Cryptographic Agility and its Relation to Circular Encryption

Tolga Acar[1] Mira Belenkiy[2] Mihir Bellare[3] David Cash[4]

Abstract

We initiate a provable-security treatment of cryptographic *agility*. A primitive (for example PRFs, authenticated encryption schemes or digital signatures) is agile when multiple, individually secure schemes can securely share the same key. We provide a surprising connection between two seemingly unrelated but challenging questions. The first, new to this paper, is whether wPRFs (weak-PRFs) are agile. The second, already posed several times in the literature, is whether every secure (IND-R) encryption scheme is secure when encrypting cycles. We resolve the second question in the negative and thereby the first as well. We go on to provide a comprehensive treatment of agility, with definitions for various different primitives. We explain the practical motivations for agility. We provide foundational results that show to what extent it is achievable and practical constructions to achieve it to the best extent possible. On the theoretical side our work uncovers new notions and relations and settles stated open questions, and on the practical side it serves to guide developers.

Keywords: Circular encryption.

[1] eXtreme Computing Group, Microsoft Research, Microsoft, One Microsoft Way, Redmond, WA 98052, USA. Email: tolga@microsoft.com. URL: http://research.microsoft.com/en-us/people/tolga.

[2] eXtreme Computing Group, Microsoft Research, Microsoft, One Microsoft Way, Redmond, WA 98052, USA. Email: mibelenk@microsoft.com.

[3] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: mihir@cs.ucsd.edu. URL: http://cseweb.ucsd.edu/~mihir/.

[4] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: cdc@gatech.edu. URL: http://cseweb.ucsd.edu/~cdc/.

Contents

1 Introduction

This paper initiates a provable-security treatment of cryptographic *agility*. Agility considers a set of schemes, all meeting some base notion of security, and requires that security is maintained when multiple schemes use the same key. Agility may be considered for any cryptographic primitive: PRFs, authenticated symmetric encryption, collision-resistant hashing, IND-CCA public-key encryption, whatever. To illustrate let us jump right to the example where we have the most interesting results. Then we will back up and discuss motivation and other results.

ARE WPRFS AGILE? Let F be a family of functions and consider a game that picks a random key K and challenge bit b and gives the adversary an oracle **Fn** that takes no inputs. Each time it is called, **Fn** picks random x, y, returning $(x, F_K(x))$ if $b = 1$ and (x, y) otherwise. Naor and Reingold [25] call F a weak-PRF (wPRF) if the adversary can't guess b.

Why this notion? Being a wPRF is, in practice, a much weaker assumption on a blockcipher than the usual PRF or PRP one. Yet powerful results by Naor and Reingold [25], Maurer and Sjödin [21] and Maurer and Tessaro [22] show that symmetric cryptography can be efficiently and securely based on wPRFs.

Letting F^1, F^2 be wPRFs having keys of the same length, consider a game that picks a *single* random key K and challenge bit b and gives the adversary an oracle **Fn** that, on input $i \in \{1, 2\}$, picks random x, y, returning $(x, F^i_K(x))$ if $b = 1$ and (x, y) otherwise. It's just like the previous game, but with two function families, and *both are being evaluated with the same key.* We say that the pair $\{F^1, F^2\}$ is agile if an adversary can't guess b in the above game. Now consider the following statement or conjecture:

wPRF-A : EVERY PAIR $\{F^1, F^2\}$ OF WPRFS is agile.

Is the statement true? Our first guess was yes, because the randomness of the inputs means the two functions are unlikely to ever be evaluated on the same point, and then it is hard to see what harm there is in their using the same key. But attempts to prove this failed. It is unclear how to reduce the agility of $\{F^1, F^2\}$ to their individual, assumed wPRF securities because reduction-based proof methods break down totally when the key is the same for both functions. Does that mean the statement is false? To demonstrate that, we need a counter-example, meaning specific families F^1, F^2 that (under some assumption) are wPRFs but we have an attack showing $\{F^1, F^2\}$ is not agile. However, an example is not immediate, again due to the fact that the attacker has no control on the inputs to the functions, these being chosen at random by the game.

We clarify that the question is *not* whether there *exists* a pair $\{F^1, F^2\}$ that is agile. We are not asking for a construction of F^1, F^2 that can securely share a key. Indeed, such a construction is trivial: just let F be a wPRF and let $F^1 = F^2 = F$. The question is whether *all* pairs $\{F^0, F^1\}$ of wPRFs are agile.

We have still to motivate *why* we should care whether security is maintained when two schemes use the same key, a practice that cryptographers would typically frown upon. But we will soon explain important practical reasons for this concern. Furthermore, our focus on wPRFs is not arbitrary. We will see that wPRFs are "agility catalysts" in the sense that if they are agile then we can make a host of other primitives agile as well. So the above question —is **wPRF-A** true or not— is central.

We find it intriguing that so basic and simply stated a question is hard to answer. We will obtain the answer by turning to something that seems different but eventually isn't.

ARE IND-R ENCRYPTION SCHEMES CYC-SECURE? IND-R (INDistinguishability from Random) [27] is a strong notion of CPA-privacy for symmetric encryption schemes that is met by common blockcipher modes of operation (CBC, CTR). It implies IND-CPA. CYC asks for privacy when encrypting "cycles" of the form $\mathcal{E}(K_1, K_2), \mathcal{E}(K_2, K_1)$. Cyclic security was introduced by Camenisch and Lysyanskaya [13] and is of interest as a simple and basic instance of the security of encrypting key-dependent messages

concurrently considered by Black, Rogaway and Shrimpton [10]. Now consider the following statement or conjecture:

IND-is-CYC : EVERY IND-R SYMMETRIC ENCRYPTION SCHEME IS CYC-SECURE.

Broadly speaking, this asks whether "normal" security implies security when encrypting cycles. In their work presenting a particular, public-key encryption scheme shown to securely encrypt cycles if the DDH assumption holds, Boneh, Halevi, Hamburg and Ostrovsky [11] explicitly ask, and leave open, the above question. (Up to details of the definitions.) Black, Rogaway and Shrimpton [10] pose it too. Haitner and Holenstein's black-box separations for key-dependent message security [16] only consider stronger forms of security, and do not apply to this question.

THE CONNECTION. We have just stated two open problems that on the face of it are quite different. The first is about wPRFs and the second about symmetric encryption, which are different primitives. In the first case, the issue is sharing a key between two schemes. In the second, no key sharing is involved and we refer to standard notions. Yet, we show the two problems are related. Specifically, we show in Theorem 4.2 that

wPRF-A $\Rightarrow$ **IND-is-CYC**.

That is, if *every* pair $\{F^1, F^2\}$ of wPRFs is agile (can securely share a key) then *every* IND-R symmetric encryption scheme is CYC-secure (can securely encrypt cycles).

SETTLING BOTH QUESTIONS. So far the above is an instance of what Karp called the "If pigs could whistle then horses could fly" approach that aims to understand open questions in cryptography and complexity theory by relating them to each other. As above, this approach can turn up interesting relations between seemingly unrelated problems. But it doesn't settle them. However, in this case, we can go further. We provide in Theorem 4.6 a direct and explicit counter-example to show that **IND-is-CYC** is false, resolving the above-mentioned open problem. Our **wPRF-A** $\Rightarrow$ **IND-is-CYC** connection then implies that **wPRF-A** is also false, settling the question of whether wPRFs are agile. The counter-example is a symmetric encryption scheme shown to be IND-R under the SXDH assumption of [2] but shown by attack to not be CYC-secure.

This result refuting **IND-is-CYC** is strengthened by the fact that IND-R is a very strong version of (CPA) privacy and our formalization of CYC is a very weak one. (The formal definitions are in the body of the paper.) Thus, even strong "normal" security fails to imply weak security for encrypting cycles. Interest in this question is witnessed by the work of Backes, Pfitzmann and Scedrov [4] who have previously shown that IND-CPA does not imply a formalization CYC-BPS of cyclic security. However, their counter-example encryption scheme is stateful while ours is stateless, and also IND-R $\Rightarrow$ IND-CPA and CYC-BPS $\Rightarrow$ CYC, making their result weaker than ours.

EXTENSIONS AND IMPLICATIONS. Above we discussed **IND-is-CYC** in the symmetric setting. Our result showing it is false, however, extends to the public-key setting, showing that IND-CPA (semantic security) does not imply security of encrypting cycles, answering the open question of Boneh, Halevi, Hamburg and Ostrovsky [11]. Our result confirms that to achieve circular-security, one needs novel, dedicated schemes and analyses, vindicating work in this line [11, 12, 1].

CONTEXT. Let us now back up to provide some context for agility and describe our other contributions in this area. Cryptographic code usually has a suite of allowed schemes of any particular type. (For example, authenticated encryption.) But new standards or proposed standards appear at a rapid rate. Cryptographic code written today needs to be able to easily incorporate schemes that will appear in the future. This has been recognized and enunciated in developer forums, where the term "agility" has been used to refer to the ability to easily add schemes to an existing suite by structuring code to allow schemes to be substituted in a blackbox manner. The IETF is currently considering adding agility to the widely deployed RADIUS protocol [26]. Resources for software professionals, like a recent Microsoft Developer Network Magazine article [29], encourage agility.

Keys for use with the existing schemes will, however, already have been distributed. Changing them or getting new ones distributed and certified is difficult and error-prone. Agility, thus, would ask that it be possible to maintain the existing key, using this single key with multiple schemes, both new and old. (The presence of new schemes will not preclude use of the old ones. Data encrypted under old schemes and then stored still has to be decrypted, and legacy systems must be supported.)

Agility is of course possible only among schemes that have keys of the same type or length. (An algorithm with a 128-bit key and another with a 256-bit key shouldn't share a key.) But key-compatibility is common given that many schemes will use the same underlying blockciphers or hash functions. Popular proposed or standardized symmetric authenticated encryption (AE) schemes like CCM [30], OCB [27], CWC [20], GCM [23], and EAX [9], for example, all use a 128-bit AES key.

Cryptographers have always recommended key separation, usually interpreted as asking that schemes for different purposes not use the same key. Thus, MAC and symmetric encryption should use different keys, as should public-key encryption and digital signatures. Assuming this type of separation is in place, the issue is whether different schemes for the *same* goal, for example authenticated encryption or PRF, may securely use the same key. This is what agility captures.

We clarify that agility is about *individually secure* schemes sharing a key. It is not about what happens when a scheme is broken and replaced by another that is (hopefully) secure. When that happens, you should not retain the old key since the attacks on the old scheme may already have compromised it.

In their work on chosen-protocol attacks, Kelsey, Schneier and Wagner [19] point both to the danger of using the same key across different schemes and the pressures that are likely to make this happen, the latter including the cost of certification of new keys, the spread of cryptographic APIs, and the limits posed on key-storage by smartcards. They are concerned mostly with different primitives (they call them protocols, for example, encryption and digital signature) sharing a key. Agility can be viewed as a class of chosen-protocol attacks in which the schemes, or protocols, are all for the same goal.

The present paper can serve as a developer guide for agility, pointing out what is possible and what is not. We provide formal definitions that enable a rigorous treatment of agility. With regard to results, on the positive side, we provide practical constructions, showing how to use PRFs and wPRFs as catalysts to confer agility on higher-level primitives like authenticated encryption. On the negative side we show that agility for the full set of schemes meeting some notion is usually unlikely. Let us now expand on all this.

Definitions. Agility is novel, definitionally, in that, unlike standard definitions of security, which apply to individual schemes, agility is a property of a *set* $\mathsf{\Gamma}$ of schemes that individually already meet some base notion of security. Thus, $\mathsf{\Gamma}$ might be the set of all PRFs or some subset thereof. It is as though one moves up one level in "types."

We appropriately extend the game defining base security so that the key is chosen just once yet an adversary can, via a *scheme argument*, pass in different schemes that will all use this key. The set $\mathsf{\Gamma}$ is said to be a-agile ($a \in \mathbb{N}$) with respect to the base security notion if, for all compatible, size a subsets $\mathsf{\Pi}$ of $\mathsf{\Gamma}$, the adversary advantage is negligible when its scheme arguments are drawn from $\mathsf{\Pi}$. (Compatible means the schemes in the set have keys of the same type and length.) In the body of the paper we exemplify with detailed definitions for the case of PRFs, wPRFs and authenticated encryption. In this framework, what we called wPRF agility above is the 2-agility of the set $\mathsf{\Gamma}$ of all wPRFs.

Foundations. The most basic theoretical question is whether a primitive (for example, PRF, AE) is agile, by which we mean that the set of *all* schemes that are individually secure is a-agile for $a \geq 2$. We answer this for a variety of primitives. Figure 1 summarizes our findings. As it shows, collision-resistant hash functions, when formalized as keyed families, are agile. (Practical functions like MD5, SHA1, SHA256 being unkeyed are trivially agile.) IND-CPA-secure public-key encryption schemes

Primitive	Agile?
PRFs, wPRFs, MACs, IND-CPA symmetric encryption, symmetric authenticated encryption, IND-CCA public-key encryption, digital signatures	No
Collision-resistant hash functions, IND-CPA public-key encryption	Yes

Figure 1: Agility status of some basic primitives.

are also agile. So two RSA-based public-key encryption schemes can share the same keys as long as only IND-CPA-security is desired. PRFs, MACs, IND-CPA secure symmetric encryption schemes, AE schemes, IND-CCA-secure public-key encryption schemes and digital signatures are *not* agile. We present counter-examples in the body of the paper for some of these, and others are similar.

The above results are relatively straightforward. The most interesting question was whether wPRFs are agile. As discussed at length above, we have answered the question in the negative by first making a connection to cyclic encryption and then answering an open question there.

