Cryptographic Agility and its Relation to Circular Encryption

Tolga Acar[1], Mira Belenkiy[1], Mihir Bellare[2], and David Cash[2]

[1] eXtreme Computing Group, Microsoft Research, Redmond, WA ; {tolga,mibelenk}@microsoft.com ; http://research.microsoft.com/en-us/people/tolga

[2] Department of Computer Science & Engineering, University of California San Diego, CA ; {mihir,cdc}@cs.ucsd.edu ; http://www.cs.ucsd.edu/users/{mihir,cdc}

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.

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 $\mathbf{Fn}$ that takes no inputs. Each time it is called, $\mathbf{Fn}$ picks random x, y, returning $(x, F_K(x))$ if $b = 1$ and (x, y) otherwise. Naor and Reingold [24] 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 [24], 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) [26] 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 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 5 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 [3] 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 [5] 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, 2].

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 [25]. Resources for software professionals, like a recent Microsoft Developer Network Magazine article [28], 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 [29], OCB [26], 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

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

Fig. 1. Agility status of some basic primitives.

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. Fig. 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 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 primitve 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 [24, 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, 4, 12, 2]. 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 $x_1, x_2, \ldots$.

GAMES. Our definitions and proofs use the language of code-based games [8]. Recall that a game —look at Fig. 2 for examples— has an (optional) **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. 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. Boolean flags are assumed initialized to false. 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

proc KeySetup(ff)	**proc KeySetup**(ff)
$pars \xleftarrow{\$} \mathsf{ff.Pg}(1^k)$; $K \xleftarrow{\$} \mathsf{ff.Kg}(pars)$	$pars \xleftarrow{\$} \mathsf{ff.Pg}(1^k)$; $K \xleftarrow{\$} \mathsf{ff.Kg}(pars)$
$b \xleftarrow{\$} \{0,1\}$	$b \xleftarrow{\$} \{0,1\}$
Return $pars$	Return $pars$
proc Fn(ff, x)	**proc Fn**(ff)
If $\mathsf{ff.DomR}(pars, x) = 0$ then return $\perp$	$x \xleftarrow{\$} \mathsf{ff.Dom}(pars)$
If $b = 1$ then $y \leftarrow \mathsf{ff.f}(pars, K, x)$	If $b = 1$ then $y \leftarrow \mathsf{ff.f}(pars, K, x)$
Else $y \xleftarrow{\$} \mathsf{ff.Rng}(pars)$	Else $y \xleftarrow{\$} \mathsf{ff.Rng}(pars)$
Return y	Return (x, y)
proc Finalize(b')	**proc Finalize**(b')
Return $(b' = b)$	Return $(b' = b)$

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

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 $pars \in [\mathsf{es.Pg}(1^k)]$, the key generator outputs a triple (ek, dk, 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 $pk = \perp$. (In which case we may assume wlog $ek = dk$.) We say it is asymmetric if it is always the case that $ek = pk$. We require that there is a polynomial $r(\cdot)$, called the *number of coins used by* es.Enc, such that es.Enc draws its coins from $\{0,1\}^{r(k)}$ whenever its first input is $pars \in [\mathsf{es.Pg}(1^k)]$. We require that $\mathsf{es.Enc}(pars, ek, \cdot; R)\colon\ \mathsf{es.Msg}(pars) \to \mathsf{es.Ctxts}(pars)$ and $\mathsf{es.Dec}(pars, dk, \mathsf{es.Enc}(pars, ek, M; R)) = M$ for all $R \in \{0,1\}^{r(k)}$, all $M \in \mathsf{es.Msg}(pars)$, all $(ek, dk, pk) \in [\mathsf{es.Kg}(pars)]$, 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 Fig. 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, but it is likely to be hard to parse. Instead, 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.

