter can leak a valid message-tag pair that has not been queried to oracles before, and thus it seems hard to achieve a sound white-box version of existential unforgeability. It might be possible to define a white-box version of weaker notions such as universal unforgeability, but such notions are out of the scope of this paper. Studying weaker notions is a future work.

whIND$-CPA. Though wh$\widetilde{\text{SPRP}}$ is naturally derived from whPRI, many popular conventional encryption schemes such as CTR and CBC are not $\widetilde{\text{SPRP}}$-secure even in the black-box setting. Thus we seek for another definition extending ones that CTR and CBC meet in the black-box setting. Since CTR and CBC cannot achieve indistinguishability against CCAs, we focus on security against CPAs.

In the black-box setting, we have three scenarios depending on how IVs for encryption queries are chosen.

1. Arbitrary IV (or, nonce-misuse) scenario: IVs are chosen by adversaries completely arbitrarily.
2. Nonce IV (or, nonce-respecting) scenario: IVs are chosen by adversaries arbitrarily, but repeated uses are prohibited (i.e., once an IV value is used for a query, it is never be used again).
3. Random IV scenario: An IV is chosen uniformly at random for every encryption query.

CTR and CBC cannot achieve indistinguishability in the first scenario. The second scenario is popular in the black-box setting but not suitable in our context since a lifter may leak information on a valid message-ciphertext pair w.r.t. an unused nonce. Thus we focus on the random IV scenario.

We follow [69] for the black-box security notion against CPAs for conventional random-IV encryption scheme. The notion is defined by real and ideal experiments. In the real experiment, an adversary has an access to a modified version of the encryption oracle $\mathcal{E}_K$, which we denote by $\mathcal{E}_{K,\mathrm{rnd}}$. For each encryption query, $\mathcal{E}_{K,\mathrm{rnd}}$ chooses IV uniformly at random, and returns $(IV, \mathcal{E}_K(IV))$. In the ideal experiment, $\mathcal{E}_{K,\mathrm{rnd}}$ is replaced with an oracle $\$(\cdot)$ that just returns a random IV and a random ciphertext of the same length as the message. A scheme is defined to be secure if an adversary with a reasonable amount of computational resources cannot distinguish the two experiments. We call this black-box security notion IND$-CPA[22].

Our new notion whIND$-CPA is defined by extending IND$-CPA in the same way as whPRI is defined extending PRI security. In the real world, the black-box oracle given to $\mathcal{A}$ is $\mathcal{E}_{K,\mathrm{rnd}}$ only. In the ideal world, the oracle $\$(\cdot)$ is given to both of $\mathcal{A}$ and $\mathcal{S}$. We show the real and ideal experiments in Fig. 3 in the appendix. The advantage $\mathbf{Adv}^{\mathrm{whIND\$-CPA}}_{\Pi,\mathcal{C}_\Pi,\mathcal{S}}(\mathcal{A})$ is defined as before. We assume that the white-box program given to a lifter contains an implementation of not only encryption but also decryption.

6 Weak Public Indifferentiability and White-Box Security Reductions

This section first introduces a weaker version of public indifferentiability which we name *weak public indifferentiability*. Second, we show that weak public indif-

[22] This name is from [72], though it is defined for nonce-based scheme rather than random-IV schemes.

ferentiability implies reductions between our white-box security notions introduced in Sections 4 and 5. Third, we provide feasibility results that our white-box security notions on various schemes can be reduced to whPRP, by showing weak public indifferentiable constructions.

6.1 Weak Public Indifferentiability and Compositions

An important point to be aware of about public indifferentiability is that it seems quite hard to prove a composition of two arbitrary public indifferentiable scheme become again public indifferentiable. This is because the general "composition theorem" is not applicable to show public indifferentiability of composite schemes due to the following reason. Suppose a scheme U^{Q} (Q is an ideally random primitive) is public indifferentiable from a random object P (e.g., a random oracle). Then, what the general "composition theorem" for public indifferentiability says is that we can safely replace P in a protocol or construction with U^{Q} if the security of the protocol/construction is defined by single-stage games satisfying the following condition: Queries to P by any party involved in the security games can be made public without affecting the security. (We denote this condition by (C).) Now, assume that there is another scheme T^{P} that is public indifferentiable from a random object R. If (C) were satisfied by the security games of T^{P} (i.e., by the security games of public indifferentiability), public indifferentiability of U^{Q} and the "composition theorem" would imply public indifferentiability of $\mathsf{T}^{\mathsf{U}^{\mathsf{Q}}}$. However, (C) is not satisfied because queries by a simulator must not be visible to an adversary in the ideal game. Thus the general "composition theorem" is not applicable to prove public indifferentiability of $\mathsf{T}^{\mathsf{U}^{\mathsf{Q}}}$. (See also Remark 2.)

However, infeasibility of compositions is inconvenient because security proofs cannot be provided in a modular way. To remedy this, we introduce a weaker variant which we name *weak public indifferentiability*. Let T^{P} be a construction querying to an ideally random primitive P, and let R be a random object of which input-output interfaces are compatible with T^{P}. Now, let $\mathsf{Rev}'[\mathsf{R}]$ be a variant of the revealing interface $\mathsf{Rev}[\mathsf{R}]$ that returns the list of all the queries made so far by $\mathcal{A}$, but not by $\mathcal{S}$, together with the responses[23]. We define *weak public indifferentiability* in the same way as public indifferentiability is defined except that the revealing interface is $\mathsf{Rev}'[\mathsf{R}]$ instead of $\mathsf{Rev}[\mathsf{R}]$. Weak public indifferentiability advantage is defined as $\mathsf{Adv}^{\text{weak-pub-indiff}}_{\mathsf{T},\mathsf{R},\mathcal{S}}(\mathcal{A}) := \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{T}^{\mathsf{P}},\mathsf{P}}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{R},\mathcal{S}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}}\right]$.

[23] Note that lists returned by $\mathsf{Rev}'[\mathsf{R}]$ contain more useful information for $\mathcal{S}$ than lists returned by $\mathsf{Rev}[\mathsf{R}]$. This is because (1) $\mathcal{S}$ can record what it has queried to R so far by itself, and (2) Sometimes $\mathcal{S}$ cannot tell which queries recorded in a list by $\mathsf{Rev}[\mathsf{R}]$ have been queried by $\mathcal{A}$: If a value x had been queried to R for the first time by $\mathcal{S}$ but not $\mathcal{A}$, there is no means for $\mathcal{S}$ to know whether $\mathcal{A}$ queried x to R afterwards.

A public indifferentiable scheme is weak public indifferentiable[24]. This is because a simulator $\mathcal{S}$ for public indifferentiability can be converted into a one for weak public indifferentiability just by recording queries that $\mathcal{S}$ makes to R.

On Compositions of Two Weak Public Indifferentiable Schemes. Here we explain that a composition of two weak public indifferentiable schemes become weak public indifferentiable if a few additional conditions are satisfied. To explain this, we formally define *random-IV schemes*. Note that we say a construction T^{P} is deterministic if, for an arbitrary input X, the output value $\mathsf{T}^{\mathsf{P}}(X)$ is unchanged during each game.

Definition 4. *A construction* T^{P} *is a* random-IV scheme *if it is a public-coin protocol. Namely, there exists a deterministic construction* $\tilde{\mathsf{T}}^{\mathsf{P}}$ *and a set* $\mathbf{IV}$ *such that, on arbitrary input* X*,* T^{P} *runs as follows: (1) Take a value* IV *from* $\mathbf{IV}$ *uniformly at random. (2) Return* $(IV, \tilde{\mathsf{T}}^{\mathsf{P}}(IV, X))$*.*

The following lemma shows the composition of two weak public indifferentiable schemes is again weak public indifferentiable if a few additional conditions are satisfied. Here we provide only an informal version due to page limitation. See Section E in the appendix for a formal version and a proof.

Lemma 1 (Composition of weak public indifferentiable schemes, informal). *Suppose the following (1)-(3) hold: (1)* T^{P} *is a* deterministic or random-IV *scheme calling an ideally random primitive* P *and is weak public indifferentiable from* R*, (2)* U^{Q} *is another* deterministic *construction calling an ideally random primitive* Q *and is weak public indifferentiable from* P*, and (3)* P *and* Q *are deterministic. Then* $\mathsf{T}^{\mathsf{U}^{\mathsf{Q}}}$ *is also weak public indifferentiable from* R*, regarding* Q *as the primitive oracle.*

All compositions of (weak public) indifferentiable schemes appearing in this paper satisfy (1)-(3).

Intuition of the Proof. Here we explain a sketch of the proof when all the functions and constructions are deterministic. Suppose the ideal game for T^{U} is being executed with an adversary $\mathcal{A}$.

Let $\mathcal{S}_{\mathsf{T}}$ (resp., $\mathcal{S}_{\mathsf{U}}$) be a "good" simulator for T (resp., U) making the indifferentiability advantage small. Then, a "good" simulator $\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}$ for T^{U} is defined as follows, by using $\mathcal{S}_{\mathsf{T}}$ and $\mathcal{S}_{\mathsf{U}}$ as subroutines: When a value x is queried to $\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}$, it first runs $\mathcal{S}_{\mathsf{U}}$ on the input x as a subroutine. Intuitively, $\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}$ tries to convince the subroutine $\mathcal{S}_{\mathsf{U}}$ that "now $\mathcal{S}_{\mathsf{U}}$ is run as a part of $\mathcal{A}^{\mathsf{T}^{\mathsf{P}},\mathcal{S}_{\mathsf{U}}^{\mathsf{P},\mathsf{Rev}'[\mathsf{P}]}}$". When $\mathcal{S}_{\mathsf{U}}$ returns an output, $\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}$ returns it to $\mathcal{A}$ as its own output. To achieve this, $\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}$ simulates the oracles P and $\mathsf{Rev}'[\mathsf{P}]$ for $\mathcal{S}_{\mathsf{U}}$. P is simulated just by running $\mathcal{S}_{\mathsf{T}}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}$. (Note

[24] It seems hard to prove weak public indifferentiability implies public indifferentiability, but currently we are not aware of any separation example that is weak public indifferentiable but not public indifferentiable.

that $\mathcal{S}_{\mathsf{T}^\cup}$ is given oracle access to R and $\mathsf{Rev}'[\mathsf{R}]$.) The non-trivial part is how to simulate the oracle $\mathsf{Rev}'[\mathsf{P}]$.

What the subroutine $\mathcal{S}_\mathsf{U}$ is expecting to receive when it makes a query to the revealing interface is a list storing queries (and the responses) to P that are made so far by $\mathcal{A}$ through T (but not by $\mathcal{S}_\mathsf{U}$) while running $\mathcal{A}^{\mathsf{T}^\mathsf{P},\mathcal{S}_\mathsf{U}^{\mathsf{P},\mathsf{Rev}'[\mathsf{P}]}}$. Hence, when the subroutine $\mathcal{S}_\mathsf{U}$ makes a query to the revealing interface, $\mathcal{S}_{\mathsf{T}^\cup}$ simulates the oracle $\mathsf{Rev}'[\mathsf{P}]$ as follows. First, $\mathcal{S}_{\mathsf{T}^\cup}$ queries to $\mathsf{Rev}'[\mathsf{R}]$ to get the list $\mathsf{List}_\mathcal{A}[\mathsf{R}]$ of queries made so far to R by $\mathcal{A}$ (but not by $\mathcal{S}_{\mathsf{T}^\cup}$). Then $\mathcal{S}_{\mathsf{T}^\cup}$ computes the function $\mathsf{T}^{\mathcal{S}_\mathsf{T}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}}$ on the input X for each entry (X, Y) in $\mathsf{List}_\mathcal{A}[\mathsf{R}]$, recording all the queries by T to $\mathcal{S}_\mathsf{T}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}$ into a list $\mathsf{List}_{\mathrm{prim}}$, together with the responses. Finally, $\mathcal{S}_{\mathsf{T}^\cup}$ returns $\mathsf{List}_{\mathrm{prim}}$ to $\mathcal{S}_\mathsf{U}$ as a response. The simulation works well because $\mathcal{S}_{\mathsf{T}^\cup}$ can tell which value has been queried to R so far by $\mathcal{A}$ (but not by $\mathcal{S}_{\mathsf{T}^\cup}$). See Section E in the appendix for further details.

Remark 2. The above idea does not work for (original) public indifferentiability. Here we explain which part fails for public indifferentiability. The non-trivial part of the proof is again how to simulate $\mathsf{Rev}[\mathsf{P}]$ for $\mathcal{S}_\mathsf{U}$. The issue in simulating $\mathsf{Rev}[\mathsf{P}]$ is also again how to determine the values queried to P through T by $\mathcal{A}$ *but not by* $\mathcal{S}_\mathsf{U}$. Now, the procedure "First, $\mathcal{S}_{\mathsf{T}^\cup}$ queries to $\mathsf{Rev}'[\mathsf{R}]$ to get..." does not work for public indifferentiability due to the property (2) in Footnote 23.

6.2 Weak Public Indifferentiability Implies White-Box Reduction

Let $(\pi, \mathcal{C}_\pi)$ be a white-box symmetric-key scheme that are either of a keyed function, block cipher, AEAD, or a conventional IV-based encryption scheme. In addition, let $(\Sigma^\pi, \mathcal{C}_{\Sigma^\pi})$ be another white-box symmetric-key scheme built on $(\pi, \mathcal{C}_\pi)$. We assume Σ calls π in a black-box manner not only at a level of syntax but also at a level of implementation, i.e., the following conditions are satisfied.

1. The implementation of π (denoted by $[\![\pi]\!]$) is included into the implementation of Σ^π (denoted by $[\![\Sigma^\pi]\!]$). In particular, $[\![\pi]\!]$ and an implementation of an oracle-aided algorithm Σ (which is independent from π) is explicitly separated in $[\![\Sigma^\pi]\!]$.
2. The implementation of Σ calls $[\![\pi]\!]$ in a black-box manner.

Our goal is to reduce the security of $(\Sigma^\pi, \mathcal{C}_{\Sigma^\pi})$ to the security of $(\pi, \mathcal{C}_\pi)$. By sec-const (resp., sec-prim) we denote the security notion corresponding to $(\Sigma^\pi, \mathcal{C}_{\Sigma^\pi})$ (resp., $(\pi, \mathcal{C}_\pi)$), which is whPRI, whPRP, whPRF, wh$\widetilde{\text{SPRP}}$, or whIND\$-CPA[25].

By abuse of notations, we use the same symbols Σ^π and π to denote the corresponding keyed black-box oracles given to $(\mathcal{A}_{\mathrm{create}}, \mathcal{A}_{\mathrm{dist}})$ in the white-box security definitions. We assume Σ^π is a deterministic or random-IV scheme: If

[25] We assume the interfaces of π that Σ accesses to are only those given to $\mathcal{A}$ as black-box oracles in the security games of sec-prim. For instance, if π is a block cipher E_K and sec-prim is whPRP, we assume that Σ calls only E_K and does not call E_K^{-1} (though simulators in the ideal game of whPRP access to both of P and P^{-1}).

it is a random-IV scheme, there exists a scheme $\tilde{\Sigma}^{\pi}$ and a set $\mathbf{IV}$ such that Σ^{π} runs as follows on arbitrary input X: (1) IV is chosen uniformly at random from $\mathbf{IV}$. (2) Return $(IV, \tilde{\Sigma}^{\pi}(IV, X))$.

Let R and P denote the ideal oracles given to a simulator in the ideal games of the security definition of $(\Sigma^{\pi}, \mathcal{C}_{\Sigma^{\pi}})$ and $(\pi, \mathcal{C}_{\pi})$, respectively. Suppose there exist non-decreasing functions $q_{\Sigma}(\cdot,\cdot)$ and $\sigma_{\Sigma}(\cdot,\cdot)$ satisfying the following property: If Σ^{π} is evaluated on q inputs of which lengths [26] are σ in total during a game, Σ makes at most $q_{\Sigma}(q,\sigma)$ queries to π and the lengths of the queries are at most $\sigma_{\Sigma}(q,\sigma)$ in total. In addition, assume we have the following three algorithms.

1. An adversary $\mathcal{A} = (\mathcal{A}_{\mathrm{create}}, \mathcal{A}_{\mathrm{dist}})$ against $(\Sigma^{\pi}, \mathcal{C}_{\Sigma^{\pi}})$. The running time is at most $t_{\mathcal{A}}$. The number of black-box oracle queries by $\mathcal{A}$ is at most $q_{\mathcal{A}}$ and the lengths of queries are at most $\sigma_{\mathcal{A}}$ in total. $\mathcal{A}$ creates a lifter running in time t_{lif} and outputs at most λ-bit leakage.
2. A simulator $\mathcal{S}_{\mathrm{prim}}$ for $(\pi, \mathcal{C}_{\pi})$ on sec-prim. $\mathcal{S}_{\mathrm{prim}}$ makes at most $q_{\mathcal{S}_{\mathrm{prim}}}$ queries to the ideal oracle P. The lengths of queries are at most $\sigma_{\mathcal{S}_{\mathrm{prim}}}$ in total.
3. A simulator $\mathcal{S}_{\mathrm{indiff}}$ for weak public indifferentiability of Σ^{P} from R[27]. There exist non-decreasing functions $q_{\mathcal{S}_{\mathrm{indiff}}}(\cdot,\cdot,\cdot,\cdot)$ and $\sigma_{\mathcal{S}_{\mathrm{indiff}}}(\cdot,\cdot,\cdot,\cdot)$ satisfying the following properties: If an adversary makes at most q_c (resp., q_p) construction (resp., primitive) queries of which lengths are at most σ_c (resp., σ_p) in total in the ideal game of weak public indifferentiability, $\mathcal{S}_{\mathrm{indiff}}$ makes at most $q_{\mathcal{S}_{\mathrm{indiff}}}(q_c, \sigma_c, q_p, \sigma_p)$ queries to the ideal oracle R. The lengths of the queries are $\sigma_{\mathcal{S}_{\mathrm{indiff}}}(q_c, \sigma_c, q_p, \sigma_p)$ in total.

Theorem 2. *Let $\mathcal{A}$, $\mathcal{S}_{\mathrm{prim}}$, and $\mathcal{S}_{\mathrm{indiff}}$ be as above. Then there exists an adversary $\mathcal{A}' = (\mathcal{A}'_{\mathrm{create}}, \mathcal{A}'_{\mathrm{dist}})$ against $(\pi, \mathcal{C}_{\pi})$, a simulator $\mathcal{S}_{\mathrm{const}}$ for $(\Sigma^{\pi}, \mathcal{C}_{\Sigma^{\pi}})$, and an algorithm $\mathcal{A}''$ against weak public indifferentiability of Σ such that*

$$\mathbf{Adv}^{\text{sec-const}}_{\Sigma^{\pi}, \mathcal{C}_{\Sigma^{\pi}}, \mathcal{S}_{\mathrm{const}}}(\mathcal{A}) = \mathbf{Adv}^{\text{sec-prim}}_{\pi, \mathcal{C}_{\pi}, \mathcal{S}_{\mathrm{prim}}}(\mathcal{A}') + \mathbf{Adv}^{\text{weak-pub-indiff}}_{\Sigma, \mathsf{R}, \mathcal{S}_{\mathrm{indiff}}}(\mathcal{A}'') \quad (1)$$

holds. Here, we can construct $\mathcal{S}_{\mathrm{const}}$, $\mathcal{A}'$, and A'' so that (a) $\mathcal{A}'$ does not depend on $\mathcal{S}_{\mathrm{prim}}$ and $\mathcal{S}_{\mathrm{indiff}}$, (b) $\mathcal{S}_{\mathrm{const}}$ does not depend on $\mathcal{A}$, (c) $\mathcal{A}''$ does not depend on $\mathcal{S}_{\mathrm{indiff}}$, and the following conditions hold: (1) $\mathcal{S}_{\mathrm{const}}$ makes at most $q_{\mathcal{S}_{\mathrm{indiff}}}(q_{\mathcal{A}}, \sigma_{\mathcal{A}}, q'_{\Sigma}+q_{\mathcal{S}_{\mathrm{prim}}}, \sigma'_{\Sigma}+\sigma_{\mathcal{S}_{\mathrm{prim}}})$ queries to R. The lengths of the queries are at most $\sigma_{\mathcal{S}_{\mathrm{indiff}}}(q_{\mathcal{A}}, \sigma_{\mathcal{A}}, q'_{\Sigma}+q_{\mathcal{S}_{\mathrm{prim}}}, \sigma'_{\Sigma}+\sigma_{\mathcal{S}_{\mathrm{prim}}})$ in total. Here, $q'_{\Sigma} := q_{\Sigma}(q_{\mathcal{A}}, \sigma_{\mathcal{A}})$ and $\sigma'_{\Sigma} := q_{\Sigma}(q_{\mathcal{A}}, \sigma_{\mathcal{A}})$ (2) $\mathcal{A}'$ runs in time $O(t_{\mathcal{A}} + \sigma'_{\Sigma})$ and makes at most q'_{Σ} queries to a black-box oracle. The lengths of the queries are at most σ'_{Σ} in total. $\mathcal{A}'$ creates a lifter $\mathcal{L}'$ that runs in time $O(t_{\mathrm{lif}})$ and outputs at most λ-bit leakage. (3) $\mathcal{A}''$ makes at most $q_{\mathcal{A}}$ construction queries of which lengths are at most $\sigma_{\mathcal{A}}$ in total, and makes at most $q'_{\Sigma} + q_{\mathcal{S}_{\mathrm{prim}}}$ primitive queries of which lengths are at most $\sigma'_{\Sigma} + \sigma_{\mathcal{S}_{\mathrm{prim}}}$ in total.

[26] The unit of length can be set arbitrarily (e.g., bit or block) depending on the context.

[27] Since P is the oracle given to a simulator while π is the black-box oracle given to an adversary in the security games of sec-prim, Σ may access to only a part of the interfaces of P: If sec-prim is whPRP and $\pi = E_K$, P is the pair (P, P^{-1}) (here, P is a random permutation) but Σ accesses only to P (and not to P^{-1}) because the black-box oracle interface given to an adversary $\mathcal{A}$ in the definition of whPRP is only E_K (and E_K^{-1} is not given to $\mathcal{A}$).

Interpretation of Theorem 2. The above theorem indeed shows that $(\Sigma^\pi, \mathcal{C}_{\Sigma^\pi})$ is a secure white-box scheme w.r.t. sec-const if the underlying scheme $(\pi, \mathcal{C}_\pi)$ is secure w.r.t. sec-prim and Σ^{P} is weak public indifferentiable from R: Let $\mathcal{A}$ be an adversary attacking $(\Sigma^\pi, \mathcal{C}_{\Sigma^\pi})$. Then, we can construct an adversary $\mathcal{A}'$ to attack $(\pi, \mathcal{C}_\pi)$ as in Theorem 2. If $(\pi, \mathcal{C}_\pi)$ is secure (w.r.t. sec-prim), then there is a simulator $\mathcal{S}_{\mathrm{prim}}$ for $(\pi, \mathcal{C}_\pi)$ that makes $\mathbf{Adv}^{\text{sec-prim}}_{\pi,\mathcal{C}_\pi,\mathcal{S}_{\mathrm{prim}}}(\mathcal{A}')$ small. In addition, if Σ^{P} is weak public indifferentiable from R, then there exists a simulator $\mathcal{S}_{\mathrm{indiff}}$ making $\mathbf{Adv}^{\text{weak-pub-indiff}}_{\Sigma,\mathcal{S}_{\mathrm{indiff}}}(\mathcal{A}'')$ small, where $\mathcal{A}''$ is the adversary built from $\mathcal{A}$ and $\mathcal{S}_{\mathrm{prim}}$ as in the theorem. Again, Theorem 2 assures that we can construct $\mathcal{S}_{\mathrm{const}}$ from $\mathcal{S}_{\mathrm{prim}}$ and $\mathcal{S}_{\mathrm{indiff}}$ such that $\mathbf{Adv}^{\text{sec-const}}_{\Sigma^\pi,\mathcal{C}_{\Sigma^\pi},\mathcal{S}_{\mathrm{const}}}(\mathcal{A})$ satisfies Eq. (1). If all the parameters appearing in Theorem 2 are not so large, the advantage $\mathbf{Adv}^{\text{sec-const}}_{\Sigma^\pi,\mathcal{C}_{\Sigma^\pi},\mathcal{S}_{\mathrm{const}}}(\mathcal{A})$ is sufficiently small.

Intuition of the Proof. Here we provide a rough sketch on why Σ^π becomes secure if π is secure and Σ^{P} satisfies the *original* indifferentiability. Let $\mathcal{S}_{\mathrm{indiff}}$ be a simulator making the indifferentiability advantage of Σ^{P} small. We consider the following three games.

1. [The real world (for Σ^π on sec-const).] The adversary $\mathcal{A} = (\mathcal{A}_{\mathrm{create}}, \mathcal{A}_{\mathrm{dist}})$ is given a black-box oracle access to Σ^π. A lifter L is given a white-box implementation of Σ^π.
2. [Intermediate world.] The black-box oracle of π and the lifter L in the real world are replaced with a random permutation P and a simulator $\mathcal{S}_{\mathrm{prim}}$ (for π on sec-prim), respectively. The adversary $\mathcal{A} = (\mathcal{A}_{\mathrm{create}}, \mathcal{A}_{\mathrm{dist}})$ and $\S_{\mathrm{prim}}$ are given oracle access to Σ^{P} and P, respectively. Especially, this game executes three algorithms $\mathcal{A}^{\Sigma^{\mathsf{P}}}_{\mathrm{create}}$, $\mathcal{S}^{\mathsf{P}}_{\mathrm{prim}}$, and $\mathcal{A}^{\Sigma^{\mathsf{P}}}_{\mathrm{dist}}$.
3. [The ideal world] The black-box oracle given to $\mathcal{A} = (\mathcal{A}_{\mathrm{create}}, \mathcal{A}_{\mathrm{dist}})$ is R. In addition, the simulator (for Σ^π on sec-const) is defined to be $\mathcal{S}^{\mathcal{S}_{\mathrm{indiff}}}_{\mathrm{prim}}$. $\mathcal{S}^{\mathcal{S}_{\mathrm{indiff}}}_{\mathrm{prim}}$ is also given an oracle access to R. Especially, this game executes three algorithms $\mathcal{A}^{\mathsf{R}}_{\mathrm{create}}$, $\mathcal{S}^{\mathcal{S}^{\mathsf{R}}_{\mathrm{indiff}}}_{\mathrm{prim}}$, and $\mathcal{A}^{\mathsf{R}}_{\mathrm{dist}}$.

If π is secure, then we can replace π (in Σ^π) and a lifter in the real world with P and a simulator $\mathcal{S}_{\mathrm{prim}}$, respectively, with a small security loss. That is, the difference between the first and the second worlds is small. Next, regarding the tuple $(\mathcal{A}_{\mathrm{create}}, \mathcal{S}_{\mathrm{prim}}, \mathcal{A}_{\mathrm{dist}})$ as a single algorithm, we can regard the intermediate world as a game where a single-stage adversary $(\mathcal{A}_{\mathrm{create}}, \mathcal{S}_{\mathrm{prim}}, \mathcal{A}_{\mathrm{dist}})$ runs relative to the oracles $(\Sigma^{\mathsf{P}}, \mathsf{P})$. Moreover, we can also regard the ideal wold as a game where the single algorithm $(\mathcal{A}_{\mathrm{create}}, \mathcal{S}_{\mathrm{prim}}, \mathcal{A}_{\mathrm{dist}})$ runs relative to the oracles $(\mathsf{R}, \mathcal{S}^{\mathsf{R}}_{\mathrm{indiff}})$. Especially, the difference between the intermediate and ideal worlds matches the indifferentiability advantage of the *single* algorithm $(\mathcal{A}_{\mathrm{create}}, \mathcal{S}_{\mathrm{prim}}, \mathcal{A}_{\mathrm{dist}})$ against T^{P} and R with respect to $\mathcal{S}_{\mathrm{indiff}}$[28]. Since Σ^{P} is indifferentiable from R by $\mathcal{S}_{\mathrm{indiff}}$, the difference between the intermediate world and the ideal world is also small.

[28] This is the reason that we can utilize the indifferentiability of Σ^{P} from R to show the security of Σ^π although the security games of Σ^π are not single-stage games.

In fact there are some subtleties on how to simulate list of queries passed to $\mathcal{S}_{\mathrm{prim}}$. Moreover, when we consider weak public indifferentiability instead of original indifferentiability, we also have to consider how to simulate the revealing interface $\mathsf{Rev}'[\mathsf{R}]$. See Section F in the appendix for details.

6.3 Feasibility Results

This section shows feasibility results that various white-box security notions can be reduced to that of block ciphers (whPRP) and FIL/FOL keyed functions (whPRF) like in the black-box setting. We only prove (weak) public indifferentiability of the constructions because Theorem 2 shows white-box security reductions follow from (weak) public indifferentiability.

whPRP-whPRF Switch. Let P be an n-bit random permutation. Then, regarding (P, P^{-1}) as a primitive oracle, P is public indifferentiable from a random function $\mathsf{RF} : \{0,1\}^n \to \{0,1\}^n$. (In the real world, the construction oracle is P and the primitive oracle is (P, P^{-1}).) Specifically, the proposition below holds.

Proposition 1. *There is a simulator $\mathcal{S}$ making at most q_p queries to RF satisfying $\mathbf{Adv}^{\text{pub-indiff}}_{P,\mathsf{RF},\mathcal{S}}(\mathcal{A}) \leq \frac{(q_c+q_p)^2}{2^n}$ for any adversary $\mathcal{A}$ making at most q_c and q_p queries to the construction and primitive oracles, respectively.*

The proof is quite straightforward. See Section G in the appendix for a complete proof. Together with Theorem 2, this proposition implies that a whPRP-secure BC is a whPRF-secure keyed function.

