Security Under Key-Dependent Inputs

Shai Halevi Hugo Krawczyk
IBM T.J. Watson Research Center
shaih@alum.mit.edu, hugo@ee.technion.ac.il

August 13, 2007

Abstract

In this work we re-visit the question of building cryptographic primitives that remain secure even when queried on inputs that depend on the secret key. This was investigated by Black, Rogaway, and Shrimpton in the context of randomized encryption schemes and in the random oracle model. We extend the investigation to deterministic symmetric schemes (such as PRFs and block ciphers) and to the standard model. We term this notion "*security against key-dependent-input attack*", or KDI-security for short. Our motivation for studying KDI security is the existence of significant real-world implementations of deterministic encryption (in the context of storage encryption) that actually rely on their building blocks to be KDI secure.

We consider many natural constructions for PRFs, ciphers, tweakable ciphers and randomized encryption, and examine them with respect to their KDI security. We exhibit inherent limitations of this notion and show many natural constructions that fail to be KDI secure in the standard model, including some schemes that have been proven in the random oracle model. On the positive side, we demonstrate examples where some measure of KDI security can be provably achieved (in particular, we show such examples in the standard model).

1 Introduction

Does it make sense for an application to self-encrypt an encryption key? That is, if E_s represents an encryption function with key s, would it ever be the case that an application needs to store or transmit $E_s(s)$? Cryptographers typically see this as a *dangerous abuse* of an encryption scheme, and standard security criteria for encryption scheme do not take this possibility into account. Still, there are applications where such form of security is helpful. This security concern was formally defined and studied by Black, Rogaway, and Shrimpton [5] in the context of symmetric encryption under the name KDM-security (for Key-Dependent-Messages). In particular Black et al. proved that KDM-secure symmetric encryption can be achieved in the random-oracle model.

If "encrypting your own key" is abusive for randomized encryption, using this practice with deterministic constructions (such as pseudo-random functions and permutations) seems even more dangerous. The present work was motivated by a real-world application that turned out to be doing just that: the IEEE P1619 standard group was developing a standard for "sector level encryption" [15], which must be length-preserving and hence must be deterministic. The group was considering a scheme based on the tweakable cipher of Liskov et al. [18], but some members objected, citing

an attack that can be mounted when the proposed scheme is applied to its own secret key. An argument ensued as to whether or not this "self-encryption" scenario is a real problem or just a theoretical possibility that would never happen in the real world. The argument was decided when the group was informed that the implementation of disk encryption in Windows Vista™ can store to the disk an encryption of its own secret keys in some situations. Consequently, the group switched to a different scheme, based on Rogaway's work [23], for which the particular attack in question does not seem to apply (see more details in Section 5 and in [14]).

Another reason to study KDI-security arises in the context of anonymous credentials: Camenisch and Lysyanskaya [7] introduced the notion of "circular security", which is similar to (but somewhat weaker than) KDI security, and used it as a tool to discourage delegation of credentials in an anonymous credential system. Also, in the formal-methods community the definitional work of Black et al. from [5] was used to strengthen the definition of computational encryption and prove it equivalent to the Dolev-Yao formulation [17, 1].

1.1 Our results

In this work we re-visit KDM security, with emphasis on deterministic constructions and analysis in the standard model. We rename the notion to *KDI security*, to stress that we are not talking only about encryption (and hence the Input is not necessarily a Message). We demonstrate some inherent limitations of these notions and present positive and negative results concerning the KDI security of encryption, pseudo-random functions and (tweakable) pseudo-random permutations, with respect to this notion, both in the standard model and in the "ideal cipher model".

Definition and inherent limitations. We begin in Section 2 by exploring the notion of KDI-security and its limitations, specifically as they pertain to deterministic constructions. We observe that KDI-security of deterministic schemes cannot be achieved (even in an idealized models) without restricting the key-dependent queries that the attacker can make: Allowing the attacker to query a function f_s on multiple functions of the key necessarily translates into a KDI attack that *recovers the full key*, and this attack works even if the underlying primitive is an "ideal cipher" or a random oracle! (This is similar to the setting of "related key attacks" [3].) In practical terms this means that an application must restrict the types of information on the key that may potentially be "encrypted" under the key itself.

For this reason, we parametrize KDI security by the set of functions of the key that the attacker can use in its queries. Given the impossibility mentioned above when the set of functions is too rich, it makes sense to even investigate the minimal notion of KDI-security with respect to just a *single* function, as we may at least hope that even an abusive implementation that "encrypts its own key" will only do so in one form, rather than "encrypting" many copies of the key in many different forms (e.g., it may encrypt the key itself or a hash of the key, but not multiple, arbitrary functions of the key). In particular, negative results obtained in this minimalistic setting imply impossibility of stronger notions of KDI security. In this light, we investigate the existence of schemes that are KDI-secure with respect to *all* efficient deterministic functions of the key, as long as the attacker is restricted to query a *single* function of its choice in the attack.