The following shows why our focus on PRFs and wPRFs is not arbitrary and also shows how, despite the above, to get strong agility in practice.

PRF-DERIVED AGILITY. Our DtE (Derive-then-Encrypt) transform associates to a given PRF ff and a given AE scheme es a new AE scheme $\mathsf{es}_{\mathsf{ff}}$ in which ff under the base key is used to derive a subkey that is then used for es. This turns out to have strong agility properties. Specifically, let Γ be the set of all AE schemes $\mathsf{es}_{\mathsf{ff}}$ as es ranges over *all* AE schemes. Then, for any a, the set Γ is a-agile with respect to AE. The short rendition of this is that AE has now, effectively, become agile. The lack of agility in the primitive itself has been circumvented by using it not directly but within the scope of our construction which can in fact maintain a single key and yet be able to swap in and securely use *any* AE scheme. This is of direct interest in practice where, as we have seen, there are numerous existing and emerging options for AE such as CCM, OCB, CWC, GCM and EAX. Even this (small) set of schemes is probably not agile. But it becomes so when used via our construction.

The above requires that the ff scheme be fixed. But it too is a primitive for which agility may be desirable. If we want to be able to use arbitrary PRFs, the above-noted lack of agility of the primitive means we are out of luck. But in practice these are blockciphers for which there may be only a small set of relevant choices. (For example, all AES finalists.) This set may in fact be agile.

WPRF-DERIVED AGILITY. DtE uses a PRF to make AE agile. Could we use a wPRF instead? This is attractive for two reasons. The first is that a wPRF is a weaker assumption on a blockcipher than a PRF. The second is that, as a result, a set of blockciphers is more likely to be agile with respect to wPRF than to PRF. (We cannot of course hope for agility with respect to all wPRFs since that class is not agile. As above, however, we'd like to get it for as large a subset of the class as possible.) However, the obvious way to extend the construction, namely upon encryption to pick a random R, use $F_K(R)$ as the AE key where K is the base key and F our wPRF, and return R with the ciphertext, fails to achieve AE, even in the absence of agility. What we instead observe is that some of the existing transforms of wPRFs to PRFs from [25, 21, 22] have a form that make them agility preserving, meaning that if the wPRF is drawn from an agile set then the result is an agile set of PRFs. This yields a construct that is more robust and in practice more agile than DtE but also more expensive.

RELATED WORK. Agility is part of the broader issue of the security of key reuse [19]. Haber and Pinkas [15] analyze the security of several specific constructions of public-key encryption and digital signatures when a single public/secret key pair is used both for encryption and signing or for two encryption schemes or digital signature schemes simultaneously. They do not consider the general

problem of key reuse and focus on public-key primitives.

Key-dependent message security and its special case, circular security, were defined in concurrent works [10, 13] and recent work gives several constructions of various primitives meeting different flavors of security [17, 18, 11, 3, 12, 1]. At the end of Section 4 we discuss exactly how our counterexample for circular security fits into prior work.

2 Preliminaries

NOTATION AND CONVENTIONS. If x is a string then $|x|$ denotes its length, and if S is a set then $|S|$ denotes its size. The empty string is denoted ε. If $a = (a_1, \ldots, a_n)$ then $(a_1, \ldots, a_n) \leftarrow a$ means we parse a as shown. Unless otherwise indicated, an algorithm may be randomized. "PT" stands for "polynomial time." By $y \leftarrow A(x_1, x_2, \ldots; r)$ we denote the operation of running A on inputs $x_1, x_2, \ldots$ and coins $r \in \{0,1\}^*$. We denote by $y \xleftarrow{\$} A(x_1, x_2, \ldots)$ the operation of picking r at random and letting $y \leftarrow A(x_1, x_2, \ldots; r)$. (The coins are chosen from a space that may depend on the inputs.) We denote by $[A(x_1, x_2, \ldots)]$ the set of all possible outputs of A on inputs $x_1, x_2, \ldots$. We denote by $k \in \mathbb{N}$ the security parameter and by 1^k its unary encoding.

GAMES. Our definitions and proofs use the language of code-based games [8]. Recall that a game — look at Figure 2 for examples— has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. If **Initialize** is missing it is understood to be the trivial procedure that merely returns 1^k. A game G is executed with an adversary A as follows. First, **Initialize**(if present) executes, and its outputs are the inputs to A. Then A executes, its oracle queries being answered by the corresponding procedures of G. When A terminates, its output becomes the input to the **Finalize** procedure. The output of the latter, denoted G^A, is called the output of the game, and we let "G^A" denote the event that this game output takes value true. The running time of an adversary is the worst case time of the execution of the adversary with the game defining its security, so that the execution time of the called game procedures is included.

FUNCTION FAMILIES. The (common) syntax we use for PRFs and wPRFs is more general than may be usual because we will (later) need to consider schemes defined via families of groups. An FF-scheme ("FF" stands for "Function Family") $\mathsf{ff} = (\mathsf{ff.Pg}, \mathsf{ff.Kg}, \mathsf{ff.f}, \mathsf{ff.DomR}, \mathsf{ff.RngR})$ consists of a parameter generator, a key generator, an evaluator, a domain recognizer and a range recognizer, all PT algorithms, the last three deterministic. We require that $\mathsf{ff.f}(\mathit{pars}, K, \cdot)$: $\mathsf{ff.Dom}(\mathit{pars}) \to \mathsf{ff.Rng}(\mathit{pars})$ for every $k \in \mathbb{N}$, $\mathit{pars} \in [\mathsf{Pg}(1^k)]$ and $K \in [\mathsf{Kg}(\mathit{pars})]$, where $\mathsf{ff.Dom}(\mathit{pars}) = \{x : \mathsf{ff.DomR}(\mathit{pars}, x) = 1\}$ and $\mathsf{ff.Rng}(\mathit{pars}) = \{y : \mathsf{ff.RngR}(\mathit{pars}, y) = 1\}$. We require that one can sample from $\mathsf{ff.Dom}(\mathit{pars})$ and $\mathsf{ff.Rng}(\mathit{pars})$ in PT on input pars. We also require that $|\mathsf{ff.Dom}(\mathit{pars})| \geq 2^k$ for all $\mathit{pars} \in [\mathsf{ff.Pg}(1^k)]$ and all $k \in \mathbb{N}$. (PRFs on tiny domains are trivially constructed and don't even imply one-way functions. This convention rules them out.)

ENCRYPTION SYNTAX. Our syntax for encryption is general enough to cover both symmetric and asymmetric encryption, which will save us from repeating similar security definitions for both cases. A ENC-scheme ("ENC" stands for encryption) $\mathsf{es} = (\mathsf{es.Pg}, \mathsf{es.Kg}, \mathsf{es.Enc}, \mathsf{es.Dec}, \mathsf{es.MsgR}, \mathsf{es.CtxtR})$ consists of 6 PT algorithms: a parameter generator, a key generator, encryption and decryption algorithms, a plaintext recognizer and a ciphertext recognizer. The decryption algorithm is deterministic. On input $\mathit{pars} \in [\mathsf{es.Pg}(1^k)]$, the key generator outputs a triple $(\mathit{ek}, \mathit{dk}, \mathit{pk})$, where ek is an encryption key, dk is a decryption key and pk is a public key. We say that the encryption scheme is symmetric if it is always the case that $\mathit{pk} = \perp$. (In which case we may assume wlog $\mathit{ek} = \mathit{dk}$.) We say it is asymmetric if it is always the case that $\mathit{ek} = \mathit{pk}$. We require that there is a polynomial $r(\cdot)$, called the *number of coins used by* $\mathsf{es.Enc}$, such that $\mathsf{es.Enc}$ draws its coins from $\{0,1\}^{r(k)}$ whenever its first input is $\mathit{pars} \in [\mathsf{es.Pg}(1^k)]$. We require that $\mathsf{es.Enc}(\mathit{pars}, \mathit{ek}, \cdot\,; R)$: $\mathsf{es.Msg}(\mathit{pars}) \to \mathsf{es.Ctxts}(\mathit{pars})$ and $\mathsf{es.Dec}(\mathit{pars}, \mathit{dk}, \mathsf{es.Enc}(\mathit{pars}, \mathit{ek}, M; R)) = M$ for all $R \in \{0,1\}^{r(k)}$, all $M \in \mathsf{es.Msg}(\mathit{pars})$, all

proc KeySetup(ff)
$pars \xleftarrow{\$} \mathsf{ff.Pg}(1^k)$; $K \xleftarrow{\$} \mathsf{ff.Kg}(pars)$
$b \xleftarrow{\$} \{0,1\}$
Return $pars$

proc Fn(ff, x)
If $b = 1$ then $y \leftarrow \mathsf{ff.f}(pars, K, x)$
Else $y \xleftarrow{\$} \mathsf{ff.Rng}(pars)$
Return y

proc Finalize(b')
Return $(b' = b)$

proc KeySetup(ff)
$pars \xleftarrow{\$} \mathsf{ff.Pg}(1^k)$; $K \xleftarrow{\$} \mathsf{ff.Kg}(pars)$
$b \xleftarrow{\$} \{0,1\}$
Return $pars$

proc Fn(ff)
$x \xleftarrow{\$} \mathsf{ff.Dom}(pars)$
If $b = 1$ then $y \leftarrow \mathsf{ff.f}(pars, K, x)$
Else $y \xleftarrow{\$} \mathsf{ff.Rng}(pars)$
Return (x, y)

proc Finalize(b')
Return $(b' = b)$

Figure 2: Game $\mathsf{FF.PR.Gm}_k$, on the left, and game $\mathsf{FF.wPR.Gm}_k$, on the right, for $k \in \mathbb{N}$.

$(ek, dk, pk) \in [\mathsf{es.Kg}(pars)]$, all $pars \in [\mathsf{es.Pg}(1^k)]$ and all $k \in \mathbb{N}$, where $\mathsf{es.Msg}(pars) = \{M : \mathsf{es.MsgR}(pars, M) = 1\}$ and $\mathsf{es.Ctxts}(pars) = \{C : \mathsf{es.CtxtR}(pars, C) = 1\}$. We require that one can sample from $\mathsf{es.Msg}(pars)$ and $\mathsf{es.Ctxts}(pars)$ in PT on input $pars$.

3 Agility definitions

The notions of security we usually define apply to schemes, saying what it means for the scheme to be secure. (For example, a function family is a PRF if ...). Agility is different. It is not a property of an individual scheme but of a set of schemes relative to some (standard) security notion for these schemes. Thus, we might have a set of PRFs and talk of their agility with respect to the PRF notion.

The template for an agility definition is as follows. We start with a syntax. (For example, FF for function families or ENC for encryption schemes.) We then provide a sequence of games to capture agility with respect to a (usually standard) underlying notion of security for individual schemes. (For example, games $\mathsf{FF.PR.Gm}_k$, $k \in \mathbb{N}$, on the left side of Figure 2. The underlying notion here, called PR for pseudorandomness, is the standard PRF notion.) The unusual feature of the games is to have a *scheme argument*, meaning the adversary may provide procedures a scheme (of the syntax being considered) whose algorithms the game then uses. Agility of a set Π of schemes is measured by allowing the adversary to use *different* members of Π (the choice at its discretion) in the role of scheme argument, *with the underlying key remaining the same*. (For this to be possible, Π must be consistent in the sense that all its schemes have keys of the same syntactic form.) Some advantage will be associated to an adversary and Π, and thence we will get a definition of agility for Π. Restricting attention to a set Π consisting of a single scheme (corresponding to an adversary whose scheme argument is always this one scheme) recovers the base underlying security notion (for example, PRF) for this scheme, thereby saving us from defining it separately and also confirming that agility is a natural extension of the base notion.

We could carry through the above in a fully general way. For concreteness and simplicity, however, we exemplify with agility definitions for three primitives important to this paper, namely PRFs, wPRFs and authenticated encryption. To expose the underlying unity, however, we use a uniform notation, where Sec-security of Syntax-schemes, for example, refers to security of schemes of the shown syntax with regard to the shown base notion of security. We hope the reader will forgive the standard notion of a PRF ending up, for this reason, being called PR-security of an FF-scheme. The definitional templates here can be easily lifted and adapted to define agility of other primitives.

PRF Agility. Say a set Π of FF-schemes is *compatible* if all $\mathsf{ff} \in \Pi$ have the same parameter generator

and also all $\mathsf{ff} \in \Pi$ have the same key generator. Consider the games $\mathsf{FF.PR.Gm}_k$ ($k \in \mathbb{N}$) on the left side of Figure 2. Call an adversary A Π-restricted if the scheme arguments in its queries are all drawn from Π, it makes only one **KeySetup** query, this being its first oracle query, it never repeats an oracle query, and any $\mathbf{Fn}(\mathsf{ff}, x)$ query it makes satisfies $x \in \mathsf{ff.Dom}(pars)$. (All this must hold with probability 1 regardless of how queries are answered. Π being compatible means the parameter and key generation algorithms invoked by **KeySetup** will be the same regardless of the FF-scheme that it is provided as input.) Let $\mathbf{Adv}^{\mathsf{PR}}_{\Pi,A}(k) = 2\Pr[\mathsf{FF.PR.Gm}^A_k] - 1$ for any compatible set Π of FF-schemes and Π-restricted adversary A. ("PR" stands for "pseudorandom".)

We say that a finite, compatible set Π of FF-schemes is agile with respect to PR if the function $\mathbf{Adv}^{\mathsf{PR}}_{\Pi,A}(\cdot)$ is negligible for all PT, Π-restricted adversaries A. We say that a (not necessarily compatible or finite) set Γ of FF-schemes is a-agile with respect to PR ($a \in \mathbb{N}$) if every size a, compatible subset $\Pi \subseteq \Gamma$ of Γ is agile with respect to PR. We say that Γ is agile with respect to PR if it is a-agile with respect to PR for every $a \in \mathbb{N}$.