Reduction from whPRP to whPRF. The 6-round Feistel construction is public indifferentiable from a random invertible permutation when round functions are random functions [61]. Thus we can build a whPRP-secure BC from a whPRF-secure keyed function.

Reduction from VIL/VOL-whPRF to FIL/FOL-whPRF. The indifferentiability result of the sponge construction (Theorem 1) implies that we can build VIL/VOL-whPRF from FIL/FOL-whPRF. We can also build VIL/FOL-whPRF from FIL/FOL-whPRF by the Merkle-Damgård construction since it is public indifferentiable [40].

Reduction from whPRI to FIL/FOL-whPRF. By the result of Barbosa and Farshim [6], an indifferentiable AEAD can be constructed from a FIL/FOL random function by a scheme based on (unbalanced) 3-round Feistel that uses the sponge construction as round functions. (See Theorem 5 in the appendix and the explanation below for more details.) Thus we can build a whPRI-secure AEAD from a whPRF-secure FIL/FOL keyed function. In Section 7 we show a more practical construction.

Reduction from wh$\widetilde{\text{SPRP}}$ to whPRF. A weak public indifferentiable VIL tweakable ideally random permutation can be built from a FIL/FOL random function f, by a (balanced) 6-round Feistel construction of which round functions are the sponge construction using f as an underlying primitive. See Section H in the appendix for more details.

Reduction from whIND$-CPA to whPRF. Let us modify the encryption oracle of CTR in such a way that (1) a uniformly random IV is chosen for each encryption query (rather than IV is chosen by adversary) and IV is returned together with the ciphertext, and (2) the underlying keyed function of CTR is replaced with a random function ρ. We denote the resulting encryption oracle by $\mathcal{E}^{\rho}_{\mathrm{rnd}}$. Then $\mathcal{E}^{\rho}_{\mathrm{rnd}}$ is public indifferentiable from $\$(\cdot)$ that appears in the ideal experiment of whIND$-CPA. More precisely, the following proposition holds.

Proposition 2. *There exists a simulator $\mathcal{S}$ making at most q_c queries to the $\$(\cdot)$, where the lengths of queries are at most σ blocks in total, that satisfies $\mathbf{Adv}^{\text{pub-indiff}}_{\mathcal{E}_{\mathrm{rnd}},\$(\cdot),\mathcal{S}}(\mathcal{A}) \leq \frac{\sigma^2}{2^m} + \frac{\sigma(\sigma+q_p)}{2^m}$ for any adversary $\mathcal{A}$ making at most q_c queries to the construction oracle of which lengths are at most σ blocks in total and q_p queries to the primitive oracle.*

Intuition of the Proof. For simplicity, we assume $\mathrm{len}(M)$ is always a multiple of n and denote the i-th block of M by M_i, i.e., $M = M_1||\cdots||M_{\mathrm{len}(M)/n}$. Roughly speaking, the simulator $\mathcal{S}$ runs as follows: Let $\mathsf{List}[\$(\cdot)]$ be the list storing queries made so far to $\$(\cdot)$ and the responses. When a fresh value x is queried to the interface corresponding to ρ, the simulator first queries to the revealing interface to get $\mathsf{List}[\$(\cdot)]$. If there exists $(M,(IV,C)) \in \mathsf{List}[\$(\cdot)]$ such that $x = IV + i - 1$ for $1 \leq i \leq \mathrm{len}(M)/n$, the adversary may be trying to compute the i-th block of C itself. Thus the simulator sets the value $\rho(x)$ as $\rho(x) := M_i \oplus C_i$ so that the adversary cannot notice that ρ is simulated. If such $(M,(IV,C))$ does not exist in $\mathsf{List}[\$(\cdot)]$, the value $\rho(x)$ is just randomly sampled.

The simulator may not be able to sample the value $\rho(x)$ in compatible with C and fail if the following (a) or (b) happen: (a) when a message M is queried to $\$(\cdot)$ and IV is randomly chosen, $IV+i = IV'+j$ holds for some $(M',(IV',C')) \in \mathsf{List}[\$'(\cdot)]$, where $0 \leq i < \mathrm{len}(M)/n$ and $0 \leq j < \mathrm{len}(M')/n$. (b) when a message M is queried to $\$(\cdot)$ and IV is randomly chosen, the value $IV + i$ $(0 \leq i < \mathrm{len}(M)/n)$ collides with a value x on which the output value of ρ is already defined. The events (a) and (b) correspond to the terms $\frac{\sigma^2}{2^m}$ and $\frac{\sigma(\sigma+q_p)}{2^m}$ in the security bound, respectively. If both of (a) and (b) do not happen in the ideal world, then outputs of ρ are appropriately simulated in compatible with ciphertexts, and thus an adversary cannot distinguish the ideal world from the real world. See Section J in the appendix for a complete proof.

On Reduction of Pairs and Generic Compositions We also observe feasibility of reductions of pairs (i.e., providing proof in a modular way), and infeasibility of generic compositions for AEADs. See Section I in the appendix for details.

7 A Search for a Practical whPRI-Secure AEAD Mode

This section shows a practical AEAD mode to convert whPRP into whPRI. Section 7.1 shows that SIV with CTR is public indifferentiable from a fixed-key random injection when the tag-generation (or, MAC) part is a single VIL/FOL random function f and the underlying keyed function of CTR is a FIL/FOL random function ρ. Then, in Section 7.2, we replace ρ and f with keyed functions built from a single whPRP, and observe that the resulting scheme is a whPRI-secure AEAD.

7.1 Public Indifferentiability of SIV+CTR

Let $\mathsf{CTR}^{\rho}(IV, M)$ denote the encryption function of the counter mode with the underlying keyed function being replaced with a random function $\rho : \{0,1\}^{\tau} \to \{0,1\}^{n}$ ($\tau \leq n$). In addition, let $\mathit{\Pi} = (\mathcal{E}^{f,\rho}, \mathcal{D}^{f,\rho})$ be the SIV construction of which keyed function for tag-generation is replaced with a random function $f : \mathbf{N} \times \mathbf{A} \times \{0,1\}^{*} \to \{0,1\}^{\tau}$ and conventional encryption scheme is replaced with CTR^{ρ}. Let $\mathsf{enc} : \mathbf{N} \times \mathbf{A} \times \{0,1\}^{*} \to \{0,1\}^{*}$ be an arbitrary encoding function that encodes each tuple (N, A, X) into a single bit string in a uniquely decodable manner. We let $\mathrm{len}(N, A, X) := \mathrm{len}(\mathsf{enc}(N, A, X))$ and call $\lceil \mathrm{len}(X)/n \rceil$ the block length of a bit string of X. The following theorem shows $\mathit{\Pi}$ is public indifferentiable from a random injection.

Theorem 3. *Let $F : \mathbf{N} \times \mathbf{A} \times \{0,1\}^{*} \to \{0,1\}^{*}$ be a fixed-key random injection with message space $\{0,1\}^{*}$ and such that $\mathrm{len}(F(N, A, M)) = \mathrm{len}(M) + \tau$. There exists a simulator $\mathcal{S}$ for public indifferentiability of $\mathit{\Pi}$ from $F^{\pm}$, where a primitive oracle is (f, ρ), such that the number of queries by $\mathcal{S}$ to the construction oracle is at most q_f and the block lengths of the queries are at most σ_f in total, and*

$$\mathbf{Adv}^{\text{pub-indiff}}_{\mathit{\Pi}, F^{\pm}, \mathcal{S}}(\mathcal{A}) \leq \frac{(\sigma_c + \sigma_f)^2}{2^{\tau}} + \frac{(\sigma_c + \sigma_f)(q_{\rho} + \sigma_c)}{2^{\tau}} + \frac{3q_c}{2^{\tau}} + \frac{(q_c + q_f)^2}{2^{\tau}} \quad (2)$$

holds for any adversary $\mathcal{A}$ of which computational resources are as follows: To the construction oracle, $\mathcal{A}$ makes at most q_c queries of which block lengths are at most σ_c in total. To the first primitive oracle (corresponding to f), $\mathcal{A}$ makes at most q_f queries of which block lengths are at most σ_f in total. To the second primitive oracle (corresponding to ρ), $\mathcal{A}$ makes at most q_{ρ} queries. Here, we assume $(q_c + q_f) \leq 2^{\tau - 1}$.

Intuition of the Proof. For simplicity, we assume $\mathrm{len}(M)$ is always a multiple of n. For each (N, A, M), we assume $F(N, A, M)$ is divided as $F(N, A, M) = IV || C_1 || \cdots || C_{\ell}$, where $IV \in \{0,1\}^{\tau}$ and $C_1, \ldots, C_{\ell} \in \{0,1\}^{n}$. The simulation of ρ is almost the same as that for the proof of random-IV CTR (See the explanation below Proposition 2. Here, $\$(\cdot)$ in Proposition 2 is replaced with F.). Simulation of $f(N, A, M)$ is done just by querying (N, A, M) to F and return $F_0(N, A, M)$. Intuitively, the simulation does not work well if the simulation of ρ fails (the events (a) and (b) in the explanation below Proposition 2), or (c) an

adversary computes $\mathcal{D}^{f,\rho}(N,A,C)$ itself for a tuple (N,A,C) such that C has never been returned from F, and $\mathcal{D}^{f,\rho}(N,A,C)$ happens to be a value that is *not* $\perp$. The events (a) and (b) correspond to the terms $\frac{(\sigma_c+\sigma_f)^2}{2^\tau}$ and $\frac{(\sigma_c+\sigma_f)(q_\rho+\sigma_c)}{2^\tau}$ of Eq. (14), respectively. Due to (c), an additional term $\frac{q_c}{2^\tau}$ is added. Moreover, we need another term $\frac{(q_c+q_f)^2+2q_c}{2^\tau}$ to deal with lazy sampling of a random injection. See Section K in the appendix for a complete proof.

7.2 Instantiation with Block Ciphers

This section discusses how to combine the scheme in the previous subsection with a whPRP-secure block cipher to build a whPRI-secure AEAD.

Assume $\tau < n$ and let $P : \{0,1\}^n \to \{0,1\}^n$ a random permutation. Define $P_0 : \{0,1\}^{n-1} \to \{0,1\}^{n-1}$ and $P_1 : \{0,1\}^\tau \to \{0,1\}^n$ by $P_0(x) :=$ (The lower $(n-1)$ bits of $P(0||x)$) and $P_1(x) := P(1^{n-\tau}||x)$. Set $\mathbf{N} = \{0,1\}^{n/2}$. Let enc be an encoding function such that $\mathsf{enc}(N,A,M) = N||A||M||\mathrm{len}(M)$. (We assume $\mathrm{len}(M)$ is represented as an $n/4$-bit string.) In addition, define $\mathrm{MAC}^P(N,A,M) := \mathsf{Sponge}^{P_0}(\mathsf{enc}(N,A,M))$. Replace f (tag generation function) and ρ (underlying function of CTR) of Π in Theorem 3 with MAC^P and P_1, and denote the resulting scheme by $\Pi[P]$.

Then, $\Pi[P]$ is weak public indifferentiable from a fixed-key random injection $F^\pm$ when $P^\pm$ is regarded as a primitive oracle[29]. Furthermore, if we replace P with a whPRP-secure block cipher E_K, the resulting scheme $\Pi[E_K]$ becomes a whPRI-secure AEAD by Theorem 2. (Here, we assume $\Pi[E_K]$ is implemented in such a way that the implementation of the mode is explicitly separated from the implementation of E_K and the former calls the latter in a black-box manner, so that Theorem 2 can be applied.) More precisely, the following corollary holds. (See Section L in the appendix for more details on how to derive the corollary.)

Corollary 1. *Let $\mathcal{A}$ be an adversary against $(\Pi[E_K], \mathcal{C}_{\Pi[E_K]})$ on whPRI-security. The running time of $\mathcal{A}$ is at most t. The number of queries by $\mathcal{A}$ to a black-box oracle is at most q and the block lengths of the queries are at most σ in total. $\mathcal{A}$ creates a lifter running in time t_{lif} and outputs at most λ-bit leakage. In addition, let $\mathcal{S}_{\mathrm{prim}}$ be a simulator for $(E,\mathcal{C}_E)$ on whPRP-security. $\mathcal{S}_{\mathrm{prim}}$ makes at most q_{sim} queries to $P^\pm$. Then there exists a simulator $\mathcal{S}_{\mathrm{const}}$ for $(\Pi[E_K], \mathcal{C}_{\Pi[E_K]})$ on whPRI-security and an adversary $\mathcal{A}'$ against $(E,\mathcal{C}_E)$ on whPRP-security such*

[29] This is because (1) an invertible permutation is public indifferentiable from a random function (Proposition 1), (2) the sponge construction is indifferentiable from a random oracle (Theorem 1), (3) the scheme Π in Theorem 3 is public indifferentiable from a fixed-key random injection, (4) composition of deterministic weak public indifferentiable schemes are again weak public indifferentiable (Lemma 1, or its formal version Lemma 2 in the appendix).

that $\mathbf{Adv}^{\mathrm{whPRI}}_{\Pi[E_K],\mathcal{C}_{\Pi[E_K]},\mathcal{S}_{\mathrm{const}}}(\mathcal{A})$ *is upper bounded by*

$$\mathbf{Adv}^{\mathrm{whPRP}}_{E_K,\mathcal{C}_{E_K},\mathcal{S}_{\mathrm{prim}}}(\mathcal{A}') + \frac{\lceil \frac{n}{r} \rceil^6 (10\lceil \frac{n}{r} \rceil \sigma + q_{\mathrm{sim}})^4}{2^\tau} + \frac{\lceil \frac{n}{r} \rceil^4 (10\lceil \frac{n}{r} \rceil \sigma + q_{\mathrm{sim}})^3}{2^\tau}$$
$$+ \frac{3q_c}{2^\tau} + \frac{\lceil \frac{n}{r} \rceil^2 (10\lceil \frac{n}{r} \rceil \sigma + q_{\mathrm{sim}})^2}{2^\tau} + \epsilon\left(2\left\lceil \frac{n}{r} \right\rceil \left(8\left\lceil \frac{n}{r} \right\rceil \sigma + q_{\mathrm{sim}}\right)\right) + \frac{\lceil \frac{n}{r} \rceil^2 (9\lceil \frac{n}{r} \rceil \sigma + 2q_{\mathrm{sim}})^2}{2^n},$$

where $\epsilon(j) = 1 - \Pi_{i=1}^{j}(1 - \frac{1}{2^c})$. $\mathcal{S}_{\mathrm{const}}$ *makes at most* $\left\lceil \frac{n}{r} \right\rceil (9\left\lceil \frac{n}{r} \right\rceil \sigma + q_{\mathrm{sim}})$ *queries to* $F^{\pm}$ *and the lengths of the queries are at most* $(\left\lceil \frac{n}{r} \right\rceil^3 (9\left\lceil \frac{n}{r} \right\rceil \sigma + q_{\mathrm{sim}})^2)$ *blocks in total.* $\mathcal{A}'$ *runs in time* $O(t + \sigma)$ *and makes at most* $(2\sigma + \left\lceil \frac{n}{r} \right\rceil q)$ *black-box oracle queries.* $\mathcal{A}'$ *outputs a lifter that runs in time* $O(t_{\mathrm{lif}})$ *and outputs at most* λ *bits of leakage.*

Interpretation of Corollary 1. Let us set $\tau = n - 1$ and $(r, c) = (n/2, n/2 - 1)$. Then, Corollary 1 says that $\mathbf{Adv}^{\mathrm{whPRI}}_{\Pi[E_K],\mathcal{C}_{\Pi[E_K]},\mathcal{S}_{\mathrm{const}}}(\mathcal{A})$ becomes small as long as $\mathbf{Adv}^{\mathrm{whPRP}}_{E,\mathcal{C}_E,\mathcal{S}_{\mathrm{prim}}}(\mathcal{A})$ is small and $q_{\mathrm{sim}}, q, \sigma \ll 2^{n/4}$. This means the following: Let λ and t_{lif} be some reasonable parameters ($\ll 2^{n/4}$) and assume the underlying block cipher E_K is a secure whPRP. More concretely, let $\mathcal{A}'$ be an adversary attacking E_K with $t \ll 2^\kappa$ and $q \ll 2^{n/4}$, and $\mathcal{L}$ be a lifter running in time $t_{\mathrm{lif}}(< 2^{n/4})$ that leaks at most λ-bit leakage. Suppose, for any such $\mathcal{A}'$ and $\mathcal{L}$, there exists $\mathcal{S}_{\mathrm{prim}}$ with making $q_{\mathrm{sim}}(\ll 2^{n/4})$ queries such that $\mathbf{Adv}^{\mathrm{whPRP}}_{E,\mathcal{C}_E,\mathcal{S}_{\mathrm{prim}}}(\mathcal{A}')$ is sufficiently small. Then $\Pi[E_K]$ is whPRI-secure against an adversary $\mathcal{A}$ as long as (1) the running time of $\mathcal{A}$ is $\ll 2^\kappa$, (2) the length of messages processed by $\Pi[E_K]$ is $\ll 2^{n/4}$ blocks in total while running $\mathcal{A}$ in the real world, and (3) the running time and leakage of a lifter (output by $\mathcal{A}$) are at most t_{lif} and λ bits [30].

On Underlying Block Cipher. The above discussions show that $\Pi[E_K]$ is whPRI-secure if E_K is whPRP-secure and the amount of data processed by $\Pi[E_K]$ is $\ll 2^{n/4}$. As a candidate of whPRP-secure BC, we conjecture that SPACE is whPRP-secure for some parameter settings (see Section 5.1). However, the block length of SPACE is basically $n = 128$ only, when $2^{n/4} = 2^{32}$. In practical use cases, the limitation of 2^{32} is inconvenient and unsatisfactory.

Thus, we propose a 256-bit block variant of SPACE-16, which we name SPACE256-16. Its details are provided in Section N in the appendix, where we discuss its security against various attacks following the convention of block cipher designs. We conjecture[31] that SPACE256-16 is a $(\lambda, t, q, t_{\mathrm{lif}}, q_{\mathrm{sim}}, \epsilon)$-secure whPRP with $\lambda \approx 2^{20}$, $t \approx 2^{128}$, $q \approx 2^{64}$, $q_{\mathrm{sim}} \approx 2^{64}$, and $\epsilon \ll 1$, as long as $t_{\mathrm{lif}} \ll q_{\mathrm{sim}}$. Assuming our conjecture is true, $\Pi[E_K]$ with E_K instantiated with

[30] Note that λ does not explicitly appear in the upper bound of $\mathbf{Adv}^{\mathrm{whPRI}}_{\Pi[E_K],\mathcal{C}_{\Pi[E_K]},\mathcal{S}_{\mathrm{const}}}(\mathcal{A})$ in the corollary. This is because we (implicitly) assume $\lambda \leq q_{\mathrm{sim}}$ and the effect of λ is absorbed into q_{sim} in the security bound.

[31] This conjecture is obtained by changing the settings of n and n_a in our conjecture in Section 5.1 to $n = 256$ and $n_a = 16$. κ is still 128.

SPACE256-16 is secure until the amount of processed data is $\ll 2^{64}$ (and the amount of leakage is $< 2^{20}$).

To evaluate the performance, we implemented $\Pi[E_K]$ using SPACE256-16 on a single core in a laptop PC with Intel Core i7-1065G7, being Turbo Boost and hyperthreading disabled. The implementation size is in the order of KB or MB. As a result, the performance reaches about 530 CPB when a 1KB message is processed. Considering the performance of raw SPACE-16 is 305.11 cpb [25], we believe our mode of operation achieves relevant performance.

The limit of leakage for SPACE256-16 is not large. Still, in the same way as (the original, 128-bit-block) SPACE-32 and SPACE-24 provide better security than SPACE-16 does, a better limit could be achieved by 256-bit-block versions of SPACE-32 or SPACE-24 (at the cost of performance). We introduced a 256-bit-block version of SPACE-16 rather than SPACE-32 or SPACE-24 to balance security and performance. Improving the performance and the limit of the leakage is an interesting future work. This could be achieved by improving SPACE-hard block ciphers, modes of operations, or both.

References

1. Agrawal, S., Bock, E.A., Chen, Y., Watson, G.J.: White-box cryptography with device binding from token-based obfuscation and more. IACR Cryptology ePrint Archive 2021/767 (2021)
2. Alliance, S.C.: A smart card alliance mobile & nfc council white paper, host card emulation (hce) 101 (2014)
3. Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-key encryption in the bounded-retrieval model. In: Gilbert, H. (ed.) EUROCRYPT 2010, Proceedings. LNCS, vol. 6110, pp. 113–134. Springer (2010)
4. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) CRYPTO 2009, Proceedings. LNCS, vol. 5677, pp. 36–54. Springer (2009)
5. Banik, S., Bogdanov, A., Isobe, T., Jepsen, M.B.: Analysis of software countermeasures for whitebox encryption. IACR Trans. Symmetric Cryptol. **2017**(1), 307–328 (2017)
6. Barbosa, M., Farshim, P.: Indifferentiable authenticated encryption. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Proceedings, Part I. LNCS, vol. 10991, pp. 187–220. Springer (2018)
7. Barwell, G., Martin, D.P., Oswald, E., Stam, M.: Authenticated encryption in the face of protocol and side channel leakage. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Proceedings, Part I. LNCS, vol. 10624, pp. 693–723. Springer (2017)
8. Bellare, M., Dai, W.: Defending against key exfiltration: Efficiency improvements for big-key cryptography via large-alphabet subkey prediction. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017, Proceedings. pp. 923–940. ACM (2017)
9. Bellare, M., Kane, D., Rogaway, P.: Big-key symmetric encryption: Resisting key exfiltration. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Proceedings, Part I. LNCS, vol. 9814, pp. 373–402. Springer (2016)

10. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006, Proceedings. LNCS, vol. 4004, pp. 409–426. Springer (2006)
11. Berti, F., Guo, C., Pereira, O., Peters, T., Standaert, F.: Tedt, a leakage-resist AEAD mode for high physical security applications. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1), 256–320 (2020)
12. Berti, F., Pereira, O., Peters, T., Standaert, F.: On leakage-resilient authenticated encryption with decryption leakages. IACR Trans. Symmetric Cryptol. **2017**(3), 271–293 (2017)
13. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008, Proceedings. LNCS, vol. 4965, pp. 181–197. Springer (2008)
14. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) EUROCRYPT '99, Proceedings. LNCS, vol. 1592, pp. 12–23. Springer (1999)
15. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO '90, Proceedings. LNCS, vol. 537, pp. 2–21. Springer (1990)
16. Billet, O., Gilbert, H., Ech-Chatbi, C.: Cryptanalysis of a White Box AES Implementation. In: SAC 2004, Revised Selected Papers. pp. 227–240 (2004)
17. Biryukov, A., Bouillaguet, C., Khovratovich, D.: Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key (extended abstract). In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Proceedings, Part I. LNCS, vol. 8873, pp. 63–84. Springer (2014)
18. Biryukov, A., Perrin, L.: Symmetrically and asymmetrically hard cryptography. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Proceedings, Part III. LNCS, vol. 10626, pp. 417–445. Springer (2017)
19. Biryukov, A., Udovenko, A.: Attacks and countermeasures for white-box designs. In: Peyrin, T., Galbraith, S.D. (eds.) ASIACRYPT 2018, Proceedings, Part II. LNCS, vol. 11273, pp. 373–402. Springer (2018)
20. Biryukov, A., Udovenko, A.: Dummy shuffling against algebraic attacks in white-box implementations. In: Canteaut, A., Standaert, F. (eds.) EUROCRYPT 2021, Proceedings, Part II. LNCS, vol. 12697, pp. 219–248. Springer (2021)
21. Bock, E.A., Amadori, A., Bos, J.W., Brzuska, C., Michiels, W.: Doubly half-injective prgs for incompressible white-box cryptography. In: Matsui, M. (ed.) CT-RSA 2019, Proceedings. LNCS, vol. 11405, pp. 189–209. Springer (2019)
22. Bock, E.A., Amadori, A., Brzuska, C., Michiels, W.: On the security goals of white-box cryptography. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(2), 327–357 (2020)
23. Bock, E.A., Brzuska, C., Fischlin, M., Janson, C., Michiels, W.: Security reductions for white-box key-storage in mobile payments. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Proceedings, Part I. LNCS, vol. 12491, pp. 221–252. Springer (2020)
24. Bogdanov, A., Isobe, T.: White-box cryptography revisited: Space-hard ciphers. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015, Proceedings. pp. 1058–1069. ACM (2015)
25. Bogdanov, A., Isobe, T., Tischhauser, E.: Towards practical whitebox cryptography: Optimizing efficiency and space hardness. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Proceedings, Part I. LNCS, vol. 10031, pp. 126–158 (2016)

26. Bos, J.W., Hubain, C., Michiels, W., Teuwen, P.: Differential computation analysis: Hiding your white-box designs is not enough. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016, Proceedings. LNCS, vol. 9813, pp. 215–236. Springer (2016)
27. Cash, D., Ding, Y.Z., Dodis, Y., Lee, W., Lipton, R.J., Walfish, S.: Intrusion-resilient key exchange in the bounded retrieval model. In: Vadhan, S.P. (ed.) TCC 2007, Proceedings. LNCS, vol. 4392, pp. 479–498. Springer (2007)
28. Cho, J., Choi, K.Y., Dinur, I., Dunkelman, O., Keller, N., Moon, D., Veidberg, A.: WEM: A new family of white-box block ciphers based on the even-mansour construction. In: Handschuh, H. (ed.) CT-RSA 2017, Proceedings. LNCS, vol. 10159, pp. 293–308. Springer (2017)
29. Chow, S., Eisen, P.A., Johnson, H., van Oorschot, P.C.: White-box cryptography and an AES implementation. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002, Revised Papers. LNCS, vol. 2595, pp. 250–270. Springer (2002)
30. Chow, S., Eisen, P.A., Johnson, H., van Oorschot, P.C.: A white-box DES implementation for DRM applications. In: Feigenbaum, J. (ed.) Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Revised Papers. LNCS, vol. 2696, pp. 1–15. Springer (2002)
31. Coron, J., Patarin, J., Seurin, Y.: The random oracle model and the ideal cipher model are equivalent. In: Wagner, D.A. (ed.) CRYPTO 2008, Proceedings. LNCS, vol. 5157, pp. 1–20. Springer (2008)
32. Crescenzo, G.D., Lipton, R.J., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Halevi, S., Rabin, T. (eds.) TCC 2006, Proceedings. LNCS, vol. 3876, pp. 225–244. Springer (2006)
33. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher square. In: Biham, E. (ed.) FSE '97, Proceedings. LNCS, vol. 1267, pp. 149–165. Springer (1997)
34. Daemen, J., Rijmen, V.: Probability distributions of correlation and differentials in block ciphers. J. Math. Cryptol. **1**(3), 221–242 (2007)
35. Degabriele, J.P., Janson, C., Struck, P.: Sponges resist leakage: The case of authenticated encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Proceedings, Part II. LNCS, vol. 11922, pp. 209–240. Springer (2019)
36. Delerablée, C., Lepoint, T., Paillier, P., Rivain, M.: White-box security notions for symmetric encryption schemes. In: Lange, T., Lauter, K.E., Lisonek, P. (eds.) SAC 2013, Revised Selected Papers. LNCS, vol. 8282, pp. 247–264. Springer (2013)
37. Dinur, I., Dunkelman, O., Kranz, T., Leander, G.: Decomposing the ASASA block cipher construction. IACR Cryptology ePrint Archive 2015/507
38. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Unterluggauer, T.: ISAP - towards side-channel secure authenticated encryption. IACR Trans. Symmetric Cryptol. **2017**(1), 80–105 (2017)
39. Dobraunig, C., Mennink, B.: Leakage resilience of the duplex construction. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Proceedings, Part III. LNCS, vol. 11923, pp. 225–255. Springer (2019)
40. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging merkle-damgård for practical applications. In: Joux, A. (ed.) EUROCRYPT 2009, Proceedings. LNCS, vol. 5479, pp. 371–388. Springer (2009)
41. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) TCC 2006, Proceedings. LNCS, vol. 3876, pp. 207–224. Springer (2006)
42. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS 2008, Proceedings. pp. 293–302. IEEE Computer Society (2008)