Pseudo-random functions. We investigate in Sections 3 and 4 the existence of pseudo-random functions (PRFs) that are KDI secure with respect to any (single) function of the key, and present the following results:

1. We show that in the "ideal cipher model", KDI security is achievable if one restricts the functions of the key that can be queried to be independent of the ideal cipher itself.
2. In contrast we prove that this goal cannot be achieved in the standard model, by showing that for each (deterministic) PRF family there is a function of the key relative to which the given family in *not* KDI secure.
3. On the positive side, if we allow the PRF construction to depend on a fixed public random value, often called a "salt" (and do not allow the function of the key to depend on the same salt value), then we can get KDI-security in the standard model (assuming that standard PRFs exist).
4. The construction from 3 also implies a non-constructive proof that for every function g there is a PRF (whose description depends on g) that is KDI-secure against this particular function g. We show also a constructive proof for the case where g is a "well spread" function.
5. Finally, we describe an "arguably more natural" PRF (based on the Blum-Micali pseudo-random generator) that is KDI-secure with respect to the identity function.

Tweakable ciphers. We return in Section 5 to the "ideal cipher model" to study the KDI security of tweakable ciphers (which are the basis for the IEEE P1619 standard that provided the initial motivation of this work). We establish a definition of KDI security for tweakable ciphers, describe an attack on a scheme of Liskov et al. [18] thus demonstrating that it is not KDI-secure (even in the "ideal cipher model"), and then show that some other schemes (including the one by Rogaway [23]) are KDI-secure in this model.

Randomized encryption. We conclude in Section 6 by taking another look at randomized encryption, in particular PRF-based symmetric encryption. Black et al. proved in [5] that the encryption scheme $\mathrm{Enc}_s(x) = (r, f_s(r) \oplus x)$ is KDI-secure when f_s is implemented using a random oracle as $f_s(x) = H(s|x)$. We observe that this encryption scheme is not KDI-secure in general when we only require that f_s is a secure PRF, not even if f_s is itself KDI-secure. More surprisingly, this construction fails even for "natural" choices of the PRF f_s, such as when instantiated using the Davies-Meyer construction. (This serves as an interesting reminder of the caution one has to exercise when basing security on idealized models.)

On the positive side, we show that if f_s is a KDI-secure *invertible* PRF, then the encryption scheme $\mathrm{Enc}_s(x) = (r, f_s(r) \oplus f_s(x))$ is also KDI secure. Unfortunately our salted PRF from Item 3 above seems to be inherently non-invertible, but the "g-specific construction" from Item 4 can be made invertible and, if randomized, can work for any function g, not just a "well spread" one. Hence we obtain for every function g an encryption scheme $\mathrm{Enc}^{(g)}$ that is KDI secure with respect to that function g.

The moral. We view the results in this work as lending support to the "common cryptographic wisdom" that the practice of self encryption of a key is a dangerous abuse of a cryptosystem. We demonstrate that many security goals that can be stated with respect to this practice inherently cannot be achieved; in some cases not even in an idealized model. Our counter-example for the case of randomized encryption (Section 6) is particularly troubling: *We show a failure of a textbook construction for symmetric encryption with respect to a very natural implementation of its components. Moreover, this is the case in spite of the fact that almost the same construction was previously proved secure in the random oracle model!* We also show that similar warnings apply to other secret-key primitives, such as a PRF, when applied to the key or, more generally, to a function of the key. On the positive side, we show explicit constructions that achieve limited notions of KDI security (even in the standard model). Two interesting open questions that remain are finding, for each function g of the key, a (deterministic) PRF that is KDI secure with respect to g and, more fundamentally, finding a symmetric encryption scheme that can be proven KDI-secure in the standard model with respect to any function g.

Terminology. To simplify the presentation we state our results in the language of asymptotic security, using the terms "feasible" for probabilistic polynomial-time, "non-negligible"or "noticeable" for larger than some polynomial fraction, and "negligible" or "insignificant" for smaller than any polynomial fraction. It is clear, however, that all the results could also be stated in the language of "exact security" (at a price of a somewhat more cluttered presentation).[1]

2 Definitional approach and some intrinsic limitations

Roughly, to define security with respect to key-dependent input attacks we modify the standard attack scenarios for the various primitives that we study, by allowing the attacker to query its oracles not only on explicit strings but also on functions of the secret key. That is, where the original notion provided the attacker access to an oracle $O(\cdot)$, we add an oracle $O'(\cdot)$ that gets as input a description of a function g (e.g., in the form of a circuit that computes the function) and outputs $O(g(s))$ where s is the secret key of the construction in question. We will refer to the queries to O' as *functional queries*. We extend this definitional approach to the "ideal cipher model" by allowing oracle access to keyed random permutations (and their inverses) and by possibly allowing the functional queries to depend on these oracles.