We recover the usual notion of an FF-scheme ff being a PRF —which we call PR-security here for uniformity— as agility of the singleton set $\{\mathsf{ff}\}$ with respect to PR. To spell it out, FF-scheme ff is PR-secure if the function $\mathbf{Adv}^{\mathsf{PR}}_{\{\mathsf{ff}\},A}(\cdot)$ is negligible for all PT $\{\mathsf{ff}\}$-restricted adversaries A. Then ff is PR-secure iff it is a PRF, and the set FF.PR.Sch of all PR-secure FF-schemes is the set of all PRFs.

wPRF AGILITY. The above is easily adapted to wPRFs. The games $\mathsf{FF.wPR.Gm}_k$ ($k \in \mathbb{N}$) are now those on the right side of Figure 2. Call an adversary A Π-restricted if the scheme arguments in its queries are all drawn from Π, it makes only one **KeySetup** query, this being its first oracle query, and it never repeats an oracle query. Let $\mathbf{Adv}^{\mathsf{wPR}}_{\Pi,A}(k) = 2\Pr[\mathsf{FF.wPR.Gm}^A_k] - 1$ for any compatible set Π of FF-schemes and Π-restricted adversary A. ("wPR" stands for "weakly pseudorandom".) We say that a finite compatible set Π of FF-schemes is agile with respect to wPR if the function $\mathbf{Adv}^{\mathsf{wPR}}_{\Pi,A}(\cdot)$ is negligible for all PT, Π-restricted adversaries A. We say that a (not necessarily compatible or finite) set Γ of FF-schemes is a-agile with respect to wPR ($a \in \mathbb{N}$) if every size a, compatible subset $\Pi \subseteq \Gamma$ of Γ is agile with respect to wPR. We say that Γ is agile with respect to wPR if it is a-agile with respect to wPR for every $a \in \mathbb{N}$. As before we recover the usual notion of an FF-scheme ff being a wPRF, which we call wPR-security here, as agility of the singleton set $\{\mathsf{ff}\}$ with respect to wPR, and let FF.wPR.Sch be the set of all wPR-secure FF schemes.

AGILITY FOR AUTHENTICATED ENCRYPTION. Early definitions of AE [7] gave separate privacy and integrity requirements. Our agility games $\mathsf{ENC.AuE.Gm}_k$ ($k \in \mathbb{N}$) given in Figure 3, where $\mathsf{es} = (\mathsf{es.Pg}, \mathsf{es.Kg}, \mathsf{es.Enc}, \mathsf{es.Dec}, \mathsf{es.MsgR}, \mathsf{es.CtxtR})$ is a ENC-scheme, are instead based on a unified definition in the style of Rogaway and Shrimpton [28]. The privacy requirement is indistinguishability from random [27] (IND-R). This strengthening of the usual notion of [6] tends to be naturally achieved by block cipher modes of operation [6]. The games of course have the scheme argument that is central to agility. The definitions proceed in direct analogy to the above. To detail them, first say a set Π of ENC-schemes is *compatible* if all $\mathsf{es} \in \Pi$ have the same parameter generator and also all $\mathsf{es} \in \Pi$ have the same key generator. Call an adversary A Π-restricted if the scheme arguments in its queries are all drawn from Π, it makes only one **KeySetup** query, this being its first oracle query, and any $\mathbf{RoR}(\mathsf{es}, M)$ query it makes satisfies $M \in \mathsf{es.Msg}(pars)$. Let $\mathbf{Adv}^{\mathsf{AuE}}_{\Pi,A}(k) = 2\Pr[\mathsf{ENC.AuE.Gm}^A_k] - 1$ for any finite, compatible set Π of ENC-schemes and Π-restricted adversary A. ("AuE" stands for "authenticated encryption".) We say that a compatible set Π of ENC-schemes is agile with respect to AuE if the function $\mathbf{Adv}^{\mathsf{AuE}}_{\Pi,A}(\cdot)$ is negligible for all PT, Π-restricted adversaries A. We say that a (not necessarily compatible or finite) set Γ of ENC-schemes is a-agile with respect to AuE ($a \in \mathbb{N}$) if every size a, compatible subset $\Pi \subseteq \Gamma$ of Γ is agile with respect to AuE. We say that Γ is agile with respect to AuE if it is a-agile with respect to AuE for every $a \in \mathbb{N}$. We recover the usual notion of an ENC-scheme es being an authenticated encryption scheme, which we call AuE-security here, as agility of the singleton set $\{\mathsf{es}\}$ with respect to AuE and let ENC.AuE.Sch be the set of all AuE-secure ENC schemes.

proc KeySetup(es)
$pars \xleftarrow{\$} \mathsf{es.Pg}(1^k)$; $(ek, dk, pk) \xleftarrow{\$} \mathsf{es.Kg}(pars)$
$S \leftarrow \emptyset$; $b \xleftarrow{\$} \{0,1\}$
Return $(pars, pk)$

proc RoR(es, M)
If $b = 1$ Then $C \xleftarrow{\$} \mathsf{es.Enc}(pars, ek, M)$
Else $C \xleftarrow{\$} \mathsf{es.Ctxts}(pars)$
$S \leftarrow S \cup \{(\mathsf{es}, C)\}$
Return C

proc Dec(es, C)
If $(\mathsf{es}, C) \in S$ Then Return $\perp$
If $b = 1$ Then $M \leftarrow \mathsf{es.Dec}(pars, dk, C)$
Else $M \leftarrow \perp$
Return M

proc Finalize(b')
Return $(b' = b)$

Figure 3: Game $\mathsf{ENC.AuE.Gm}_k$ for $k \in \mathbb{N}$.

This definition is for both symmetric and asymmetric schemes, even though the latter can only meet it if no **Dec** queries are allowed. The latter restriction results in the IND-R notion of privacy under CPA that will be useful later.

Discussion. Our definition of a (possibly infinite) set Γ of schemes being agile is that every *finite* compatible subset Π of it is agile, meaning the advantage of any Π-restricted adversary is negligible. One might ask why we did not simply define Γ to be agile if the advantage of any Π-restricted adversary is negligible for every compatible but not necessarily finite subset Π of Γ. The reason is that this notion (let's call it strong agility) is generically unachievable, meaning unachievable regardless of what is the base notion of security, whether PRF, wPRF, AE or any other. To explain why, say S is a scheme meeting the base notion of security. We show how to build an infinite, compatible set $\Pi = \{\mathsf{S}^i\}_{i=1}^{\infty}$ of schemes such that each S^i, taken individually, continues to meet the base notion of security, but Π is not strongly agile because there exists a Π-restricted adversary with non-negligible advantage. The idea is that S^i behaves insecurely when the security parameter k equals i, and otherwise behaves like S. For each i the scheme S^i is still secure because only finitely many values of k lie below i. On the other hand, an adversary against the agility of Π could, when the security parameter is k, provide S^k as the scheme argument, attacking only this scheme and exploiting its insecurity to get a high advantage.

4 Negative results

We consider the central foundational question about agility, namely whether it can be achieved for the set of *all* secure schemes of a given type. We begin by showing how to rule this out quite simply for PRFs and AE. Similar methods yield negative results for many other primitives, but *not* for wPRFs. We establish the connection between the latter and circular encryption, and then provide our negative result on circular encryption, namely that IND-R does not imply CYC. This will be used to establish non-agility of wPRFs.

4.1 Some simple non-agility results

Non-agility of PRFs. PRF agility is important because PRFs model blockciphers, for which agility is important in practice, and also (cf. Section 5) because PRFs are "universal" with regard to providing agility in the sense that if a set of PRFs is agile we can use it to build a class of authenticated encryption schemes that is agile with respect to *arbitrary* substitution of the encryption. The following says the set FF.PR.Sch of all PRFs is not a-agile for $a \geq 2$ under the minimal assumption that PRFs exist.

Proposition 4.1 *Let $a \geq 2$. If the set* FF.PR.Sch *of all* PR*-secure* FF*-schemes is not empty then it is not a-agile with respect to* PR*.*

Proof of Proposition 4.1: Let $\mathsf{ff} \in \mathsf{FF.PR.Sch}$. We construct a PR-secure scheme $\overline{\mathsf{ff}}$ such that the set $\Pi = \{\mathsf{ff}, \overline{\mathsf{ff}}\}$ is not 2-agile, meaning the two PRFs cannot securely use the same key. The Proposition follows.

For the construction, we assume points in the range of ff are bitstrings. This is wlog since they can always be encoded as such. The parameter generator, key generator and domain recognizer of $\overline{\mathsf{ff}}$ are the same as those of ff. On input $pars, K, x$, the evaluator $\overline{\mathsf{ff}}.\mathsf{f}$ lets $y \leftarrow \mathsf{ff}.\mathsf{f}(pars, K, x)$ and returns the bitwise complement $\overline{y}$ of y. The new FF-scheme has range defined by $\overline{\mathsf{ff}}.\mathsf{Rng}(pars) = \{\, \overline{y} \;:\; y \in \mathsf{ff}.\mathsf{Rng}(pars) \,\}$.

It is easy to see that $\overline{\mathsf{ff}}$ is PR-secure (meaning, is a PRF) assuming ff is. The interesting question is what happens when they share a key. Consider the Π-restricted adversary A that on input $pars$, begins with a $\mathbf{KeySetup}(\mathsf{ff})$ query. Then it lets $x \xleftarrow{\$} \mathsf{ff}.\mathsf{Dom}(pars)$ and lets $y \leftarrow \mathbf{Fn}(\mathsf{ff}, x)$ and $z \leftarrow \mathbf{Fn}(\overline{\mathsf{ff}}, x)$. (Note the definition of a Π-restricted adversary required it to not repeat an oracle query. This condition is met because $(\mathsf{ff}, x) \neq (\overline{\mathsf{ff}}, x)$.) If $z = \overline{y}$ it outputs 1, else 0.

We assume $|\mathsf{ff}.\mathsf{Rng}(pars)| \geq 2$. This is wlog because there are standard ways to extend the range of a PRF. Now we claim that $\mathbf{Adv}^{\mathsf{PR}}_{\Pi,A}(\cdot) \geq 1/2$, which shows that Π is not 2-agile as desired. We justify the claim as follows. If $b = 1$ in game $\mathsf{FF.PR.Gm}_k$ then $z = \overline{y}$ and A returns 1. If $b = 0$, it returns 1 with the probability that $z = \overline{y}$ when z is drawn at random from $\overline{\mathsf{ff}}.\mathsf{Rng}(pars)$ and y is drawn at random from $\mathsf{ff}.\mathsf{Rng}(pars)$. But both sets $\mathsf{ff}.\mathsf{Rng}(pars)$ and $\overline{\mathsf{ff}}.\mathsf{Rng}(pars)$ have size at least two, so the probability is at most $1/2$. ▌

The above says the class $\mathsf{FF.PR.Sch}$ of all PRFs is not a-agile for $a \geq 2$. But it is still possible that some proper subsets Γ of $\mathsf{FF.PR.Sch}$ are a-agile for some $a \geq 2$. This is interesting for practice, where one may be interested in a certain specific and quite small collection of schemes, and is why we defined agility for subsets of $\mathsf{FF.PR.Sch}$ rather than merely for the whole.

Extensions. Similar ideas exclude agility for many other primitives. Let us illustrate by sketching a counterexample to show that $\mathsf{ENC.AuE.Sch}$ is not a-agile with respect to AuE for any $a > 1$. Given $\mathsf{es} \in \mathsf{ENC.AuE.Sch}$ we construct $\overline{\mathsf{es}} \in \mathsf{ENC.AuE.Sch}$ which given $pars, K, M$ lets $C \xleftarrow{\$} \mathsf{es}(pars, K, M)$ and returns $\overline{C}$. We claim $\{\mathsf{es}, \overline{\mathsf{es}}\}$ is not agile. This is because an attacker can query $\mathbf{RoR}(\mathsf{es}, M)$ to get back C and then query $\mathsf{Dec}(\overline{\mathsf{es}}, \overline{C})$ to get back a message that will be M if the challenge bit b was 1 and is unlikely to be M otherwise.

wPRFs? Our counter-example for PRF agility relied on having two functions compute related outputs on the same, adversary-given input. This type of approach fails to yield a counter-example for wPRFs because the inputs are not under adversary control. In particular, there seems to be no way to attack the wPRF-agility of the two PRFs built in the proof of Proposition 4.1 because the attack relied on the ability of the adversary to feed the same input to the two functions.

The fact that only random, non-adversarially controlled inputs are used seems to give wPRFs a lot of agility potential. Yet, if we try to prove agility, we get stuck because it is unclear how to reduce the agility of a pair of functions to the individual ones in a blackbox way when they share a key. We could explore blackbox separation results, but these are only weak indications of separation since they rule out proving the result by certain proof technique rather than ruling out the result itself. Instead, we will get a counter-example just like we did for PRFs, by making a connection with circular encryption and then establishing a negative result on the latter. Both steps are of independent interest.