43. Fouque, P., Karpman, P., Kirchner, P., Minaud, B.: Efficient and provable white-box primitives. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Proceedings, Part I. LNCS, vol. 10031, pp. 159–188 (2016)
44. Gilbert, H., Plût, J., Treger, J.: Key-recovery attack on the ASASA cryptosystem with expanding s-boxes. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Proceedings, Part I. LNCS, vol. 9215, pp. 475–490. Springer (2015)
45. Gueron, S., Lindell, Y.: GCM-SIV: full nonce misuse-resistant authenticated encryption at under one cycle per byte. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015, Proceedings. pp. 109–119. ACM (2015)
46. Guo, C., Pereira, O., Peters, T., Standaert, F.: Towards low-energy leakage-resistant authenticated encryption from the duplex sponge construction. IACR Trans. Symmetric Cryptol. **2020**(1), 6–42 (2020)
47. Halevi, S., Rogaway, P.: A tweakable enciphering mode. In: Boneh, D. (ed.) CRYPTO 2003, Proceedings. LNCS, vol. 2729, pp. 482–499. Springer (2003)
48. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Proceedings, Part I. LNCS, vol. 9056, pp. 15–44. Springer (2015)
49. Inc., G.O.: Gurobi optimizer, official webpage, `http://www.gurobi.com/`
50. intertrust: Intertrust white paper, taking steps to protect financial mobile applications (2018)
51. Ishai, Y., Sahai, A., Wagner, D.A.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003, Proceedings. LNCS, vol. 2729, pp. 463–481. Springer (2003)
52. Kalai, Y.T., Reyzin, L.: A survey of leakage-resilient cryptography. In: Goldreich, O. (ed.) Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali, pp. 727–794. ACM (2019)
53. Kalai, Y.T., Reyzin, L.: A survey of leakage-resilient cryptography. IACR Cryptol. ePrint Arch. p. 302 (2019)
54. Knudsen, L.R., Wagner, D.A.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002, Revised Papers. LNCS, vol. 2365, pp. 112–127. Springer (2002)
55. Koike, Y., Sakamoto, K., Hayashi, T., Isobe, T.: Galaxy: A family of stream-cipher-based space-hard ciphers. In: Liu, J.K., Cui, H. (eds.) ACISP 2020, Proceedings. LNCS, vol. 12248, pp. 142–159. Springer (2020)
56. Krämer, J., Struck, P.: Leakage-resilient authenticated encryption from leakage-resilient pseudorandom functions. In: Bertoni, G.M., Regazzoni, F. (eds.) COSADE 2020, Revised Selected Papers. LNCS, vol. 12244, pp. 315–337. Springer (2020)
57. Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: Joux, A. (ed.) FSE 2011, Revised Selected Papers. LNCS, vol. 6733, pp. 306–327. Springer (2011)
58. Kwon, J., Lee, B., Lee, J., Moon, D.: FPL: white-box secure block cipher using parallel table look-ups. In: Jarecki, S. (ed.) CT-RSA 2020, Proceedings. LNCS, vol. 12006, pp. 106–128. Springer (2020)
59. Lepoint, T., Rivain, M., Mulder, Y.D., Roelse, P., Preneel, B.: Two Attacks on a White-Box AES Implementation. In: SAC 2013, Revised Selected Papers. pp. 265–285 (2013)
60. Longo, J., Martin, D.P., Oswald, E., Page, D., Stam, M., Tunstall, M.: Simulatable leakage: Analysis, pitfalls, and new constructions. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Proceedings, Part I. LNCS, vol. 8873, pp. 223–242. Springer (2014)

61. Mandal, A., Patarin, J., Seurin, Y.: On the public indifferentiability and correlation intractability of the 6-round feistel construction. In: Cramer, R. (ed.) TCC 2012, Proceedings. LNCS, vol. 7194, pp. 285–302. Springer (2012)
62. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT '93, Proceedings. LNCS, vol. 765, pp. 386–397. Springer (1993)
63. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004, Proceedings. LNCS, vol. 2951, pp. 21–39. Springer (2004)
64. McGrew, D.A., Viega, J.: The security and performance of the galois/counter mode (GCM) of operation. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004, Proceedings. LNCS, vol. 3348, pp. 343–355. Springer (2004)
65. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004, Proceedings. LNCS, vol. 2951, pp. 278–296. Springer (2004)
66. Minaud, B., Derbez, P., Fouque, P., Karpman, P.: Key-recovery attacks on ASASA. J. Cryptol. **31**(3), 845–884 (2018)
67. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C., Yung, M., Lin, D. (eds.) Inscrypt 2011. LNCS, vol. 7537, pp. 57–76. Springer (2011)
68. Mulder, Y.D., Roelse, P., Preneel, B.: Cryptanalysis of the Xiao - Lai White-Box AES Implementation. In: SAC 2012, Revised Selected Papers. pp. 34–49 (2012)
69. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering generic composition. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014, Proceedings. LNCS, vol. 8441, pp. 257–274. Springer (2014)
70. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009, Proceedings. LNCS, vol. 5479, pp. 462–482. Springer (2009)
71. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. In: Paterson, K.G. (ed.) EUROCRYPT 2011, Proceedings. LNCS, vol. 6632, pp. 487–506. Springer (2011)
72. Rogaway, P.: Authenticated-encryption with associated-data. In: Atluri, V. (ed.) ACM CCS 2002, Proceedings. pp. 98–107. ACM (2002)
73. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: a block-cipher mode of operation for efficient authenticated encryption. In: Reiter, M.K., Samarati, P. (eds.) ACM CCS 2001, Proceedings. pp. 196–205. ACM (2001)
74. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006, Proceedings. LNCS, vol. 4004, pp. 373–390. Springer (2006)
75. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In: Coron, J., Nielsen, J.B. (eds.) EUROCRYPT 2017, Proceedings, Part III. LNCS, vol. 10212, pp. 185–215 (2017)
76. Standaert, F., Pereira, O., Yu, Y.: Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Proceedings, Part I. LNCS, vol. 8042, pp. 335–352. Springer (2013)
77. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Proceedings, Part I. LNCS, vol. 9056, pp. 287–314. Springer (2015)
78. Todo, Y., Isobe, T.: Hybrid code lifting on space-hard block ciphers: Application to Yoroi and SPNbox. IACR Trans. Symmetric Cryptol. **2022**(3), 368–402 (2022)
79. Whiting, D., Housley, R., Ferguson, N.: AES Encryption & Authentication Using CTR Mode & CBC-MAC. IEEE P802.11 Wireless LNAs (2002)

80. Wyseur, B., Michiels, W., Gorissen, P., Preneel, B.: Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings. In: SAC 2007, Revised Selected Papers. pp. 264–277 (2007)

81. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Proceedings, Part I. LNCS, vol. 10031, pp. 648–678 (2016)

82. Yoneyama, K., Miyagawa, S., Ohta, K.: Leaky random oracle. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **92-A**(8), 1795–1807 (2009)

A Additional Figures

Experiment $\mathbf{Exp}^{\boxed{\text{whIND\$-CPA}}\text{-real}}_{\Pi,\mathcal{C}_\Pi,\mathcal{A}}$	**9:**
$K \xleftarrow{\$} \mathbf{K}$, $\mathcal{P} \leftarrow \mathcal{C}_\Pi(K)$	
$(\mathcal{L}, S) \leftarrow {\mathcal{A}_{\text{create}}}^{\mathcal{E}_K}()$	
$L \leftarrow \mathcal{L}(\mathcal{P})$	
$\beta \leftarrow {\mathcal{A}_{\text{dist}}}^{\mathcal{E}_K}(S, L)$	
return β	

Experiment $\mathbf{Exp}^{\boxed{\text{whIND\$-CPA}}\text{-ideal}}_{\mathcal{S},\mathcal{A}}$	**10:**
$(\mathcal{L}, S) \leftarrow {\mathcal{A}_{\text{create}}}^{\$(\cdot)}()$	
$L \leftarrow \mathcal{S}^{\$(\cdot)}(\mathcal{L}, \mathsf{List}_{\text{create}})$	
$\beta \leftarrow {\mathcal{A}_{\text{dist}}}^{\$(\cdot)}(S, L)$	
return β	

Fig. 3: Experiments for whIND$-CPA. $\mathsf{List}_{\text{create}}$ denotes the list of queries to $\$(\cdot)$ by $\mathcal{A}_{\text{create}}$ and the responses.

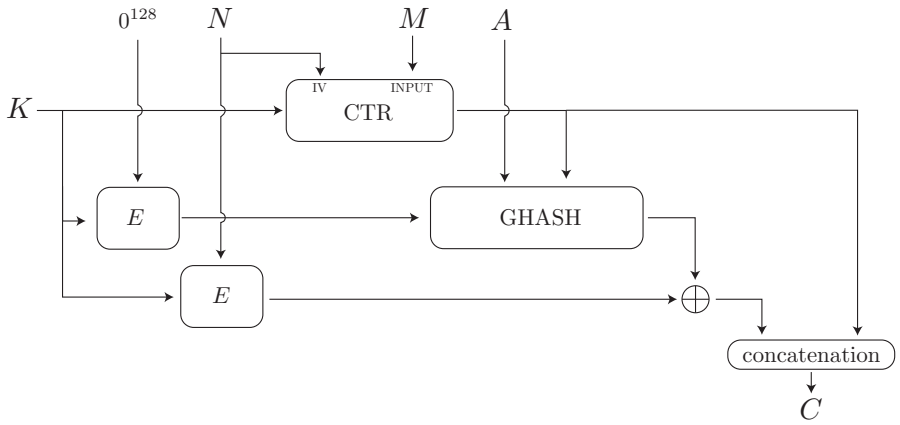

Fig. 4: A simplified sketch of GCM mode of operation when $N \in \{0,1\}^{96}$. (N is padded before fed to E and CTR but we omit to write details because it does not affect our attack.)

B Code Lifting on Existing Modes Other than GCM Using White-Box Block Ciphers

Section 3 showed that an efficient lifter, given an unlimited access to a white-box implementation of GCM, can leak the information on exponentially many valid message-ciphertext pairs by a small amount of leakage. The information is so much that an adversary can perform universal forgery. The attack works regardless of whether the underlying block cipher resists code lifting or not, and it intuitively shows that GCM is vulnerable to code lifting attacks. This section shows similar attack exist on white-box implementations of GCM-SIV, CCM, and OCB (under some reasonable assumptions), and that these modes are essentially not resistant to code lifting attacks.

Our goal of this section is to provide intuitions that the modes are unlikely to resist code lifting, rather than to provide formal discussions. For readability, we sometimes simplify (a part of) the descriptions of the modes that does not essentially affect our attacks.

B.1 Code Lifting on GCM-SIV

GCM is proven secure only if nonces are never repeated for encryption (i.e., secure in the nonce-respecting scenario). Once a nonce is repeatedly used, the scheme is easily broken even in the black-box setting. However, it is not always easy to completely ensure that users do not repeat nonces.

GCM-SIV [45] is a variant of GCM resolving this issue. It is proven secure even if nonces are repeated for encryption (i.e., secure in the nonce-misuse scenario). Figure 5 shows the high-level (and a simplified) overview of GCM-SIV using two keys.

We expected that GCM-SIV would be immune from code lifting as it resists nonce-repeating attacks. However, we observe that GCM-SIV is broken if an efficient and reasonable lifter is given an unlimited access to a white-box implementation.

We reasonably assume K_1 and a white-box implementation of E_{K_2} can efficiently be recovered once a white-box implementation of GCM-SIV is given: Readers may wonder if they could be protected by software or hardware countermeasures. However, the effectiveness of such countermeasures would be limited, given that existing white-box implementations of AES ensure security only when adversaries have limited access to implemented algorithms. (Note that K_1 is used only in GHASH, which is essentially a polynomial in K_1 and some other variables.)

Universal Forgery. Suppose that an attacker knows

1. K_1,
2. a valid message-ciphertext pair $((N, A, M), C)$.

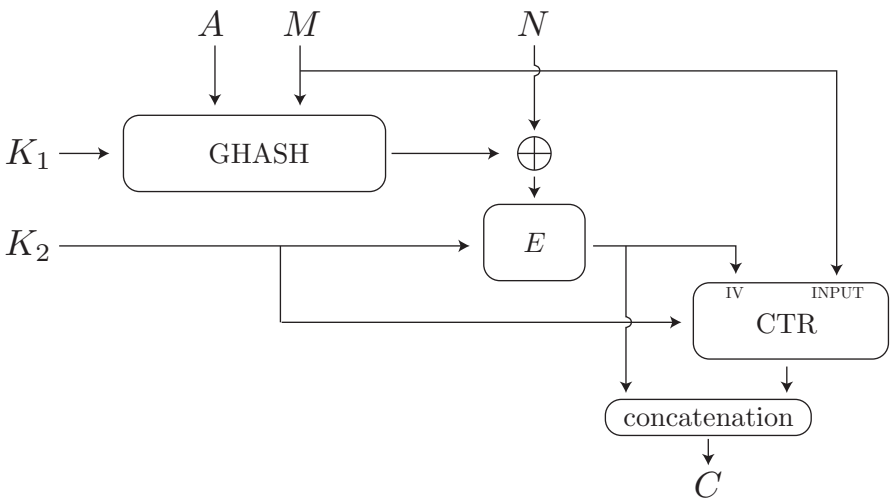

Fig. 5: GCM-SIV mode of operation.

Then, for arbitrarily chosen associated data A' and any message M' with $\mathrm{len}(M') \leq \mathrm{len}(M)$, the attacker can compute N' and C' such that $((N', A', M'), C')$ is a valid message-ciphertext pair without making encryption or decryption queries as follows.

1. $IV \leftarrow$ the upper n bits of C, $\tilde{C} \leftarrow$ the remaining part of C
2. $N' \leftarrow \mathrm{GHASH}(K_1, A, M) \oplus \mathrm{GHASH}(K_1, A', M') \oplus N$
3. $\tilde{C}' \leftarrow M' \oplus$ (the upper $\mathrm{len}(M')$ bits of $\tilde{C} \oplus M$)
4. $C \leftarrow IV || \tilde{C}'$

That is, the attacker can perform universal forgery.

The above discussions intuitively show that GCM-SIV is not secure against code lifting attacks: An efficient lifter can leak information on exponentially many message-ciphertext pairs just by leaking a small amount of data, i.e., K_1 and a valid message-ciphertext pair. The attack works regardless of whether the underlying white-box block cipher is secure against code lifting or not.

Remark on Single-Key GCM-SIV GCM-SIV supports a single-key variant, where it takes a single key K_0 and derive two keys K_1 and K_2 as $K_1 = E_{K_0}(0^{128})$ and $K_2 = E_{K_0}(0^{127} \| 1)$. A natural way to implement this single-key variant is to (1) implement a white-box implementation of E_{K_0} and (2) compute $K_1 \leftarrow E_{K_0}(0^{128})$ and $K_2 \leftarrow E_{K_0}(0^{127} \| 1)$ at every encryption/decryption function call (i.e., E_{K_0} is used as a deterministic key derived function).

Unfotunately, it is a disaster. Given an unlimited access to such a white-box implementation, an attacker (lifter) can simply compute and copy K_1 and K_2.

Once K_1 and K_2 are retrieved, arbitrary plaintext (resp., ciphertext) can be encrypted (resp., decrypted).

Remark on GCM-SIV$^+$ AES-GCM-SIV described in RFC8452 is based on GCM-SIV$^+$, a variant of the original GCM-SIV. The POLYVAL hash function is used instead of GHASH, and the initial counter of CTR is generated from the tag in a different way. The two differences do not affect our attack, and the universal forgery attack works on GCM-SIV$^+$ too.

The single-key GCM-SIV$^+$ is further different from the single-key GCM-SIV, where it derives two keys K_1 and K_2 by encrypting a nonce (with some padding functions) with E_{K_0}. Thus, K_1 and K_2 dynamically change depending on nonces. Let $K_{1,N}$ and $K_{2,N}$ denote the keys derived from a nonce N. Again, an attacker (lifter) can copy $K_{1,N}$ and $K_{2,N}$ for arbitrary N. Once $K_{1,N}$ and $K_{2,N}$ are retrieved, arbitrary plaintext can be encrypted with nonce N, and arbitrary ciphertext encrypted with nonce N can be decrypted.

B.2 Code Lifting on CCM

CCM [79] is also used in many applications as the authenticated encryption, e.g., wireless LAN described in IEEE 802.11i. CCM uses CBC-MAC for the authentication and CTR for the encryption. Figure 6 shows a rough and simplified sketch of the CCM mode of operation. Both CTR_0 and A_0 are set by loading the nonce with different encodings. In addition, A_0 depends on the byte length of the message. (A_0 is not proper associated data since it depends on nonce, but we treat it as associated data for ease of explanation.)

Unlike GCM or GCM-SIV, universal forgery is infeasible. This is because, if an attacker wants to be able to evaluate chaining values of the CBC-MAC on an arbitrarily chosen message, then the attacker has to copy the whole implementation of E_K beforehand. However, existential forgery is still possible.

Existential Forgery. First, suppose an attacker knows the following information.

(A) The value of $E_K(CTR_0)$ and $E_K(A_0)$ that are derived from a nonce N (and a message length ℓ_M).
(B) The ciphertext C corresponding to an input (N, A_0, M). Here, M is an $\ell_M \cdot n$ bit (ℓ_M-block) message.
(C) The ciphertext C' corresponding to (N, A_0, M'), where M' is another $\ell_M \cdot n$ bit (ℓ_M-block) message.

Let X_i (resp., X'_i) be the input to E_K in the CBC-MAC part just after XORing M_i (resp., M'_i). In addition, let $Y_i := E_K$ and $Y'_i := E_K(X'_i)$. Then the attacker knowing (A)-(C) can compute all of X_i, X'_i, Y_i, and Y'_i.

Define $\tilde{\mathcal{M}} = \tilde{M}_1 || \cdots || \tilde{M}_{\ell_M}$ be the set of all the ℓ_M-bit block messages $\tilde{M}$ satisfying the following conditions:

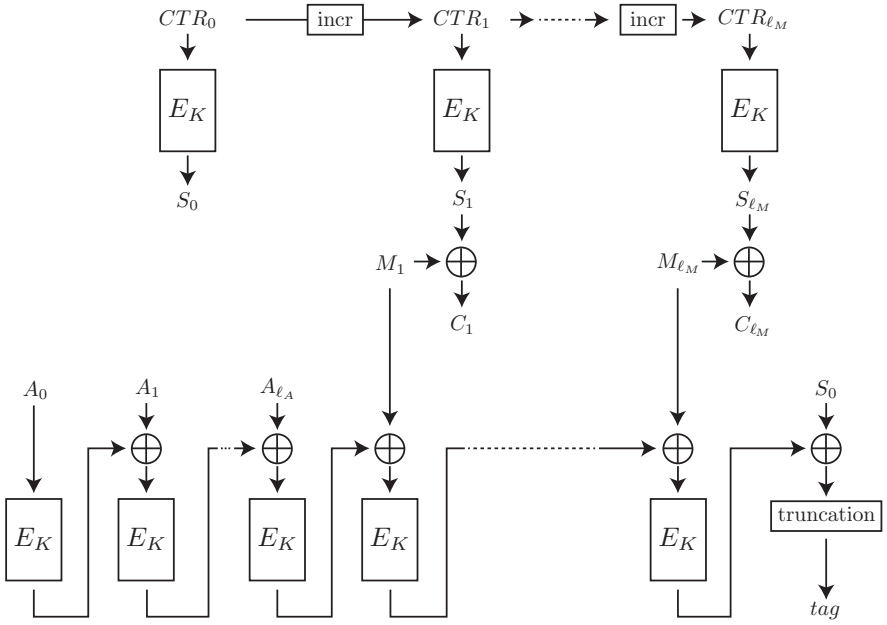

Fig. 6: CCM mode of operation.

1. $\tilde{M}_1 = E_K(A_0) \oplus X_1$ or $E_K(A_0) \oplus X'_1$
2. $\tilde{M}_i = \tilde{Y}_{i-1} \oplus X_i$ or $\tilde{Y}_{i-1} \oplus X'_1$ for $i = 2, 3, \ldots \ell_M$. (here, $\tilde{Y}_i$ denotes the output of E_K in the CBC-MAC part just after XORing $\tilde{M}_i$ in encrypting $\tilde{M}$, which is either of Y_i of Y'_i).

We can expect that the size of the set $\tilde{\mathcal{M}}$ is $\approx 2^{\ell_M}$.

Now we observe that an attacker can compute a valid ciphertext $\tilde{C}$ corresponding to $(N, A_0, \tilde{M})$ for arbitrary $\tilde{M} \in \tilde{\mathcal{M}}$ if the attacker knows (A)-(C), because the attacker can compute X_i and X'_i. (Note that the keystream $S_1||S_2||\cdots||S_{\ell_M}$ used in CTR is common for M, M', and $\tilde{M}$ and it can be recovered from M and C.)

The above discussions shows that CCM is vulnerable to code lifting: If a lifter leaks only a small amount of data (A)-(C), an attacker obtains the information on exponentially many (2^{ℓ_M}) input-output pairs of CCM. This attack works even if it is hard to copy the full implementation of E_K.

B.3 Code Lifting on OCB

OCB is an authenticated encryption designed by Rogaway et al. [73, 57]. Figure 7 shows a rough sketch of the OCB mode of operation when associated data is empty, where each Δ_i is computed from certain two values $Ktop, L_* \in \{0,1\}^{128}$,

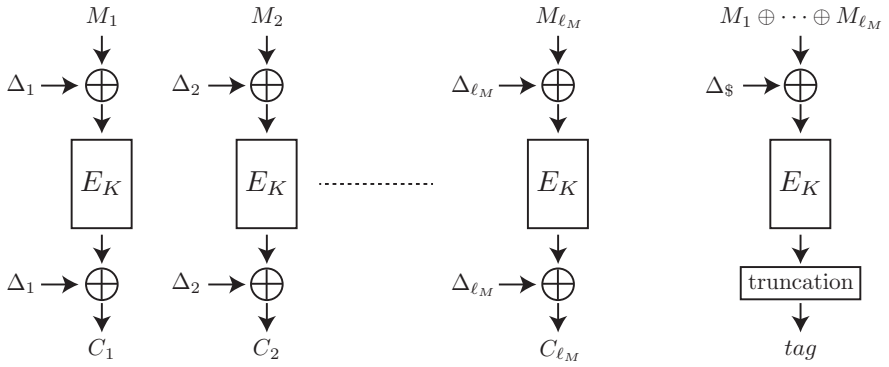

Fig. 7: OCB mode of operation.

which are derived from a nonce and E_K. (Here we assume the block length of E_K is $n = 128$.)

We show that existential forgery on OCB is possible by code lifting: Let c be a small positive integer (e.g., $c = 2$) and $\mathcal{X} := \{x \in \{0,1\}^{128} | x < 2^c\}$. A lifter first choose a nonce N, and compute corresponding $Ktop$, L_*, and $S = \Delta_1 \oplus \Delta_2 \oplus \cdots \Delta_{\ell_M} \oplus \Delta_\$$. The lifter then copies and leaks $Ktop$, L_*, 2^c outputs of E_K whose corresponding inputs has the form of $\mathcal{X}$, and 2^c outputs of E_K whose corresponding inputs has the form of $S \oplus \mathcal{X}$. In total, $2^{c+1} + 2$ blocks are copied.

Given the leakage by the lifter, an attacker can construct exponentially many message-ciphertext pairs, i.e., existential forgeries. Let X_i and Y_i be the input and output of the ith block cipher E_K, respectively. For arbitrary choice of $X_i \in \mathcal{X}$, the attacker can compute corresponding M_i and C_i because $Ktop$ and L_* are known. Moreover,

$$M_1 \oplus \cdots \oplus M_{\ell_M} \oplus \Delta_\$ = \bigoplus_{i=1}^{\ell_M} X_i \oplus S \in S \oplus \mathcal{X},$$

holds and they can also compute the output of E_K of the last block thanks to code lifting. Similarly to CCM, the number of existential forgeries (c^{ℓ_M}) is exponentially large.

C On Details of Results by Barbosa-Farshim

This section explains details of the results by Barbosa and Farshim on indifferentiabile AEAD schemes [6], where the ideal oracle is a *variable-key* random injection F and its inverse F^{-1}. Note that a variable random injection takes not only nonce, associated data, and message (or ciphertext) but also a key as an input. In what follows, we explain the results by Barbosa and Farshim especially relevant to our results: the impossibility of generic compositions, and indifferentiable constructions by Encode-then-Encipher (EtE) or 3-round Feistel.

C.1 Impossibility of Generic Compositions

A common way to build a secure AEAD scheme is to glue a secure MAC (or PRF) with a secure conventional encryption scheme. Such conversions are referred to as *generic compositions*. Possible generic compositions are comprehensively classified in a previous paper by Namprepre et al. [69]. However, Barbosa and Farshim observed that the generic compositions in Namprepre et al.'s paper, which includes SIV, are not indifferentiable AEAD schemes. Below we explain this, taking SIV as an example.

For simplicity, we assume that there is no associated data and nonces are n-bit strings. First, the underlying MAC (or PRF) is modeled as a random function $\mathsf{RF} : \{0,1\}^\kappa \times \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^\tau$, and the underlying conventional IV-based encryption scheme is modeled as a family of VIL random permutations $E : \{0,1\}^\kappa \times \{0,1\}^\tau \times \{0,1\}^* \to \{0,1\}^*$, i.e., $E(K, IV, \cdot)$ is a random permutation satisfying $\mathrm{len}(E(K, IV, M)) = \mathrm{len}(M)$ for each $K \in \{0,1\}^\kappa$ and $IV \in \{0,1\}^\tau$. Then the encryption function of SIV is defined as $\mathcal{E}((K_1, K_2), N, M) := \mathsf{RF}(K_1, N, M) || E(K_2, \mathsf{RF}(K_1, N, M), M)$. The tuple (RF, E, D) are regarded as primitive oracles, where D is the decryption function corresponding to E. Here, the important point is that a two-key construction is considered, i.e., two different keys K_1 and K_2 are used for RF and E, following the common setting for generic compositions.

Now it is easy to see that a simple differentiator exists against SIV: First, query an arbitrary tuple $((K_1, K_2), N, M)$ to the encryption oracle of SIV (real world) or a variable-key random injection $F^{\pm}$ (ideal world). Let T_1 be the tag part of the response, i.e., the most significant τ bits of the ciphertext. Second, query the tuple again to the same oracle, keeping (K_1, N, M) unchanged while changing K_2 to another value. Let T_2 be the tag part of the response. Then $T_1 = \mathsf{RF}(K_1, N, M) = T_2$ always holds in the real world but T_1 does not match T_2 with an overwhelming probability in the ideal world.

The above attack does not work in the single-key setting where $K_1 = K_2$, but still another attack differentiates SIV. First, choose random (K_1, N, M) and query $((K_1, K_1), N, M)$ to the encryption oracle of SIV (or variable-key random injection). Let $C = T||C'$ be the response, where T is the tag part. Second, query (K_1, T, C') to the decryption oracle of the underlying conventional IV-based encryption scheme (or the corresponding interface of a simulator), and let M' be the response. Then $M' = M$ always holds in the real world. However, in the ideal world, any simulator $\mathcal{S}$ cannot efficiently compute M and returns M' of another value with high probability even if $\mathcal{S}$ has an access to the ideal oracle $F^{\pm}$ since $\mathcal{S}$ does not have any information on N.

Remark 3. Barbosa and Farshim showed all the generic compositions by Namprempre et al. are not indifferentiable in the *two-key* setting. They also showed some of them including SIV are not indifferentiable even in the single-key setting, but do not show indifferentiability nor differentiator for others.

C.2 Indifferentiable Construction by EtE

Let $\mathsf{RO} : \{0,1\}^* \to \{0,1\}^\kappa$ be a VIL random oracle, and let $E : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^*$ be an ideally random tweakable enciphering scheme. That is, E_K is a length-preserving random permutation for each $K \in \{0,1\}^\kappa$. Denote E_K^{-1} by D_K, and define an AEAD $\Pi = (\mathcal{E}, \mathcal{D})$ as follows.

> **Encryption.** $\mathcal{E}(K, N, A, M) := E(\mathsf{RO}(K, N, A), M||0^\tau)$, where we assume (K, N, A) is encoded into a single bit string in a uniquely decodable manner.
> **Decryption.** $\mathcal{D}(K, N, A, C) := M$ if $D(\mathsf{RO}(K, N, A), C) = M||0^\tau$ for some M and $\mathcal{D}(K, N, A, C) := \perp$ otherwise.

The following theorem, which is a combination of Theorem 4, Lemma 2, and Proposition 3 of [6], shows the EtE scheme is indifferentiable froma a variable-key random injection. Here, the construction oracle is $(\mathcal{E}, \mathcal{D})$ and the primitive oracle is (RO, E, D) in the real world.

Theorem 4 (Indifferentiability of AEAD by EtE [6]). *There exists an expected $O(q)$-query simulator $\mathcal{S}$ such that $\mathsf{Adv}^{\mathrm{indiff}}_{\Pi, F^\pm, \mathcal{S}}(\mathcal{A}) \leq O(q^2/2^\kappa)$ holds for any adversary $\mathcal{A}$ making at most q queries.*

C.3 Indifferentiable Construction by 3-round Feistel

An indifferentiable AEAD can be built upon an enciphering scheme as in Theorem 4, but it is unclear how we can build an efficient enciphering scheme from a small primitive of fixed input lengths (FIL) and fixed output lengths (FOL) in an indifferentiably secure manner. The Feistel construction is known to be indifferentiable from a random permutation, but at least 6 rounds are required [31]. Barbosa and Farshim instead showed an indifferentiable AEAD scheme by an (unbalanced) 3-round Feistel with the message encoding $M \mapsto M||0^\tau$.

More precisely, let $f_1, f_3 : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^\tau$ and $f_2 : \{0,1\}^\kappa \times \{0,1\}^\tau \to \{0,1\}^\infty$ be random functions, and $\Phi_3(K, M||M')$ be the unbalanced 3-round Feistel construction computed as $X := M' \oplus f_1(K, M)$, $Y := M \oplus f_2(K, X)$, $Z := X \oplus f_3(K, Y) \oplus X$, $\Phi_3(K, M) = Z||Y$, where $M \in \{0,1\}^*, M' \in \{0,1\}^\tau$, and the output of $f_2(K, X)$ is truncated to $\mathrm{len}(M)$ bits. (see also Fig. 8).

Again, let $\mathsf{RO} : \{0,1\}^* \to \{0,1\}^\kappa$ be a random oracle, and $\Pi = (\mathcal{E}, \mathcal{D})$ denote the AEAD defined in the same way as the EtE scheme in Theorem 4 except that Φ_3 is used instead of a random enciphering scheme E. Then the following theorem, which is a combination of Theorem 5, Lemma 2, and Proposition 3 of [6], shows Π is indifferentiable from a variable-key random injection. Here, the construction oracle is $\Pi = (\mathcal{E}, \mathcal{D})$ and the primitive oracle is $(\mathsf{RO}, F_1, F_2, F_3)$ in the real world.