Ideally, we would like to find constructions that remain secure even when the attacker can query the primitive on any efficient function of the key. There are, however, some inherent limitations to this approach. For example, letting the attacker query a cipher E_s on input $g(s) = E_s^{-1}(s)$, the key would be obviously exposed. A more general limitation arises in the context of deterministic primitives, as we show next.

KDI-insecurity against unrestricted queries. The idea of this argument is that an attacker can try to apply many different functions to the key s, and use collisions of the form $g(s) = g'(s)$ to do a binary search for the key s. That is, the attacker uses two different functional queries g, g', and

[1] On the other hand, the "exact security" language may be somewhat more natural when talking about block ciphers and tweakable block ciphers.

checks if it gets the same answer on both. This (in essence) tells the attacker whether $g(s) = g'(s)$, which cuts the key-space by two. Here we describe a simple example of this argument, which is essentially the same as the one described in [3] in the context of related-key attacks.

Let Ψ be any deterministic construction that has a secret key (such that the disclosure of s compromises the security of Ψ). For simplicity (this is not essential for the general argument), assume that both the key space and the input space of Ψ is $\{0,1\}^n$, and that for every fixed key $s \in \{0,1\}^n$ the function $\Psi_s(\cdot)$ is injective. Consider now a set of functions $\{g_i, g'_i : 1 \leq i \leq n\}$ with the property that for every i and every s we have $g_i(s) = g'_i(s)$ if and only if the i'th bit of s is zero. (An example is a set of functions containing additions of constants modulo 2^n as well as xor with constants from $\{0,1\}^n$. For example, for all $i < n$ we set g_i to be xor with $0^{n-i}10^{i-1}$ and g'_i to be addition of 2^i modulo 2^n.)

The attacker then simply queries its oracle Ψ' on the inputs g_i and g'_i for all i. If the i'th bit of the secret key is 0 then $g_i(s) = g'_i(s)$ and therefore $\Psi'(g) = \Psi(g_i(s)) = \Psi(g'_i(s)) = \Psi'(g')$ (because Ψ is deterministic). On the other hand, if the i'th bit of the secret key is 1 then $g_i(s) \neq g'_i(s)$ and since Ψ is injective it follows that also $\Psi'(g) = \Psi(g_i(s)) \neq \Psi(g'_i(s)) = \Psi'(g')$. The attacker can therefore determine all the bits of the secret key s in violation of the security of Ψ.

Parametrized definition. As a consequence of the above observations, and similar to the case of key-related attacks [3], the definition of KDI-security will be parametrized by a class of function descriptions $\mathcal{C}$, and all the queries to the O' oracle will be restricted to functions from $\mathcal{C}$. The question of whether KDI security with respect to a certain class $\mathcal{C}$ provides a meaningful level of security depends heavily on the application. In some cases anything less than "all polynomial-size circuits" may be insufficient while in others having $\mathcal{C}$ restricted to the identity function only (i.e., one is allowed to query the primitive on the key itself but not on other functions of the key) may suffice.

In many cases, providing security assurance against one function of the key, i.e., the case where $|\mathcal{C}| = 1$, will be of significant value: we may at least hope that even an abusive implementation that "encrypts its own key" will only do so in one form, rather than encrypting many copies of the key in many different forms. Given the limitations discussed above (and more to be shown in the sequel) we will judge different constructions under the "modest" requirement that they resist singleton classes $|\mathcal{C}| = 1$. We would like to get a construction that is KDI secure against *all* singleton classes (i.e., the attacker is allowed to choose a single function $g(s)$ to query but the function g could be any efficient function of s). Unfortunately, examples such as the one with the function $g(s) = E_s^{-1}(s)$ demonstrate that even this modest goal cannot always be achieved. In such a case we will study the "minimalist" requirement that a construction is KDI secure against *one specific function*.