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 Fig. 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. (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 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) 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 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 $\mathsf{FF.PR.Sch}$ of all PR-secure FF-schemes is the set of all PRFs.

wPRF agility. The games $\mathsf{FF.wPR.Gm}_k$ ($k \in \mathbb{N}$) are now those on the right side of Fig. 2. 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 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) 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. 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 $\mathsf{FF.wPR.Sch}$ 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 Fig. 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 [27]. The privacy requirement is indistinguishability from random [26] (IND-R), a strengthening of the usual notion of [6] that 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 and it makes only one **KeySetup** query, this being its first oracle query. Let $\mathbf{Adv}^{\mathsf{AuE}}_{\Pi,A}(k) = 2\Pr[\mathsf{ENC.AuE.Gm}^{A}_{k}] - 1$ for any 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) 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 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 $\mathsf{ENC.AuE.Sch}$ be the set of all AuE-secure ENC schemes.

This definition is for both symmetric and asymmetric schemes even though the latter can only meet it if no **Dec** queries are allowed because the $\mathsf{IND\text{-}R}$ notion of privacy obtained by incorporating the latter restriction will be useful later.

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.

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 $M \notin \mathsf{es.Msg}(pars)$ Then Return $\bot$
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 $\bot$
If $b = 1$ Then $M \leftarrow \mathsf{es.Dec}(pars, dk, C)$
Else $M \leftarrow \bot$
Return M

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

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

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 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 (Proposition 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 $y \leftarrow \mathsf{ff.f}(pars, K, x)$ and returns the bitwise complement $\overline{y}$ of y. The new FF-scheme has range defined by $\overline{\mathsf{ff}}.\mathsf{Rng}(pars) = \{\, \overline{y} \,:\, y \in \mathsf{ff.Rng}(pars) \,\}$.

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 **KeySetup**(ff) query. Then it lets $x \xleftarrow{\$} \mathsf{ff.Dom}(pars)$ 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 $|\mathsf{ff.Rng}(pars)| \geq 2$. 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 $\mathsf{ff.Rng}(pars)$. But both sets $\mathsf{ff.Rng}(pars)$ 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 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 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 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 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. However, this type of approach does not work to prove non-agility of wPRFs because the inputs to the functions are not under adversary control. On the other hand, a proof that wPRFs are agile also seems out of reach of reduction-based techniques. We turn to resolving this.

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, {es}-restricted adversaries A that make no **Dec** queries. This strong version of privacy under CPA from [26] implies the standard IND-CPA and is achieved by blockcipher modes of operation like CTR and CBC [6].

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 Fig. 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. We will first be interested in the former.

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 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-

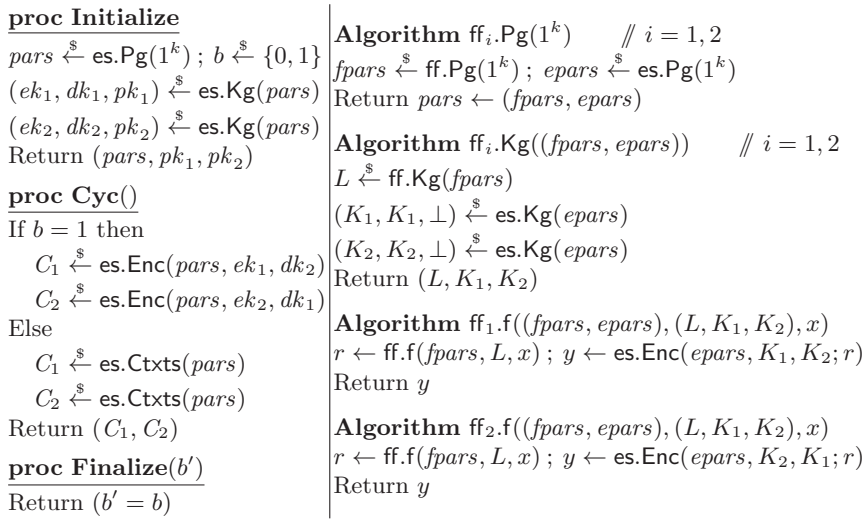

Fig. 4. Game $\mathsf{ENC.CYC.Gm}_k$ on the right, for $k \in \mathbb{N}$. On the right, algorithms for FF-schemes $\mathsf{ff}_1, \mathsf{ff}_2$ of the proof of Theorem 2.