Theorem 5 (Indifferentiability of AEAD by 3-round Feistel [6]). *Suppose $q \leq O(\min\{\sqrt{2^{n+\tau}}, 2^n\})$. Then, there exists a simulator $\mathcal{S}$ making $O(q^2)$ queries such that $\mathsf{Adv}^{\mathrm{indiff}}_{\Pi^{\mathrm{F3}}, F^\pm, \mathcal{S}}(\mathcal{A}) \leq O(q^3/2^\tau + q^2/2^\kappa)$ holds for any adversary $\mathcal{A}$ making at most q queries in total.*

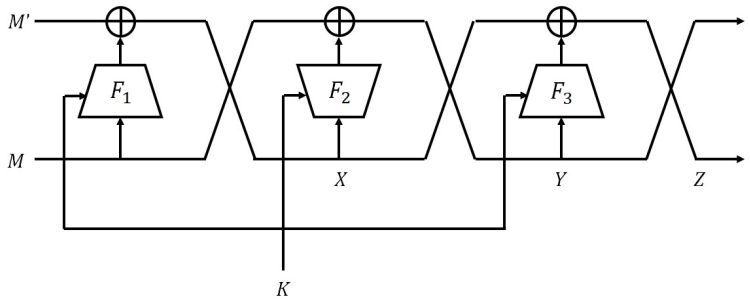

Fig. 8: 3-round (unbalanced) Feistel Construction $\Phi_3(K, M||M')$.

A VIL and variable output lengths (VOL) function that is indifferentiable from a random oracle can be built from a FIL/FOL random function by the sponge construction (Theorem 1). Thus, by replacing $\mathsf{RO}, F_1, F_2, F_3$ of Π with the sponge construction, we can build an indifferentiable AEAD from a FIL/FOL random function. (Compositions of indifferentiable schemes are again indifferentiable. See Remark 1.)

D On the Assumption that $\mathcal{S}$ can Observe Queries by $\mathcal{A}_{\mathbf{create}}$.

Here we explain why we assume that $\mathcal{S}$ can observe queries by $\mathcal{A}_{\mathrm{create}}$ in our white-box security notions. We assume the readers have already read Section 5.3 before reading this section, and know how we define whIND$-CPA. (See also the last paragraph of Section 4.2).

Let $\Pi = (\mathcal{E}, \mathcal{D})$ denote CTR that is based on an n-bit block cipher E_K. We assume there is a white-box implementation of Π such that the white-box implementation of E_K can efficiently be retrieved from the implementation of Π.

Intuitively, Π is white-box secure in the sense that $\mathcal{E}_K$ is indistinguishable from $\$(\cdot)$ in the following situation.

1. As an black-box oracle, an adversary is given access to $\mathcal{E}_K$ (in the real world) or $\$(\cdot)$ (in the ideal world).
2. Every time a message is queried to the encryption oracle, IV is always chosen uniformly at random (rather than chosen by adversary) and returned together with the ciphertext.
3. The underlying block cipher is E_K white-box secure.

This is because, even if an adversary $\mathcal{A}$ obtains a (partial) information L of the implementation of Π, it will not help $\mathcal{A}$ distinguish $\mathcal{E}_K$ from $\$(\cdot)$ because an IV is always randomly chosen, and L will not contain useful information on the value $E_K(\mathrm{IV})||E_K(IV+1)||\cdots$ with a high probability (as long as E_K is white-box secure). Thus we argue that a sound white-box security definition should be such a one that judges the random-IV CTR secure.

Indeed, we can show Π is whIND\$-CPA-secure if E_K is a whPRP (see Section 6.3). However, there exists an efficient adversary that "breaks" the scheme if the assumption that $\mathcal{S}$ can observe queries by $\mathcal{A}_{\mathrm{create}}$ is removed from the definition of whIND\$-CPA. The adversary runs as follows.

1. $\mathcal{A}_{\mathrm{create}}$ queries 0^n to the encryption oracle to obtain (IV, C). Then, $\mathrm{A}_{\mathrm{create}}$ builds a lifter $\mathcal{L}$ that, given an implementation of Π, computes the value $E_K(\mathrm{IV})$ by itself[32] and sends $L := E_K(IV)$ to $\mathcal{A}_{\mathrm{dist}}$ as a leakage ($\mathrm{A}_{\mathrm{create}}$ does not include the information of C into $\mathcal{L}$). In addition, $\mathcal{A}_{\mathrm{create}}$ passes both of C to $\mathcal{A}_{\mathrm{dist}}$.
2. Given L and C, finally $\mathcal{A}_{\mathrm{dist}}$ outputs 1 if $L = C$ and 0 if $L \neq C$.

Note that $\mathcal{A}_{\mathrm{dist}}$ always outputs 1 in the real world. Now, assume a simulator $\mathcal{S}$ in the ideal world cannot observe the queries by $\mathcal{A}_{\mathrm{create}}$ and the responses. Then $\mathcal{S}$ cannot return a leakage L which is equal to C with high probability because $\mathcal{S}$ has no information on C. Hence $\mathcal{A}_{\mathrm{dist}}$ outputs 0 with a high probability in the ideal world, and Π is deemed "insecure".

E On Composition of Weak Public Indifferentiable Schemes

This section explains details about the fact that a composition of two weak public indifferentiable schemes become weak public indifferentiable, if the schemes satisfy a few additional conditions.

Let T^{P} be a deterministic or random-IV construction calling an ideally random primitive P, and let R be the ideally random oracle of which interface is compatible with T^{P}. In addition, let U^{Q} be a deterministic construction calling an ideally random primitive Q, and assume the interfaces of P and U are compatible. Suppose P and Q are deterministic.

If T^{P} is a random-IV construction, we assume there is a deterministic construction $\tilde{\mathsf{T}}^{\mathsf{P}}$ and a set $\mathbf{IV}$ such that T^{P} runs as follows on arbitrary input X: (1) Choose a value IV uniformly at random from $\mathbf{IV}$. (2) Return $(IV, \tilde{\mathsf{T}}^{\mathsf{P}}(IV, X))$.

Suppose there exist non-decreasing functions $q_{\mathsf{T}}(\cdot,\cdot)$ and $\sigma_{\mathsf{T}}(\cdot,\cdot)$ such that, if T^{P} is evaluated on q inputs of which lengths are σ bits in total in a security game, then T makes at most $q_{\mathsf{T}}(q,\sigma)$ queries to P and the lengths of the queries are at most $\sigma_{\mathsf{T}}(q,\sigma)$ in total. In addition, suppose we have the following algorithms.

1. A simulator $\mathcal{S}_{\mathsf{T}}$ (resp., $\mathcal{S}_{\mathsf{U}}$) on weak public indifferentiability of T from R (resp., U from P), where the primitive oracle is P (resp., Q). There exist non-decreasing functions $q_{\mathcal{S}_{\mathsf{T}}}(\cdot,\cdot,\cdot,\cdot)$ and $\sigma_{\mathcal{S}_{\mathsf{T}}}(\cdot,\cdot,\cdot,\cdot)$ (resp., $q_{\mathcal{S}_{\mathsf{U}}}(\cdot,\cdot,\cdot,\cdot)$ and $\sigma_{\mathcal{S}_{\mathsf{U}}}(\cdot,\cdot,\cdot,\cdot)$) satisfying the following properties: If an adversary makes at most q_c (resp., q_p) queries to a construction (resp., primitive) oracle in security games of weak public indifferentiability, and the lengths of the queries are at most σ_c (resp., σ_p) in total, then $\mathcal{S}_{\mathsf{T}}$ (resp., $\mathcal{S}_{\mathsf{U}}$) makes at most

[32] Note that we are assuming the implementation E_K can efficiently be retrieved from the implementation of Π.

$q_{\mathcal{S}_\mathsf{T}}(q_c, \sigma_c, q_p, \sigma_p)$ queries to the ideal oracle R (resp., $q_{\mathcal{S}_\mathsf{U}}(q_c, \sigma_c, q_p, \sigma_p)$ queries to P). The lengths of the queries are $\sigma_{\mathcal{S}_\mathsf{T}}(\sigma_c, \sigma_p)$ at most (resp., $\sigma_{\mathcal{S}_\mathsf{U}}(\sigma_c, \sigma_p)$) in total.
2. An adversary $\mathcal{A}$ against weak public indifferentiability of T^U from R. The number of queries by $\mathcal{A}$ to the construction oracle (resp., primitive oracle) is at most $q_{\mathcal{A},c}$ (resp., $q_{\mathcal{A},p}$) and the lengths of the queries are at most $\sigma_{\mathcal{A},c}$ (resp., $\sigma_{\mathcal{A},p}$) in total.

Then the following lemma holds.

Lemma 2 (Formal version of Lemma 1). *Given the above situation, there exists (1) a simulator $\mathcal{S}_{\mathsf{T}^\cup}$ on weak public indifferentiability of T^U from R, where the primitive oracle is Q, and (2) an adversary $\mathcal{A}'$ (resp., $\mathcal{A}''$) against weak public indifferentiability of T from R (resp., U from P) such that*

$$\mathbf{Adv}^{\text{weak-pub-indiff}}_{\mathsf{T}^\cup,\mathsf{R},\mathcal{S}_{\mathsf{T}^\cup}}(\mathcal{A}) = \mathbf{Adv}^{\text{weak-pub-indiff}}_{\mathsf{T},\mathsf{R},\mathcal{S}_\mathsf{T}}(\mathcal{A}') + \mathbf{Adv}^{\text{weak-pub-indiff}}_{\mathsf{U},\mathsf{P},\mathcal{S}_\mathsf{U}}(\mathcal{A}'') \quad (3)$$

holds. Here, the number of queries by $\mathcal{S}_{\mathsf{T}^\cup}$ to R is at most $q_{\mathcal{S}_\mathsf{T}}(q_{\mathcal{A},c}, \sigma_{\mathcal{A},c}, q'_\mathsf{T} + q_{\mathcal{S}_\mathsf{U}}(q'_\mathsf{T}, \sigma'_\mathsf{T}, q_{\mathcal{A},p}, \sigma_{\mathcal{A},p}), \sigma'_\mathsf{T} + \sigma_{\mathcal{S}_\mathsf{U}}(q'_\mathsf{T}, \sigma'_\mathsf{T}, q_{\mathcal{A},p}, \sigma_{\mathcal{A},p}))$ and the lengths of the queries are at most $\sigma_{\mathcal{S}_\mathsf{T}}(q_{\mathcal{A},c}, \sigma_{\mathcal{A},c}, q'_\mathsf{T} + q_{\mathcal{S}_\mathsf{U}}(q'_\mathsf{T}, \sigma'_\mathsf{T}, q_{\mathcal{A},p}, \sigma_{\mathcal{A},p}), \sigma'_\mathsf{T} + \sigma_{\mathcal{S}_\mathsf{U}}(q'_\mathsf{T}, \sigma'_\mathsf{T}, q_{\mathcal{A},p}, \sigma_{\mathcal{A},p}))$ in total, where $q'_\mathsf{T} := q_\mathsf{T}(q_{\mathcal{A},c}, \sigma_{\mathcal{A},c})$ and $\sigma'_\mathsf{T} := \sigma_\mathsf{T}(q_{\mathcal{A},c}, \sigma_{\mathcal{A},c})$. The number of queries by $\mathcal{A}'$ to the construction oracle (resp., primitive oracle) is at most $q_{\mathcal{A},c}$ (resp., $q'_\mathsf{T} + q_{\mathcal{S}_\mathsf{U}}(q'_\mathsf{T}, \sigma'_\mathsf{T}, q_{\mathcal{A},p}, \sigma_{\mathcal{A},p})$) and the lengths of the queries are at most $\sigma_{\mathcal{A},c}$ (resp., $\sigma'_\mathsf{T} + \sigma_{\mathcal{S}_\mathsf{U}}(q'_\mathsf{T}, \sigma'_\mathsf{T}, q_{\mathcal{A},p}, \sigma_{\mathcal{A},p})$) in total. The number of queries by $\mathcal{A}''$ to the construction oracle (resp., primitive oracle) is at most q'_T (resp., $q_{\mathcal{A},p}$) and the lengths of the queries are at most σ'_T (resp., $\sigma_{\mathcal{A},p}$) in total.

Proof. Define $\mathcal{S}_{\mathsf{T}^\cup}$, $\mathcal{A}'$, and $\mathcal{A}''$ as follows.

$\underline{\mathcal{S}_{\mathsf{T}^\cup}}$. Note that $\mathcal{S}_{\mathsf{T}^\cup}$ is given oracle access to R and the revealing interface $\mathsf{Rev}'[\mathsf{R}]$. $\underline{\mathcal{S}_{\mathsf{T}^\cup}}$ keeps lists $\mathsf{List}_{\text{tmp}}$ and $\mathsf{List}_{\text{prim}}$, which are empty at the beginning[33]. When a query x is made to $\mathcal{S}_{\mathsf{T}^\cup}$, it runs as follows, using $\mathcal{S}_\mathsf{T}$ and $\mathcal{S}_\mathsf{U}$ as subroutines.

1. Query to the revealing interface $\mathsf{Rev}'[\mathsf{R}]$ and get the list $\mathsf{List}_\mathcal{A}[\mathsf{R}]$ storing queries made so far by $\mathcal{A}$ to R and the responses.
   If T is deterministic: For each $(X, Y) \in \mathsf{List}_\mathcal{A}[\mathsf{R}] \backslash \mathsf{List}_{\text{tmp}}$, do the following procedure:
   (a) Compute $\mathsf{T}^{\mathcal{S}_\mathsf{T}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}}$ on the input X, storing all the queries made by T to $\mathcal{S}_\mathsf{T}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}$ during the computation, together with the responses, into $\mathsf{List}_{\text{prim}}$.

[33] $\mathsf{List}_{\text{prim}}$ is a list to simulate the revealing interface given to $\mathcal{S}_\mathsf{U}$. $\mathsf{List}_{\text{tmp}}$ is a list constructed in such a way that $\mathsf{List}_\mathcal{A}[\mathsf{R}] \setminus \mathsf{List}_{\text{tmp}}$ in Step 1-(a) contains the queries (and the responses) that are made by $\mathcal{A}$ to R before x is queried to $\mathcal{S}_{\mathsf{T}^\cup}$ and after $\mathcal{S}_{\mathsf{T}^\cup}$ returned something to $\mathcal{A}$ in response to the previous query. (*Weak* public indifferentiability is introduced so that we can indeed build $\mathsf{List}_{\text{tmp}}$ in such a way. If lists returned by the revealing interface of R contain queries by $\mathcal{S}_\mathsf{T}$, it seems to hard to build a desired $\mathsf{List}_{\text{tmp}}$.)

(b) Add (X, Y) to $\mathsf{List}_{\mathrm{tmp}}$.

If T is a random-IV scheme: For each $(X, (IV, Y)) \in \mathsf{List}_{\mathcal{A}}[\mathsf{R}] \setminus \mathsf{List}_{\mathrm{tmp}}$, do the following procedure:

(a) Compute $\tilde{\mathsf{T}}^{\mathcal{S}_{\mathsf{T}}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}}$ on the input (IV, X), storing all the queries made to $\mathcal{S}_{\mathsf{T}}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}$ during the computation, together with the responses, into $\mathsf{List}_{\mathrm{prim}}$.
(b) Add $(X, (IV, Y))$ to $\mathsf{List}_{\mathrm{tmp}}$.

2. Run $\mathcal{S}_{\mathsf{U}}$ on x, simulating the ideal oracle P by running $\mathcal{S}_{\mathsf{T}}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}$. When $\mathcal{S}_{\mathsf{U}}$ makes a query to the revealing interface, return $\mathsf{List}_{\mathrm{prim}}$.
3. Finally, return the output of $\mathcal{S}_{\mathsf{U}}$.

$\underline{\mathcal{A}'.}$ Note that $\mathcal{A}'$ is given oracle access to $\mathcal{O}_{\mathrm{const}}$ (either of T^{P} or R) and $\mathcal{O}_{\mathrm{prim}}$ (either of P or $\mathcal{S}_{\mathsf{T}}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}$). $\mathcal{A}'$ keeps three lists $\mathsf{List}_{\mathrm{const}}$, $\mathsf{List}_{\mathrm{tmp}}$, and $\mathsf{List}_{\mathrm{prim}}$, which are empty at the beginning.

First, $\mathcal{A}'$ runs $\mathcal{A}$. Queries by $\mathcal{A}$ to a construction oracle is just forwarded to $\mathcal{O}_{\mathrm{const}}$ and recorded into $\mathsf{List}_{\mathrm{const}}$ together with the responses. When $\mathcal{A}$ makes a primitive query x, a primitive oracle for $\mathcal{A}$ is simulated as follows.

1. **If T is deterministic:** For each $(X, Y) \in \mathsf{List}_{\mathrm{const}} \setminus \mathsf{List}_{\mathrm{tmp}}$, do the following procedure:

   (a) Compute $\mathsf{T}^{\mathcal{O}_{\mathrm{prim}}}$ on the input X, storing all the queries made to $\mathcal{O}_{\mathrm{prim}}$ during the computation together with the responses into $\mathsf{List}_{\mathrm{prim}}$.
   (b) Add (X, Y) to $\mathsf{List}_{\mathrm{tmp}}$.

   If T is a random-IV scheme: For each $(X, (IV, Y)) \in \mathsf{List}_{\mathrm{const}} \setminus \mathsf{List}_{\mathrm{tmp}}$, do the following procedure:

   (a) Compute $\tilde{\mathsf{T}}^{\mathcal{O}_{\mathrm{prim}}}$ on the input (IV, X), storing all the queries made to $\mathcal{O}_{\mathrm{prim}}$ during the computation together with the responses into $\mathsf{List}_{\mathrm{prim}}$.
   (b) Add $(X, (IV, Y))$ to $\mathsf{List}_{\mathrm{tmp}}$.

2. Run $\mathcal{S}_{\mathsf{U}}$ on x. Queries by $\mathcal{S}_{\mathsf{U}}$ to an ideal oracle are forwarded to $\mathcal{O}_{\mathrm{prim}}$. When $\mathcal{S}_{\mathsf{U}}$ makes a query to the revealing interface, return $\mathsf{List}_{\mathrm{prim}}$.
3. Finally, return the output of $\mathcal{S}_{\mathsf{U}}$ (to $\mathcal{A}$).

$\underline{\mathcal{A}''.}$ Given oracle access to $\mathcal{O}_{\mathrm{const}}$ (either of U^{Q} or P) and $\mathcal{O}_{\mathrm{prim}}$ (either of Q or $\mathcal{S}_{\mathsf{U}}^{\mathsf{P},\mathsf{Rev}'[\mathsf{P}]}$), $\mathcal{A}''$ runs $\mathcal{A}^{\mathsf{T}^{\mathcal{O}_{\mathrm{const}}},\mathcal{O}_{\mathrm{prim}}}$. When $\mathcal{A}$ returns an output, $\mathcal{A}''$ returns it as its own output.

Then, since P and Q are deterministic, we have

$$\begin{aligned}
\mathbf{Adv}^{\text{weak-pub-indiff}}_{\mathsf{T}^{\mathsf{U}},\mathsf{R},\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}}(\mathcal{A}) &= \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{T}^{\mathsf{U}^{\mathsf{Q}}},\mathsf{Q}}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{R},\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}}\right] \\
&= \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{T}^{\mathsf{U}^{\mathsf{Q}}},\mathsf{Q}}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{T}^{\mathsf{P}},\mathcal{S}_{\mathsf{U}}^{\mathsf{P},\mathsf{Rev}'[\mathsf{P}]}}\right] \\
&\quad + \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{T}^{\mathsf{P}},\mathcal{S}_{\mathsf{U}}^{\mathsf{P},\mathsf{Rev}'[\mathsf{P}]}}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{R},\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}}\right] \\
&= \Pr\left[1 \leftarrow \mathcal{A}''^{\mathsf{U}^{\mathsf{Q}},\mathsf{Q}}\right] - \Pr\left[1 \leftarrow \mathcal{A}''^{\mathsf{P},\mathcal{S}_{\mathsf{U}}^{\mathsf{P},\mathsf{Rev}'[\mathsf{P}]}}\right] \\
&\quad + \Pr\left[1 \leftarrow \mathcal{A}'^{\mathsf{T}^{\mathsf{P}},\mathsf{P}}\right] - \Pr\left[1 \leftarrow \mathcal{A}'^{\mathsf{R},\mathcal{S}_{\mathsf{T}}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}}\right] \\
&= \mathbf{Adv}^{\text{weak-pub-indiff}}_{\mathsf{U},\mathsf{P},\mathcal{S}_{\mathsf{U}}}(\mathcal{A}'') + \mathbf{Adv}^{\text{weak-pub-indiff}}_{\mathsf{T},\mathsf{R},\mathcal{S}_{\mathsf{T}}}(\mathcal{A}').
\end{aligned}$$

Hence Eq. (3) holds.

Next, we analyze the number of queries by $\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}$ and their lengths.

Suppose the ideal game of weak public indifferentiability on U^{Q} is executed with an adversary $\mathcal{B}$ and $\mathcal{S}_{\mathsf{U}}^{\mathsf{P},\mathsf{Rev}'[\mathsf{P}]}$. By assumption, the number of queries by S_{U} to P and their lengths are upper bounded by $q_{\mathcal{S}_{\mathsf{U}}}(q_c, \sigma_c, q_p, \sigma_p)$ and $\sigma_{\mathcal{S}_{\mathsf{U}}}(q_c, \sigma_c, q_p, \sigma_p)$, where

q_c: the number of construction queries by $\mathcal{B}$,
σ_c: the total length of the construction queries by $\mathcal{B}$,
q_p: the number of primitive queries by $\mathcal{B}$, and
σ_p: the total length of the primitive queries by $\mathcal{B}$.

In fact, $\mathcal{S}_{\mathsf{U}}$ can get information on construction queries by $\mathcal{B}$ only through lists returned by the revealing interface. In particular, when analyzing the number of queries by $\mathcal{S}_{\mathsf{U}}$ and their lengths, q_c and σ_c can be replaced with

$\tilde{q}_c$: the maximum number of queries stored in lists returned by the revealing interface, and
$\tilde{\sigma}_c$: the maximum of the total length of queries stored in lists returned by the revealing interface.

Recall $q'_{\mathsf{T}} := q_{\mathsf{T}}(q_{\mathcal{A},c}, \sigma_{\mathcal{A},c})$ and $\sigma'_{\mathsf{T}} := \sigma_{\mathsf{T}}(q_{\mathcal{A},c}, \sigma_{\mathcal{A},c})$. When the ideal game of weak public indifferentiability for T^{U} is executed with $\mathcal{A}$ and $\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}$, The number of queries made by T to $\mathcal{S}_{\mathsf{T}}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}$ at Step 1-(a) of $\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}$ is at most q'_{T} in total and their total length is at most σ'_{T}. In addition, when $\mathcal{S}_{\mathsf{U}}$ makes a query to the revealing interface in Step 2 of $\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}$, $\mathsf{List}_{\text{prim}}$ is returned. The queries stored in $\mathsf{List}_{\text{prim}}$ are those recorded at Step 1-(a). The number of those queries are at most q'_{T} and the length is at most σ'_{T} in total. Hence (by the above discussions on $\tilde{q}_c$ and $\tilde{\sigma}_c$), the number of queries made by $\mathcal{S}_{\mathsf{U}}$ to $\mathcal{S}_{\mathsf{T}}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}$ in Step 2 of $\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}$ is at most $q_{\mathcal{S}_{\mathsf{U}}}(q'_{\mathsf{T}}, \sigma'_{\mathsf{T}}, q_{\mathcal{A},c}, \sigma_{\mathcal{A},p})$ and the total length is at most $\sigma_{\mathcal{S}_{\mathsf{U}}}(q'_{\mathsf{T}}, \sigma'_{\mathsf{T}}, q_{\mathcal{A},c}, \sigma_{\mathcal{A},p})$. Therefore the number of queries by $\mathcal{S}_{\mathsf{T}^{\mathsf{U}}}$ to R is at most

$$q_{\mathcal{S}_{\mathsf{T}}}(q_{\mathcal{A},c}, \sigma_{\mathcal{A},c}, q'_{\mathsf{T}} + q_{\mathcal{S}_{\mathsf{U}}}(q'_{\mathsf{T}}, \sigma'_{\mathsf{T}}, q_{\mathcal{A},p}, \sigma_{\mathcal{A},p}), \sigma'_{\mathsf{T}} + \sigma_{\mathcal{S}_{\mathsf{U}}}(q'_{\mathsf{T}}, \sigma'_{\mathsf{T}}, q_{\mathcal{A},p}, \sigma_{\mathcal{A},p}))$$

and the lengths of the queries are at most

$$\sigma_{\mathcal{S}_\mathsf{T}}(q_{\mathcal{A},c}, \sigma_{\mathcal{A},c}, q'_\mathsf{T} + q_{\mathcal{S}_\mathsf{U}}(q'_\mathsf{T}, \sigma'_\mathsf{T}, q_{\mathcal{A},p}, \sigma_{\mathcal{A},p}), \sigma'_\mathsf{T} + \sigma_{\mathcal{S}_\mathsf{U}}(q'_\mathsf{T}, \sigma'_\mathsf{T}, q_{\mathcal{A},p}, \sigma_{\mathcal{A},p}))$$

in total.

The upper bounds of queries by $\mathcal{A}'$ and their lengths can be derived in the same way. Analysis of the upper bounds on $\mathcal{A}''$ is trivial. □

F Proof of Theorem 2

Proof (of Theorem 2). Define $\mathcal{A}'$, $\mathcal{S}_{\mathrm{const}}$, and $\mathcal{A}''$ as follows.

$\underline{\mathcal{A}' = (\mathcal{A}'_{\mathrm{create}}, \mathcal{A}'_{\mathrm{dist}})}$: Note that $\mathcal{A}$ is given access to an oracle $\mathcal{O}_{\mathrm{prim}}$, which is expected to be π or P. $\mathcal{A}' = (\mathcal{A}'_{\mathrm{create}}, \mathcal{A}'_{\mathrm{dist}})$ runs as follows, utilizing $\mathcal{A}$.

1. First, $\mathcal{A}'_{\mathrm{create}}$ runs $\mathcal{A}_{\mathrm{create}}^{\Sigma^{\mathcal{O}_{\mathrm{prim}}}}$. When $\mathcal{A}_{\mathrm{create}}$ outputs a state S and a lifter $\mathcal{L}$ (for $\mathcal{C}_{\Sigma^\pi}$), $\mathcal{A}'_{\mathrm{create}}$ builds and outputs a lifter $\mathcal{L}'$ (for $\mathcal{C}_\pi$) running as follows.
   (a) Build a white-box implementation $[\![\Sigma^\pi]\!]$ of $(\Sigma^\pi, \mathcal{C}_{\Sigma^\pi})$ using the white-box implementation $[\![\pi]\!]$ of $(\pi, \mathcal{C}_\pi)$ given to $\mathcal{L}'$, and run $\mathcal{L}$ on the input $[\![\Sigma^\pi]\!]$.
   (b) When $\mathcal{L}$ outputs a leakage, forward it to $\mathcal{A}_{\mathrm{dist}}$.
   In addition, $\mathcal{A}'_{\mathrm{create}}$ passes the state S to $\mathcal{A}'_{\mathrm{dist}}$ as its own state.
2. The second stage adversary $\mathcal{A}'_{\mathrm{dist}}$ receives a state S from $\mathcal{A}'_{\mathrm{create}}$ and L from a lifter (or a simulator), and then runs $\mathcal{A}_{\mathrm{dist}}^{\Sigma^{\mathcal{O}_{\mathrm{prim}}}}(S, L)$. Finally $\mathcal{A}'_{\mathrm{dist}}$ returns the output of $\mathcal{A}_{\mathrm{dist}}$.

$\underline{\mathcal{S}_{\mathrm{const}}}$: Note that, as inputs, $\mathcal{S}_{\mathrm{const}}$ receives the description of a lifter $\mathcal{L}$ (for $\mathcal{C}_{\Sigma^\pi}$) and a list $\mathsf{List}_{\mathrm{create}}$ containing all the queries by $\mathcal{A}_{\mathrm{create}}$ and the responses. In addition, $\mathcal{S}_{\mathrm{const}}$ is given oracle access to R. $\mathcal{S}_{\mathrm{const}}$ runs as follows, keeping an additional list $\mathsf{List}_{\mathrm{prim}}$ which is empty at the beginning.

1. From $\mathcal{L}$, build a lifter $\mathcal{L}'$ for $\mathcal{C}_\pi$, like $\mathcal{A}'_{\mathrm{create}}$ does.
2. **If Σ is deterministic:** For each $(X, Y) \in \mathsf{List}_{\mathrm{create}}$, compute the function $\Sigma^{\mathcal{S}_{\mathrm{indiff}}^{\mathsf{R},\mathsf{Rev\text{-}Sim}}}$ on the input X, recording all the queries by Σ to $\mathcal{S}_{\mathrm{indiff}}^{\mathsf{R},\mathsf{Rev\text{-}Sim}}$ into $\mathsf{List}_{\mathrm{prim}}$ together with the responses. Here, $\mathsf{Rev\text{-}Sim}$ is an oracle that returns $\mathsf{List}_{\mathrm{create}}$.
   If Σ is a random-IV scheme: For each $(X, (IV, Y)) \in \mathsf{List}_{\mathrm{create}}$, compute the function $\tilde{\Sigma}^{\mathcal{S}_{\mathrm{indiff}}^{\mathsf{R},\mathsf{Rev\text{-}Sim}}}$ on the input (IV, X), recording all the queries by $\tilde{\Sigma}$ to $\mathcal{S}_{\mathrm{indiff}}^{\mathsf{R},\mathsf{Rev\text{-}Sim}}$ into $\mathsf{List}_{\mathrm{prim}}$ together with the responses. Here, $\mathsf{Rev\text{-}Sim}$ is an oracle that returns $\mathsf{List}_{\mathrm{create}}$.
3. Run $\mathcal{S}_{\mathrm{prim}}^{\mathcal{S}_{\mathrm{indiff}}^{\mathsf{R},\mathsf{Rev\text{-}Sim}}}(\mathcal{L}', \mathsf{List}_{\mathrm{prim}})$.
4. Return the output of $\mathcal{S}_{\mathrm{prim}}$.