4.2 Auxiliary definitions for encryption

We say that ENC-scheme es is IND-R-secure if $\mathbf{Adv}^{\mathsf{AuE}}_{\{\mathsf{es}\},A}(\cdot)$ is negligible for all PT, $\{\mathsf{es}\}$-restricted adversaries A that make no **Dec** queries. This strong version of privacy under CPA from [27] implies the standard IND-CPA and is achieved by blockcipher modes of operation like CTR and CBC [6].

proc Initialize
$pars \xleftarrow{\$} \mathsf{es.Pg}(1^k)$; $b \xleftarrow{\$} \{0,1\}$
$(ek_1, dk_1, pk_1) \xleftarrow{\$} \mathsf{es.Kg}(pars)$
$(ek_2, dk_2, pk_2) \xleftarrow{\$} \mathsf{es.Kg}(pars)$
Return $(pars, pk_1, pk_2)$

proc Cyc()
If $b = 1$ then
  $C_1 \xleftarrow{\$} \mathsf{es.Enc}(pars, ek_1, dk_2)$
  $C_2 \xleftarrow{\$} \mathsf{es.Enc}(pars, ek_2, dk_1)$
Else
  $C_1 \xleftarrow{\$} \mathsf{es.Ctxts}(pars)$
  $C_2 \xleftarrow{\$} \mathsf{es.Ctxts}(pars)$
Return (C_1, C_2)

proc Finalize(b')
Return $(b' = b)$

Algorithm $\mathsf{ff}_i.\mathsf{Pg}(1^k)$ // $i = 1, 2$
$fpars \xleftarrow{\$} \mathsf{ff.Pg}(1^k)$; $epars \xleftarrow{\$} \mathsf{es.Pg}(1^k)$
Return $pars \leftarrow (fpars, epars)$

Algorithm $\mathsf{ff}_i.\mathsf{Kg}((fpars, epars))$ // $i = 1, 2$
$L \xleftarrow{\$} \mathsf{ff.Kg}(fpars)$
$(K_1, K_1, \bot) \xleftarrow{\$} \mathsf{es.Kg}(epars)$
$(K_2, K_2, \bot) \xleftarrow{\$} \mathsf{es.Kg}(epars)$
Return (L, K_1, K_2)

Algorithm $\mathsf{ff}_1.\mathsf{f}((fpars, epars), (L, K_1, K_2), x)$
$r \leftarrow \mathsf{ff.f}(fpars, L, x)$; $y \leftarrow \mathsf{es.Enc}(epars, K_1, K_2; r)$
Return y

Algorithm $\mathsf{ff}_2.\mathsf{f}((fpars, epars), (L, K_1, K_2), x)$
$r \leftarrow \mathsf{ff.f}(fpars, L, x)$; $y \leftarrow \mathsf{es.Enc}(epars, K_2, K_1; r)$
Return y

Figure 4: Game $\mathsf{ENC.CYC.Gm}_k$ on the right, for $k \in \mathbb{N}$. On the right, algorithms for FF-schemes $\mathsf{ff}_1, \mathsf{ff}_2$ of the proof of Theorem 4.2.

Say es can *encrypt its own keys* if $dk \in \mathsf{es.Msg}(pars)$ for every $(ek, dk, pk) \in [\mathsf{es.Kg}(pars)]$, every $pars \in [\mathsf{es.Pg}(1^k)]$ and every $k \in \mathbb{N}$. For such an encryption scheme, let $\mathbf{Adv}^{\mathsf{CYC}}_{\mathsf{es},A}(k) = \Pr[\mathsf{ENC.CYC.Gm}^A_k]$ where the game in question is shown in Figure 4. Say es is CYC-secure if $\mathbf{Adv}^{\mathsf{CYC}}_{\mathsf{es},A}(\cdot)$ is negligible for all PT A. This asks that es have pseudorandom ciphertexts under a weak type of circular-encryption attack. The adversary is given access to samples, each of which is either a circular encryption of two keys or a pair of random strings, and is challenged to distinguish these two cases. Normal chosen-plaintext queries are not allowed, so CYC-security does not imply IND-CPA. Since our results are negative, this only strengthens them.

The definitions we have just given are for both the symmetric and asymmetric case.

4.3 Relating wPRF agility and encryption security

We show that if every pair of wPRFs is 2-agile, then every IND-R-secure symmetric ENC-scheme is CYC-secure. We say that an FF-scheme ff has *bit-output* if there is a polynomial $r(k) \geq k$ such that $\mathsf{ff.Rng}(pars) = \{0,1\}^{r(k)}$ for all $pars \in [\mathsf{ff.Pg}(1^k)]$ and all $k \in \mathbb{N}$.

Theorem 4.2 (**wPRF-A $\Longrightarrow$ IND-is-CYC**) *Suppose the set* FF.wPR.Sch *of all* wPR*-secure* FF*-schemes is* 2*-agile with respect to* wPR*, and further that* wPR*-secure* FF*-schemes with bit-output exist. Then every* IND-R*-secure symmetric encryption scheme that can encrypt its own keys is also* CYC*-secure.*

To prove this, we start with an IND-R-secure symmetric ENC-scheme es and then build a pair of wPRFs. Assuming wPRFs are 2-agile, this pair is 2-agile as a special case. We will then prove CYC-security of es based on the 2-agility of the wPRF pair.

Accordingly, let $\mathsf{es} = (\mathsf{es.Pg}, \mathsf{es.Kg}, \mathsf{es.Enc}, \mathsf{es.Dec}, \mathsf{Enc.MsgR}, \mathsf{Enc.CtxtR})$ be a symmetric ENC-scheme, and let $r(\cdot)$ be the number of coins used by es.Enc. Let $\mathsf{ff} = (\mathsf{ff.Pg}, \mathsf{ff.Kg}, \mathsf{ff.f}, \mathsf{ff.DomR}, \mathsf{ff.RngR})$ be a FF-scheme such that $\mathsf{ff.Rng}(pars) = \{0,1\}^{r(k)}$ for all $pars \in [\mathsf{ff.Pg}(1^k)]$ and all $k \in \mathbb{N}$. That an FF-scheme with such range exists follows from the assumption that FF-schemes with bit-output exist, for we can reduce output size by truncation or increase it by application of a PRG.

proc KeySetup(gg) // G_0, G_1, G_2

001 $fpars \xleftarrow{\$} \mathsf{ff.Pg}(1^k)$; $epars \xleftarrow{\$} \mathsf{es.Pg}(1^k)$
002 $L \xleftarrow{\$} \mathsf{ff.Kg}(fpars)$; $b \xleftarrow{\$} \{0,1\}$
003 $(K_1, K_1, \bot) \xleftarrow{\$} \mathsf{es.Kg}(epars)$
004 $(K_2, K_2, \bot) \xleftarrow{\$} \mathsf{es.Kg}(epars)$
005 Return $(fpars, epars)$

proc Fn(gg) // G_0

011 $x \xleftarrow{\$} \mathsf{ff.Dom}(fpars)$
012 $r \leftarrow \mathsf{ff.f}(fpars, L, x)$
013 $y \leftarrow \mathsf{es.Enc}(epars, K_i, K_j; r)$
014 Return (x, y)

proc Fn(gg) // G_1

101 $x \xleftarrow{\$} \mathsf{ff.Dom}(fpars)$
102 $r \xleftarrow{\$} \mathsf{ff.Rng}(fpars)$
103 $y \leftarrow \mathsf{es.Enc}(epars, K_i, K_j; r)$
104 Return (x, y)

proc Fn(gg) // G_2

201 $x \xleftarrow{\$} \mathsf{ff.Dom}(fpars)$
202 $y \xleftarrow{\$} \mathsf{es.Ctxts}(epars)$
203 Return (x, y)

proc Finalize(b') // G_0, G_1, G_2

020 Return $(b' = 1)$

Figure 5: Games for proof of Lemma 4.3.

For $i = 1, 2$ we now define FF-scheme $\mathsf{ff}_i = (\mathsf{ff}_i.\mathsf{Pg}, \mathsf{ff}_i.\mathsf{Kg}, \mathsf{ff}_i.\mathsf{f}, \mathsf{ff}_i.\mathsf{DomR}, \mathsf{ff}_i.\mathsf{RngR})$. The parameter, key-generation and evaluator algorithms are in Figure 4. Since es is symmetric, we are assuming wlog that the encryption and decryption keys are the same, so that output of the j-th execution of $\mathsf{es.Kg}(epars)$ in the code of $\mathsf{ff}_i.\mathsf{Kg}((fpars, epars))$ has the form $(K_j, K_j, \bot)$ $(j = 1, 2)$. The FF-schemes ff_1 and ff_2 are identical except for how their evaluators compute the output value y, where the roles of K_1 and K_2 are reversed. We let $\mathsf{ff}_1.\mathsf{DomR} = \mathsf{ff}_2.\mathsf{DomR} = \mathsf{ff.DomR}$, and $\mathsf{ff}_1.\mathsf{RngR} = \mathsf{ff}_2.\mathsf{RngR} = \mathsf{es.CtxtR}$. Since $\mathsf{ff}_1, \mathsf{ff}_2$ have the same parameter and key-generation algorithms, $\{\mathsf{ff}_1, \mathsf{ff}_2\}$ is compatible. The following says that each of ff_1 and ff_2, *taken individually*, is wPR-secure.

Lemma 4.3 *Suppose symmetric* ENC*-scheme* es *is* IND-R*-secure and* FF*-scheme* ff *is* wPR*-secure. Let* $\mathsf{ff}_1, \mathsf{ff}_2$ *be constructed from them as described above. Then* ff_1 *and* ff_2 *are both* wPR*-secure.*

Proof of Lemma 4.3: Let $i \in \{1, 2\}$ and let A_i be a $\{\mathsf{ff}_i\}$-restricted adversary against the wPR-security of ff_i making $q(\cdot)$ **Fn** queries. We design a $\{\mathsf{ff}\}$-restricted adversary A against the wPR-security of ff and a $\{\mathsf{es}\}$-restricted adversary B against the AuE-security of es such that

$$\mathbf{Adv}^{\mathsf{wPR}}_{\{\mathsf{ff}_i\},A_i}(k) \;\le\; \mathbf{Adv}^{\mathsf{wPR}}_{\{\mathsf{ff}\},A}(k) + \mathbf{Adv}^{\mathsf{AuE}}_{\{\mathsf{es}\},B}(k) \tag{1}$$

for all $k \in \mathbb{N}$. Adversary A makes $q(\cdot)$ **Fn** queries. Adversary B makes $q(\cdot)$ **RoR** queries and zero **Dec** queries. The running times of A, B are that of A_i. The lemma follows.

The games of Figure 5 will be executed with A_i. Remember this adversary begins with a **KeySetup** query and then makes $q(k)$ **Fn** queries. In all these queries, the scheme argument, denoted gg in the games, will take value ff_i. Game G_0 is game $\mathsf{FF.wPR.Gm}_k$ with challenge bit set to 1, specialized to our case by using the definition of ff_i in terms of ff, es, and returning true when the adversary output is 1 rather than when it equals the challenge bit. We are letting $j = 2$ if $i = 1$ and $j = 1$ if $i = 2$, so that es encrypts message K_j under key K_i at lines 013, 103 as per the definition of ff_i. The formal scheme argument gg is never actually used in the code since we know it will always equal ff_i. We have

$$\begin{aligned}\mathbf{Adv}^{\mathsf{wPR}}_{\{\mathsf{ff}_i\},A_i}(k) &= \Pr[G_0^{A_i}] - \Pr[G_2^{A_i}] \\ &= (\Pr[G_0^{A_i}] - \Pr[G_1^{A_i}]) + (\Pr[G_1^{A_i}] - \Pr[G_2^{A_i}]) \;. \end{aligned} \tag{2}$$

Game G_1 picks r at random at line 102 rather than letting it be $\mathsf{ff.f}(fpars, L, x)$ as at line 012. We build adversary A so that

$$\Pr[G_0^{A_i}] - \Pr[G_1^{A_i}] \;\le\; \mathbf{Adv}^{\mathsf{wPR}}_{\{\mathsf{ff}\},A}(k) \;. \tag{3}$$

proc Initialize // G_0, G_1, G_2

001 $fpars \xleftarrow{\$} \mathsf{ff.Pg}(1^k)$; $epars \xleftarrow{\$} \mathsf{es.Pg}(1^k)$
002 $L \xleftarrow{\$} \mathsf{ff.Kg}(fpars)$
003 $(K_1, K_1, \bot) \xleftarrow{\$} \mathsf{es.Kg}(epars)$
004 $(K_2, K_2, \bot) \xleftarrow{\$} \mathsf{es.Kg}(epars)$
005 Return $(epars, \bot, \bot)$

proc Cyc() // G_0

010 $r_1 \xleftarrow{\$} \mathsf{ff.Rng}(fpars)$
011 $r_2 \xleftarrow{\$} \mathsf{ff.Rng}(fpars)$
012 $y_1 \leftarrow \mathsf{es.Enc}(epars, K_1, K_2; r_1)$
013 $y_2 \leftarrow \mathsf{es.Enc}(epars, K_2, K_1; r_2)$
014 Return (y_1, y_2)

proc Cyc() // G_1

101 $x_1 \xleftarrow{\$} \mathsf{ff.Dom}(fpars)$; $r_1 \leftarrow \mathsf{ff.f}(fpars, L, x_1)$
102 $x_2 \xleftarrow{\$} \mathsf{ff.Dom}(fpars)$; $r_2 \leftarrow \mathsf{ff.f}(fpars, L, x_2)$
103 $y_1 \leftarrow \mathsf{es.Enc}(epars, K_1, K_2; r_1)$
104 $y_2 \leftarrow \mathsf{es.Enc}(epars, K_2, K_1; r_2)$
105 Return (y_1, y_2)

proc Cyc() // G_2

201 $y_1 \xleftarrow{\$} \mathsf{es.Ctxts}(epars)$
202 $y_2 \xleftarrow{\$} \mathsf{es.Ctxts}(epars)$
203 Return (y_1, y_2)

proc Finalize(b') // G_0, G_1, G_2

020 Return $(b' = 1)$

Figure 6: Games for proof of Lemma 4.4.