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 $\mathsf{ff.Rng}(\mathit{pars}) = \{0,1\}^{r(k)}$ for all $\mathit{pars} \in [\mathsf{ff.Pg}(1^k)]$ 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.

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 Fig. 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 $\mathsf{es.Kg}(\mathit{epars})$ in the code of $\mathsf{ff}_i\mathsf{.Kg}((\mathit{fpars}, \mathit{epars}))$ 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.

proc Initialize

$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2) \xleftarrow{\$} \mathsf{GS}(1^k)$; $x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p$; $b \xleftarrow{\$} \{0,1\}$
If $(b = 1)$ then $(X_1, Y_1, Z_1, X_2, Y_2, Z_2) \leftarrow (g_1^{x_1}, g_1^{y_1}, g_1^{x_1 y_2}, g_2^{x_2}, g_2^{y_2}, g_2^{x_2 y_2})$
Else $(X_1, Y_1, Z_1, X_2, Y_2, Z_2) \leftarrow (g_1^{x_1}, g_1^{y_1}, g_1^{z_1}, g_2^{x_2}, g_2^{y_2}, g_2^{z_2})$
Return $((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2), X_1, Y_1, Z_1, X_2, Y_2, Z_2)$

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

Fig. 5. Game $\mathsf{SXDH}_{\mathsf{GS},k}$, for $k \in \mathbb{N}$, used to define the hardness of the SXDH problem in group scheme GS.

Lemma 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.*

The proof is in [1]. 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. *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 2 follows from these two lemmas. The proof of Lemma 4 is in [1].

The SXDH assumption. Our counterexample encryption scheme that is IND-R-secure but not CYC-secure relies on the SXDH assumption [3] 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 involved as well evaluate $\mathbf{e}(\cdot, \cdot)$ in time polynomial in k. The Symmetric External Diffie-Hellman (SXDH) assumption [3] 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 Fig. 5. 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. We also 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.

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.

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)

Fig. 6. 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 $\mathsf{es}_j.\mathsf{Ctxts}(pars) = \mathbb{G}_1^3 \times \mathbb{G}_2^3 \times \mathbb{Z}_p^2$.

Theorem 5. (**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 $j \in \{1, 2\}$, Fig. 6 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.

Lemma 6. *Let* $\mathsf{es}_1, \mathsf{es}_2$ *be the* ENC*-schemes associated to group scheme* GS *via Fig. 6. Suppose the SXDH problem is hard in* GS*. Then* $\mathsf{es}_1, \mathsf{es}_2$ *are* IND-R*-secure.*

The proof is in [1]. Now we show, however, that the schemes are not circular secure.

Lemma 7. *Let* $\mathsf{es}_1, \mathsf{es}_2$ *be the* ENC*-schemes associated to group scheme* GS *via Fig. 6. 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 5 follows from these two lemmas.

Non-agility of wPRFs. We can now combine Theorems 2 and 5 to rule out agility of wPRFs:

Theorem 8. *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 5, 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 $\{\mathsf{es}_1, \mathsf{es}_2\}$ gives an adversary high advantage.

One might ask what is the value of Theorem 2 given this direct counterexample. First, we believe Theorem 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 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.

Separating semantic and circular security of PKE. The public-key case of Theorem 5 resolves the following question posed by Boneh, Halevi, Hamburg and Ostrovsky [11]: does there exist an IND-CPA-secure public-key encryption that becomes insecure when a 2-cycle is published? (Our es_2, being IND-R-secure, is of course IND-CPA-secure. Two-cycle security as per [11], was, however, a weaker requirement than ours, being of an IND-CPA flavor rather than our IND-R flavor. But it is not hard to show that our es_2 fails this cyclicity notion as well.)