$\mathcal{A}''$: Note that this adversary is given oracle access to $\mathcal{O}_{\mathrm{const}}$ and $\mathcal{O}_{\mathrm{prim}}$, which correspond to a construction oracle and a primitive oracle (or simulator) in the games of weak public indifferentiability. $\mathcal{A}''$ runs as follows, keeping two lists $\mathsf{List}_{\mathrm{create}}$ and $\mathsf{List}_{\mathrm{prim}}$ which are empty at the beginning.

1. Run $\mathcal{A}_{\mathrm{create}}^{\mathcal{O}_{\mathrm{const}}}$, recording all the queries to $\mathcal{O}_{\mathrm{const}}$ by $\mathcal{A}_{\mathrm{create}}$ into a list $\mathsf{List}_{\mathrm{create}}$ together with the responses. Let $\mathcal{L}$ and S denote the lifter and the state returned by $\mathcal{A}_{\mathrm{create}}$, and $\mathcal{L}'$ be the lifter constructed from $\mathcal{L}$ as is done in $\mathcal{A}'$.
2. **If Σ is deterministic:** For each $(X, Y) \in \mathsf{List}_{\mathrm{create}}$, compute the function $\Sigma^{\mathcal{O}_{\mathrm{prim}}}$ on the input X, recording all the queries by Σ to $\mathcal{O}_{\mathrm{prim}}$ a list $\mathsf{List}_{\mathrm{prim}}$ together with the responses.
   If Σ is a random-IV scheme: For each $(X, (IV, Y)) \in \mathsf{List}_{\mathrm{create}}$, compute the function $\tilde{\Sigma}^{\mathcal{O}_{\mathrm{prim}}}$ on the input (IV, X), recording all the queries by $\tilde{\Sigma}$ to $\mathcal{O}_{\mathrm{prim}}$ a list $\mathsf{List}_{\mathrm{prim}}$ together with the responses.
3. Run $L \leftarrow \mathcal{S}_{\mathrm{prim}}^{\mathcal{O}_{\mathrm{prim}}}(\mathcal{L}', \mathsf{List}_{\mathrm{prim}})$, and then $b \leftarrow \mathcal{A}_{\mathrm{dist}}^{\mathcal{O}_{\mathrm{const}}}(S, L)$.
4. Return b.

We show they satisfy Eq. (1) by introducing a sequence of games.

Game 0. This game is the real game on $(\Sigma^{\pi}, \mathcal{C}_{\Sigma^{\pi}})$ w.r.t. sec-const when the adversary is $\mathcal{A}$.

Game 1.

0. Sample an ideal primitive P corresponding to π. Let $\mathsf{List}_{\mathrm{create}}$ and $\mathsf{List}_{\mathrm{prim}}$ be empty lists.
1. Run $(\mathcal{L}, S) \leftarrow \mathcal{A}_{\mathrm{create}}^{\Sigma^{\mathsf{P}}}$, recording all the queries by $\mathcal{A}_{\mathrm{create}}$ to Σ^{P} into a list $\mathsf{List}_{\mathrm{create}}$ together with the responses. Then, build $\mathcal{L}'$ from $\mathcal{L}$ like $\mathcal{A}'$ does.
2. **If Σ is deterministic:** For each $(X, Y) \in \mathsf{List}_{\mathrm{create}}$, compute Σ^{P} on the input X, recording all the queries by Σ to P into a list $\mathsf{List}_{\mathrm{prim}}$ together with the responses.
   If Σ is a random-IV scheme: For each $(X, (IV, Y)) \in \mathsf{List}_{\mathrm{create}}$, compute $\tilde{\Sigma}^{\mathsf{P}}$ on the input (IV, X), recording all the queries by $\tilde{\Sigma}$ to P into a list $\mathsf{List}_{\mathrm{prim}}$ together with the responses.
3. Run $L \leftarrow \mathcal{S}_{\mathrm{prim}}^{\mathsf{P}}(\mathcal{L}', \mathsf{List}_{\mathrm{prim}})$
4. Run $b \leftarrow \mathcal{A}_{\mathrm{dist}}^{\Sigma^{\mathsf{P}}}(S, L)$.
5. Return b.

We see the output distributions of Game 0 and Game 1 are identical to that of $\mathsf{Exp}_{\pi, \mathcal{C}_{\pi}, \mathcal{A}'}^{\boxed{\text{sec-prim}}\text{-real}}$ and $\mathsf{Exp}_{\mathcal{S}_{\mathrm{prim}}, \mathcal{A}'}^{\boxed{\text{sec-prim}}\text{-ideal}}$. Hence

$$\Pr[\text{Game } 0 = 1] - \Pr[\text{Game } 1 = 1] = \mathbf{Adv}_{\pi, \mathcal{C}_{\pi}, \mathcal{S}_{\mathrm{prim}}}^{\text{sec-prim}}(\mathcal{A}') \tag{4}$$

holds.

Game 2.

0. Sample an ideal primitive R corresponding to Σ^π. Let $\mathsf{List}_{\mathrm{create}}$ and $\mathsf{List}_{\mathrm{prim}}$ be empty lists.
1. Run $(\mathcal{L}, S) \leftarrow \mathcal{A}^{\mathsf{R}}_{\mathrm{create}}$, recording all the queries to R by $\mathcal{A}_{\mathrm{create}}$ and the responses into a list $\mathsf{List}_{\mathrm{create}}$. Then, build $\mathcal{L}'$ from $\mathcal{L}$ like $\mathcal{A}'$ does.
2. **If Σ is deterministic:** For each $(X, Y) \in \mathsf{List}_{\mathrm{create}}$, compute the function $\Sigma^{\mathcal{S}^{\mathsf{R},\mathsf{Rev\text{-}Sim}}_{\mathrm{indiff}}}$ on the input X, recording all the queries by Σ to $\mathcal{S}^{\mathsf{R},\mathsf{Rev\text{-}Sim}}_{\mathrm{indiff}}$ into $\mathsf{List}_{\mathrm{prim}}$ together with the responses. Here, $\mathsf{Rev\text{-}Sim}$ is an oracle that returns $\mathsf{List}_{\mathrm{create}}$.
   If Σ is a random-IV scheme: For each $(X, (IV, Y)) \in \mathsf{List}_{\mathrm{create}}$, compute the function $\tilde{\Sigma}^{\mathcal{S}^{\mathsf{R},\mathsf{Rev\text{-}Sim}}_{\mathrm{indiff}}}$ on the input (IV, X), recording all the queries by $\tilde{\Sigma}$ to $\mathcal{S}^{\mathsf{R},\mathsf{Rev\text{-}Sim}}_{\mathrm{indiff}}$ into $\mathsf{List}_{\mathrm{prim}}$ together with the responses. Here, $\mathsf{Rev\text{-}Sim}$ is an oracle that returns $\mathsf{List}_{\mathrm{create}}$.
3. Run $L \leftarrow \mathcal{S}^{\mathcal{S}^{\mathsf{R},\mathsf{Rev\text{-}Sim}}_{\mathrm{indiff}}}_{\mathrm{prim}}(\mathcal{L}', \mathsf{List}_{\mathrm{prim}})$.
4. Run $b \leftarrow \mathcal{A}^{\mathsf{R}}_{\mathrm{dist}}(S, L)$.
5. Return b.

Now, we see that the output distribution of Game 1 (resp., Game 2) is identical to that of the real (resp., ideal) game for weak public indifferentiability of Σ^{P} from R when the adversary is $\mathcal{A}''$ and the simulator is $\mathcal{S}_{\mathrm{indiff}}$. Therefore

$$\Pr[\text{Game } 1 = 1] - \Pr[\text{Game } 2 = 1] = \mathbf{Adv}^{\text{pub-indiff}}_{\Sigma,\mathsf{R},\mathcal{S}_{\mathrm{indiff}}}(\mathcal{A}'') \tag{5}$$

holds.

Game 3. This is the ideal game on $(\Sigma^\pi, \mathcal{C}_{\Sigma^\pi})$ w.r.t. sec-const where the adversary is $\mathcal{A}$ and the simulator is $\mathcal{S}_{\mathrm{const}}$. Then

$$\Pr[\text{Game } 2 = 1] - \Pr[\text{Game } 3 = 1] = 0 \tag{6}$$

holds.

From Eq. (4)-(6),

$$\begin{aligned}
\mathbf{Adv}^{\text{sec-const}}_{\Sigma^\pi,\mathcal{C}_{\Sigma^\pi},\mathcal{S}_{\mathrm{const}}}(\mathcal{A}) &= \Pr\left[\mathbf{Exp}^{\boxed{\text{sec-const}}\text{-real}}_{\Sigma^\pi,\mathcal{C}_{\Sigma^\pi},\mathcal{A}} = 1\right] - \Pr\left[\mathbf{Exp}^{\boxed{\text{sec-const}}\text{-ideal}}_{\mathcal{S}_{\mathrm{const}},\mathcal{A}} = 1\right] \\
&= \Pr[\text{Game } 0 = 1] - \Pr[\text{Game } 3 = 1] \\
&= \mathbf{Adv}^{\text{sec-prim}}_{\pi,\mathcal{C}_\pi,\mathcal{S}_{\mathrm{prim}}}(\mathcal{A}') + \mathbf{Adv}^{\text{weak-pub-indiff}}_{\Sigma,\mathsf{R},\mathcal{S}_{\mathrm{indiff}}}(\mathcal{A}'')
\end{aligned}$$

follows.

Next, we analyze the number of queries by $\mathcal{S}_{\mathrm{const}}$ and their lengths.

Suppose the ideal game of weak public indifferentiability on Σ^{R} is executed with an adversary $\mathcal{B}$ and $\mathcal{S}^{\mathsf{R},\mathsf{Rev}'[\mathsf{R}]}_{\mathrm{indiff}}$. By assumption, the number of queries by $\mathcal{S}_{\mathrm{indiff}}$ to R and their lengths are upper bounded by $q_{\mathcal{S}_{\mathrm{indiff}}}(q_c, \sigma_c, q_p, \sigma_p)$ and $\sigma_{\mathcal{S}_{\mathrm{indiff}}}(q_c, \sigma_c, q_p, \sigma_p)$, where

q_c: the number of construction queries by $\mathcal{B}$,

σ_c: the total length of the construction queries by $\mathcal{B}$,
q_p: the number of primitive queries by $\mathcal{B}$, and
σ_p: the total length of the primitive queries by $\mathcal{B}$.

In fact, $\mathcal{S}_{\text{indiff}}$ can get information on construction queries by $\mathcal{B}$ only through lists returned by the revealing interface. In particular, when analyzing the number of queries by $\mathcal{S}_{\text{indiff}}$ and their lengths, q_c and σ_c can be replaced with

$\tilde{q}_c$: the maximum number of queries stored in lists returned by the revealing interface, and
$\tilde{\sigma}_c$: the maximum of the total length of queries stored in lists returned by the revealing interface.

Recall $q'_\Sigma := q_\Sigma(q_\mathcal{A}, \sigma_\mathcal{A})$ and $\sigma'_\Sigma := \sigma_\Sigma(q_{\mathcal{A},c}, \sigma_{\mathcal{A},c})$. When the ideal game for $(\Sigma^\pi, \mathcal{C}_{\Sigma^\pi})$ is executed with $\mathcal{A}$ and $\mathcal{S}_{\text{const}}$, the number of queries made by Σ to $\mathcal{S}_{\text{indiff}}^{\mathsf{R},\mathsf{Rev\text{-}Sim}}$ at Step 2 of $\mathcal{S}_{\text{const}}$ is at most q'_Σ in total and their total length is at most σ'_Σ. The number of queries made by $\mathcal{S}_{\text{prim}}$ to $\mathcal{S}_{\text{indiff}}^{\mathsf{R},\mathsf{Rev\text{-}Sim}}$ at Step 3 of $\mathcal{S}_{\text{const}}$ is at most $q_{\mathcal{S}_{\text{prim}}}$ in total and their total length is at most $\sigma_{\mathcal{S}_{\text{prim}}}$. In addition, when $\mathcal{S}_{\text{indiff}}$ makes a query to the revealing interface in Step 2 or Step 3 of $\mathcal{S}_{\text{const}}$, $\mathsf{List}_{\text{create}}$ is returned. The queries stored in $\mathsf{List}_{\text{create}}$ are those made by $\mathcal{A}_{\text{create}}$. Thus, the number of these queries are at most $q_\mathcal{A}$ and the lengths are at most $\sigma_\mathcal{A}$ in total. Hence (by the above discussions on $\tilde{q}_c$ and $\tilde{\sigma}_c$), the number of queries by $\mathcal{S}_{\text{const}}$ to R is at most

$$q_{\mathcal{S}_{\text{indiff}}}(q_\mathcal{A}, \sigma_\mathcal{A}, q'_\Sigma + q_{\mathcal{S}_{\text{prim}}}, \sigma'_\Sigma + \sigma_{\mathcal{S}_{\text{prim}}})$$

and the lengths of the queries are at most

$$\sigma_{\mathcal{S}_{\text{indiff}}}(q_\mathcal{A}, \sigma_\mathcal{A}, q'_\Sigma + q_{\mathcal{S}_{\text{prim}}}, \sigma'_\Sigma + \sigma_{\mathcal{S}_{\text{prim}}})$$

in total.

The upper bounds of queries by $\mathcal{A}'$ and their lengths can be derived in the same way. Analysis of the upper bounds on $\mathcal{A}''$ is trivial.

□

G Proof of Proposition 1

This section shows Proposition 1.

Intuition of the Proof. Suppose we execute the ideal game with an adversary $\mathcal{A}$. The simulator $\mathcal{S}$ runs as follows, recording all the queries made to $\mathcal{S}$ together with the responses that $\mathcal{S}$ return to $\mathcal{A}$:
Simulation of P. When $\mathcal{A}$ queries a fresh value x to the first primitive interface (corresponding to P), $\mathcal{S}$ queries x to RF and returns $\mathsf{RF}(x)$ to $\mathcal{A}$.
Simulation of P^{-1}. Let `Dom-Defined` be the set of the values x such that (1) $\mathcal{A}$ has queried x to the construction oracle or the first primitive interface (corresponding to P) before, or (2) $\mathcal{A}$ has received x as a response from the second

primitive interface (corresponding to P^{-1}) before. When $\mathcal{A}$ queries a fresh value y to the second primitive interface (corresponding to P^{-1}), $\mathcal{S}$ queries to the revealing interface $\mathsf{Rev}'[\mathsf{RF}]$ to construct $\texttt{Dom-Defined}$. Then, $\mathcal{S}$ chooses x uniformly at random from $\{0,1\}^n \setminus \texttt{Dom-Defined}$, and returns x to $\mathcal{A}$ as a response.

The above simulation works well as long as no collision of RF is observed, and we obtain the desired result. A complete proof is as follows.

Proof (of Proposition 1). Define a simulator $\mathcal{S}$ as follows.

$\underline{\mathcal{S}:}$ Note that $\mathcal{S}$ is given an oracle access to RF and the revealing interface $\mathsf{Rev}[\mathsf{RF}]$. $\mathcal{S}$ keeps a list $\mathsf{List}[P]$, which is empty at the beginning of the game, and runs as follows.

1. When an adversary queries x to the primitive interface corresponding to P, check if $(x, y) \in \mathsf{List}[P]$ for some y. If there exists such y, respond with y. Otherwise, query x to RF, respond with $\mathsf{RF}(x)$, and add the pair $(x, \mathsf{RF}(x))$ into $\mathsf{List}[P]$.
2. When an adversary queries y to the primitive interface corresponding to P^{-1}, check if $(x, y) \in \mathsf{List}[P]$ for some x. If there exists such x, respond with x. Otherwise, query to the revealing interface to get the list $\mathsf{List}[\mathsf{RF}]$ of queries made so far to RF and the responses. If $\mathsf{List}[\mathsf{RF}]$ contains a collision of RF, return $\perp$. If $\mathsf{List}[\mathsf{RF}]$ does not contain collisions, execute the following procedure.
   (a) $\mathsf{List}[P] \leftarrow \mathsf{List}[P] \cup \mathsf{List}[\mathsf{RF}]$.
   (b) Check if $(x, y) \in \mathsf{List}[P]$ for some x. If there exists such x, return x. Otherwise, proceed to the next step.
   (c) $x \xleftarrow{\$} \{0,1\}^n \setminus X$, where $X := \{x' | (x', y') \in \mathsf{List}[P]\}$.
   (d) Add (x, y) to $\mathsf{List}[P]$ and return x.

Let $\mathcal{A}$ be an adversary making at most q_c construction queries and q_p primitive queries, respectively. Then the above simulator $\mathcal{S}$ makes at most q_p queries to RF. In what follows we show

$$\mathbf{Adv}^{\text{pub-indiff}}_{P,\mathsf{RF},\mathcal{S}}(\mathcal{A}) \leq \frac{(q_c + q_p)^2}{2^n} \tag{7}$$

holds by introducing games and using the game-playing technique [10].

$\underline{\text{Game 1.}}$
This is the real game of public indifferentiability with adversary $\mathcal{A}$ where the construction oracle is P and the primitive oracle is (P, P^{-1}).

$\underline{\text{Game 2.}}$
In this game, the construction oracle P and primitive oracle (P, P^{-1}) are replaced with $\mathcal{O}_{\text{const}}$ and $(\mathcal{O}^+_{\text{prim}}, \mathcal{O}^-_{\text{prim}})$ in Fig. 9, respectively, where the single-boxed part of Fig. 9 is executed but the double-boxed part is not executed. $(\mathcal{O}_{\text{const}}, (\mathcal{O}^+_{\text{prim}}, \mathcal{O}^-_{\text{prim}}))$ is completely indistinguishable from $(P, (P, P^{-1}))$ and

Algorithm 11: $\mathcal{O}_{\mathrm{const}}(x)$.

if $\exists y, (x, y) \in \mathsf{List}[\mathsf{RF}]$ **then**
  return y
else
  $y \xleftarrow{\$} \{0,1\}^n$
  if $\exists (x', y') \in \mathsf{List}[\mathsf{RF}]$ s.t. $y = y'$ **then**
    $\mathbf{bad} \leftarrow 1$
    $\boxed{y \xleftarrow{\$} \{0,1\}^n \setminus \{y' | (x', y') \in \mathsf{List}[\mathsf{RF}]\}}$
  Add (x, y) to $\mathsf{List}[\mathsf{RF}]$
  return y

Algorithm 12: $\mathcal{O}^{+}_{\mathrm{prim}}(x)$.

if $\exists y, (x, y) \in \mathsf{List}[P]$ **then**
  return y
else
  $y \leftarrow \mathsf{RF}(x)$
Add (x, y) to $\mathsf{List}[P]$
return y

Algorithm 13: $\mathcal{O}^{-}_{\mathrm{prim}}(y)$.

if $\exists x, (x, y) \in \mathsf{List}[P]$ **then**
  return x
else if $\mathbf{bad} = 1$ **then**
  $\boxed{\boxed{\textbf{return } \perp}}$
else
  $\mathsf{List}[P] \leftarrow \mathsf{List}[P] \cup \mathsf{List}[\mathsf{RF}]$
  if $\exists x, (x, y) \in \mathsf{List}[P]$ **then**
    return x
  else
    $x \xleftarrow{\$} \{0,1\}^n \setminus \{x' | (x', y') \in \mathsf{List}[P]\}$
    Add (x, y) to $\mathsf{List}[P]$
    return y

Fig. 9: Oracles in Games 2 and 3. The single-boxed part is executed in Game 2 but not in Game 3. The double-boxed part is executed in Game 3 but not in Game 2. $\mathcal{O}^{+}_{\mathrm{prim}}(x)$ and $\mathcal{O}^{-}_{\mathrm{prim}}(y)$ are primitive oracles corresponding to $P(x)$ (as a primitive oracle) and $P^{-1}(y)$ in the real game.

we have

$$\Pr[\text{Game1} = 1] = \Pr[\text{Game2} = 1].$$

Game 3.
In this game, the construction and primitive oracles are still $\mathcal{O}_{\text{const}}$ and $(\mathcal{O}^{+}_{\text{prim}}, \mathcal{O}^{-}_{\text{prim}})$ in Fig. 9, respectively, but now the double-boxed part is executed while the single-boxed part is not. Games 2 and 3 are identical until the flag **bad** is set, and

$$\Pr[\text{Game2} = 1] - \Pr[\text{Game3} = 1] \leq \Pr[\textbf{bad} \text{ is set to } 1] \leq \frac{(q_c + q_p)^2}{2^n}$$

holds.

Game 4.
This is the ideal game of public indifferentiability with adversary $\mathcal{A}$ and simulator $\mathcal{S}$. We see that $(\mathsf{RF}, \mathcal{S}^{\mathsf{RF},\mathsf{Rev}[\mathsf{RF}]})$ is completely indistinguishable from $(\mathcal{O}_{\text{const}}, (\mathcal{O}^{+}_{\text{prim}}, \mathcal{O}^{-}_{\text{prim}}))$ in Game 3 and

$$\Pr[\text{Game3} = 1] = \Pr[\text{Game4} = 1]$$

holds.

Due to the above arguments,

$$\mathbf{Adv}^{\text{pub-indiff}}_{P,\mathsf{RF},\mathcal{S}}(\mathcal{A}) = \Pr[\text{Game1} = 1] - \Pr[\text{Game4} = 1] \leq \frac{(q_c + q_p)^2}{2^n}$$

follows.

□

H On Reduction from wh$\widetilde{\text{SPRP}}$ to FIL-whPRF

Here explain the details that a VIL tweakable ideally random permutation can be built from a FIL random function f in a secure manner (w.r.t. weak public indifferentiability), by the 6-round Feistel construction of which round functions are built by the sponge construction using f as an underlying function. (In particular, by Theorem 2, it follows that wh$\widetilde{\text{SPRP}}$-secure enciphering schemes can be built from whPRF-secure FIL/FOL keyed functions.)

Let $\{0,1\}^{2*}$ denote the set of all the bit strings of which lengths are a multiple of 2, and $\mathbf{IV}$ be a finite set. Let $F : \mathbf{IV} \times \mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0} \times \{0,1\}^* \to \{0,1\}^\infty$ be a function such that $\mathrm{len}(F(IV, i, m, X)) = m$.

Now, let $\Phi'_6 : \mathbf{IV} \times \{0,1\}^{2*} \to \{0,1\}^{2*}$ be a function such that, for each IV and $M_L, M_R \in \{0,1\}^n$, $\Phi'_6(IV, M_L||M_R)$ is the output of the $2n$-bit 6-round Feistel construction of which i-th round function is $F(IV, i, n, \cdot)$. That is, $\Phi'_6(IV, M_L||M_R)$ is computed as follows:

1. $x_0 \leftarrow M_R$ and $x_1 \leftarrow M_L$
2. For $i = 2, 3, \ldots, 6$, $x_i \leftarrow F(IV, i-1, n, x_{i-1}) \oplus x_{i-2}$
3. $\Phi'_6(IV, M_L||M_R) \leftarrow x_5||x_6$.

For each fixed IV and n, $\Phi'_6(IV, \cdot)$ is public indifferentiable from a $2n$-bit invertible random permutation when the domain of $\Phi'_6(IV, \cdot)$ is restricted to $2n$ and F is a random function (the primitive oracle is F) [61]. In addition, outputs of $F(IV, i, m, X)$ for different (IV, m) are independent, and thus outputs of Φ'_6 for different IV and different message lengths are independent. Therefore the indifferentiability of Φ'_6 from a VIL ideally random tweakable permutation $E : \mathbf{IV} \times \{0,1\}^{2*} \to \{0,1\}^{2*}$ (when regarding F as a primitive) follows from a simple hybrid arguments on IV and the message length.

Let f be a FIL/FOL random function. When F in Φ'_6 is replaced with Sponge^f (with some appropriate encodings), the resulting scheme is weak public indifferentiable from a VIL ideally random tweakable permutation, where f is regarded as a primitive (by Lemma 2).

I Discussions: Reduction of Pairs and Generic Compositions

This section shows details on reductions of pairs and generic compositions.

I.1 Reduction of Pairs.

Suppose we have two white-box schemess, say, a block cipher $(E, \mathcal{C}_E)$ and an AEAD $(\Pi, \mathcal{C}_\Pi)$. Then the white-box security of the pair $((E, \Pi), (\mathcal{C}_E, \mathcal{C}_\Pi))$ is naturally defined by combining whPRP and whPRI as follows:

1. Independent two keys K and K' are chosen for E and Π.

2. Both of the two black-box oracles (E_K and $(\mathcal{E}_K, \mathcal{D}_K)$ in the real experiment, and P and (F, F^{-1}) in the ideal experiment, where P is a random permutation and F is a random injection) are given to an adversary $\mathcal{A} = (\mathcal{A}_{\text{create}}, \mathcal{A}_{\text{dist}})$.
3. A lifter is given the two implementations $\mathcal{C}_E(K)$ and $\mathcal{C}_\Pi(K')$.
4. A simulator is allowed to make queries to both of the two ideal oracles, i.e., (P, P^{-1}) and (F, F^{-1}).

Similarly we can define the security of pairs of a whPRF and wh$\widetilde{\text{SPRP}}$, two whPRPs, and so on.

In the black-box setting, usually such a pair becomes automatically secure if each of the two schemes are secure: We can bound the advantage of an adversary for pairs by the sum of the advantage for one scheme and the advantage for another. However, this is not the case for our white-box security notions. That is, even if each of the two schemes are secure, it is unclear whether we can deduce the pair is secure.

Still, we can show security reductions in a modular way through weak public indifferentiability: Suppose we have a white-box security reduction from a scheme X_1 to another Y_1 and a reduction from X_2 to Y_2, and assume they are given weak through public indifferentiability (and Theorem 2). That is, assume X_1 (resp., X_2) is weak public indifferentiable from an ideally random object when Y_1 (resp., Y_2) is replaced with an ideally random object. Then, the pair (X_1, X_2) is again weak public indifferentiable. Thus the white-box security of (X_1, X_2) can be reduced to the security of (Y_1, Y_2) as pairs by Theorem 2.

More precisely, the following proposition holds.

Proposition 3. *For $i = 1, 2$, let $\mathcal{S}_i$ be a simulator for weak public indifferentiability of a construction $\mathsf{T}_i^{\mathsf{P}_i}$ from an ideally random object R_i (P_i is the underlying ideal primitive). In addition, let $\mathcal{A}_3$ be an adversary for weak public indifferentiability of $(\mathsf{T}_1^{\mathsf{P}_1}, \mathsf{T}_2^{\mathsf{P}_2})$ from $(\mathsf{R}_1, \mathsf{R}_2)$ that makes at most $q_{const,i}$ queries to T_i of which lengths are at most $\sigma_{const,i}$ in total, and $q_{prim,i}$ queries to P_i of which lengths are at most $\sigma_{prim,i}$ in total. Then there exists an adversary $\mathcal{A}_i$ for weak public indifferentiability of $\mathsf{T}_i^{\mathsf{P}_i}$ for $i = 1, 2$ satisfying*

$$\mathbf{Adv}^{\text{weak-pub-indiff}}_{(\mathsf{T}_1,\mathsf{T}_2),(\mathsf{R}_1,\mathsf{R}_2),\mathcal{S}_3}(\mathcal{A}_3) \leq \mathbf{Adv}^{\text{weak-pub-indiff}}_{\mathsf{T}_1,\mathsf{R}_1,\mathcal{S}_1}(\mathcal{A}_1) + \mathbf{Adv}^{\text{weak-pub-indiff}}_{\mathsf{T}_2,\mathsf{R}_2,\mathcal{S}_2}(\mathcal{A}_2).$$

Here, $\mathcal{S}_3 := (\mathcal{S}_1, \mathcal{S}_2)$. In addition, $\mathcal{A}_1$ and $\mathcal{A}_2$ can be built in such a way that that (1) the number of construction and primitive queries by $\mathcal{A}_i$ ($i = 1, 2$) is at most $q_{const,i}$ and $q_{prim,i}$, respectively, (2) the lengths of queries by $\mathcal{A}_i$ ($i = 1, 2$) to the construction and primitive oracles are at most $\sigma_{const,i}$ and $\sigma_{prim.i}$, respectively, and (3) $\mathcal{A}_1$ does not depend on $\mathcal{S}_1$ and $\mathcal{S}_2$, and $\mathcal{A}_2$ does not depend on $\mathcal{S}_2$.

Proof. Define $\mathcal{A}_1$ and $\mathcal{A}_2$ as follows.

$\underline{\mathcal{A}_1}$: First, $\mathcal{A}_1$ runs $\mathcal{A}_3$, sampling the function P_2 by itself. When $\mathcal{A}_3$ makes a query x to the interfaces corresponding to T_2 or P_2, $\mathcal{A}_1$ responds with $\mathsf{T}_2^{\mathsf{P}_2}(x)$

or $\mathsf{P}_2(x)$, respectively, by using the sampled function. When $\mathcal{A}_3$ makes queries to the interfaces corresponding to T_1 or P_1, $\mathcal{A}_1$ just forwards them to the construction oracle and the primitive oracle given to $\mathcal{A}_1$.