3 Pseudo-random Functions

Below we use the convention that for security parameter n, the key for a pseudo-random function is a random n-bit string, and that the function is from $\{0,1\}^{\ell_{\text{in}}(n)}$ to $\{0,1\}^{\ell_{\text{out}}(n)}$ where ℓ_{in} and ℓ_{out} are efficiently computable and polynomially bounded. Then a family of pseudo-random functions

is an ensemble

$$\mathcal{F} = \left\{ f_s : \{0,1\}^{\ell_{\text{in}}(n)} \to \{0,1\}^{\ell_{\text{out}}(n)} \;\middle|\; s \in \{0,1\}^n \right\}_{n \in \mathcal{N}}$$

and we require that there is an efficient evaluation procedure that given any $s \in \{0,1\}^n$ and any $x \in \{0,1\}^{\ell_{\text{in}}(n)}$ computes $y = f_s(x)$.

The standard security definition for pseudo-random functions as defined in [10] asserts that no feasible attacker $\mathcal{A}^{\phi}(1^n)$ (with oracle access to ϕ) can distinguish with non-negligible advantage the case where $\phi = f_s$ for a random $s \in_R \{0,1\}^n$ from the case where ϕ is chosen as a random function from $\{0,1\}^{\ell_{\text{in}}(n)}$ to $\{0,1\}^{\ell_{\text{out}}(n)}$.

In order to capture KDI security of pseudo-random functions, we augment the standard definition of pseudo-random functions by letting the adversary also access another oracle ϕ' that takes as input a description of a function g, and outputs $\phi(g(s))$. (The output size of the functions g considered here is assumed to match the size of inputs to the pseudorandom function f.)

Definition 1 (KDI-secure PRFs) *A family $\mathcal{F}$ of pseudo-random functions is* KDI-secure *with respect to a class $\mathcal{C}$ of circuits if no feasible attacker $\mathcal{A}^{\phi,\phi'}(1^n)$ (with oracle access to ϕ, ϕ') can distinguish with non-negligible advantage between the following two cases:*

1. *$\phi = f_s$ for a random $s \in_R \{0,1\}^n$ and for any $g \in \mathcal{C}$ $\phi'(g) = \phi(g(s))$;*
2. *ϕ is chosen as a random function $\phi : \{0,1\}^{\ell_{\text{in}}(n)} \to \{0,1\}^{\ell_{\text{out}}(n)}$, s is chosen at random in $\{0,1\}^n$, and for any $g \in \mathcal{C}$, $\phi'(g) = \phi(g(s))$.*

Note: Many of our results apply to the case where $\mathcal{C}$ includes a single function g; in this case, one can dispense of the ϕ' oracle and simply assume that the attacker is given the value of $\phi(g(s))$.

On KDI-insecure PRFs. We first observe that secure PRFs (or block ciphers) are not necessarily KDI-secure, not even with respect to the identity function. Indeed, given any secure PRF family $F = \{F_s\}$, one can trivially modify it as follows:

$$F'_s(x) = \begin{cases} s & \text{if } x = s \\ F_s(x) & \text{otherwise} \end{cases}$$

Clearly, the family $F' = \{F'_s\}$ is still a secure PRF, but it is not KDI-secure with respect to the identity function. Similarly, if we start with a secure cipher E (a strong pseudo-random permutation) we can build another secure cipher E' that is not KDI-secure with respect to the identity function:

$$E'_s(x) = \begin{cases} s & \text{if } x = s \\ E_s(s) & \text{if } x = E_s^{-1}(s) \\ E_s(x) & \text{otherwise} \end{cases}$$

Negative Example 1 *There exist secure PRFs and secure block ciphers that are KDI-insecure with respect to the identity function.* ■

3.1 Constructions in the "ideal-cipher model"

We saw above that the construction $f_s(x) = E_s(x)$ where E is a secure block cipher is not necessarily KDI-secure. Here we show that this construction is at least KDI secure in the "ideal-cipher model". We begin by adapting our definition of KDI security to Shannon's "ideal cipher model".

Recall that in the ideal-cipher model, all the parties (including the attacker) are given black-box access to two tables $\Pi(\cdot,\cdot)$ and $\Pi^{-1}(\cdot,\cdot)$. These tables are chosen at random subject to the condition that for every "key" s, $\Pi(s,\cdot)$ is a permutation and $\Pi^{-1}(s,\cdot)$ is its inverse (and all these permutations are over the same domain). For simplicity of presentation we assume that on security parameter n, the key that selects the permutation is of length n bits and the permutations themselves are over $\{0,1\}^n$. Namely, for each $s \in \{0,1\}^n$, $\Pi(s,\cdot)$ is a random permutation over $\{0,1\}^n$, and $\Pi^{-1}(s,\cdot)$ is the inverse permutation.