On input 1^k, adversary A begins with the initializations

$$epars \xleftarrow{\$} \mathsf{es.Pg}(1^k)\ ;\ (K_1, K_1, \bot) \xleftarrow{\$} \mathsf{es.Kg}(epars)\ ;\ (K_2, K_2, \bot) \xleftarrow{\$} \mathsf{es.Kg}(epars)\ .$$

It then runs $A_i(1^k)$. When the latter makes its **KeySetup**(ff_i) query, A calls its own **KeySetup** oracle on input ff to get back $fpars$ and returns $(fpars, epars)$ to A_i. When A_i now makes a **Fn**(ff_i) query, A responds via

$(x, r) \xleftarrow{\$} \mathbf{Fn}(\mathsf{ff})$; $y \leftarrow \mathsf{es.Enc}(epars, K_i, K_j; r)$
Return (x, y)

where **Fn** in the code is A's own **Fn** oracle. When A_i halts with output b', adversary A returns b'.

We build adversary B so that

$$\Pr[G_1^{A_i}] - \Pr[G_2^{A_i}] \ \le\ \mathbf{Adv}^{\mathsf{AuE}}_{\{\mathsf{es}\},B}(k)\ . \tag{4}$$

On input 1^k, adversary B begins with the initializations

$$fpars \xleftarrow{\$} \mathsf{ff.Pg}(1^k)\ ;\ (K_j, K_j, \bot) \xleftarrow{\$} \mathsf{es.Kg}(epars)\ .$$

It then runs $A_i(1^k)$. When the latter makes its **KeySetup**(ff_i) query, B calls its own **KeySetup** oracle on input es to get back $epars$ and returns $(fpars, epars)$ to A_i. When A_i now makes a **Fn**(ff_i) query, A responds via

$y \xleftarrow{\$} \mathbf{RoR}(\mathsf{es}, K_j)$
Return (x, y)

When A_i halts with output b', adversary B returns b'.

Equation (1) follows from Equations (2), (3) and (4). ▌

The next lemma says that if $\{\mathsf{ff}_1, \mathsf{ff}_2\}$ is agile with respect to wPR, then es is CYC-secure.

Lemma 4.4 *Suppose* $\mathsf{ff}_1, \mathsf{ff}_2$ *are constructed as described above from symmetric* ENC*-scheme* es *and* wPR*-secure* FF*-scheme* ff. *Suppose* $\{\mathsf{ff}_1, \mathsf{ff}_2\}$ *is agile with respect to* wPR. *Then* es *is* CYC*-secure.*

Theorem 4.2 follows from these two lemmas.

Proof of Lemma 4.4: Let C be an adversary against the CYC-security of es making $q(\cdot)$ **Cyc** queries. We design a $\{\mathsf{ff}\}$-restricted adversary A against the wPR-security of ff and a $\{\mathsf{ff}_1, \mathsf{ff}_2\}$-restricted adversary B against the wPR agility of $\{\mathsf{ff}_1, \mathsf{ff}_2\}$ such that

$$\mathbf{Adv}^{\mathsf{CYC}}_{\mathsf{es},C}(k) \;\le\; \mathbf{Adv}^{\mathsf{wPR}}_{\{\mathsf{ff}\},A}(k) + \mathbf{Adv}^{\mathsf{wPR}}_{\{\mathsf{ff}_1,\mathsf{ff}_2\},B}(k) \tag{5}$$

for all $k \in \mathbb{N}$. Adversaries A, B make $2q(\cdot)$ **Fn** queries and have the same running time as C. The lemma follows.

The games of Figure 6 will be executed with C. Remember this adversary has input the output $(\mathit{epars}, \bot, \bot)$ of **Initialize** and then makes $q(k)$ **Cyc** queries. Game G_0 is game $\mathsf{ENC.CYC.Gm}_k$ with the challenge bit set to 1 and returning true when the adversary output equals 1 rather than when it equals the challenge bit. (There are some extra steps in **Initialize** that will be used in later games but don't affect G_0.) We have

$$\begin{aligned}\mathbf{Adv}^{\mathsf{CYC}}_{\mathsf{es},C}(k) &= \Pr[G_0^C] - \Pr[G_2^C] \\ &= (\Pr[G_0^C] - \Pr[G_1^C]) + (\Pr[G_1^C] - \Pr[G_2^C])\;. \end{aligned}\tag{6}$$

At lines 101, 102, game G_1 picks r_1, r_2 as the results of $\mathsf{ff.f}(\mathit{fpars}, L, \cdot)$ at random inputs rather then directly at random as at lines 016, 017. We build adversary A so that

$$\Pr[G_0^C] - \Pr[G_1^C] \;\le\; \mathbf{Adv}^{\mathsf{wPR}}_{\{\mathsf{ff}\},A}(k)\;. \tag{7}$$

On input 1^k, adversary A begins with the initializations

$$\mathit{epars} \xleftarrow{\$} \mathsf{es.Pg}(1^k)\;;\; (K_1, K_1, \bot) \xleftarrow{\$} \mathsf{es.Kg}(\mathit{epars})\;;\; (K_2, K_2, \bot) \xleftarrow{\$} \mathsf{es.Kg}(\mathit{epars})\;.$$

It then runs $C((\mathit{epars}, \bot, \bot))$. When the latter makes a $\mathbf{Cyc}()$ query, A responds via

$(x_1, r_1) \xleftarrow{\$} \mathbf{Fn}(\mathsf{ff})\;;\; y_1 \leftarrow \mathsf{es.Enc}(\mathit{epars}, K_1, K_2; r_1)$
$(x_2, r_2) \xleftarrow{\$} \mathbf{Fn}(\mathsf{ff})\;;\; y_2 \leftarrow \mathsf{es.Enc}(\mathit{epars}, K_2, K_1; r_2)$
Return (y_1, y_2)

where **Fn** in the code is A's own **Fn** oracle. When C halts with output b', adversary A returns b'.

We build adversary B so that

$$\Pr[G_1^C] - \Pr[G_2^C] \;\le\; \mathbf{Adv}^{\mathsf{wPR}}_{\{\mathsf{ff}_1,\mathsf{ff}_2\},B}(k)\;. \tag{8}$$

On input 1^k, adversary B begins with a $\mathbf{KeySetup}(\mathsf{ff}_1)$ query, obtaining $(\mathit{fpars}, \mathit{epars})$ in response. It then runs $C(\mathit{epars})$. When the latter makes a **Cyc** query, B responds via

$(x_1, y_1) \xleftarrow{\$} \mathbf{Fn}(\mathsf{ff}_1)\;;\; (x_2, y_2) \xleftarrow{\$} \mathbf{Fn}(\mathsf{ff}_2)$
Return (y_1, y_2)

When C halts with output b', adversary B returns b'.

Equation (5) follows from Equations (6), (7) and (8). ∎

4.4 IND-R-but-not-IND-CYC encryption schemes

THE SXDH ASSUMPTION. Our counterexample encryption scheme that is $\mathsf{IND\text{-}R}$-secure but not CYC-secure relies on the SXDH assumption [2] which we now formalize. A *group scheme* is a PT algorithm GS that on input 1^k outputs $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2)$, where p is a k-bit prime, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are descriptions of groups of order p, $\mathbf{e}$: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear map, and g_i is a generator for G_i, $i = 1, 2$. We assume that one can recognize and multiply elements of the groups

proc Initialize

$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2) \xleftarrow{\$} \mathsf{GS}(1^k)$; $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p^*$; $b \xleftarrow{\$} \{0,1\}$
Return $((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2), X_1, X_2)$

proc 2dh()

$y_1, y_2, z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p$
If $(b = 1)$ then $(Y_1, Z_1, Y_2, Z_2) \leftarrow (g_1^{y_1}, g_1^{x_1 y_1}, g_2^{y_2}, g_2^{x_2 y_2})$
Else $(Y_1, Z_1, Y_2, Z_2) \leftarrow (g_1^{y_1}, g_1^{z_1}, g_2^{y_2}, g_2^{z_2})$
Return (Y_1, Z_1, Y_2, Z_2)

proc Finalize(b')

Return $(b' = b)$

Figure 7: Game $\mathsf{SXDH}_{\mathsf{GS},k}$, for $k \in \mathbb{N}$, used to define the hardness of the SXDH problem in group scheme GS.

involved as well evaluate $\mathbf{e}(\cdot,\cdot)$ in time polynomial in k. The Symmetric External Diffie-Hellman (SXDH) assumption [2] is that the Decisional Diffie-Hellman problem is hard in both G_1 and G_2. Formally, let $\mathbf{Adv}^{\mathsf{SXDH}}_{\mathsf{GS},A}(k) = 2\Pr[\mathsf{SXDH}^A_{\mathsf{GS},k}] - 1$ where the game is in Figure 7. The SXDH problem is said to be hard for GS if $\mathbf{Adv}^{\mathsf{SXDH}}_{\mathsf{GS},A}(\cdot)$ is negligible for every PT A that makes only one **2dh** query.

The restriction to one query is not important. The following lemma, which will be useful later, says that more queries won't help. The proof is in Appendix A. It exploits a self-reducibility property of the DH problem. The version of the latter we need, stated in [5], is a variant of one from [24].

Lemma 4.5 *Let* GS *be a group scheme and* A *an adversary making* $q(\cdot)$ ***2dh*** *queries. Then there is an adversary* A_1 *making 1* ***2dh*** *query such that for all* $k \in \mathbb{N}$ *we have*

$$\mathbf{Adv}^{\mathsf{SXDH}}_{\mathsf{GS},A}(k) \;\le\; \mathbf{Adv}^{\mathsf{SXDH}}_{\mathsf{GS},A_1}(k) + \frac{2q(k)}{2^{k-1}} \; . \tag{9}$$

The running time of A_1 *is that of* A *plus the time for* $O(q)$ *exponentiations in the first two groups.*

We assume that a group scheme comes equipped with a PT "key derivation function" H that, for $i = 1, 2$, takes input $pars = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2)$ together with i and $Z \in \mathbb{G}_i$, and returns a point $H(pars, i, Z) \in \mathbb{Z}_p$. The only requirement we place on H is that for all $pars \in [\mathsf{GS}(1^k)]$ and both $i = 1, 2$, if Z is uniformly distributed over $\mathbb{G}_i$, then $H(pars, i, Z)$ is uniformly distributed over $\mathbb{Z}_p$. This requirement can be relaxed to allow a negligible deviation from uniform, and, at the cost of complications in the constructions, can even be dropped altogether.

IND-R-BUT-NOT-IND-CYC ENCRYPTION SCHEMES. The following says that if SXDH is true then we can build counterexample encryption schemes, both symmetric and asymmetric, which are IND-R-secure (and hence IND-CPA-secure) but are not CYC-secure.

Theorem 4.6 (**SXDH** $\Longrightarrow$ NOT **IND-is-CYC**) *Suppose there exists a group scheme in which the SXDH problem is hard. There there exist symmetric and asymmetric* ENC*-schemes which are* IND-R*-secure but not* CYC*-secure.*

To prove this, let GS be a group scheme for which SXDH is hard. For $j \in \{1, 2\}$, Figure 8 associates to GS the ENC-scheme $\mathsf{es}_j = (\mathsf{es}_j.\mathsf{Pg}, \mathsf{es}_j.\mathsf{Kg}, \mathsf{es}_j.\mathsf{Enc}, \mathsf{es}_j.\mathsf{Dec}, \mathsf{es}_j.\mathsf{MsgR}, \mathsf{es}_j.\mathsf{CtxtR})$. ENC-scheme es_1 is symmetric and ENC-scheme es_2 is asymmetric. Notice both schemes can encrypt their own keys. (That is, the decryption keys are in the message space.) As described the schemes do not use coins that are bitstrings of some length $r(\cdot)$ depending only on the security parameter as our definition requires, but they may be easily modified to do this while retaining the attributes given by the Lemmas below. The following says that both schemes are IND-R-secure (and hence IND-CPA secure) assuming SXDH.

Algorithm $\mathsf{es}_j.\mathsf{Pg}(1^k)$ // $j = 1, 2$
$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2) \xleftarrow{\$} \mathsf{GS}(1^k)$
$pars \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2)$
Return $pars$

Algorithm $\mathsf{es}_1.\mathsf{Kg}(pars)$
$x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p^*$
$X_1 \leftarrow g_1^{x_1}$; $X_2 \leftarrow g_2^{x_2}$
$dk \leftarrow (x_1, x_2)$; $ek \leftarrow (X_1, X_2)$
Return $(ek, dk, \perp)$

Algorithm $\mathsf{es}_2.\mathsf{Kg}(pars)$
$x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p^*$
$X_1 \leftarrow g_1^{x_1}$; $X_2 \leftarrow g_2^{x_2}$
$dk \leftarrow (x_1, x_2)$; $ek \leftarrow (X_1, X_2)$
Return (ek, dk, ek)

Algorithm $\mathsf{es}_j.\mathsf{Enc}(pars, ek, (m_1, m_2))$ // $j = 1, 2$
$(X_1, X_2) \leftarrow ek$; $y_1, y_2, u_1, u_2 \xleftarrow{\$} \mathbb{Z}_p$
$Y_1 \leftarrow g_1^{y_1}$; $U_1 \leftarrow g_1^{u_1}$; $Z_1 \leftarrow X_1^{y_1}$; $T_1 \leftarrow X_1^{u_1/m_2}$
$Y_2 \leftarrow g_2^{y_2}$; $U_2 \leftarrow g_2^{u_2}$; $Z_2 \leftarrow X_2^{y_2}$; $T_2 \leftarrow X_2^{u_2/m_1}$
$c_1 \leftarrow m_1 + H(pars, 1, Z_1)$
$c_2 \leftarrow m_2 + H(pars, 2, Z_2)$
$C \leftarrow (Y_1, U_1, T_1, Y_2, U_2, T_2, c_1, c_2)$
Return C

Algorithm $\mathsf{es}_j.\mathsf{Dec}(pars, dk, C)$ // $j = 1, 2$
$(Y_1, U_1, T_1, Y_2, U_2, T_2, c_1, c_2,) \leftarrow C$
$(x_1, x_2) \leftarrow dk$
$m_1 \leftarrow c_1 - H(pars, 1, Y_1^{x_1})$
$m_2 \leftarrow c_2 - H(pars, 2, Y_2^{x_2})$
Return (m_1, m_2)

Figure 8: ENC-scheme es_1 is symmetric and es_2 is asymmetric. For $j = 1, 2$ the message space is $\mathsf{es}_j.\mathsf{Msg}(pars) = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ and the ciphertext space is $\mathsf{es}_j.\mathsf{Ctxts}(pars) = \mathbb{G}_1^3 \times \mathbb{G}_2^3 \times \mathbb{Z}_p^2$.