$\underline{\mathcal{A}_2}$: First, $\mathcal{A}_2$ samples the function R_1 by itself and runs $\mathcal{A}_3$. $\mathcal{A}_2$ uses $\mathcal{S}_1$ as a subroutine. When $\mathcal{A}_3$ makes queries to the interfaces corresponding to T_1 or P_1, $\mathcal{A}_2$ respond with the sampled function R_1 or by using $\mathcal{S}_1^{\mathsf{R}_1,\mathsf{Rev}'[\mathsf{R}]}$, respectively. When $\mathcal{A}_3$ makes queries to the interfaces corresponding to T_2 or P_2, $\mathcal{A}_2$ just forwards them to the construction oracle and the primitive oracle given to $\mathcal{A}_2$.

Now we have

$$\begin{aligned}
\mathbf{Adv}^{\text{weak-pub-indiff}}_{(\mathsf{T}_1,\mathsf{T}_2),(\mathsf{R}_1,\mathsf{R}_2),\mathcal{S}_3}(\mathcal{A}_3) &= \Pr\left[1 \leftarrow \mathcal{A}_3^{(\mathsf{T}_1^{\mathsf{P}_1},\mathsf{T}_2^{\mathsf{P}_2}),(\mathsf{P}_1,\mathsf{P}_2)}\right] - \Pr\left[1 \leftarrow \mathcal{A}_3^{(\mathsf{R}_1,\mathsf{R}_2),(\mathcal{S}_1^{\mathsf{R}_1,\mathsf{Rev}'[\mathsf{R}_1]},\mathcal{S}_2^{\mathsf{R}_2,\mathsf{Rev}'[\mathsf{R}_2]})}\right] \\
&= \Pr\left[1 \leftarrow \mathcal{A}_3^{(\mathsf{T}_1^{\mathsf{P}_1},\mathsf{T}_2^{\mathsf{P}_2}),(\mathsf{P}_1,\mathsf{P}_2)}\right] - \Pr\left[1 \leftarrow \mathcal{A}_3^{(\mathsf{R}_1,\mathsf{T}_2^{\mathsf{P}_2}),(\mathcal{S}_1^{\mathsf{R}_1,\mathsf{Rev}'[\mathsf{R}_1]},\mathsf{P}_2)}\right] \\
&\quad + \Pr\left[1 \leftarrow \mathcal{A}_3^{(\mathsf{R}_1,\mathsf{T}_2^{\mathsf{P}_2}),(\mathcal{S}_1^{\mathsf{R}_1,\mathsf{Rev}'[\mathsf{R}_1]},\mathsf{P}_2)}\right] - \Pr\left[1 \leftarrow \mathcal{A}_3^{(\mathsf{R}_1,\mathsf{R}_2),(\mathcal{S}_1^{\mathsf{R}_1,\mathsf{Rev}'[\mathsf{R}_1]},\mathcal{S}_2^{\mathsf{R}_2,\mathsf{Rev}'[\mathsf{R}_2]})}\right] \\
&= \Pr\left[1 \leftarrow \mathcal{A}_1^{\mathsf{T}_1^{\mathsf{P}_1},\mathsf{P}_1}\right] - \Pr\left[1 \leftarrow \mathcal{A}_1^{\mathsf{R}_1,\mathcal{S}_1^{\mathsf{R}_1,\mathsf{Rev}'[\mathsf{R}_1]}}\right] \\
&\quad + \Pr\left[1 \leftarrow \mathcal{A}_2^{\mathsf{T}_2^{\mathsf{P}_2},\mathsf{P}_2}\right] - \Pr\left[1 \leftarrow \mathcal{A}_2^{\mathsf{R}_2,\mathcal{S}_2^{\mathsf{R}_2,\mathsf{Rev}'[\mathsf{R}_2]}}\right] \\
&= \mathbf{Adv}^{\text{weak-pub-indiff}}_{\mathsf{T}_1,\mathsf{R}_1,\mathcal{S}_1}(\mathcal{A}_1) + \mathbf{Adv}^{\text{weak-pub-indiff}}_{\mathsf{T}_2,\mathsf{R}_2,\mathcal{S}_2}(\mathcal{A}_2),
\end{aligned}$$

and $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the desired properties. □

I.2 On Generic Compositions for AEADs.

After seeing our new security notions, a natural question is whether it is possible to build a whPRI-secure AEAD by generic compositions like [69] or not.

Indeed, it seems possible to show *public indifferentiability* of SIV when the tag-generation function is a completely random function and the conventional encryption scheme is a family of VIL random permutations in the single-key setting (i.e., the keys for tag-generation and conventional encryption are identical)[34] If this is the case, by Theorem 2 we can reduce whPRI security of SIV to (the pair of) whPRF security and $\widetilde{\text{whSPRP}}$ security of the underlying schemes. This is in contrast to the result by Barbosa and Farshim on the impossibility of *indifferentiability* of generic compositions, especially SIV.

Nevertheless, such compositions would be of less interest in practice because it is unclear how to build efficient and secure $\widetilde{\text{whSPRP}}$-secure (or, public indifferentiable) enciphering scheme. As mentioned in Section 6, we can rely on

[34] The differentiators in Section 2 does not seem to work in the single-key setting if simulators can observe construction queries. See the paragraph above Remark 3 for details.

an 6-round (balanced) Feistel-based construction to use it to realize wh$\widetilde{\text{SPRP}}$-secure enciphering scheme, but the resulting AEAD scheme would be much less efficient than the scheme based on 3-round Feistel (see Section 2 for details).

Moreover, it seems hard to reduce the whPRI-security of an AEAD scheme by generic composition to the whIND$-CPA-security of the underlying conventional encryption scheme (and the whPRF-security of the underlying keyed function). Regardless of whether we rely on public indifferentiability, what we have to show is to reduce a security notion of AEAD against CCAs to a security notion of underlying encryption scheme against CPAs. This essentially means we have to prove that adversaries (in the real world) cannot notice if we replace the decryption oracle with an oracle that, on an input (N, A, C), (1) returns the original message M if (N, A, M) has already been queried to the encryption oracle and the response was C, and (2) returns $\perp$ otherwise. However, showing such a claim is impossible in the white-box setting because a leakage may contain some information on a valid ciphertext that has not been returned from the encryption oracle.

Instead of generic compositions, the next section directly shows the security of SIV when the underlying conventional encryption scheme is CTR.

J Proof of Proposition 2

Proof (of Proposition 2). We show the proposition by introducing a sequence of games and utilizing the code-based game-playing technique [10].

Game 1. This is the real game for random-IV CTR built on ρ (denoted by $\mathcal{E}^{\rho}_{\mathrm{rnd}}$). We assume outputs of ρ is sampled via lazy sampling.

Game 2. In this game we introduce an oracle $\$'(\cdot)$ that runs as in Algorithm 14. (The output distribution of $\$'(\cdot)$ is the same as that of $\$(\cdot)$ but we do not give $\$'(\cdot)$ to adversaries as the construction oracle.) The sampling of the primitive oracle ρ is changed depending on $\$'(\cdot)$ as in Algorithm 15. The construction oracle (random-IV counter mode) is also modified as in Algorithm 16, where the resulting construction is denoted by ${\mathcal{E}'_{\mathrm{rnd}}}^{\$',\rho}$. The list of queries to ρ (resp., $\$'(\cdot)$) made so far and the responses is denoted by $\mathsf{List}[\rho]$ (resp., $\mathsf{List}[\$'(\cdot)]$).

Since the output distribution of $\$'$ is completely random, the output distributions of the construction oracle and the primitive oracle are the same as those in Game 1. Hence

$$\Pr\left[\mathrm{Game1} = 1\right] = \Pr\left[\mathrm{Game2} = 1\right] \tag{8}$$

holds.

Game 3. This game is the same as Game 2 only except that Line 4 of Algorithm 16 is executed. Games 2 and 3 are identical until the flag **bad2** is set, and thus

$$\Pr\left[\mathrm{Game2} = 1\right] - \Pr\left[\mathrm{Game3} = 1\right] \leq \Pr\left[\mathbf{bad2}\right] \tag{9}$$

Algorithm 14: Sampling of $\$'(M)$.

1: $IV \xleftarrow{\$} \{0,1\}^m$, $C \xleftarrow{\$} \{0,1\}^{\mathrm{len}(M)}$
2: **if** $\exists (M', (IV', C'))$ $\in$ $\mathsf{List}[\$'(\cdot)]$ s.t. $IV + i = IV' + j$ for some $0 \leq i < \lceil \mathrm{len}(M)/n \rceil$ and $0 \leq j < \lceil \mathrm{len}(M')/n \rceil$, or $\exists (X, y) \in \mathsf{List}[\rho]$ s.t. $x = IV + i$ for some $0 \leq i < \lceil \mathrm{len}(M)/n \rceil$ **then**
3: **bad2** $\leftarrow 1$
4: Add $(M, (IV, C))$ to $\mathsf{List}[\$'(\cdot)]$
5: **return** (IV, C)

Fig. 10: Sampling of the oracle $\$'(\cdot)$, of which output distribution is the same as that of the ideal construction oracle $\$(\cdot)$.

holds. In addition, we have

$$\Pr[\mathbf{bad2}] \leq \frac{\sigma^2}{2^m} + \frac{\sigma(\sigma + q_p)}{2^m} \tag{10}$$

because the length of queries to $\$'(\cdot)$ is at most σ blocks in total and the size of $\mathsf{List}[\rho]$ does not exceed $(\sigma + q_p)$ during the games[35]. Therefore

$$\Pr\left[\mathrm{Game2} = 1\right] - \Pr\left[\mathrm{Game3} = 1\right] \leq \frac{\sigma^2}{2^m} + \frac{\sigma(\sigma + q_p)}{2^m} \tag{11}$$

holds.

In addition, we can show the following claim.

Claim. Suppose a message M is queried to ${\mathcal{E}'_{\mathrm{rnd}}}^{\$',\rho}$. If the flag **bad2** is not set while running Algorithm 16, the value $(IV, C'_1 || \cdots || C'_\ell)$ returned at line 9 of Algorithm 16 is equal to the value (IV, C) at line 2 (in both of Game 2 and Game 3). In particular, ${\mathcal{E}'_{\mathrm{rnd}}}^{\$',\rho}$ in Game 3 is perfectly indistinguishable from $\$'(\cdot)$.

Proof. Since the flag **bad2** is not set, the conditions at line 2 of Algorithm 14 are not satisfied when $\$'(\cdot)$ is called at line 2 of Algorithm 16. This means, at line 6 of Algorithm 16, the value $(IV + i - 1)$ is a fresh query to ρ, and the value $\rho(IV + i - 1)$ is sampled at line 7 or line 9 of Algorithm 15 by using the

[35] The first term $\frac{\sigma^2}{2^m}$ corresponds to the first condition "$\exists (M', (IV', C)) \in \mathsf{List}[\$'(\cdot)]$..." at line 2 of the Algorithm 16 and the second term corresponds to the second condition "or, $\exists (X, y) \in \mathsf{List}[\rho]$...".

Algorithm 15: Sampling of $\rho(X)$ depending on $\$'(\cdot)$.

if $\exists y$ s.t. $(X, y) \in \mathsf{List}[\rho]$ **then**
  return y
if $\exists (M, (IV, C)) \in \mathsf{List}[\$'(\cdot)]$ s.t. $X = IV + i - 1$ for some $1 \leq i \leq \lceil \mathrm{len}(M)/n \rceil$ **then**
  $(M_1, M_2, \ldots, M_{\lceil \mathrm{len}(M)/n \rceil}) \xleftarrow{n} M$
  $(C_1, C_2, \ldots, C_{\lceil \mathrm{len}(M)/n \rceil}) \xleftarrow{n} C$
  if $i < \lceil \mathrm{len}(M)/n \rceil$ **then**
    $y \leftarrow C_i \oplus M_i$
  else
    $y' \xleftarrow{\$} \{0,1\}^{n-(\mathrm{len}(M) \bmod n)}$, $y \leftarrow (C_i \oplus M_i) || y'$
  \\ `If there exist two or more candidates for the pair` $(M, (IV, C))$ `in` $\mathsf{List}[\$'(\cdot)]$`, we take the smallest one with respect to a lexicographical order.`
else
  $y \xleftarrow{\$} \{0,1\}^n$
Add (X, y) to $\mathsf{List}[\rho]$
return y

Algorithm 16: $\mathcal{E}'_{\mathrm{rnd}}{}^{\$',\rho}(M)$

1: $(M_1, M_2, \ldots, M_\ell) \xleftarrow{n} M$
2: $(IV, C) \leftarrow \$'(M)$
3: **if** **bad2 = 1 then**
4: $\quad$ **return** (IV, C)
5: **for** $i = 1$ **to** ℓ **do**
6: $\quad y \leftarrow \rho(IV + i - 1)$,
7: $\quad y' \leftarrow$ (the upper $\mathrm{len}(M_i)$ bits of y),
8: $\quad C'_i \leftarrow M_i \oplus y'$
9: **return** $(IV, C'_1 \,\|\, C'_2 \,\|\, \cdots \,\|\, C'_\ell)$

Fig. 11: The construction oracle in Game 2 and 3. The boxed part is executed only in Game 3.

tuple $(M, (IV, C))$ (but not another tuple $(M', (IV', C'))$ in $\mathsf{List}[\$'(\cdot)]$). Hence the claim follows. □

Game 4. In this game, the construction oracle is $\$'(\cdot)$. The primitive oracle is not changed from Game 3 (i.e., Algorithm 15). By the claim we proved below Eq. (11), we have

$$\Pr[\text{Game3} = 1] - \Pr[\text{Game4} = 1] = 0. \tag{12}$$

Game 5. This is the ideal game. We define a simulator $\mathcal{S}$ running as follows.

1. When an adversary makes a query X to the interface corresponding to ρ, query to the revealing the interface to get the list $\mathsf{List}[\$(\cdot)]$ storing the queries made so far to $\$(\cdot)$ and the responses.
2. Sample and return the value $\rho(X)$ as in Algorithm 15, replacing $\mathsf{List}[\$'(\cdot)]$ with $\mathsf{List}[\$(\cdot)]$.

Then obviously

$$\Pr[\text{Game4} = 1] - \Pr[\text{Game5} = 1] = 0 \tag{13}$$

holds.

From Eq. (8)-(13),

$$\mathbf{Adv}^{\text{pub-indiff}}_{\mathcal{E}^{\rho}_{\mathrm{rnd}},\$(\cdot),\mathcal{S}} = \Pr[\text{Game1} = 1] - \Pr[\text{Game5} = 1] \leq \frac{\sigma^2}{2^m} + \frac{\sigma(\sigma + q_p)}{2^m}$$

follows, and the simulator satisfies the desired properties.

K Security Proof for SIV+CTR

This section proves Theorem 3, which is restated in Theorem 6 below. In what follows, we just write "random injection" to refer to fixed-key random injection.

Recall that $\mathsf{CTR}^{\rho}(IV, M)$ denotes the encryption function of the counter mode with the underlying keyed function being replaced with a random function $\rho : \{0,1\}^{\tau} \to \{0,1\}^{n}$. In addition, $\Pi = (\mathcal{E}^{f,\rho}, \mathcal{D}^{f,\rho})$ is the SIV construction of which keyed function for tag-generation is replaced with a random function $f : \mathbf{N} \times \mathbf{A} \times \{0,1\}^{*} \to \{0,1\}^{\tau}$ and conventional encryption scheme is replaced with CTR^{ρ}. Let $\mathsf{enc} : \mathbf{N} \times \mathbf{A} \times \{0,1\}^{*} \to \{0,1\}^{*}$ be an arbitrary encoding function that encodes each tuple (N, A, X) into a single bit string in a uniquely decodable manner. We let $\mathrm{len}(N, A, X) := \mathrm{len}(\mathsf{enc}(N, A, X))$ and call $\lceil \mathrm{len}(N, A, M)/n \rceil$ the block length of (N, A, X).

Theorem 6 (Theorem 3, restated). *Let $F : \mathbf{N} \times \mathbf{A} \times \{0,1\}^{*} \to \{0,1\}^{*}$ be a fixed-key random injection with message space $\{0,1\}^{*}$ and such that $\mathrm{len}(F(N, A, M)) = \mathrm{len}(M) + \tau$. There exists a simulator $\mathcal{S}$ for public indifferentiability of Π from $F^{\pm}$ such that the number of queries by $\mathcal{S}$ to the construction oracle is at most q_f and the block lengths of the queries are at most σ_f in total, and*

$$\mathbf{Adv}^{\text{pub-indiff}}_{\Pi, F^{\pm}, \mathcal{S}}(\mathcal{A}) \leq \frac{(\sigma_c + \sigma_f)^2}{2^{\tau}} + \frac{(\sigma_c + \sigma_f)(q_{\rho} + \sigma_c)}{2^{\tau}} + \frac{3q_c}{2^{\tau}} + \frac{(q_c + q_f)^2}{2^{\tau}} \quad (14)$$

holds for any adversary $\mathcal{A}$ of which computational resources are as follows: To the construction oracle, $\mathcal{A}$ makes at most q_c queries of which block lengths are at most σ_c in total. To the first primitive oracle (corresponding to f), $\mathcal{A}$ makes at most q_f queries of which block lengths are at most σ_f in total. To the second primitive oracle (corresponding to ρ), $\mathcal{A}$ makes at most q_{ρ} queries. Here, we assume $(q_c + q_f) \leq 2^{\tau - 1}$.

Notation for the list of input-output pairs of functions. Suppose that output values of a function g are sampled on the fly when inputs are queried to g (e.g., by lazy sampling) rather than determined all at once at the beginning of games. For such a function g, by $\mathsf{List}[g]$ we denote the list of input-output pairs $(x_0, g(x_0)), (x_1, g(x_1)), \ldots$ of g that have been defined so far.

For a bit string x, by $[x]_{\mathsf{Upper}(m)}$ and $[x]_{\mathsf{Lower}(m)}$ we denote the upper and lower m bits of x, respectively. By abuse of notations we denote $\min_{1 \leq \alpha \leq n} \{\alpha \equiv x \bmod n\}$ by $x \bmod n$.

In what follows We prove Theorem 6 by introducing games and utilizing the code-based game-playing technique [10],

K.1 Game 1

This is the real game for Π.

Algorithm 17: $\rho(x)$, lazy sampling

1: **if** $\exists y, (x, y) \in \mathsf{List}[\rho]$ **then**
2: **return** y
3: **else**
4: $y \xleftarrow{\$} \{0,1\}^n$, add (x, y) to $\mathsf{List}[\rho]$
5: **return** y

Algorithm 18: $f(N, A, M)$, lazy sampling

1: **if** $\exists T, ((N, A, M), T) \in \mathsf{List}[f]$ **then**
2: **return** T
3: **else**
4: $T \xleftarrow{\$} \{0,1\}^\tau$, add $((N, A, M), T)$ to $\mathsf{List}[f]$
5: **return** T

K.2 Game 2

Here we modify the oracles ρ and f so that outputs are sampled by lazy sampling. As mentioned before, we maintain lists $\mathsf{List}[\rho]$ and $\mathsf{List}[f]$ which record "already-defined" input values of ρ and f, respectively. See also Algorithm 17 and 18.

Apparently, the following proposition holds.

Proposition 4.

$$\Pr[\text{Game } 1 = 1] - \Pr[\text{Game } 2 = 1] = 0.$$

K.3 Game 3

Here we introduce a random function $F'(N, A, M)$ of which domain and codomain are the same as the ideal encryption oracle F and that satisfies $\mathrm{len}(F'(N, A, M)) = \mathrm{len}(M) + \tau$ for each input (N, A, M). See Fig. 12 for details. We assume the value $F'(N, A, M)$ is represented as a concatenation $F'_0(N, A, M) \| \cdots \| F'_{\lceil \mathrm{len}(M)/n \rceil}(N, A, M)$, where $F'_0(N, A, M) \in \{0,1\}^\tau$, $F'_i(N, A, M) \in \{0,1\}^n$ for $i = 1, \ldots, \lceil \mathrm{len}(M)/n \rceil -$

Algorithm 19: $F'(N, A, M)$, lazy sampling

1: **if** $((N, A, M), C) \in \mathsf{List}[F']$ **then**
2: **return** C
3: **else**
4: $C \xleftarrow{\$} \{0,1\}^{\mathrm{len}(M)+\tau}$
5: Add $((N, A, M), C)$ to $\mathsf{List}[F']$
6: **return** C

Fig. 12: Lazy sampling of F'.

Algorithm 20: Sampling of $f(N, A, M)$ depending on F'.

1: **if** $((N, A, M), T) \in \mathsf{List}[f]$ **then**
2: **return** T
3: **else**
4: $T \leftarrow F'_0(N, A, M)$
5: Add $((N, A, M), T)$ to $\mathsf{List}[f]$
6: **return** T

1, and $F'_{\lceil \mathrm{len}(M)/n \rceil}(N, A, M) \in \{0,1\}^{\mathrm{len}(M) \bmod n}$. As mentioned before, we assume $\mathsf{List}[F'_i(N, A, \cdot)]$ is constructed and automatically updated when $\mathsf{List}[F'(N, A, \cdot)]$ is updated, for each (N, A).

Sampling of f and ρ depending on F'. From this game, we assume outputs of f and ρ are sampled depending on F' so that $\mathcal{E}^{f,\rho}(N, A, M)$ will match $F'(N, A, M)$ as much as possible.

The sampling of f for a fresh query (N, A, M) is performed by first querying (N, A, M) to F' and setting $f(N, A, M) \leftarrow F'_0(N, A, M)$ (see also Algorithm. 20).

The sampling of $\rho(X)$ for a fresh query X is performed roughly as follows. See Algorithm 21 for a precise description.

1. If $\mathcal{A}$ seems to compute the i-th ciphertext block C_i that corresponds to an input (N, A, M) which has already been queried to F' before and $i <$

$\lceil \mathrm{len}(M)/n \rceil$, $\rho(X)$ is set to be $F'_i(N, A, M) \oplus M_i$. (Here, M_i is the i-th message block.) If $i = \lceil \mathrm{len}(M)/n \rceil$, $\rho(X)$ is set to be $(F'_i(N, A, M) \oplus M_i)||y'$, where y' is an $(n - (M \bmod n))$-bit random string.
2. Otherwise $\rho(X)$ is sampled as before.

Algorithm 21: Sampling of $\rho(X)$ depending on F'.

if $\exists y$ s.t. $(X, y) \in \mathsf{List}[\rho]$ **then**
  return y
else if $\exists((N, A, M), IV||C') \in \mathsf{List}[F']$ s.t. $X = IV + i - 1$ for some $1 \leq i \leq \lceil \mathrm{len}(M)/n \rceil$ (here, $IV = F'_0(N, A, M)$) **then**
  $(M_1, M_2, \ldots, M_{\lceil \mathrm{len}(M)/n \rceil}) \xleftarrow{n} M$
  if $i < \lceil \mathrm{len}(M)/n \rceil$ **then**
    $y \leftarrow F'_i(N, A, M) \oplus M_i$
  else
    $y' \xleftarrow{\$} \{0,1\}^{n-(\mathrm{len}(M) \bmod n)}$, $y \leftarrow (F'_i(N, A, M) \oplus M_i)||y'$
  \\ `If there exist two or more candidates for` (N, A, M)`, we take the smallest one with respect to a lexicographical order.`
else
  $y \xleftarrow{\$} \{0,1\}^n$
Add (X, y) to $\mathsf{List}[\rho]$
return y

Fig. 13: Sampling of $\rho(X)$ depending on F'. Note that a fresh query to F' is never made in this algorithm.

Description of Game 3. Game 3 is defined in the same way as Game 2, except that the sampling of f and ρ are done as in Algorithm 20 and Algorithm 21. The encryption and decryption oracles given to $\mathcal{A}$ are still $\mathcal{E}^{f,\rho}$ and $\mathcal{D}^{f,\rho}$.

We define an event **bad3** by **bad3** := **bad3**$\mathcal{E}$ $\vee$ **bad3**$\mathcal{D}$, where **bad3**$\mathcal{E}$ and **bad3**$\mathcal{D}$ are defined as follows.

bad3$\mathcal{E}$: Just after the value $\mathcal{E}^{f,\rho}(N, A, M)$ is computed for a query (N, A, M) to $\mathcal{E}^{f,\rho}$, F' is already defined on (N, A, M) but $F'(N, A, M) \neq \mathcal{E}^{f,\rho}(N, A, M)$.

bad3$\mathcal{D}$: When (N, A, C) is queried to $\mathcal{D}^{f,\rho}$,
(a) $((N, A, M), C) \notin \mathsf{List}[F']$ for any M but $\mathcal{D}^{f,\rho}$ returns $M' \neq \perp$, or
(b) $((N, A, M), C) \in \mathsf{List}[F']$ for a unique M but $\mathcal{D}^{f,\rho}$ returns $M' \neq M$ or $\perp$.
(c) There exist $M \neq M'$ such that $((N, A, M), C), ((N, A, M'), C) \in \mathsf{List}[F']$.

The output distributions of f and ρ are unchanged from Game 2 (this is because F' is a random function and each tuple (i, N, A, M) is used at most once in Algorithm 21). Hence the following proposition holds.

Proposition 5.

$$\Pr[\text{Game2} = 1] - \Pr[\text{Game3} = 1] = 0.$$

Next, we show the following proposition.

Proposition 6. *It holds that*

$$\Pr[\mathbf{bad3}] \leq \frac{(\sigma_c + \sigma_f)^2}{2^\tau} + \frac{(\sigma_c + \sigma_f)(q_\rho + \sigma_c)}{2^\tau} + \frac{q_c}{2^\tau}.$$

In what follows, we often use the property $\Pr[A \wedge B] \leq \Pr[A|B]$ for events A and B without any notice. To prove Proposition 6, we additionally introduce the following bad events.

bad3$_{F'}$: When a fresh value (N, A, M) is queried to F' and $IV := F'_0(N, A, M)$ is sampled, the value $(IV + i - 1)$ for some $1 \leq i \leq \lceil \mathrm{len}(M)/n \rceil$ happens to collide with $(IV' + j - 1)$ for some existing $((N', A', M'), IV') \in \mathsf{List}[F'_0]$ and $1 \leq j \leq \lceil \mathrm{len}(M')/n \rceil$.

bad3$_\rho$: When a fresh value (N, A, M) is queried to F' and $IV := F'_0(N, A, M)$ is sampled, the value $(IV + i - 1)$ for some $1 \leq i \leq \lceil \mathrm{len}(M)/n \rceil$ happens to collide with x such that $(x, y) \in \mathsf{List}[\rho]$ for some y.

We say that the value $\mathcal{E}^{f,\rho}(N, A, M)$ is defined if all the values of ρ and f required to compute $\mathcal{E}^{f,\rho}(N, A, M)$ are already sampled and defined. Then the following lemma holds.

Lemma 3. *If the value $f(N, A, M)$ (resp., $F'(N, A, M)$) is already defined, then the value $F'(N, A, M)$ (resp., $f(N, A, M)$) is also already defined and*

$$F'_0(N, A, M) = f(N, A, M)$$

holds. In addition, if **bad3**$_{F'} \vee$ **bad3**$_\rho$ *does not happen and the value $\mathcal{E}^{f,\rho}(N, A, M)$ is already defined, then the value $F'(N, A, M)$ is also already defined and*

$$F'(N, A, M) = \mathcal{E}^{f,\rho}(N, A, M)$$

holds. In particular, the event **bad3**$\mathcal{E}$ *does not occur if* **bad3**$_{F'} \vee$ **bad3**$_\rho$ *does not happen.*

Proof. We first show the statement for F'_0 and f, and then for F' and $\mathcal{E}^{f,\rho}$.

F'_0 and f. Suppose the value $F'(N, A, M)$ is already defined. Due to the definition of the samplings in Algorithm 20 and Algorithm 21, a fresh query (N, A, M) to F' is made only when (N, A, M) is queried to f as a fresh value. Hence $f(N, A, M)$ is already defined iff $F'_0(N, A, M)$ is already defined, and $f(N, A, M) = F'_0(N, A, M)$ holds as long as they are defined.

F' and $\mathcal{E}^{f,\rho}$ Next, assume the value $\mathcal{E}^{f,\rho}(N, A, M)$ is defined for a tuple (N, A, M) and $\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho}$ does not hold. Then, by the claim on F'_0 and f, the value $F'(N, A, M)$ is already defined and $F'_0(N, A, M) = f(N, A, M)$ holds.

Let us fix $i \in \{1, \ldots, \lceil \mathrm{len}(M)/n \rceil\}$ arbitrarily and let $IV := F'_0(N, A, M)$. Since the value $\mathcal{E}^{f,\rho}(N, A, M)$ is already defined, there exists a pair $(IV+i-1, y)$ in $\mathsf{List}[\rho]$.

Now we can deduce that the value y was sampled at line 6 or 8 of Algorithm 21 with the tuple (i, N, A, M) (but not another tuple (j, N', A', M')): If z had been sampled at line 6 or 8 but by using another tuple (j, N', A', M'), then $F'_0(N', A', M') + j - 1 = F_0(N, A, M) + i - 1$ would hold and $\mathbf{bad3}_{F'}$ would happen. However, this contradicts our assumption that $\mathbf{bad3}_{F'}$ does not happen. In addition, if y had been sampled at line 11 of Algorithm 21, the value $F'(N, A, M)$ would have been sampled after the sampling of y, and $\mathbf{bad3}_{\rho}$ would happen (when $F'(N, A, M)$ is sampled). However, this again contradicts our assumption that $\mathbf{bad3}_{\rho}$ does not happen.