We augment the definition of KDI-security to the ideal-cipher model by providing the attacker with oracle-access to Π, Π^{-1}, and more importantly by potentially allowing the class of function-descriptions in $\mathcal{C}$ to depend on Π and/or Π^{-1}. Specifically, in this case we allow the circuits in $\mathcal{C}$ to include also Π-gates that on input (s,x) return $\Pi(s,x)$ (and similarly also Π^{-1}-gates). When stating a result in this paper in the context of the ideal cipher model we will specify whether we assume the functional queries $g(s)$ to depend or not in the oracles Π and Π^{-1}.

Note that when adapting Definition 1 to the "ideal cipher model", the attacker's advantage is measured with respect to the probability distribution where for each $s \in \{0,1\}^n$, $\Pi(s,\cdot)$ is a random permutation over $\{0,1\}^n$ and $\Pi^{-1}(s,\cdot)$ is the inverse permutation (also, this advantage is parametrized in terms of the number of queries the attacker performs to these oracles as well as the number of such queries performed while computing the function g, if the latter depends on Π, Π^{-1}).

Remark. The distinction between circuits that include Π and Π^{-1} gates and circuits that do not is one of the main reasons for using the "ideal cipher model" in the KDI context. Indeed, in some cases we would like to argue that a cipher is KDI-secure with respect to any function g that "does not depend on the cipher itself". This restriction is generally not well defined in the standard model but can be captured in the "ideal cipher model" by specifying that the function g is described by a circuit that does *not* include Π or Π^{-1} gates.

KDI-security of $f_s(x) = \Pi_s(x)$. It is easy to see that even in the "ideal cipher model", we can find functions g that depend on Π such that the construction $f_s(x) = \Pi(s,x)$ is not KDI-secure with respect to g. For example, if we set $g(s) = \Pi^{-1}(s,s)$ then

$$f_s(g(s)) = \Pi(s, \Pi^{-1}(s,s)) = s \tag{1}$$

However, we can show that this construction is KDI secure with respect to every function g that does not depend on Π, specifically:

Theorem 1 *Let g be any Boolean circuit* with no Π-gates or Π^{-1}-gates. *Then the construction $f_s(x) = \Pi_s(x)$ is a KDI-secure pseudo-random function in the "ideal cipher model" with respect to the singleton class $\mathcal{C} = \{g\}$.*

Proof Sketch The attacker $\mathcal{A}$ has access to three oracles: $\Pi(\cdot,\cdot)$, and $\Pi^{-1}(\cdot,\cdot)$ that represent the ideal cipher and $f(\cdot)$ which is either $\Pi(s,\cdot)$ for a random s (the "real case"), or an independent

random function (the "random case"). In addition, the attacker is given the value $f(g(s))$, where s is the key in the "real case" and just a random string in the "random case".

We consider a "hybrid case" which is just like the "random case", except that f is chosen as a random permutation rather than a random function. Clearly, the "hybrid" and the "random" cases cannot be distinguished upto the birthday bound. The heart of the proof is in showing that the attacker cannot distinguish the "hybrid" from the "real" case.

Next we argue that the attacker has only a negligible probability to ever query its Π or Π^{-1} oracles with the correct key s: since g is independent of Π, Π^{-1} then all the values that the attacker sees are entries of Π, Π^{-1} that by themselves are independent of s. (This is where the counterexample $g(s) = \Pi^{-1}(s, s)$ comes in, dependent on Π^{-1} allows the attacker to ask for "the value in the entry in which s is written".) As long as the attacker still did not query Π or Π^{-1} with the right key s, then the answer that it got so far can be completely simulated by the attacker itself, save for cases where a query $f(x)$ on some string x happened to return the same value as $f(g(s))$. This last event either happens with negligible probability (if the pre-image of $g(s)$ is smaller than $2^{n/2}$) or they still leave exponentially many possibilities for s (if the pre-image of $g(s)$ is larger). Hence the attacker only has an exponentially small probability of hitting the right key s in the next query that it makes.

But short of querying Π, Π^{-1} on the right s (and since g is independent of Π, Π^{-1}), the answers that the attacker gets in both the "hybrid" and the "real" cases are drawn from the same probability distribution. Namely the initial value of $f(g(s))$ and the answers to all the queries to f are computed using a random permutation which is independent of the queries that the attacker makes to Π, Π^{-1}. ■

Similar claims can be made for many of the published PRP-to-PRF constructions in the literature, e.g., the schemes from [4], the truncation construction [12], the XOR construction [20], etc.[2]

4 KDI-Secure PRFs in the Standard Model

We have shown that an ideal cipher is also a KDI-secure PRF with respect to any function g that does not depend on the cipher itself. On the other hand, we saw, in the examples following Definition 1, that in the standard model a secure cipher (or PRF) does not have to be KDI-secure, not even with respect to simple functions such as the identity function. Here we investigate the existence (and constructibility) of KDI-secure schemes in the standard model. One obstacle is the fact that in the standard model it is harder to impose independence between a PRF (or cipher) scheme and the function g.