Lemma 4.7 *Let* $\mathsf{es}_1, \mathsf{es}_2$ *be the* ENC*-schemes associated to group scheme* GS *via Figure 8. Suppose the SXDH problem is hard in* GS*. Then* $\mathsf{es}_1, \mathsf{es}_2$ *are* IND-R*-secure.*

The tricky thing about the proof that follows is that the SXDH assumption will need to be invoked twice, once to say that the T_1, T_2 components of the ciphertext look random and then again to say the c_1, c_2 components also look random.

Proof of Lemma 4.7: Let $i \in \{1, 2\}$ and let A_i be ad $\{\mathsf{es}_i\}$-restricted adversary against the AuE-security of es_i making $q(\cdot)$ **RoR** queries and zero **Dec** queries. We design adversaries B_1, B_2 against the SXDH-security of GS such that

$$\mathbf{Adv}^{\mathsf{AuE}}_{\{\mathsf{es}_i\},A_i}(k) \;\le\; \mathbf{Adv}^{\mathsf{SXDH}}_{\mathsf{GS},B_1}(k) + \mathbf{Adv}^{\mathsf{SXDH}}_{\mathsf{GS},B_2}(k) \tag{10}$$

for all $k \in \mathbb{N}$. Adversaries B_1, B_2 make $q(\cdot)$ **2dh** queries and have running time that of A_i. The lemma follows from Lemma 4.5.

The games of Figure 9 will be executed with A_i. Remember this adversary begins with a **KeySetup** query and then makes $q(k)$ **RoR** queries. It makes no **Dec** queries, so the games omit this procedure. In all A_i's queries, the scheme argument, denoted es in the games, will take value es_i. Game G_0 is game $\mathsf{ENC.AuE.Gm}_k$ with challenge bit set to 1, specialized to our case by using the definition of es_i and returning true when the adversary output is 1 rather than when it equals the challenge bit. **Initialize** returns $pk = \perp$ in the symmetric case $\mathsf{es} = \mathsf{es}_1$ and $pk = ek$ in the asymmetric case $\mathsf{es} = \mathsf{es}_2$. We have

$$\begin{aligned}\mathbf{Adv}^{\mathsf{AuE}}_{\{\mathsf{es}_i\},A_i}(k) &= \Pr[G_0^{A_i}] - \Pr[G_4^{A_i}] \\ &= \textstyle\sum_{n=0}^{3}(\Pr[G_n^{A_i}] - \Pr[G_{n+1}^{A_i}])\ .\end{aligned} \tag{11}$$

Game G_1 picks Z_1, Z_2 at random at line 103 rather than the way they were picked at line 013. We build adversary B_1 so that

$$\Pr[G_0^{A_i}] - \Pr[G_1^{A_i}] \;\le\; \mathbf{Adv}^{\mathsf{SXDH}}_{\mathsf{GS},B_1}(k)\ . \tag{12}$$

On input $(pars, X_1, X_2)$ where $pars = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2)$, adversary B_1 lets $pk \leftarrow \perp$ if $i = 1$ and $pk \leftarrow (X_1, X_2)$ otherwise and then runs $A_i(1^k)$. When the latter makes its $\mathbf{KeySetup}(\mathsf{es}_i)$ query, B_1 returns $(pars, pk)$. When A_i now makes a $\mathbf{RoR}(\mathsf{es}_i, (m_1, m_2))$ query, B_1 responds via

proc KeySetup(es) // G_0, G_1, G_2, G_3, G_4

001 $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2) \xleftarrow{\$} \mathsf{GS}(1^k)$
002 $pars \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2)$
003 $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p^*$
004 $X_1 \leftarrow g_1^{x_1}$; $X_2 \leftarrow g_2^{x_2}$
005 $dk \leftarrow (x_1, x_2)$; $ek \leftarrow (X_1, X_2)$
006 If $\mathsf{es} = \mathsf{es}_1$ then $pk \leftarrow \perp$ else $pk \leftarrow ek$
007 Return $(pars, pk)$

proc RoR$(\mathsf{es}, (m_1, m_2))$ // G_0

011 For $j = 1, 2$ do
012 $y_j, u_j \xleftarrow{\$} \mathbb{Z}_p$; $l \leftarrow 1 + (j \bmod 2)$
013 $Y_j \leftarrow g_j^{y_j}$; $Z_j \leftarrow X_j^{y_j}$
014 $U_j \leftarrow g_j^{u_j}$; $S_j \leftarrow X_j^{u_j}$
015 $T_j \leftarrow S_j^{1/m_l}$
016 $r_j \leftarrow H(pars, j, Z_j)$; $c_j \leftarrow m_j + r_j$
017 $C \leftarrow (Y_1, U_1, T_1, Y_2, U_2, T_2, c_1, c_2)$
018 Return C

proc RoR$(\mathsf{es}, (m_1, m_2))$ // G_4

401 $C \xleftarrow{\$} \mathbb{G}_1^3 \times \mathbb{G}_2^3 \times \mathbb{Z}_p^2$
402 Return C

proc Finalize(b') // G_0, G_1, G_2, G_3, G_4

020 Return $(b' = 1)$

proc RoR$(\mathsf{es}, (m_1, m_2))$ // G_1

101 For $j = 1, 2$ do
102 $y_j, u_j, z_j \xleftarrow{\$} \mathbb{Z}_p$; $l \leftarrow 1 + (j \bmod 2)$
103 $Y_j \leftarrow g_j^{y_j}$; $Z_j \leftarrow g_j^{z_j}$
104 $U_j \leftarrow g_j^{u_j}$; $S_j \leftarrow X_j^{u_j}$
105 $T_j \leftarrow S_j^{1/m_l}$
106 $r_j \leftarrow H(pars, j, Z_j)$; $c_j \leftarrow m_j + r_j$
107 $C \leftarrow (Y_1, U_1, T_1, Y_2, U_2, T_2, c_1, c_2)$
108 Return C

proc RoR$(\mathsf{es}, (m_1, m_2))$ // G_2

201 For $j = 1, 2$ do
202 $y_j, u_j, z_j, t_j \xleftarrow{\$} \mathbb{Z}_p$; $l \leftarrow 1 + (j \bmod 2)$
203 $Y_j \leftarrow g_j^{y_j}$; $Z_j \leftarrow g_j^{z_j}$
204 $U_j \leftarrow g_j^{u_j}$; $S_j \leftarrow g_j^{t_j}$
205 $T_j \leftarrow S_j^{1/m_l}$
206 $r_j \leftarrow H(pars, j, Z_j)$; $c_j \leftarrow m_j + r_j$
207 $C \leftarrow (Y_1, U_1, T_1, Y_2, U_2, T_2, c_1, c_2)$
208 Return C

proc RoR$(\mathsf{es}, (m_1, m_2))$ // G_3

301 For $j = 1, 2$ do
302 $y_j, u_j \xleftarrow{\$} \mathbb{Z}_p$; $l \leftarrow 1 + (j \bmod 2)$
303 $Y_j \leftarrow g_j^{y_j}$; $U_j \leftarrow g_j^{u_j}$; $T_j \xleftarrow{\$} \mathbb{G}_j$
304 $r_j \xleftarrow{\$} \mathbb{Z}_p$; $c_j \leftarrow m_j + r_j$
305 $C \leftarrow (Y_1, U_1, T_1, Y_2, U_2, T_2, c_1, c_2)$
306 Return C

Figure 9: Games for proof of Lemma 4.7.

$(Y_1, Z_1, Y_2, Z_2) \xleftarrow{\$} \mathbf{2dh}()$
$u_1, u_2 \xleftarrow{\$} \mathbb{Z}_p$; $U_1 \leftarrow g_1^{u_1}$; $T_1 \leftarrow X_1^{u_1/m_2}$; $U_2 \leftarrow g_2^{u_2}$; $T_2 \leftarrow X_2^{u_2/m_1}$
$r_1 \leftarrow H(pars, 1, Z_1)$; $r_2 \leftarrow H(pars, 2, Z_2)$; $c_1 \leftarrow m_1 + r_1$; $c_2 \leftarrow m_2 + r_2$
$C \leftarrow (Y_1, U_1, T_1, Y_2, U_2, T_2, c_1, c_2)$
Return C

When A_i halts with output b', adversary B_1 returns b'.

Game G_2 picks S_1, S_2 at random at line 204 rather than the way they were picked at line 104. We build adversary B_2 so that

$$\Pr[G_1^{A_i}] - \Pr[G_2^{A_i}] \;\leq\; \mathbf{Adv}^{\mathsf{SXDH}}_{\mathsf{GS}, B_2}(k)\ . \tag{13}$$

On input $(pars, X_1, X_2)$ where $pars = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2)$, adversary B_2 lets $pk \leftarrow \perp$ if $i = 1$ and $pk \leftarrow (X_1, X_2)$ otherwise and then runs $A_i(1^k)$. When the latter makes its **KeySetup**(es_i) query, B_1 returns $(pars, pk)$. When A_i now makes a **RoR**$(\mathsf{es}_i, (m_1, m_2))$ query, B_2 responds via

$(U_1, S_1, U_2, S_2) \xleftarrow{\$} \mathbf{2dh}()$
$y_1, y_2, z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p$; $Y_1 \leftarrow g_1^{y_1}$; $Z_1 \leftarrow g_1^{z_1}$; $Y_2 \leftarrow g_2^{y_2}$; $Z_2 \leftarrow g_2^{z_2}$; $T_1 \leftarrow S_1^{1/m_2}$; $T_2 \leftarrow S_2^{1/m_1}$
$r_1 \leftarrow H(pars, 1, Z_1)$; $r_2 \leftarrow H(pars, 2, Z_2)$; $c_1 \leftarrow m_1 + r_1$; $c_2 \leftarrow m_2 + r_2$
$C \leftarrow (Y_1, U_1, T_1, Y_2, U_2, T_2, c_1, c_2)$

Return C

When A_i halts with output b', adversary B_2 returns b'.

Game G_3 picks r_1, r_2 at random at line 304 rather than getting them by applying H to Z_1, Z_2 as at line 206. The uniformity assumption we made on H ensures that the distributions of r_1, r_2 are the same. At line 303 it picks T_j at random rather than computing it as at line 205, but since $m_l \in \mathbb{Z}_p^*$, the distribution of T_j is the same in both cases. Hence

$$\Pr[G_2^{A_i}] \;=\; \Pr[G_3^{A_i}]\;. \tag{14}$$

Now we compare the distributions of C from lines 305 and 401 and see that they are the same, meaning

$$\Pr[G_3^{A_i}] \;=\; \Pr[G_4^{A_i}]\;. \tag{15}$$

Equation (10) follows from Equations (11), (12), (13), (14), and (15). ▌

Now we show, however, that the schemes are not circular secure.

Lemma 4.8 *Let* $\mathsf{es}_1, \mathsf{es}_2$ *be the* ENC*-schemes associated to group scheme* GS *via Figure 8. Then* $\mathsf{es}_1, \mathsf{es}_2$ *are not* CYC*-secure.*

Proof: We describe a PT adversary A such that $\mathbf{Adv}^{\mathsf{CYC}}_{\mathsf{es}_j,A}(k) \geq 1 - 2^{-k+1}$ for both $j = 1, 2$. The adversary ignores its input public key pk and hence works against both the symmetric and asymmetric versions of the scheme. $A(pk)$ issues a single query to **Cyc** and receives a pair (C_1, C_2) whose component ciphertexts it parses as $(Y_1, U_1, T_1, Y_2, U_2, T_2, c_1, c_2) \leftarrow C_1$ and $(\hat{Y}_1, \hat{U}_1, \hat{T}_1, \hat{Y}_2, \hat{U}_2, \hat{T}_2, \hat{c}_1, \hat{c}_2) \leftarrow C_2$. A returns 1 if $\mathbf{e}(U_1, \hat{U}_2) = \mathbf{e}(T_1, \hat{T}_2)$ and 0 otherwise. For the analysis, let $dk_1 = (x_1, x_2)$ and $dk_2 = (\hat{x}_1, \hat{x}_2)$ be the decryption keys chosen in the game. If $b = 1$, then

$$\mathbf{e}(T_1, \hat{T}_2) = \mathbf{e}(X_1^{u_1/\hat{x}_2}, \hat{X}_2^{\hat{u}_2/x_1}) = \mathbf{e}(U_1^{x_1/\hat{x}_2}, \hat{U}_2^{\hat{x}_2/x_1}) = \mathbf{e}(U_1, \hat{U}_2),$$

so A outputs 1. If $b = 0$, then the ciphertexts were sampled at random from $\mathbb{G}_1^3 \times \mathbb{G}_2^3 \times \mathbb{Z}_p^2$, so $\mathbf{e}(T_1, \hat{T}_2)$ and $\mathbf{e}(U_1, U_2)$ are uniformly random and independent elements of $\mathbb{G}_T$, and thus A returns 1 with probability $1/p$. ▌

Theorem 4.6 follows from these two lemmas.