The above argument shows the i-th ciphertext block of $\mathcal{E}^{f,\rho}(N, A, M)$ is equal to $F'_i(N, A, M)$ for all $i \in \{1, \ldots, \lceil \mathrm{len}(M)/n \rceil\}$. Therefore we have $\mathcal{E}^{f,\rho}(N, A, M) = F'(N, A, M)$. □

Lemma 4. $\Pr[\mathbf{bad3}_{F'}]$ *and* $\Pr[\mathbf{bad3}_{\rho}]$ *can be upper bounded as*

$$\Pr[\mathbf{bad3}_{F'}] \leq \frac{(\sigma_c + \sigma_f)^2}{2^{\tau}}, \tag{15}$$

$$\Pr[\mathbf{bad3}_{\rho}] \leq \frac{(\sigma_c + \sigma_f)(q_{\rho} + \sigma_c)}{2^{\tau}}. \tag{16}$$

Proof. We begin with showing the first inequality. f is called once at each query to $\mathcal{E}^{f,\rho}$ and $\mathcal{D}^{f,\rho}$. In addition, F' is called at most once at each query to f. Thus the block lengths of fresh queries made to F' are at most $(\sigma_c + \sigma_f)$ in total (note that a fresh query is not made while sampling output values of ρ). Since F'_0 is a random function of output length τ, the first inequality follows the definition of $\mathbf{bad3}_{F'}$.

Next, we show the second inequality. At each query of block length ℓ to $\mathcal{E}^{f,\rho}$ and $\mathcal{D}^{f,\rho}$, ρ is called at most ℓ times. In particular, the size of $\mathsf{List}[\rho]$ is at most $(q_{\rho} + \sigma_c)$ in this game. Since the block lengths of fresh queries made to F' are at most $(\sigma_c + \sigma_f)$ in total and F' is a random function, the second inequality follows from the definition of $\mathbf{bad3}_{\rho}$. □

Lemma 5. *It holds that*

$$\Pr[\mathbf{bad3}_{\mathcal{D}} \wedge \neg(\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho})] \leq \frac{q_c}{2^{\tau}}.$$

Proof. In what follows, we consider a situation where a tuple (N, A, C) is queried to $\mathcal{D}^{f,\rho}$, and study the probability that **bad3**$\mathcal{D}$-(a), **bad3**$\mathcal{D}$-(b), or **bad3**$\mathcal{D}$-(c) happen.

Recall that the decryption $\mathcal{D}^{f,\rho}$ on (N, A, C) proceeds as follows.

1. Separate C into IV and C', where IV is the upper τ-bits and C' is the rest of C.
2. Compute $M' \leftarrow \mathrm{CTR}^{\rho}(IV, C')$.
3. Compute $T \leftarrow f(N, A, M')$.
4. If $T = IV$, return M'. Otherwise, return $\perp$.

In addition, from Lemma 3, if either of the values $f(N, A, M)$ or $F'(N, A, M)$ is defined, then the other one is also defined and $f(N, A, M) = F'_0(N, A, M)$ holds.

On $\Pr[\mathbf{bad3}\mathcal{D}\text{-(a)}]$.
Assume $\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho}$ does not happen in the game. If $((N, A, M), C) = ((N, A, M), IV||C') \notin \mathsf{List}[F']$ for any M before querying (N, A, C) to $\mathcal{D}^{f,\rho}$, then the value $f(N, A, M')$ (recall $M' = \mathrm{CTR}^{\rho}(IV, C')$) is not defined yet before the query (by Lemma 3). In addition, (N, A, M') is queried to F' as a fresh query in computing $f(N, A, M')$, and $f(N, A, M') = F'_0(N, A, M')$ holds after the computation of $f(N, A, M')$ (by Lemma 3). Thus, at each query to $\mathcal{D}^{f,\rho}$, the probability that $f(N, A, M')(= F'_0(N, A, M'))$ happens to collide with IV is at most $1/2^{\tau}$. Therefore we have

$$\Pr[\mathbf{bad3}\mathcal{D}\text{-(a)}] \leq \frac{q_c}{2^{\tau}}. \tag{17}$$

On $\Pr[\mathbf{bad3}\mathcal{D}\text{-(b)} \wedge \neg(\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho})]$.
Assume $\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho}$ does not happen in the game. Suppose there exists a unique $((N, A, M), C) \in \mathsf{List}[F']$ for some M before $(N, A, C) = (N, A, IV||C')$ is queried to $\mathcal{D}^{f,\rho}$. Then, the value $F'(N, A, M)$ is defined and $f(N, A, M) = F'_0(N, A, M) = IV$ holds (by Lemma 3).

Now we observe that, for all $1 \leq i \leq \lceil \mathrm{len}(M)/n \rceil$, the value $\rho(IV + i - 1)$ is defined in Algorithm 21 as $\rho(IV + i - 1) := F'_i(N, A, M) \oplus M_i$ at line 6, or $\rho(IV + i - 1) := (F'_i(N, A, M) \oplus M_i)||y'$ with some y' at line 8: If the value $\rho(IV + i - 1)$ were sampled at line 11 of Algorithm 21, the value $F'(N, A, M)$ would have been sampled after the sampling of $\rho(IV + i - 1)$ and $\mathbf{bad3}_{\rho}$ would happen (when $F'(N, A, M)$ is sampled). However, this contradicts the assumption $\neg(\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho})$. In addition, if the value were sampled at line 6 or 8 of Algorithm 21 but with a different tuple (j, N', A', M') other than (i, N, A, M), then $F'_0(N', A', M') + j - 1 = F'_0(N, A, M) + i - 1$ holds and $\mathbf{bad3}_{F'}$ would happen. However, this contradicts the assumption $\neg(\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho})$.

The above discussion shows $\mathrm{CTR}^{\rho}(IV, C') = M$, which implies $\mathcal{D}^{f,\rho}(N, A, C) = M$. Hence we have

$$\Pr[\mathbf{bad3}\mathcal{D}\text{-(b)} \wedge \neg(\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho})] = 0. \tag{18}$$

On $\Pr[\mathbf{bad3}\mathcal{D}\text{-(c)} \wedge \neg(\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho})]$.
Suppose **bad3**$\mathcal{D}$-(c) happens, i.e., there exist $M \neq M'$ such that $((N, A, M), C), ((N, A, M'), C) \in$

$\mathsf{List}[F']$. Then $F'_0(N,A,M) = IV = F'_0(N,A,M')$ holds and $\mathbf{bad3}_{F'}$ happens. Hence

$$\Pr\left[\mathbf{bad3}\mathcal{D}\text{-(c)} \wedge \neg(\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho})\right] = 0. \tag{19}$$

The claim of the lemma follows from Eq. (17), Eq. (18), and Eq. (19). □

Proof (of Proposition 6). From Lemma 3, 4, and 5, it follows that

$$\begin{aligned}\Pr\left[\mathbf{bad3}\right] &\leq \Pr\left[\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho}\right] + \Pr\left[\mathbf{bad3}\mathcal{E} \wedge \neg(\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho})\right] \\ &\quad + \Pr\left[\mathbf{bad3}\mathcal{D} \wedge \neg(\mathbf{bad3}_{F'} \vee \mathbf{bad3}_{\rho})\right] \\ &\leq \frac{(\sigma_c+\sigma_f)^2}{2^\tau} + \frac{(\sigma_c+\sigma_f)(q_\rho+\sigma_c)}{2^\tau} + \frac{q_c}{2^\tau}, \end{aligned} \tag{20}$$

Hence the proposition holds. □

K.4 Game 4

In Game 4, the oracles $\mathcal{E}^{f,\rho}$ and $\mathcal{D}^{f,\rho}$ given to $\mathcal{A}$ is replaced with $\mathcal{E}'^{f,\rho}$ and $\mathcal{D}'^{f,\rho}$ defined as follows:

$\mathcal{E}'^{f,\rho}$: Given an input (N,A,M), compute $\mathcal{E}^{f,\rho}(N,A,M)$. After computing $\mathcal{E}^{f,\rho}(N,A,M)$, check if $\mathbf{bad3}\mathcal{E}$ happens or not. If $\mathbf{bad3}\mathcal{E}$ does not happen, return $\mathcal{E}^{f,\rho}(N,A,M)$. If $\mathbf{bad3}\mathcal{E}$ happens, query (N,A,M) to F' and return $F'(N,A,M)$. (Note that $\mathcal{E}^{f,\rho}(N,A,M) = F'(N,A,M)$ holds as long as $\mathbf{bad3}\mathcal{E}$ does not happen.)

$\mathcal{D}'^{f,\rho}$: Given an input (N,A,C), check if $((N,A,M),C) \in \mathsf{List}[F']$ for a unique M. Let $\tilde{M} := M$ if there exists such M, and $\tilde{M} := \perp$ if not. Then compute $\mathcal{D}^{f,\rho}(N,A,C)$, and check if $\mathbf{bad3}\mathcal{D}$ happens or not (by using $\tilde{M}$). If $\mathbf{bad3}\mathcal{D}$ does not happen, return $\mathcal{D}^{f,\rho}(N,A,M)$. If $\mathbf{bad3}\mathcal{D}$ happens, return $\tilde{M}$. (Note that $\mathcal{D}^{f,\rho}(N,A,M) = \tilde{M}$ as long as $\mathbf{bad3}\mathcal{D}$ does not happen.)

The behavior of $\mathcal{E}'^{f,\rho}$ and $\mathcal{D}'^{f,\rho}$ in this game is identical with that of $\mathcal{E}^{f,\rho}$ and $\mathcal{D}^{f,\rho}$ in Game 3 until $\mathbf{bad3} = \mathbf{bad3}\mathcal{E} \vee \mathbf{bad3}\mathcal{D}$ happens. Thus the following holds.

Proposition 7.

$$\Pr\left[\text{Game } 3 = 1\right] - \Pr\left[\text{Game } 4 = 1\right] \leq \Pr\left[\mathbf{bad3}\right]$$

K.5 Game 5

In Game 5, $\mathcal{E}'^{f,\rho}$ is replaced with F'. In addition, $\mathcal{D}'^{f,\rho}$ is replaced with $(F')^{-1}$ defined as follows.

$(F')^{-1}$: Given an input (N,A,C), if $((N,A,M),C) \in \mathsf{List}[F']$ for a unique M, return M. Otherwise return $\perp$.

Other functions and algorithms remain unchanged.

Since F' and $(F')^{-1}$ in this game are completely indistinguishable from $\mathcal{E}'^{f,\rho}$ and $\mathcal{D}'^{f,\rho}$ in Game 4, we have the following proposition.

Proposition 8.

$$\Pr[\text{Game } 4 = 1] - \Pr[\text{Game } 5 = 1] = 0.$$

K.6 Game 6

In this game the encryption and decryption oracles given to $\mathcal{A}$ are replaced with the lazy sampling versions of the encryption oracle F (fixed-key random injection) and the decryption oracle F^{-1}, respectively, which are defined as in Fig. 14 (the sampling of Fig. 14 is essentially the same as the lazy sampling of a random injection given in Fig. 3 of [74]). Again, we assume the value $F(N, A, M)$ is represented as $F(N, A, M) = F_0(N, A, M)||\cdots||F_{\lceil \mathrm{len}(M)/n \rceil}(N, A, M)$, where $F_0(N, A, M) \in \{0,1\}^\tau$, $F_i(N, A, M) \in \{0,1\}^n$ for $1 \leq i < \lceil \mathrm{len}(M)/n \rceil$, and $F_{\lceil \mathrm{len}(M)/n \rceil}(N, A, M) \in \{0,1\}^{\mathrm{len}(M) \bmod n}$. The sampling of the values of f and ρ (Algorithms 20 and 21) are accordingly changed so that queries to F'_i (i.e., queries to F' and truncation) are replaced with queries to F_i (queries to F and truncation).

Here we provide intuition behind Algorithms 22 and 23. First, note that the symbol $\perp$ may be assigned to $F^{-1}(N, A, C)$ when a fresh value (N, A, C) is queried to F^{-1}. Once the value $F^{-1}(N, A, C)$ is set as $F^{-1}(N, A, C) = \perp$, we cannot set $C = F(N, A, M)$ for any fresh query (N, A, M) to F made later. In particular, we have to remember which (N, A, C) the symbol $\perp$ has been assigned to. Thus we maintain the list $S^m_\perp$, which is the set of ciphertexts C such that $F^{-1}(C)$ is already defined to be $\perp$ and $\mathrm{len}(C) = m$. When sampling $F^{-1}(N, A, C)$ for a fresh query (N, A, C) to F^{-1}, first we have to decide whether to assign $\perp$ or not. This is the reason we choose a bit b at line 6 of Algorithm 23. $b = 1$ means that we do not assign $\perp$. The distribution D is chosen so that $\Pr_{b \xleftarrow{\$} \mathsf{D}}[b = 1]$ matches the fraction

$$\frac{(\text{the number of messages on which } F(N, A, \cdot) \text{ is not defined yet})}{(\text{the number of ciphertexts on which } F^{-1}(N, A, \cdot) \text{ is not defined yet})}.$$

If $b = 1$, we randomly choose a message M and define $M := F^{-1}(N, A, C)$. It is straightforward to show the oracles defined by Algorithms 22 and 23 are indeed indistinguishable from a random injection and its inverse.

As mentioned in Fig. 14, we introduce the two bad events **bad6a** and **bad6b**, and define **bad6** := **bad6a** $\vee$ **bad6b**.

If the boxed parts in Fig. 14 are not executed, then the output distributions of Algorithms 22 and 23 match those of F' and $(F')^{-1}$ in Game 5. Thus we have the following proposition.

Proposition 9.

$$\Pr[\text{Game } 5 = 1] - \Pr[\text{Game } 6 = 1] \leq \Pr[\mathbf{bad6}].$$

Algorithm 22: $F(N, A, M)$, lazy sampling

if $((N, A, M), C) \in \mathsf{List}[F]$ for some C **then**
  return C
else
  $C \xleftarrow{\$} \{0,1\}^{\mathrm{len}(M)+\tau}$
  if $\exists M', ((N, A, M'), C) \in \mathsf{List}[F]$ and $\mathrm{len}(M') = \mathrm{len}(M)$ or $C \in S_{\perp}^{\mathrm{len}(C)}$ **then**
    bad6a $\leftarrow 1$
    $C \xleftarrow{\$} \{0,1\}^{\mathrm{len}(M)+\tau} \setminus \left(\mathrm{Im}(F(N, A, \cdot), \mathrm{len}(M)) \cup S_{\perp}^{\mathrm{len}(C)}\right)$
  Add $((N, A, M), C)$ to $\mathsf{List}[F]$
  return C

Algorithm 23: $F^{-1}(N, A, C)$, lazy sampling

if $((N, A, M), C), ((N, A, M'), C) \in \mathsf{List}[F]$ for distinct M and M' **then**
  return $\perp$ \\ `In fact this part is never executed in Game 6`
else if $((N, A, M), C) \in \mathsf{List}[F]$ for some M **then**
  return M
else if $\mathrm{len}(C) < \tau$ or $\mathrm{Dom}(F(N, A, \cdot), \mathrm{len}(C) - \tau) = \{0,1\}^{\mathrm{len}(C)-\tau}$ **then**
  return $\perp$
else
  Choose $b \in \{0,1\}$ randomly according to the distribution D such that

$$\Pr_{b \xleftarrow{\$} \mathsf{D}}[b = 1] = \frac{2^{\mathrm{len}(C)-\tau} - |\mathrm{Dom}(F(N, A, \cdot), \mathrm{len}(C) - \tau)|}{2^{\mathrm{len}(C)} - |S_{\perp}^{\mathrm{len}(C)}| - |\mathrm{Dom}(F(N, A, \cdot), \mathrm{len}(C) - \tau)|}$$

  if $b = 1$ **then**
    bad6b $\leftarrow 1$
    $M \xleftarrow{\$} \{0,1\}^{\mathrm{len}(C)-\tau} \setminus \mathrm{Dom}(F(N, A, \cdot), \mathrm{len}(C) - \tau)$
    Add $((N, A, M), C)$ to $\mathsf{List}[F]$
    return M
  Add C to $S_{\perp}^{\mathrm{len}(C)}$
  return $\perp$

Fig. 14: The lazy sampling versions of the ideal encryption oracle F and its inverse F^{-1}. Here, $\mathrm{Dom}(F(N, A, \cdot), m) := \{M \in \{0,1\}^m : (N, A, M) \in \mathsf{List}[F(N, A, \cdot)]\}$ and $\mathrm{Im}(F(N, A, \cdot), m) := F(N, A, \mathrm{Dom}(F(N, A, \cdot), m))$, respectively. Note that the boxed parts are executed in Game 6. If the boxed parts are not executed, the output distributions of Algorithms 22 and 23 match those of F' and $(F')^{-1}$ in Game 5.

In addition, the following proposition holds.

Proposition 10. $\Pr[\mathbf{bad6a}] \leq (q_c + q_f)^2/2^\tau$ *and* $\Pr[\mathbf{bad6b}] \leq 2q_c/2^\tau$ *hold. In particular,*

$$\Pr[\mathbf{bad6}] \leq \frac{(q_c + q_f)^2 + 2q_c}{2^\tau} \tag{21}$$

holds.

Proof. First, we upper bound $\Pr[\mathbf{bad6a}]$. Since the number of fresh queries made to F or F^{-1} is at most $(q_c + q_f)$ in total (note that a fresh query is not made to F while sampling outputs of ρ),

$$\left|\mathrm{Im}(F(N, A, \cdot), \mathrm{len}(M)) \cup S_\perp^{\mathrm{len}(C)}\right| \leq q_c + q_f$$

holds. In addition, we always have $\mathrm{len}(C) \geq \tau$. Thus we have

$$\Pr[\mathbf{bad6a}] \leq \frac{(q_c + q_f)^2}{2^\tau}.$$

Next, we upper bound $\Pr[\mathbf{bad6b}]$. At each query (N, A, C) to F^{-1} with $\mathrm{len}(C) \geq \tau$, the bit b is set to 1 with probability

$$\begin{aligned}
&\frac{2^{\mathrm{len}(C)-\tau} - |\mathrm{Dom}(F(N, A, \cdot), \mathrm{len}(C) - \tau)|}{2^{\mathrm{len}(C)} - |S_\perp^{\mathrm{len}(C)}| - |\mathrm{Dom}(F(N, A, \cdot), \mathrm{len}(C) - \tau)|} \\
&\quad \leq \frac{2^{\mathrm{len}(C)-\tau}}{2^{\mathrm{len}(C)} - (q_c + q_f)} \\
&\quad = \frac{1}{2^\tau} \frac{1}{1 - (q_c + q_f)/2^{\mathrm{len}(C)}} \\
&\quad \leq \frac{2}{2^\tau}.
\end{aligned} \tag{22}$$

(Here, we used the assumption $q_c + q_f \leq 2^{\tau-1}$.) Since no queries are made to F^{-1} while sampling the values of f and ρ,

$$\Pr[\mathbf{bad6b}] \leq \frac{2q_c}{2^\tau}$$

holds. □

K.7 Game 7

This game is the ideal world. The encryption and decryption oracles F and F^{-1} are not implemented with lazy-sampling but their values are determined all at once at the beginning of the game. In addition, we define a simulator $\mathcal{S}$ to simulate ρ and f running as follows.

1. (Simulation of ρ) When an adversary makes a query to the interface corresponding to ρ, query to the revealing interface to get the list $\mathsf{List}[F]$ storing all the queries made so far to F (and the responses). Then, simulate ρ as in Algorithm 20 (with F' and F'_i being replaced with F and F_i), by using the list $\mathsf{List}[F]$.
2. (Simulation of f) Simulation of f is done as in Algorithm 20, with F' and F'_0 being replaced with F and F_0.

Note that the simulation of ρ can be performed without making queries to F given $\mathsf{List}[F]$.

The output distributions of Games 6 and 7 are completely the same. Thus the following proposition holds.

Proposition 11. $\Pr[\text{Game } 6 = 1] - \Pr[\text{Game } 7 = 1] = 0$

K.8 Finishing the Proof

From Propositions 4-11,

$$\begin{aligned}
&\mathbf{Adv}^{\text{pub-indiff}}_{\Pi,F^{\pm},\mathcal{S}}(\mathcal{A}) \\
&= \Pr[\text{Game1} = 1] - \Pr[\text{Game7} = 1] \\
&\leq \Pr[\mathbf{bad3}] + \Pr[\mathbf{bad6}] \\
&\leq \frac{(\sigma_c + \sigma_f)^2}{2^\tau} + \frac{(\sigma_c + \sigma_f)(q_\rho + \sigma_c)}{2^\tau} + \frac{3q_c}{2^\tau} + \frac{(q_c + q_f)^2}{2^\tau}
\end{aligned}$$

follows. In addition, the number of queries by simulator $\mathcal{S}$ to $F^{\pm}$ is at most q_f and the block lengths of the queries are at most σ_f. Hence the statement of the theorem holds.

L Details on How to Derive Corollary 1 from Theorem 3

This section explains how we can deduce that Corollary 1 holds by combining Theorem 3, Theorem 1, and Proposition 1, using Lemma 2. Note that the notations in this section follow those in Section 7.

We assume each simulator $\mathcal{S}$ for (public) indifferentiability is naturally converted into a one for weak public indifferentiability by recording queries that $\mathcal{S}$ makes to a construction oracle. Moreover, when we refer to "length" in this section, we mean block length rather than bit length (recall that the block length of a bit string X is $\lceil \mathrm{len}(X)/n \rceil$). Lemma 2 holds verbatim even if the unit of length is replaced from a bit to a block.

First, let $\Pi[\mathsf{RF}]$ denote a scheme where the random permutation P in $\Pi[P]$ is replaced with a random function $\mathsf{RF} : \{0,1\}^n \to \{0,1\}^n$. We define $\mathsf{RF}_0 : \{0,1\}^{n-1} \to \{0,1\}^{n-1}$ and $\mathsf{RF}_1 : \{0,1\}^\tau \to \{0,1\}^n$ from RF in the same way as P_0 and P_1 are defined from P. Then we can show that $\Pi[\mathsf{RF}]$ is weak public indifferentiable from a random injection when the primitive oracle is RF, by combining Theorem 3 and Theorem 1 with Lemma 2. More precisely, we apply Lemma 2 with the following setting.

1. R in Lemma 2 is a random injection $F^{\pm}$.
2. T^{P} in Lemma 2 is $\Pi = (\mathcal{E}^{f,\rho}, \mathcal{D}^{f,\rho})$ of Theorem 3. P in Lemma 2 is (f, ρ) of Theorem 3.
3. U^{Q} in Lemma 2 is $(\mathsf{Sponge}^{\mathsf{RF}_0}(\mathsf{enc}(\cdot,\cdot,\cdot)), \mathsf{RF}_1)$. Q in Lemma 2 is RF.
4. $\mathcal{S}_{\mathsf{T}}$ in Lemma 2 is the simulator of Theorem 3.
5. Let $\mathcal{S}_{\mathsf{Sponge}}$ be the simulator of Theorem 1. Then, a simulator $\mathcal{S}_{\mathsf{U}}$ is defined so that it runs as follows when a bit string x is queried to $\mathcal{S}_{\mathsf{U}}$. Note that $\mathcal{S}_{\mathsf{U}}$ is given oracle access to (f, ρ) in addition to a revealing interface.
   (a) ($\mathcal{S}_{\mathsf{U}}$ samples a random function $\mathsf{RF}' : \{0,1\}^n \to \{0,1\}^n$ at the beginning of the game.)
   (b) If x is of the form $x = 0||x'$ ($x' \in \{0,1\}^{n-1}$), run $\mathcal{S}^f_{\mathsf{Sponge}}$ on x' and respond with the output of $\mathcal{S}^f_{\mathsf{Sponge}}$.
   (c) If x is of the form $x = 1^{n-\tau}||x''$ ($x'' \in \{0,1\}^{\tau}$) is queried to $\mathcal{S}_{\mathsf{U}}$, query x'' to ρ and respond with $\rho(x'')$.
   (d) Otherwise respond with $\mathsf{RF}'(x)$.
6. The functions appearing in Lemma 2 (e.g., $q_{\mathcal{S}_{\mathsf{T}}}(\cdot,\cdot,\cdot,\cdot)$) related to the number of queries are as follows.
   (a) $q_{\mathcal{S}_{\mathsf{T}}}(q_c, \sigma_c, q_p, \sigma_p) = q_p$, $\sigma_{\mathcal{S}_{\mathsf{T}}}(q_c, \sigma_c, q_p, \sigma_p) = \sigma_p$,
   (b) $q_{\mathcal{S}_{\mathsf{U}}}(q_c, \sigma_c, q_p, \sigma_p) = q_p$, $\sigma_{\mathcal{S}_{\mathsf{U}}}(q_c, \sigma_c, q_p, \sigma_p) = \lceil \frac{n}{r} \rceil q_p^2$
   (c) $q_{\mathsf{T}}(q, \sigma) = 2\sigma$, $\sigma_{\mathsf{T}}(q, \sigma) = 2\sigma$

With the above setting, the following proposition follows from Lemma 2.

Proposition 12. *There exists a simulator $\mathcal{S}$ for weak public indifferentiability of $\Pi[\mathsf{RF}]$ from a fixed-key random injection $F^{\pm}$, where a primitive oracle is RF, such that the number of queries by $\mathcal{S}$ to the construction oracle is at most $(2\sigma_c + q_p)$ and the block lengths of the queries are at most $(2\sigma_c + \lceil \frac{n}{r} \rceil q_p^2)$ in total, and*

$$\mathbf{Adv}^{\text{weak-pub-indiff}}_{\Pi, F^{\pm}, \mathcal{S}}(\mathcal{A}) \leq \frac{(\sigma_c + (2\sigma_c + \lceil \frac{n}{r} \rceil q_p^2))^2}{2^{\tau}} + \frac{(\sigma_c + (2\sigma_c + \lceil \frac{n}{r} \rceil q_p^2))((2\sigma_c + q_p) + \sigma_c)}{2^{\tau}} + \frac{3q_c}{2^{\tau}} + \frac{(q_c + (2\sigma_c + q_p))^2}{2^{\tau}} + \epsilon(2\sigma_c + 2q_p)$$

holds for any adversary $\mathcal{A}$ of which computational resources are as follows: To the construction oracle, $\mathcal{A}$ makes at most q_c queries of which block length is at most σ_c in total. To the primitive oracle (corresponding to RF), $\mathcal{A}$ makes at most q_p queries. In addition, ϵ is the function such that $\epsilon(j) = 1 - \Pi_{i=1}^{j}(1 - \frac{1}{2^c})$.

Next, we replace RF in $\Pi[\mathsf{RF}]$ of the above proposition with P by using Proposition 1 and applying Lemma 2 again. That is, we apply Lemma 2 with the following setting.

1. R in Lemma 2 is a random injection $F^{\pm}$.
2. T^{P} in Lemma 2 is $\Pi[\mathsf{RF}]$. P in Lemma 2 is RF of Theorem 3.
3. U^{Q} in Lemma 2 is P. Q in Lemma 2 is (P, P^{-1}).
4. $\mathcal{S}_{\mathsf{T}}$ is the simulator of Proposition 12.

5. $\mathcal{S}_{\mathsf{U}}$ of Lemma 2 is the simulator of Proposition 1
6. The functions appearing in Lemma 2 (e.g., $q_{\mathcal{S}_{\mathsf{T}}}(\cdot,\cdot,\cdot,\cdot)$) related to the number of queries are as follows.
   (a) $q_{\mathcal{S}_{\mathsf{T}}}(q_c,\sigma_c,q_p,\sigma_p) = 2\sigma_c + q_p$, $\sigma_{\mathcal{S}_{\mathsf{T}}}(q_c,\sigma_c,q_p,\sigma_p) = 2\sigma_c + \lceil \frac{n}{r} \rceil q_p^2$
   (b) $q_{\mathcal{S}_{\mathsf{U}}}(q_c,\sigma_c,q_p,\sigma_p) = q_p$, $\sigma_{\mathcal{S}_{\mathsf{U}}}(q_c,\sigma_c,q_p,\sigma_p) = q_p$
   (c) $q_{\mathsf{T}}(q,\sigma) = \lceil \frac{n}{r} \rceil (3\sigma + q)$, $\sigma_{\mathsf{T}}(q,\sigma) = \lceil \frac{n}{r} \rceil (3\sigma + q)$

With the above setting, the proposition below follows from Lemma 2.

Proposition 13. *There exists a simulator $\mathcal{S}$ for weak public indifferentiability of $\Pi[\mathsf{RF}]$ from a fixed-key random injection $F^{\pm}$, where a primitive oracle is $P^{\pm}$, such that the number of queries by $\mathcal{S}$ to the construction oracle is at most $\lceil \frac{n}{r} \rceil (5\sigma_c + q_c + q_p)$ and the block lengths of the queries are at most $(2\sigma_c + \lceil \frac{n}{r} \rceil^3 (3\sigma_c + q_c + q_p)^2)$ in total, and*

$$\begin{aligned}
&\mathbf{Adv}^{\text{weak-pub-indiff}}_{\Pi,F^{\pm},\mathcal{S}}(\mathcal{A}) \\
&\leq \frac{(3\sigma_c + \lceil \frac{n}{r} \rceil^3 (3\sigma_c + q_p + q_c)^2)^2}{2^\tau} + \frac{(3\sigma_c + \lceil \frac{n}{r} \rceil^3 (3\sigma_c + q_p + q_c)^2)(3\sigma_c + \lceil \frac{n}{r} \rceil (3\sigma_c + q_p + q_c))}{2^\tau} \\
&\quad + \frac{3q_c}{2^\tau} + \frac{(q_c + (2\sigma_c + \lceil \frac{n}{r} \rceil (3\sigma_c + q_p + q_c)))^2}{2^\tau} + \epsilon\left(2\sigma_c + 2\left\lceil \frac{n}{r} \right\rceil (3\sigma_c + q_p + q_c)\right) \\
&\quad + \frac{\lceil \frac{n}{r} \rceil^2 (3\sigma_c + 2q_p)^2}{2^n}
\end{aligned}$$

holds for any adversary $\mathcal{A}$ of which computational resources are as follows: To the construction oracle, $\mathcal{A}$ makes at most q_c queries of which block length is at most σ_c in total. To the primitive oracle (corresponding to (P, P^{-1})), $\mathcal{A}$ makes at most q_p queries. In addition, ϵ is the function such that $\epsilon(N) = 1 - \Pi_{i=1}^{N}(1 - \frac{1}{2^c})$.