4.1 No single deterministic construction for all g

We begin by showing that one cannot get a single deterministic construction that is KDI secure in the standard model with respect to every singleton class $\{g\}$.

[2]For these constructions one gets KDI security in the "ideal cipher mode" but not necessarily KDI security "beyond the birthday bound". (Getting PRF security beyond the birthday bound has been the initial motivation for these constructions.)

Theorem 2 (No single construction for all g) *There exists no deterministic construction of a pseudo-random function family that is KDI-secure with respect to $\{g\}$ for all functions g.*

Proof Let $F = \{F_s\}$ be a pseudo-random family. Define $g_F(s) = F_s(0)$, and we show that F is not KDI-secure with respect to $\{g_F\}$. An attacker $\mathcal{A}$ queries its key-dependent oracle to obtain $a = F_s(g_F(s)) = F_s(F_s(0))$; then it queries the F-oracle on 0 to obtain $b = F_s(0)$; finally, it queries the F-oracle on b to obtain $c = F_s(b)$. $\mathcal{A}$ outputs 1 if $a = c$ and 0 otherwise. Clearly, when the oracle F is answered with the pseudo-random function F_s then $a = c$ and $\mathcal{A}$ outputs 1 with probability 1, while if the F-oracle is random then a and c are independent random values and hence $\mathcal{A}$ outputs 1 only with small probability. ■

Given Theorem 2, our options for obtaining positive results in the standard model are either to settle for randomized (or salted) constructions, or to come up with different constructions for different functions g. In Section 4.2 we show a randomized construction that is KDI secure with respect to g for any function g, while in Section 4.3 we show that for each "well spread" function g one can construct an explicit deterministic PRF (whose definition depends on g) that is KDI-secure with respect to the function g. This leaves open the question of whether one can construct, for *each* function g, an *explicit* deterministic PRF that is KDI secure with respect to g. (We note that the existence of non-explicit g-specific deterministic schemes for each g follows from our randomized construction.)

4.2 A single randomized construction for all g

Here we describe a construction for a pseudo-random function family F that depends on a public random "salt" r, such that for every function g, with high probability over the choice of r, the family F^r is KDI-secure with respect to g. This construction uses in an essential way strong randomness extractors [21], see a brief description and definitions in Appendix A. On a high level, our construction is as follows:

$$F^r_s(x) = f_{\mathsf{ext}_r(s)}(\mathit{Compress}(x))$$

where f is a standard PRF, ext is a randomness extractor, and *Compress* is some function whose output is much shorter than its input. The goal is to "break the dependence" between the input and the key, even in the case where the input is $g(s)$. Roughly, since *Compress* outputs very short strings, then there must be many different keys s that match any value of $\mathit{Compress}(g(s))$. Namely, s still has high entropy even given the value $\mathit{Compress}(g(s))$, so $\mathsf{ext}_r(s)$ is likely to be random and independent of $\mathit{Compress}(g(s))$. Of course, one also needs to argue *Compress* does not introduce easy to find collisions which would destroy the PRF property of F. This can be done by using a collision resistant function, but then we would have to assume that those exist. Instead, below we show that some form of universal hashing is sufficient in this context. (This complicates the proof, but let us rely on a weaker assumption.) We now provide the details.

Theorem 3 (A single randomized construction for all g) *If pseudo-random function families exist, there there exists a salted pseudo-random function family F^r, which is KDI-secure with*

respect to $\{g\}$ *for any* efficiently computable[3] *function* g*, with high probability over the choice of the salt* r*.*

Proof Let $m = n/12$. We use the following components:

- $f : \{0,1\}^m \times \{0,1\}^{6m} \to \{0,1\}^{12m}$ is a standard pseudo-random function family with m-bit keys, $6m$-bit inputs, and $12m$-bit outputs, and denote $f_s(\cdot) = f(s,\cdot)$. (Such a PRF exists under the assumption that PRFs, or one-way functions, exist.)
- $\mathsf{ext} : \{0,1\}^t \times \{0,1\}^{12m} \to \{0,1\}^m$ is a strong $(4m, 2^{-m}, 2^{-9m})$ randomness extractor,[4] and denote $\mathsf{ext}_r(\cdot) = \mathsf{ext}(r,\cdot)$.
- $H : \{0,1\}^m \times \{0,1\}^{12m} \to \{0,1\}^{6m}$ is a hash function, which is defined as follows: The m-bit key is interpreted as a non-zero element $u \in GF(2^{6m})$ (say, by padding it with $5m$ one bits), and the $12m$-bit input is interpreted as two element $x_1, x_2 \in GF(2^{6m})$. Then the function is evaluated as
$$H(u, \langle x_1, x_2\rangle) = H_u(x_1, x_2) \stackrel{\text{def}}{=} u \cdot x_1 + x_2 \tag{2}$$
when the evaluation is in $GF(2^{6m})$. (The scheme can use more general universal hash functions; see the proof of Lemma 1 for the properties of this function H that we use in our analysis.)