Separating semantic and circular security of PKE. The public-key case of Theorem 4.6 resolves the following question, which has been noted in various forms since the seminal work of Goldwasser and Micali [14] and explicitly formulated by Boneh, Halevi, Hamburg and Ostrovsky [11]: does IND-CPA security imply non-trivial "circular-security" for public-key encryption? The version of circular-security used by Boneh et al. is similar to our notion of CYC security, except that the adversary is allowed normal IND-CPA encryption queries in addition to **Cyc** queries, and instead of distinguishing these ciphertexts from random, the adversary is required to distinguish the responses from encryptions of some fixed message. This version of circular security is suitable for some applications, while our version is only intended to be a theoretical tool.

Our es_2 gives a counterexample and answers this question in the negative. Under a plausible assumption it is IND-R-secure, which implies that it is IND-CPA-secure. It is also easy to verify that our scheme does not satisfy the notion of circular-security formulated in [11], using the attack in Lemma 4.8. Intuitively, this attack is breaking the semantic security of the scheme and not the pseudorandomness of its ciphertexts.

Prior work made partial progress on answering this question. The question for "1 cycles" is much simpler, with a known counterexample (c.f. [11]), and the question for stateful symmetric encryption was resolved using a simpler counter-example by Backes et al. [4]. Boneh et al. also give a counterexample for the version of the question where one replaces semantic security with *one-way* security. All of these counterexamples exploit their specific settings, and offer no obvious route toward

resolving the more difficult and relevant form of the question. Our approach is completely different from prior work in this vein. We note that our technique does not immediately work for a version of security where larger cycles are encrypted, but it seems likely that *multi*-linear forms could yield appropriate counterexamples.

4.5 Non-agility of wPRFs

We can now combine Theorems 4.2 and 4.6 to rule out agility of wPRFs:

Theorem 4.9 *Let $a \geq 2$. Suppose there exists a group scheme in which the SXDH problem is hard and further that* wPR*-secure* FF*-schemes with bit-output exist. Then the set* FF.wPR.Sch *of all* wPR*-secure* FF*-schemes is not a-agile.*

An explicit example of a pair $\{\mathsf{ff}_1, \mathsf{ff}_2\}$ of wPR-secure FF-schemes that is not 2-agile can be obtained by combining the proofs of the two theorems. However, it turns out we can give a simpler example by directly using the techniques behind the proof of Theorem 4.6, constructing $\mathsf{ff}_1, \mathsf{ff}_2$ as follows. For $j \in \{1, 2\}$ let $\mathsf{ff}_j.\mathsf{Pg}(1^k)$ return $pars = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2) \xleftarrow{\$} \mathsf{GS}(1^k)$, let $\mathsf{ff}_j.\mathsf{Kg}(pars)$ return $x \xleftarrow{\$} \mathbb{Z}_p^*$, let $\mathsf{ff}_j.\mathsf{Dom}(pars) = \mathbb{Z}_p$ and $\mathsf{ff}_j.\mathsf{Rng}(pars) = \mathbb{G}_j^2$. For an input $y \in \mathbb{Z}_p$, let $\mathsf{ff}_1.\mathsf{f}(pars, x, y) = (g_1^y, g_1^{y/x})$ and $\mathsf{ff}_2.\mathsf{f}(pars, x, y) = (g_2^y, g_2^{xy})$. Individually, ff_1 and ff_2 can be proven to be secure wPRFs under appropriate and relatively standard assumptions. But if the same key x is simultaneously used for both function families, an obvious distinguishing attack in the same spirit as the one above against es_1 and es_2 gives an adversary high advantage.

One might ask what is the value of Theorem 4.2 given this direct counterexample. First, we believe Theorem 4.2 is interesting in its own right as a connection between seemingly unrelated primitives. Also, if there are no group schemes in which SXDH is hard, Theorem 4.2, which is unconditional, still stands, and could lead to either positive or negative results depending on the veracity of the underlying conjectures. Notice that if wPRFs are shown agile, our results not only imply that all IND-R-secure encryption schemes are CYC-secure but also that there are no group schemes where SXDH is true.

5 Positive results

The above may make us pessimistic about achieving agility but there is good news as well. First, certain primitives are agile. Second, there are steps we can take to get strong agility in practice for primitives like AE. The idea is to not use the key directly with AE but instead use a subkey derived based on the description of the AE scheme. The latter brings out the key role of PRFs and wPRFs in agility. Let us expand on these items.

5.1 Agile primitives

Collision-resistant hash functions, formalized as keyed families, are agile. IND-CPA secure public-key encryption schemes are agile. (But not IND-CCA-secure public-key encryption schemes, and not IND-CPA symmetric encryption schemes!) In both cases the reason is simple, namely that one only needs access to public information (the hashing key or public encryption key) to simulate an adversary.

5.2 PRF-based agility for AE

The prevalence of AE schemes in practice and their continued appearance in this arena makes the agility of AE important. We have seen that we can't get agility for all AE schemes. Arguably, in practice, however, it may be enough to get it for a subset of them, such as CCM, OCB, CWC, GCM, EAX. However, the designs are sufficiently related that we suspect even this small set is in fact *not* agile! (That is, using the same key for all of them at the same time is insecure.)

Algorithm $\mathsf{es_{ff}}.\mathsf{Pg}(1^k)$
$\mathit{fpars} \xleftarrow{\$} \mathsf{ff}.\mathsf{Pg}(1^k)$; $\mathit{epars} \xleftarrow{\$} \mathsf{es}.\mathsf{Pg}(1^k)$
Return $(\mathit{fpars}, \mathit{epars})$

Algorithm $\mathsf{es_{ff}}.\mathsf{Kg}((\mathit{fpars}, \mathit{epars}))$
$K \xleftarrow{\$} \mathsf{ff}.\mathsf{Kg}(\mathit{fpars})$
Return $(K, K, \bot)$

Algorithm $\mathsf{es_{ff}}.\mathsf{Enc}((\mathit{fpars}, \mathit{epars}), K, M)$
$K_{\mathsf{es}} \leftarrow \mathsf{ff}.\mathsf{f}(\mathit{fpars}, K, \langle \mathsf{es} \rangle)$
$C \xleftarrow{\$} \mathsf{es}.\mathsf{Enc}(\mathit{epars}, K_{\mathsf{es}}, M)$
Return C

Algorithm $\mathsf{es_{ff}}.\mathsf{Dec}((\mathit{fpars}, \mathit{epars}), K, C)$
$K_{\mathsf{es}} \leftarrow \mathsf{ff}.\mathsf{f}(\mathit{fpars}, K, \langle \mathsf{es} \rangle)$
$M \leftarrow \mathsf{es}.\mathsf{Dec}(\mathit{epars}, K_{\mathsf{es}}, C)$
Return M

Figure 10: The symmetric ENC-scheme scheme $\mathsf{es_{ff}}$ associated to symmetric ENC-scheme es and FF-scheme ff.

We now show how to circumvent these difficulties and achieve AE agility, not only for the above schemes, but for all AE schemes, by using the schemes not directly but inside a construction. The requirement is a PRF that is either fixed or itself drawn from a small, agile space. This requirement is not too onerous because there are more proposals and choices for higher level primitives like AE than for the blockciphers that instantiate PRFs in practice. (Typically, one just uses AES.)

For our construction and analysis that follows now, we introduce some notation to ensure that the components we are using "fit together." Let $\ell(\cdot)$ be a polynomial. Let $\mathsf{FF.PR.Sch}[\ell]$ be the set of all $\mathsf{ff} \in \mathsf{FF.PR.Sch}$ such that $\mathsf{ff}.\mathsf{Dom}(\mathit{pars}) = \{0,1\}^*$ and $\mathsf{ff}.\mathsf{Rng}(\mathit{pars}) = \{0,1\}^{\ell(k)}$ for all $\mathit{pars} \in [\mathsf{FF.Pg}(1^k)]$ and all $k \in \mathbb{N}$. Let $\mathsf{ENC.AuE.Sch}[\ell]$ be the set of all symmetric ENC-schemes $\mathsf{es} \in \mathsf{ENC.AuE.Sch}$ such that $\mathsf{es}.\mathsf{Kg}(\mathit{pars})$ returns $(K, K, \bot)$ with K uniformly distributed over $\{0,1\}^{\ell(k)}$ for all $\mathit{pars} \in [\mathsf{es}.\mathsf{Pg}(1^k)]$ and all $k \in \mathbb{N}$. We let $\langle \mathsf{es} \rangle \in \{0,1\}^*$ be some unique string-encoding of the description of es in the sense that no two schemes in $\mathsf{ENC.AuE.Sch}$ have the same encoding. Such an encoding always exists since schemes are finite tuples of algorithms and thus have finite descriptions. To $\mathsf{es} \in \mathsf{ENC.AuE.Sch}[\ell]$ and $\mathsf{FF.PR.Sch}[\ell]$ we associate the ENC-scheme $\mathsf{es_{ff}} \in \mathsf{ENC.AuE.Sch}$ defined via Figure 10. It has $\mathsf{es_{ff}}.\mathsf{Msg}((\mathit{fpars}, \mathit{epars})) = \mathsf{es}.\mathsf{Msg}(\mathit{epars})$ and $\mathsf{es_{ff}}.\mathsf{Ctxts}((\mathit{fpars}, \mathit{epars})) = \mathsf{es}.\mathsf{Ctxts}(\mathit{epars})$. The following says that the set of such schemes is agile as long as ff is drawn from an agile space, regardless of the choice of es.

Theorem 5.1 *Let $\ell(\cdot)$ be a polynomial. Let $\Gamma \subset \mathsf{FF.PR.Sch}[\ell]$ be a compatible, finite set that is agile with respect to* PR*. Then, for every $a \in \mathbb{N}$, the set $\{\, \mathsf{es_{ff}} : \mathsf{es} \in \mathsf{ENC.AuE.Sch}[\ell]$ and $\mathsf{ff} \in \Gamma \,\}$ is a-agile with respect to* AuE*.*

In particular, for any particular $\mathsf{ff} \in \mathsf{FF.PR.Sch}[\ell]$ and every $a \in \mathbb{N}$, the set $\{\mathsf{es_{ff}} : \mathsf{es} \in \mathsf{ENC.AuE.Sch}[\ell]\}$ is a-agile with respect to ENC.AuE. Thus, if we have a PRF whose stability we can trust, we can make authenticated encryption highly agile.

Proof of Theorem 5.1: Let Π be a size a, compatible subset of $\{\, \mathsf{es_{ff}} : \mathsf{es} \in \mathsf{ENC.AuE.Sch}[\ell]$ and $\mathsf{ff} \in \Gamma \,\}$. Let

$$\begin{aligned} \Pi_e &= \{\, \mathsf{es} \in \mathsf{ENC.AuE.Sch}[\ell] \,:\, \exists\, \mathsf{ff} \in \Gamma \text{ such that } \mathsf{es_{ff}} \in \Pi \,\} \\ \Pi_f &= \{\, \mathsf{ff} \in \Gamma \,:\, \exists\, \mathsf{es} \in \mathsf{ENC.AuE.Sch}[\ell] \text{ such that } \mathsf{es_{ff}} \in \Pi \,\}\,. \end{aligned}$$

List the members of Π_e as $\mathsf{es}^1, \ldots, \mathsf{es}^{a_1}$ and the members of Π_f as $\mathsf{ff}^1, \ldots, \mathsf{ff}^{a_2}$ where $a_1, a_2 \le a$. For $\mathsf{es_{ff}} \in \Pi$ let $\mathrm{Index}(\mathsf{es_{ff}})$ return the unique (i,j) such that $\mathsf{es} = \mathsf{es}^i$ and $\mathsf{ff} = \mathsf{ff}^j$. Since Π is a finite, fixed set, this index function is easily computed.

Let A be a Π-restricted adversary against the AuE-agility of Π. We design a Π_f-restricted adversary B against the PR-agility of Π_f and $\{\mathsf{es}^i\}$-restricted adversaries $A_{i,j}$ against the AuE-security of es^i for

proc KeySetup(es) // G_0

001 $S \leftarrow \emptyset$; $b \xleftarrow{\$} \{0,1\}$
002 $fpars \xleftarrow{\$} \mathsf{ff}^1.\mathsf{Pg}(1^k)$; $epars \xleftarrow{\$} \mathsf{es}^1.\mathsf{Pg}(1^k)$
003 $K \xleftarrow{\$} \mathsf{ff}^1.\mathsf{Kg}(fpars)$
004 For $i = 1, \ldots, a_1$ do
005 For $j = 1, \ldots, a_2$ do
006 $K_{i,j} \leftarrow \mathsf{ff}^j.\mathsf{f}(fpars, K, \langle \mathsf{es}^i \rangle)$
007 Return $(fpars, epars)$

proc RoR(es, M) // G_0, G_1

011 $(i,j) \leftarrow \mathrm{Index}(\mathsf{es})$
012 If $b = 1$ Then $C \xleftarrow{\$} \mathsf{es}^i.\mathsf{Enc}(epars, K_{i,j}, M)$
013 Else $C \xleftarrow{\$} \mathsf{es}^i.\mathsf{Ctxts}(epars)$
014 $S \leftarrow S \cup \{(i,j,C)\}$
015 Return C

proc KeySetup(es) // G_1

101 $S \leftarrow \emptyset$; $b \xleftarrow{\$} \{0,1\}$
102 $fpars \xleftarrow{\$} \mathsf{ff}^1.\mathsf{Pg}(1^k)$; $epars \xleftarrow{\$} \mathsf{es}^1.\mathsf{Pg}(1^k)$
103 For $i = 1, \ldots, a_1$ do
104 For $j = 1, \ldots, a_2$ do
105 $K_{i,j} \xleftarrow{\$} \{0,1\}^{\ell(k)}$
106 Return $(fpars, epars)$

proc Dec(es, C) // G_0, G_1

021 $(i,j) \leftarrow \mathrm{Index}(\mathsf{es})$
022 If $(i,j,C) \in S$ Then Return $\bot$
023 If $b = 1$ Then $M \leftarrow \mathsf{es}^i.\mathsf{Dec}(epars, K_{i,j}, C)$
024 Else $M \leftarrow \bot$
025 Return M

proc Finalize(b') // G_0, G_1

031 Return $(b' = b)$

Figure 11: Games for the proof of Theorem 5.1.