Let us review the status of this question. As noted by Goldwasser and Micali [14], it is not hard to see that semantic security does not guarantee that it is safe to encrypt a secret key under its corresponding public key. That is, one can give a scheme that is semantically-secure but is no longer secure when the adversary is given an encryption of the secret key. A natural question is if it is safe to encrypt larger cycles. Backes et al. [5] showed it is not safe for stateful symmetric encryption, but an adversary could only break circular-security when given access to encryptions of each key under its initial state. Boneh et al. [11] gave a simple public-key scheme that was one-way secure during a chosen-plaintext attack but not one-way after a circular encryption was published. Neither of the techniques in these works seemed to generalize to resolve the question for semantic-versus-circular security of public-key encryption, which was particularly relevant given recent effort towards constructing circular-secure public-key schemes.

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.

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.

PRF-based agility for AE. The existence and continued appearence of new AE schemes 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.)

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

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

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

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

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

Fig. 7. The symmetric ENC-scheme scheme $\mathsf{es_{ff}}$ associated to symmetric ENC-scheme es and FF-scheme ff.

a polynomial. Let $\mathsf{FF.PR.Sch}[\ell]$ be the set of all $\mathsf{ff} \in \mathsf{FF.PR.Sch}$ such that $\mathsf{ff.Dom}(pars) = \{0,1\}^*$ and $\mathsf{ff.Rng}(pars) = \{0,1\}^{\ell(k)}$ for all $pars \in [\mathsf{FF.Pg}(1^k)]$ and all $k \in \mathbb{N}$. Let $\mathsf{ENC.AuE.Sch}[\ell]$ be the set of all $\mathsf{es} \in \mathsf{ENC.AuE.Sch}$ such that $\mathsf{es.Kg}(pars)$ induces a uniform distribution on $\{0,1\}^{\ell(k)}$ for all $pars \in [\mathsf{es.Pg}(1^k)]$ 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 ENC.AuE.Sch have the same encoding. Such an encoding always exists since schemes are finite tuples of algorithms and thus have finite descriptions.

Theorem 9. *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 $\{ \mathsf{es_{ff}} : \mathsf{es} \in \mathsf{ENC.AuE.Sch}[\ell],\ \mathsf{ff} \in \Gamma \}$ is a-agile with respect to* AuE*.*

In particular, for $\mathsf{ff} \in \mathsf{FF.PR.Sch}[\ell]$ 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. The proof of Theorem 9 is in [1].

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 modification to the above construction when ff is a wPRF rather than a PRF is for $\mathsf{es_{ff}}((fpars, epars), K, M)$ to pick a random R, let $K_{\mathsf{es}} \leftarrow \mathsf{ff.f}(fpars, K, R)$, let $C \xleftarrow{\$} \mathsf{es.Enc}(epars, K_{\mathsf{es}}, M)$, 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, $\mathsf{es_{ff}}$ fails to be a secure AE scheme in general. Instead, we consider the constructions of PRFs from wPRFs due to Naor and Reingold [24], Maurer and Sjödin [21] and Maurer and Tessaro [22]. Some of the constructed PRFs make only blackbox appeal to a fixed number of wPRFs on independent keys. 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. Now, we can use our construction above.

Acknowledgments

The third and fourth authors were supported in part by NSF grants CNS-0627779 and CCF-0915675. We thank Tom Ristenpart for useful pointers.

References

1. T. Acar, M. Belenkiy, M. Bellare, and D. Cash. Cryptographic agility and its relation to circular encryption. Cryptology ePrint Archive, 2010. `http://eprint.iacr.org/`.
2. 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.
3. 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/`.
4. 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.
5. 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.
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.
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.
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.
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.
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.
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.
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.

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.
14. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
15. S. Haber and B. Pinkas. Securely combining public-key cryptosystems. In *ACM CCS 01*, pages 215–224. ACM Press, Nov. 2001.
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.
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.
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.
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.
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.
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.
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.
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.
24. 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.
25. 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`.
26. 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.
27. 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.
28. B. Sullivan. Cryptographic agility. Microsoft Developer Network Magazine, Aug. 2009. `http://msdn.microsoft.com/en-us/magazine/ee321570.aspx`.
29. D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). RFC 3610 (Informational), Sept. 2003.