We define $F : \underbrace{\{0,1\}^{2t}}_{\text{salt}} \times \underbrace{\{0,1\}^n}_{\text{key}} \times \underbrace{\{0,1\}^n}_{\text{input}} \to \underbrace{\{0,1\}^n}_{\text{output}}$:

$$F_s^{r_1,r_2}(x) \stackrel{\text{def}}{=} f_{\mathsf{ext}_{r_2}(s)}\left(H_{\mathsf{ext}_{r_1}(s)}(x)\right) \tag{3}$$

(recall that $n = 12m$). Roughly, the proof that F^{r_1,r_2} is KDI-secure consists of two arguments:

- In Lemma 1 we prove that the probability of collisions in the hash function H is small (this is the most technically involved part of the proof).
- Moreover, since the key and output of $H(g(s))$ together are only $7m$-bits long, then the PRF key s must have high min-entropy even conditioned on these two values, and therefore $\mathsf{ext}_{r_2}(s)$ is nearly uniform even given the extractor seeds and these two values.

Lemma 1 *Let* $g : \{0,1\}^{12m} \to \{0,1\}^{12m}$ *be any fixed function,* $H : \{0,1\}^m \times \{0,1\}^{12m} \to \{0,1\}^{6m}$ *be the function that is defined in Eq. (2), and let* $\mathsf{ext} : \{0,1\}^t \times \{0,1\}^{12m} \to \{0,1\}^m$ *be a strong* $(4m, 2^{-m}, 2^{-9m})$ *randomness extractor. We say that a seed* $r \in \{0,1\}^t$ *for the extractor is* $\underline{g\text{-bad}}$ *if there exist* $x, x' \in \{0,1\}^{12m}$ *such that*

- *either* $\Pr_s[u = \mathsf{ext}_r(s),\ H_u(x) = H_u(x')] > 2^{-m+1}$,
- *or* $\Pr_s[u = \mathsf{ext}_r(s),\ x \neq g(s)$ *and* $H_u(x) = H_u(g(s))] > 2^{-m+2}$.

[3] Hereafter, we limit our treatment to efficiently (polynomial-size) computable functions g.

[4] That is, for any distribution $\mathcal{D}$ with min entropy $4m$ or more, for all but a 2^{-9m} fractions of the seeds r, it holds that $\mathsf{ext}_r(\mathcal{D})$ is 2^{-m}-close to the uniform distribution over $\{0,1\}^m$ (see Appendix A). The 2^{-9m} parameter facilitates the proof, though it seems that one can do with a regular $(4m, 2^{-m})$-extractor.

*Otherwise, r is g-*good*. Then* $\Pr_r[r \text{ is } g\text{-bad}] < 2^{-m}$.

The proof of the lemma is presented later in this section. Using Lemma 1 we now prove the KDI-security of the PRF from Eq. (3) using a sequence-of-games argument. We fix a function g and a feasible adversary A, and consider several games between this adversary and a "challenger". We start with Game 0 that describes the "random" case in the definition of KDI-security, then go through two hybrid games (called Games 1 and 2), until we get to Game 3 that describes the "real" case in the definition of KDI-security. We show that the adversary cannot distinguish between successive games in the sequence, except with an insignificant advantage. We use Lemma 1 in the transition from Game 0 to Game 1, the security of the underlying PRF is used in the transition from Game 1 to Game 2, and the argument about "high conditional min-entropy" that was sketched above is used in the transformation from Game 2 to Game 3.

For the rest of the proof, fix an efficiently computable function $g : \{0,1\}^n \to \{0,1\}^n$ and an adversary A. We assume that A is deterministic (or else we fix also the randomness of A). Since we consider only a single function g, we can slightly simplify all the games by providing the adversary with $F_s^{r_1,r_2}(g(s))$ as input (instead of waiting for it to query F^{r_1,r_2} on $g(s)$).

Game 0. In this game the challenger chooses three random strings $r_1, r_2 \in_R \{0,1\}^t$ and $s \in_R \{0,1\}^n$, and a random function $\phi : \{0,1\}^n \to \{0,1\}^n$. It computes $x_0 = g(s)$, provides the adversary A with input $(r_1, r_2, \phi(x_0))$, and answers any further queries $x_i \in \{0,1\}^n$ of A with $\phi(x_i)$.