$1 \le i \le a_1$ and $1 \le j \le a_2$ such that

$$\mathbf{Adv}^{\mathsf{AuE}}_{\Pi,A}(k) \;\le\; 2 \cdot \mathbf{Adv}^{\mathsf{PR}}_{\Pi_f,B}(k) + \sum_{i=1}^{a_1}\sum_{j=1}^{a_2} \mathbf{Adv}^{\mathsf{AuE}}_{\{\mathsf{es}^i\},A_{i,j}}(k) \tag{16}$$

for all $k \in \mathbb{N}$. Adversary B makes $a_1 a_2 \le a^2$ **Fn** queries. The number of **RoR** and **Dec** queries made by $A_{i,j}$ is the same as A. The constructed adversaries have running time that of A plus the time for $\mathcal{O}(a^2)$ invocations of the key generation algorithm of ff^1. Now, suppose A is PT. Then the assumption that Π_f is a-agile with respect to PR means that $\mathbf{Adv}^{\mathsf{PR}}_{\Pi_f,B}(\cdot)$ is negligible. The assumption that each es^i, taken individually, is AuE-secure means that $\mathbf{Adv}^{\mathsf{AuE}}_{\{\mathsf{es}^i\},A_{i,j}}$ is negligible for each i,j. Equation (16) then implies that $\mathbf{Adv}^{\mathsf{AuE}}_{\Pi,A}(\cdot)$ is negligible. (This uses in a crucial way that there is only a finite constant number of terms in the sum.) The theorem follows.

The games of Figure 11 will be executed with A. Remember this adversary begins with a **KeySetup** query. Knowing that the argument es of this query is one of the AuE-schemes from the compatible set Π, procedure **KeySetup** of G_0 ignores it and uses $\mathsf{es}^1, \mathsf{ff}^1$ for parameter generation. It then generates a key $K_{i,j}$ for use with $\mathsf{es}^i_{\mathsf{ff}^j}$ as per the definition of the latter from Figure 10. This key is used to answer **RoR** and **Dec** queries as per game $\mathsf{ENC.AuE.Gm}_k$. We have

$$\frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\mathsf{AuE}}_{\Pi,A}(k) \;=\; \Pr[G_0^A] \;=\; \Pr[G_1^A] + (\Pr[G_0^A] - \Pr[G_1^A]) \;. \tag{17}$$

Game G_1 picks the keys $K_{i,j}$ at random at line 105 rather than based on ff^j as at line 006. The agility of Π_f, implied by the assumed agility of its superset Γ, says this makes no detectable difference. Specifically we can build Π_f-restricted adversary B such that

$$\Pr[G_0^A] - \Pr[G_1^A] \;\le\; \mathbf{Adv}^{\mathsf{PR}}_{\Pi_f,B}(k) \;. \tag{18}$$

On input 1^k, adversary B begins with the initializations

$$S \leftarrow \emptyset \;;\; b \xleftarrow{\$} \{0,1\} \;;\; epars \xleftarrow{\$} \mathsf{es}^1.\mathsf{Pg}(1^k) \;.$$

It then runs A. When the latter makes its **KeySetup** query, B calls its own **KeySetup** oracle on

input ff^1 to get back *fpars* and returns (*fpars*, *epars*) to A. For $i = 1, \ldots, a_1$ and $j = 1, \ldots, a_2$ it then lets $K_{i,j} \xleftarrow{\$} \mathbf{Fn}(\mathsf{ff}^j, \langle \mathsf{es}^i \rangle)$. It can now easily answer **RoR** and **Dec** queries of A as per the code of G_0. When A halts with output b', adversary B returns 1 if $b = b'$ and 0 otherwise.

To conclude, we need to build $A_{i,j}$ $(1 \le i \le a_1, 1 \le j \le a_2)$ such that

$$2\Pr[G_1^A] - 1 \;\le\; \sum_{i=1}^{a_1} \sum_{j=1}^{a_2} \mathbf{Adv}^{\mathsf{AuE}}_{\{\mathsf{es}^i\}, A_{i,j}}(k) \;. \tag{19}$$

This is a quite straightforward hybrid argument. We visualize it as a simultaneous attack on $\mathsf{es}^1, \ldots, \mathsf{es}^{a_1}$, furthermore with a_2 instances of each scheme. But each of the $a1a_2$ instances has its own, random dedicated key, enabling the hybrid argument to work. We omit the details. ∎

5.3 wPRF-based agility for AE

We would like to use wPRFs in place of PRFs because wPRF is a weaker assumption on a blockcipher than a PRF and, as a result, a set of blockciphers is more likely to be agile with respect to wPRF than to PRF. (We cannot of course hope for agility with respect to all wPRFs since that class is not agile. But we'd like to get it for as large a subset of the class as possible.) We show this is possible. This explains our interest in wPRFs and their importance in the agility domain.

The obvious naive modification to the above construction when ff is a wPRF rather than a PRF is for $\mathsf{es}_{\mathsf{ff}}((\mathit{fpars}, \mathit{epars}), K, M)$ to pick a random R, let $K_{\mathsf{es}} \leftarrow \mathsf{ff.f}(\mathit{fpars}, K, R)$, let $C \xleftarrow{\$} \mathsf{es.Enc}(\mathit{epars}, K_{\mathsf{es}}, M)$, and return (C, R) as the ciphertext, R being included to allow decryption. However it is easy to see that this is not secure. Even ignoring agility, $\mathsf{es}_{\mathsf{ff}}$ fails to be a secure AE scheme in general.

Instead, we consider the constructions of PRFs from wPRFs due to Naor and Reingold [25], Maurer and Sjödin [21] and Maurer and Tessaro [22]. Some of these constructed PRFs make only blackbox appeal to a single wPRF, evaluated on fixed number of independent keys. We claim that these types of constructions are agility-preserving in the sense that the set of constructed PRFs obtained by using wPRFs from a set Γ is agile with respect to PR if Γ was agile with respect to wPR. The intuition is that an attack on the a-agility of the family of constructions (where each member of the family accesses a different wPRF from Γ) will immediately imply an attack on the a-agility of Γ. The details are tedious, but the high-level proof of the claim proceeds in the same manner as Theorem 5.1, but instead of agile AE we are constructing agile PRFs. Composing the constructions from the claim with Theorem 5.1, we get an agile family of AE schemes, assuming only that we start with an agile family of wPRFs.

References

[1] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Aug. 2009. (Cited on page 2, 5.)

[2] G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. `http://eprint.iacr.org/`. (Cited on page 2, 13, 14.)

[3] M. Backes, M. Dürmuth, and D. Unruh. OAEP is secure under key-dependent messages. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 506–523. Springer, Dec. 2008. (Cited on page 5.)

[4] M. Backes, B. Pfitzmann, and A. Scedrov. Key-dependent message security under active attacks - BRSIM/UC-soundness of Dolev-Yao-style encryption with key cycles. *J. Comput. Secur.*, 16(5):497–530, 2008. (Cited on page 2, 17.)

[5] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, May 2000. (Cited on page 14, 23.)

[6] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, Oct. 1997. (Cited on page 7, 9.)

[7] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Dec. 2000. (Cited on page 7.)

[8] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, May / June 2006. (Cited on page 5.)

[9] M. Bellare, P. Rogaway, and D. Wagner. The EAX mode of operation. In B. K. Roy and W. Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 389–407. Springer, Feb. 2004. (Cited on page 3.)

[10] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In K. Nyberg and H. M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Aug. 2003. (Cited on page 2, 5.)

[11] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision diffie-hellman. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 108–125. Springer, Aug. 2008. (Cited on page 2, 5, 17.)

[12] J. Camenisch, N. Chandran, and V. Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 351–368. Springer, Apr. 2009. (Cited on page 2, 5.)

[13] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, May 2001. (Cited on page 1, 5.)

[14] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. (Cited on page 17.)

[15] S. Haber and B. Pinkas. Securely combining public-key cryptosystems. In *ACM CCS 01*, pages 215–224. ACM Press, Nov. 2001. (Cited on page 4.)

[16] I. Haitner and T. Holenstein. On the (im)possibility of key dependent encryption. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 202–219. Springer, Mar. 2009. (Cited on page 2.)

[17] S. Halevi and H. Krawczyk. Security under key-dependent inputs. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM CCS 07*, pages 466–475. ACM Press, Oct. 2007. (Cited on page 5.)

[18] D. Hofheinz and D. Unruh. Towards key-dependent message security in the standard model. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 108–126. Springer, Apr. 2008. (Cited on page 5.)

[19] J. Kelsey, B. Schneier, and D. Wagner. Protocol interactions and the chosen protocol attack. In *Security Protocols Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 91–104. Springer, Apr. 1997. (Cited on page 3, 4.)

[20] T. Kohno, J. Viega, and D. Whiting. CWC: A high-performance conventional authenticated encryption mode. In B. K. Roy and W. Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 408–426. Springer, Feb. 2004. (Cited on page 3.)

[21] U. M. Maurer and J. Sjödin. A fast and key-efficient reduction of chosen-ciphertext to known-plaintext security. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 498–516. Springer, May 2007. (Cited on page 1, 4, 21.)

[22] U. M. Maurer and S. Tessaro. Basing PRFs on constant-query weak PRFs: Minimizing assumptions for efficient symmetric cryptography. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 161–178. Springer, Dec. 2008. (Cited on page 1, 4, 21.)

[23] D. A. McGrew and J. Viega. The security and performance of the Galois/counter mode (gcm) of operation. In A. Canteaut and K. Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 343–355. Springer, Dec. 2004. (Cited on page 3.)

[24] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, Oct. 1997. (Cited on page 14, 23.)

[25] M. Naor and O. Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *J. Comput. Syst. Sci.*, 58(2):336–375, 1999. (Cited on page 1, 4, 21.)

[26] D. Nelson. Crypto-agility requirements for remote dial-in user service (radius). IETF Network Working Group Internet-Draft, Nov. 2008. `http://tools.ietf.org/html/draft-ietf-radext-crypto-agility-requirements-01`. (Cited on page 2.)

[27] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *ACM CCS 01*, pages 196–205. ACM Press, Nov. 2001. (Cited on page 1, 3, 7, 9.)

[28] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, May / June 2006. (Cited on page 7.)

[29] B. Sullivan. Cryptographic agility. Microsoft Developer Network Magazine, Aug. 2009. `http://msdn.microsoft.com/en-us/magazine/ee321570.aspx`. (Cited on page 2.)

[30] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). RFC 3610 (Informational), Sept. 2003. (Cited on page 3.)

A Proof of Lemma 4.5

We will use the following.

Lemma A.1 *Let* GS *be a group scheme. Then there is an algorithm* R *which on input* $pars = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2)$ *and* $(g_1^{x_1}, g_1^{y_1}, g_1^{z_1}, g_2^{x_2}, g_2^{y_2}, g_2^{z_2}) \in \mathbb{G}_1^3 \times \mathbb{G}_2^3$ *returns a 4-tuple* $(g_1^{y'_1}, g_1^{z'_1}, g_2^{y'_2}, g_2^{z'_2}) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2$ *such that the following hold:*

(1) *If* $z_i \equiv x_i y_i \pmod p$ *(*$i = 1, 2$*) then* y'_1, y'_2 *are uniformly and independently distributed over* $\mathbb{Z}_p$ *and* $z'_i \equiv x_i y'_i \pmod p$ *(*$i = 1, 2$*)*

(2) *If* $z_i \not\equiv x_i y_i \pmod p$ *(*$i = 1, 2$*) then* y'_1, y'_2, z'_1, z'_2 *are uniformly and independently distributed over* $\mathbb{Z}_p$

In both cases, the statement is for all inputs to the algorithm with the probability being over the coins of R *alone. The running time of* R *is that of a constant number of exponentiations in* $\mathbb{G}_1, \mathbb{G}_2$*.*

This is adapted from a lemma of [5]. The latter differs from the self-reducibility lemma of [24] in that the x_i are fixed.

Proof of Lemma 4.5: On input $pars, X_1, X_2$, where $pars = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2)$, adversary A_1 begins by making its unique **2dh** query to get (Y_1, Z_1, Y_2, Z_2). It then runs A on input $pars, X_1, X_2$. When A makes a **2dh**(), adversary A_1 returns $(Y'_1, Z'_1, Y'_2, Z'_2) \xleftarrow{\$} R(pars, (X_1, Y_1, Z_1, X_2, Y_2, Z_2))$ to A where R is the algorithm of Lemma A.1. When A halts with output b', adversary A_1 outputs b'. ∎