Finally, by combining the above proposition and Theorem 2 with the following setting, and by some simplifications of mathematical expressions, we can deduce that Corollary 1 holds.

1. R in Theorem 2 is a random injection $F^{\pm}$.
2. Σ^{π} in Theorem 2 is $\Pi[E_K]$.
3. P in Theorem 2 is a random permutation P and its inverse P^{-1}.
4. π in Theorem 2 is E_K.
5. $\mathcal{S}_{\text{indiff}}$ in Theorem 2 is the simulator of Proposition 13.
6. The functions appearing in Theorem 2 related to the number of queries are as follows.
   (a) $q_{\mathcal{S}_{\text{indiff}}}(q_c,\sigma_c,q_p,\sigma_p) = \lceil \frac{n}{r} \rceil (5\sigma_c + q_c + q_p)$, $\sigma_{\mathcal{S}_{\text{indiff}}}(q_c,\sigma_c,q_p,\sigma_p) = 2\sigma_c + \lceil \frac{n}{r} \rceil^3 (3\sigma_c + q_c + q_p)^2$.
   (b) $q_{\Sigma}(q,\sigma) = \sigma_{\Sigma}(q,\sigma) = 2\sigma + \left\lceil \frac{n}{r} \right\rceil q$.

M On the Existence of whPRP-Secure BCs

Our ensured security by the mode of operation is eventually reduced to whPRP-secure block ciphers. It is a natural question whether there exist whPRP-secure

block ciphers or not. Unfortunately, there are no existing block ciphers whose design goal is whPRP[36]. However, we expect that some SPACE-hard block ciphers can be a good candidate for whPRP-secure block ciphers. Note that this is not a provable discussion like we cannot prove a secure block cipher (such as AES) is PRP. We discuss based on the existence of real cryptanalysis.

whPRP is not natural for the claimed security of block ciphers; e.g., we usually do not have the simulator when discussing cryptanalysis of block ciphers. Therefore, in this section, we bridge from real cryptanalysis to whPRP. First, we discuss existing security notions and attack models for SPACE-hard block ciphers and introduce an extended security notion called *hybrid SPACE hardness*. We provide an intuition why the block cipher satisfying the hybrid SPACE hardness likely satisfies whPRP. Next, we discuss that SPACE likely satisfies the hybrid SPACE hardness by taking some possible cryptanalyses into consideration.

M.1 Hybrid SPACE Hardness and whPRP

SPACE Hardness. In [24], Bogdanov and Isobe introduced a SPACE hardness defined as follows:

Definition 5 ((M, Z)-SPACE hardness [24]). *The implementation of E_K is (M, Z)-space hard if it is infeasible to encrypt (decrypt) any randomly drawn plaintext (ciphertext) with a probability of more than 2^{-Z} given any code (table) of size less than M.*

We call block ciphers satisfying SPACE hardness SPACE-hard block ciphers. In practice, SPACE-hard block ciphers encrypt plaintexts by continuous table lookups, and the table is derived from a secure block cipher (such as AES) by constraining inputs and truncating outputs. Thus, even if a white-box adversary can look at the table, extracting a secret key from the table is as difficult as the key recovery attack of the secure block cipher.

Bogdanov et al. [25] pointed out that the SPACE hardness depends on attack models.

Definition 6 (Attack models for Definition 5 [25]). *Let F be a table used in E_K.*

- Known-Space Attack *(KSA) extracts M pairs of inputs and corresponding outputs of tables $(x_i, F(x_i))$ for $i \in \{1, \ldots, M\}$.*

[36] Recently, a hybrid code lifting is introduced as a new attack vector against space-hard block ciphers in [78]. The demonstrated attack is very similar to ours, and our hybrid space hardness can be regarded as the formal definition of their cryptanalysis. According to their detailed cryptanalysis, space-hard block ciphers Yoroi and SPNbox are vulnerable and do not satisfy our hybrid space hardness. On the other hand, when we increase the number of rounds in SPNbox by several rounds, it can be a good candidate.

- Chosen-Space Attack *(CSA) extracts M pairs of inputs and corresponding outputs of tables $(x_i, F(x_i))$ for $i \in \{1, \ldots, M\}$, where x_i is chosen by adversaries before mounting attacks.*
- Adaptive Chosen-Space Attack *(ACSA) extracts M pairs of inputs and corresponding outputs of tables $(x_i, F(x_i))$ for $i \in \{1, \ldots, M\}$, where x_i is adaptively chosen by adversaries, i.e., x_a can be chosen after obtaining $(x_{a-1}, F(x_{a-1}))$.*

Besides the above ones, Chow et al. suggested a more trivial attack model where some plaintext-ciphertext pairs are simply copied [28]. We call such an attack *query leaking* in short. We cannot avoid that $\lfloor \lambda/n \rfloor$ plaintext-ciphertext pairs are leaked, where λ is the total bit size of leakage and n is a block length.

Hybrid SPACE Hardness. SPACE hardness significantly deviates from wh-PRP demanded in our mode. To bridge the gap, we extend the former notion to *hybrid SPACE hardness*.

Definition 7 (Hybrid SPACE hardness). *Let $\mathcal{A}$ denote a black-box adversary running time in less than t and making q queries to E_K and D_K in total that runs as follows: $\mathcal{A}$ first makes some queries and outputs a lifter running in time t_{lif}. Then the lifter is given unlimited access to $[\![E_K]\!]$ and outputs L whose size is at most λ bits. Finally, $\mathcal{A}$ receives L, makes additional queries, and then outputs plaintext-ciphertext pairs. We say that the white-box block cipher satisfies $(t, \lambda, q, t_{\mathrm{lif}})$*-hybrid SPACE hardness *if any such $\mathcal{A}$ cannot output more than $(q + \lfloor \lambda/n \rfloor)$ correct plaintext-ciphertext pairs with sufficiently high probability.*

Hybrid SPACE hardness is a straightforward extension of SPACE hardness. The extension is two-fold: First, the adversary has a chance to make black-box queries. Second, the lifter can leak arbitrary information rather than table entries only. The time of the lifter is bounded to avoid trivial attacks by exhaustive key search. Note that it is unavoidable that $\lfloor \lambda/n \rfloor$ plaintexts or ciphertexts are leaked via a λ-bit leakage.

To study whether existing SPACE-hard block ciphers fulfill hybrid SPACE hardness, we discuss detailed cryptanalysis in Appendix M.2. We conclude that any query leaking is not helpful to attack the hybrid SPACE hardness because a black-box adversary can collect them by only oracle queries. Thus, it is enough to consider lifters that leak inside information on $[\![E_K]\!]$. We tried several-kind of such attacks. As a result, we expect the hybrid SPACE hardness to be satisfied if well-designed SPACE-hard block ciphers have a large security margin w.r.t. SPACE hardness against the KSA and CSA.

From Hybrid SPACE Hardness to whPRP. Hybrid SPACE hardness is not a formal security definition available for a security proof. Still, we expect that well-designed SPACE-hard block ciphers satisfying hybrid SPACE hardness are likely to fulfill whPRP.

The hybrid SPACE hardness guarantees that any attack cannot leak more information on plaintext-ciphertext pairs than query leaking. Besides, if an adversary can compute some plaintext-ciphertext pairs with the aid of leakage and

without the encryption oracle, it is natural to assume the lifter knows these pairs. Considering such a situation, we present an idea for constructing the simulator. With bounds coming from the mode requirements, q_{sim} must be chosen from reasonable values, e.g., $q_{\mathrm{sim}} \approx 2^{50}$, but huge t_{sim} is accepted. In addition, an adversary cannot notice even if table entries of SPACE-hard block ciphers are randomly chosen by simulators. Thus, the simulator can use the following strategy.

1. Try out all possible tables and construct ciphers using each table.
2. Apply the lifter to constructed ciphers, and list at most λ-bit information about the leaked plaintext-ciphertext query, which can be encrypted by $\mathcal{A}$ without encryption oracle.
3. Query properly chosen plaintexts, and get corresponding ciphertexts. Here, it chooses plaintexts such that it discards some tables but many tables still survive in the list.
4. Once we find a table whose λ-bit leaked query information is consistent with the random permutation P, it prepares the program $\mathcal{P}'$ using the found table and leaks $\mathcal{L}(\mathcal{P}')$.

On the procedure above, we expect there is a simulator if $t_{\mathrm{lif}} \leq q_{\mathrm{sim}}$ because of two reasons[37]. First, it is unlikely that it needs more queries than the lifter's time to obtain at most λ-bit leaked query information defined by the lifter. Next, it is unlikely that the heuristic query by the simulator in step (3) discards all table candidates because the simulator still has many tables on average after at most λ-bit leaked query information is determined, supposing λ is the quarter size of the table. For example, when a 16-bit to 112-bit table is used, the number of all possible tables is $2^{112\times 2^{16}}$. Even if $\lambda = 112 \times 2^{14}$-bit information is determined, we still have table candidates.

Again, the discussion above is not a provable one and focuses on the existence of attacks. As far as we try, there is no attack procedure such that the simulator fails. Our discussion is the initial point of reviewing the security of SPACE-hard block ciphers. We expect subsequent works to be more convinced with the security by discussing new attack strategies.

M.2 Cryptanalysis of SPACE in Context of Hybrid SPACE Hardness

Hereinafter, we concentrate on a concrete SPACE-hard block cipher SPACE. We expect SPACE fulfills the hybrid SPACE hardness. We first introduce a specification of SPACE and next review existing attacks against SPACE. We then discuss some possible attacks in the context of hybrid SPACE hardness.

[37] It is interesting that we fail to bridge from the hybrid SPACE hardness to whPRP if $q_{\mathrm{sim}} < t_{\mathrm{lif}} \leq q$. Then, we have the following attack: $\mathcal{A}_{\mathrm{create}}$ randomly chooses $x \in \{0,1\}^n$ and generates a lifter, which evaluates $y = [\![E_k]\!]^{t_{\mathrm{lif}}}(x)$ and leaks y. $\mathcal{A}_{\mathrm{dist}}$ receiving y checks if $y = E_K^{t_{\mathrm{lif}}}(x)$ holds by querying E_K t_{lif} times. The simulator fails to simulate because it cannot query to P t_{lif} times.

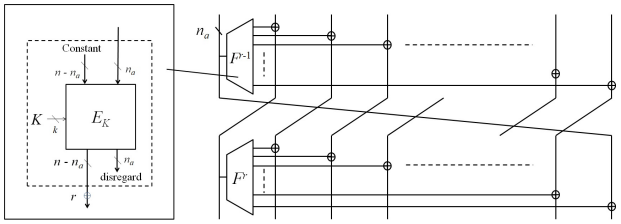

Fig. 15: Dedicated SPACE-hard block cipher SPACE

SPACE. SPACE is the first SPACE-hard block cipher designed by Bogdanov and Isobe [24][38]. After their proposal, many SPACE-hard block ciphers have been proposed [25, 43, 28, 58, 55].

SPACE is a 128-bit block cipher adopting the target-heavy generalized Feistel structure for encryption. The table called the F function (the rth round function F^r) is derived from a secure block cipher by constraining inputs, truncating outputs, and introducing a round constant r. The authors of [24] instantiated SPACE with AES-128 to generate F^r.

SPACE has four variants, SPACE-n_a, where $n_a \in \{32, 24, 16, 8\}$. The round function is iterated by $R = 128$ times for $n_a \in \{32, 24, 16\}$ and $R = 256$ times for $n_a = 8$. Figure 15 shows the $(r-1)$th and rth round functions of SPACE-n_a.

Existing Analysis by Black-Box Adversaries. We first say that security against (only) black-box adversaries was already claimed in [24]. They evaluated differential cryptanalysis [15], linear cryptanalysis [62], impossible differential [14], and so on. These attacks do not reach not only the full rounds but also the half number of the full rounds because the number of rounds for SPACE is determined by security against code lifting.

The difficulty comes from a secret, secure, and large F^r. A normal block cipher uses a combination of a short subkey and a public S-box. Thus, the guess of the short subkey is very useful because we can evaluate all 2^{n_a} mappings by guessing an n_a-bit subkey. On the other hand, guessing such a small fraction of secret information is not always powerful in SPACE. We need to guess $(128 - n_a)$ bits to get one output of one table entry.

Existing Code Lifting by White-Box Adversaries. Code lifting (by white-box adversaries) is more critical to discussing the security of SPACE. We inherit the same security level claimed in [24]: code lifting reveals quarter-size of tables to adversaries.

[38] SPACE is a dedicated design for whitebox block cipher. To the best of our knowledge, such a direction was initiated with ASASA [17], which was unfortunately broken [44, 37, 66].

In [25], Bogdanov et al. introduced three attack models: the known-space attack (KSA), chosen-space attack (CSA), and adaptive chosen space attack (ACSA) (see Definition 6 in detail). We first follows their discussion as follows.

- In the KSA and CSA, because table entries are copied non-adaptively, the probability that randomly-drawn plaintexts are successfully encrypted is $(1/4)^R$, that is 2^{-256} for SPACE-16, SPACE-24, and SPACE-32. There are 2^{128} plaintexts. Thus, there is unlikely any plaintext that can be encrypted using the copied table. Moreover, considering $(1/4)^{R/2} = (1/4)^{64} = 2^{-128}$, on average, only one plaintext can be partially encrypted up to the half number of rounds.
- In the ACSA, leaking R table entries allows a black-box adversary to encrypt at least one plaintext, where R denotes the number of rounds. $(2^{n_a-2}, 128)$-SPACE hardness means that there is no plaintext that the black-box adversary can encrypt by using the leakage. Therefore, it implies that guaranteeing $(2^{n_a-2}, 128)$-SPACE hardness is not practical when $n_a \in \{16, 24, 32\}$ because it requires at least 2^{n_a-2} rounds.

Remember that a white-box adversary can unlimitedly access the implementation. There is no reason that they copy table entries only. Practically, copying simple query information, e.g., ciphertexts for chosen plaintexts, is easy because the white-box adversary does not need to analyze the inside of the implementation. Such a trivial attack was suggested in [28]. This simple query leaking implies that sneaking $\lfloor \lambda/128 \rfloor$ plaintext-ciphertext pairs is unavoidable when λ is the total bit size of leakage[39].

After the proposal of the SPACE hardness, the attacks mentioned above have been discussed in [25, 28]. However, we do not know general attacks superior to these attacks to the best of our knowledge.

New Analysis by Hybrid Adversaries. We next discuss the security against the hybrid SPACE hardness (see Definition 7), which is our main goal. We assume code lifting of any information related to the implementation. We additionally assume the opportunity that an adversary queries to encryption and decryption oracles.

[39] Cho et al. also proposed an attack guessing unknown table entries after the KSA/CSA [28]. In SPACE, when only one table entry is unknown out of R table entries, we can successfully encrypt such a plaintext with a probability of $2^{-(128-n_a)}$ by guessing the $(128 - n_a)$-bit output of the unknown table entry. Thus, when the probability of $\binom{R}{1} \times (1/4)^{R-1} > 2^{-128}$, there is at least a plaintext encrypted with probability higher than 2^{128-n_a} using known table entries. However, an adversary cannot detect if given plaintexts belong to the set of such plaintexts. Thus, it is not clear whether it allows encrypting any randomly-drawn plaintext with a probability of more than 2^{-128}. At least, it is unlikely to be possible for adversaries running time in 2^{128}.

Black-Box Adversary with Query Leaking. We first consider a black-box adversary with the query leaking. It provides any λ-bit information related to plaintext-ciphertext pairs but does not include the inside information about the implementation. We notice that the query leaking is not helpful to break the hybrid SPACE hardness because the adversary can collect them by only oracle queries. Namely, if this combined attack works, such a cipher is already insecure against black-box adversaries.

Black-Box Adversary with KSA/CSA. We next consider a black-box adversary with the KSA and CSA. It knows 1/4 table entries. Is it possible to complement unknown table entries by using the chance of black-box oracle query? We expect it is unlikely if $(1/4)^R \ll 2^{-128}$.

Let us consider $(1/4)^R \approx 2^{-128}$. Then, there are a few plaintexts, where their encryption is possible by using known table entries with guessing a few unknown table entries. Since the time in the adversary is bounded, it is not trivial to detect such plaintexts but could be possible. If it is possible, the adversary guesses a few unknown table entries and confirms the correctness of the guess by querying encryption/decryption oracles. Once a few extra table entries are successfully recovered, the use of newly recovered table entries allows the adversary to recover more table entries. The repetition of this procedure might cause a risk of recovering all table entries by the adversary.

In SPACE, $(1/4)^R = 2^{-256}$. The security margin, i.e., the gap between 2^{-128} and 2^{-256}, is plenty enough, even considering the risk by the above guess-and-determine attack. We need to remark that any follow-up SPACE-hard block ciphers do not always have such plenty of security margin. For example, the claimed security of SPNbox is $(2^{n_a-2}, 64)$-SPACE hardness against KSA/CSA [25]. Thus, SPNbox does not satisfy the hybrid SPACE hardness unless $\lambda \ll n_a \times 2^{n_a-2}$.

The attack above uses a straightforward guess-and-determine attack. How about a combination with more advanced cryptanalyses by black-box adversaries? We tried but were not successful very well. We summarize why it is unlikely to be possible as follows.

- We usually use "(truncated) difference transitions" or "(multi-dimensional) linear mask transitions" in advanced cryptanalyses. However, the knowledge that 1/4 of table entries are known is not helpful for such statistical attacks.
- Attacks directly guessing values, like a meet-in-the-middle attack, might be more helpful than statistical attacks. However, it needs to guess too many bits because 3/4 of table entries are still unknown. Even after leakage, the target cipher still looks like having a $((128 - n_a) \times 3 \times 2^{n_a-2})$-bit key for a black-box adversary unless they search the secret key exhaustively. The size is still incredibly larger than other normal block ciphers such as AES, which have at most 11×128 bits even if all round keys are guessed independently.
- Finally, we will revisit results on black-box only and code lifting only scenarios. There is no black-box attack covering the half number of rounds. Besides, the probability is already 2^{-128} for randomly-drawn plaintexts to

encrypt up to the half round. Therefore, it is unlikely that the combined attack can cover the full rounds.

Black-Box Adversary with ACSA. The most considerable attack would combine the ACSA with the black-box oracle access. For example, a few table entries are omitted from R table entries, and the omitted table entries are later recovered by the guess-and-determine approach with black-box oracle access. When only one table entry is omitted, the entry is easily recovered by querying the corresponding plaintext to encryption oracle. When two table entries are omitted, we need to guess one table entry, $(128 - n_a)$ bits. Another table entry is derived by checking the consistency with encryption oracle, but the number of candidates is no longer one, i.e., $2^{2\cdot(128-n_a)-128}$. The number of candidates exceeds 2^{128} when three table entries are omitted. Assuming there are methods to recover three omitted table entries after code lifting, the number of additionally recovered table entries is $3 \times \frac{2^{n_a-2}}{R}$. It is unlikely to recover more table entries because the ACSA allows encrypting only preliminary-determined fixed plaintexts.

The combination with the ACSA is interesting because it reveals more table entries than 2^{n_a-2}. This attack recovers some additional table entries, and we need to review the security against the KSA and CSA again. Let us assume concrete case: assuming an adversary gets $2^{n_a-2} \times \left(1 + \frac{3}{R}\right)$ table entries after the combined attack with ACSA, the probability hitting known table entries increases from $1/4$ to $\left(1 + \frac{3}{R}\right) \times 1/4$, which is about 0.2559. The probability of encrypting randomly-drawn plaintexts (except for preliminary-determined fixed plaintexts in the ACSA) is about $2^{-251.72}$, which is low enough still.

Black-Box Adversary with Other Code Lifting. We discussed that query leakage is not helpful to break the hybrid SPACE hardness. We also discussed that table leakage (based on KSA, CSA, and ACSA) is not beneficial if it has plenty of security margin in the SPACE hardness against the KSA/CSA. But, how about other any leakage? As far as we analyze, we cannot find a more useful method than table leakage. For example, let us consider that the lifter copies some intermediate states. It could recover some table entries. However, even if possible, attackers can retrieve the corresponding table entry using each intermediate state. Thus, we believe that the direct leak of table entries would be simple and leak more information.

Some artificial designs could be possible such that another code lifting is helpful. Moreover, we might face uncertain risks when an exploitable specific structure exists in a concrete substantiation of SPACE-hard block cipher. Therefore, we need to look precisely and carefully at each SPACE-hard block cipher.

After looking at SPACE by several kinds of cryptanalysis techniques, we cannot find a more powerful attack than the combination attack with the ACSA.

Summary of security of SPACE. We finally expect that SPACE is a typical example satisfying the hybrid SPACE hardness. Concretely, it would fulfil $(t, \lambda, q, t_{\text{lif}})$-hybrid SPACE hardness with $t + t_{\text{lif}} < 2^{128}$, $\lambda = (128 - n_a) \times 2^{n_a-2}$,

and $q < 2^{128} - \lfloor \lambda/128 \rfloor$. As discussed in Appendix M.1, SPACE-hard block ciphers satisfying $(t, \lambda, q, t_{\mathrm{lif}})$-hybrid SPACE hardness is a good candidate of the block cipher satisfying $(t, \lambda, q, t_{\mathrm{lif}}, q_{\mathrm{sim}}, \epsilon)$-secure whPRP when $t_{\mathrm{lif}} \leq q_{\mathrm{sim}}$.

N Proposal of SPACE256-16

Our mode of operation ensures roughly $n/4$-bit security with an n-bit block whPRP-secure block cipher. We expect SPACE is a good candidate for whPRP-secure block ciphers, but unfortunately, the block length is 128 bits only. Other existing SPACE-hard block ciphers such as SPNbox, WhiteBlock, and WEM also have 128-bit block lengths. When we use a 128-bit block whPRP-secure block cipher, the ensured bit security is 32 bits, which we cannot recommend. Therefore, we propose a new block cipher, SPACE256-16, with the 256-bit block length and the 16-bit domain size of the table. We inherit the design rationale of SPACE and extend the block length to 256 bits. Note that we still keep the key size to 128 bits.

N.1 Specification of SPACE256-16

SPACE was initially designed to support variable branch sizes of generalized Feistel structure, and the bit length of each branch is also variable. Therefore, it is not difficult to extend the design to 256-bit block length. We simply use the same network of SPACE-8, but the bit size in each branch is extended from 8 bits to 16 bits, i.e., the number of branches is 16, and 16-bit to 240-bit table is used in the F^r of the generalized Feistel structure. Similarly to SPACE, the rth function of SPACE256-16 is specified as

$$F^r(x) = \left(\mathrm{msb}_{240}\left(AES_K(0^{112}\|x)\|AES_K(0^{111}\|1\|x)\right)\right) \oplus r. \tag{23}$$

The white-box implementation uses 2^{16} 16-bit to 240-bit lookup tables. The number of rounds to encrypt one block is 128.

N.2 Cryptanalysis in the White Box

Key Extraction. The table is generated by the AES. A white-box adversary can look at all table entries, but it is equivalent to recover the secret key by using many plaintext-ciphertext pairs. Assuming that there is no efficient key recovery attack against AES, the key extraction from the table is infeasible.

Space Hardness. We inherit existing size of leakage. i.e., a quarter-size leakage. When the KSA and CSA are mounted, the probability that we compute each round successfully is 2^{-2}. Therefore, in 128 rounds, the probability that we can successfully encrypt plaintexts is 2^{-256}, which is the same security margin as the existing SPACE-16.

When the ACSA is mounted, 128 table entries are extracted for each encryption of a chosen plaintext. Thus, the quarter-size leakage leaks $2^{n_a-2}/128 = 2^7$ chosen plaintext-ciphertext pairs. Note that the number of leaked plaintext-ciphertext pairs is smaller than the query leaking, that leaks $2^{n_a-2} \times 240/256 \approx 2^{13.9}$ chosen plaintext-ciphertext pairs.

N.3 Cryptanalysis in the Black Box

Similarly to the existing security analyses against SPACE, we evaluate the security of differential [15], linear [62], impossible differential [14], and integral attacks [33, 54]. Recently, these security analyses can be automatically evaluated by using some MILP-aided methods [67]. We use Gurobi optimizer [49] for each evaluation.

Differential Cryptanalysis. We first discuss the upper bound of the differential characteristic probability, and these probabilities are usually evaluated under the key average. The F function is generated by the secure block cipher such as AES and is a 16-bit to 240-bit function with a 128-bit secret key. The number of possible output differences is at most $\binom{2^{16}}{2} \times 2^{128} \approx 2^{159}$, but the length of the output is 240 bits. Thus, since the average of differential probability is extremely low already for only one active F function, it is unlikely to apply common differential cryptanalysis.

We next discuss very conservative security evaluation against cryptanalysis using differences. We focus on the differential characteristic with a single key instead of the key average. Again, the F function is a 16-bit to 240-bit function. It is unlikely that two different input difference causes the same output difference. Therefore, we assume the maximum differential probability of the F function is $2/2^{16} = 2^{-15}$ for a fixed key. Let $F^r(x) + Y$ be updated 15 branches in the rth round. When x is active, the difference of $F^r(x) + Y$ depends on the values of x. Attackers need to rely on probabilistic events to control each of these 15 branches. Therefore, we count the number of active F functions under the restriction of the number of inactive branches in $F^r(x) + Y$.

We first suppose at least 13 branches are active. In other words, it accepts that at most 2 branches can be inactive. Considering there are about $\binom{2^{16}}{2} \approx 2^{31}$ output differences of active F function, at most 2 branches (32 bits) can be controlled by attackers. Thus, there might be such a differential transition in the active F function. Under such a restriction, we got a result, where 40 rounds are enough to have 9 active F functions by using an MILP-aided method, and we have enough security margin.

We next suppose at least 5 branches are active. In other words, it accepts that at most 10 branches can be inactive. It is unlikely that we have such a differential transition for a randomly-chosen fixed key. However, we have 2^{128} keys, and there is the possibility to exist such a transition for 1 key out of 2^{128} keys. Even under such a restriction, we got a result, where 104 rounds are enough to have 9 active F functions by using an MILP-aided method.

Linear Cryptanalysis. In [24], the linear probability is estimated based on the result of [34].

Corollary 2 ([34]). *The linear probability of a non-trivial linear approximation over n_a-bit to n_b-bit function with $n \geq 5$ has mean 2^{-n_a} and variance 2^{-2n_a+1}.*

According to Corollary 2, the authors of [24] estimated the linear probability of the F function with 16-bit input as $2^{-12.5}$. Therefore, we inherit this estimation. We count the number of active F functions by using an MILP. As a result, r rounds ensures at least $r-1$ active F functions. Since $(2^{-12.5})^{11} < 2^{-128}$, 12 rounds have enough security margin against the linear cryptanalysis. Therefore, 128 rounds have plenty of security margin against the linear cryptanalysis.

Impossible Differential Cryptanalysis. Impossible differential is also found by using an MILP [75]. We evaluated all 32×32 combinations, where numbers of active branches in plaintexts are ciphertexts are 1. As a result, the maximum number of rounds that we got an impossible differential was 30 rounds. Therefore, 128 rounds have plenty of security margin against the impossible differential cryptanalysis.

Integral Cryptanalysis. A division property [77] is recently used as the tool to find the longest integral characteristic, and its MILP-aided method is shown in [81]. As a result, the maximum number of rounds of an integral characteristic was 18 rounds. Therefore, 128 rounds have plenty of security margin against the integral cryptanalysis.

N.4 Cryptanalysis in Context of Hybrid SPACE Hardness

As shown in M.2, we expect that SPACE-hard block cipher with significant security margin can be a good candidate satisfying hybrid SPACE hardness. We follow-up some important attacks and confirm that SPACE256-16 would satisfy the hybrid SPACE hardness.

We now consider the most powerful attack: the attack collaborated with the ACSA. In the ACSA, attackers leak 128 table entries to encrypt one plaintext. We first consider the case that attackers omit one table entry from the leakage. It is easy to recover the omitted table entry by checking the consistency with the ciphertexts given by the oracle query. We next consider the case that attackers omit two table entries from the leakage. Remember that the output size of the F function is $256 - 16 = 240$ bits. Therefore, attackers no longer guess one table entry, and such a strategy is already infeasible. For the conservative evaluation, assuming there are methods that attackers recover two omitted table entries after code lifting, the number of additionally recovered table entries is $2 \times \frac{2^{16-2}}{128} = 256$. In total, attackers can get 16640 table entries. The probability hitting known table entries increases from $1/4$ to $16640/2^{16} = 0.2539$. The probability of encrypting randomly-drawn plaintexts is about $2^{-253.14}$, which is low enough still.

Summary of security of SPACE256. Similarly to SPACE, we expect that SPACE256 is a typical example satisfying the hybrid SPACE hardness. Concretely, it would fulfil $(t, \lambda, q, t_{\mathrm{lif}})$-hybrid SPACE hardness with $t + t_{\mathrm{lif}} < 2^{128}$, $\lambda = (256 - 16) \times 2^{16-2}$, and $q < 2^{256} - \lfloor \lambda/256 \rfloor$.