Below it is convenient to view this game as if the challenger chooses ahead of time all the potential outputs of ϕ (call them $y_0, y_1, \ldots, y_q$), each chosen uniformly at random in $\{0,1\}^n$. The i'th query $\phi(x_i)$ is answered with y_i if x_i is different from all the other ϕ queries so far, and if $x_i = x_j$ for some $j < i$ then this query is answered with the same answer that was given to $\phi(x_j)$.

Game 1. The challenger chooses random strings $r_1, r_2 \in_R \{0,1\}^t$ and $s \in_R \{0,1\}^n$, and a random function $\phi' : \{0,1\}^{n/2} \to \{0,1\}^n$. It computes $u = \mathsf{ext}_{r_1}(s)$, $x_0 = g(s)$, and $h = H_u(x_0)$. Then it provides the adversary A with input $(r_1, r_2, \phi'(h))$, and answers any further queries $x_i \in \{0,1\}^n$ of A with $\phi'(H_u(x_i))$.

We can view this game as choosing $y_0, y_1, \ldots, y_q$ ahead of time, then answering the i'th query x_i with y_i if $H_u(x_i)$ is different from all previous values $H_u(x_j)$, and answering as in a previous query x_j if there is a collision. Note that as long as there are no collisions then A's questions depend only on the r, r' and the y_i's, so we can consider playing the game as if there are no collisions, and choosing s only after the fact. This will give us exactly the view of Game 0, and it will be inconsistent with Game 1 only if the choice of s implies a collision. But since all the queries of A were made before s was chosen, we can apply Lemma 1 to conclude that if r_1 is not g-bad then the probability of collisions is at most $\binom{q}{2}2^{-m+1} + q2^{-m+2}$, and we only have probability 2^{-m} of choosing a g-bad value for r_1 (where $m = n/12$).

Game 2. The challenger now chooses four random strings $r_1, r_2 \in_R \{0,1\}^t$, $s \in_R \{0,1\}^n$, and $u' \in_R \{0,1\}^{n/12}$, computes $u = \mathsf{ext}_{r_1}(s)$, $x_0 = g(s)$, and $h = H_u(x_0)$. Then it provides the

adversary A with input $(r_1, r_2, f_{u'}(h))$, and answers any further query $x_i \in \{0,1\}^n$ of A with $f_{u'}(H_u(x_i))$.

Since the PRF key u' here is chosen independently of everything else then the security of $f_{u'}$ as a (standard) pseudo-random function implies that the adversary has only a negligible advantage in distinguishing Game 2 from Game 1.

Game 3. The challenger chooses only three random strings $r_1, r_2 \in_R \{0,1\}^t$ and $s \in_R \{0,1\}^n$. It computes $u = \mathsf{ext}_{r_1}(s)$, $u' = \mathsf{ext}_{r_2}(s)$, $x_0 = g(s)$, and $h = H_u(x_0)$. The challenger provides the adversary A with input $(r_1, r_2, f_{u'}(h))$, and answers any further query $x_i \in \{0,1\}^n$ of A with $f_{u'}(H_u(x_i))$.

The crux of this step of the proof is showing that the views of the adversary in Game 2 and Game 3 are statistically close. Since we assume that the adversary A is deterministic, then the view in both games is uniquely determined by the strings r_1, u, h, r_2 and u' (since everything else is computed deterministically from them). Thus it is enough to show that the distribution over these strings is almost the same in both games.

To show this, consider the modified challenger: it chooses $r_1 \in_R \{0,1\}^t$ and $s^* \in_R \{0,1\}^n$ and computes $u = \mathsf{ext}_{r_1}(s^*)$ and $h = H_u(g(s^*))$. Then it re-chooses s conditioned on u and h, namely it chooses $s \in_R \mathsf{ext}_{r_1}^{-1}(u) \cap H_u^{-1}(h)$, then chooses $r_2 \in_R \{0,1\}^t$ and computes $u' = \mathsf{ext}_{r_2}(s)$. Clearly, this modified challenger induces an identical distribution over the vector $\langle r_1, u, h, r_2, u' \rangle$ as in Game 4.

With this modification, denote by BAD the event (over the choice of r_1, s^*) in which the set $\mathsf{ext}_{r_1}^{-1}(u) \cap H_u^{-1}(h)$ (from which s is chosen) has less than 2^{4m} elements. (Recall that $m = n/12$). Since h and u together have only $7m$ bits, then a simple counting argument shows that for every r_1, the probability of this BAD event (over the choice of s^*) is at most 2^{-m}. On the other hand, if the BAD event did not occur then s is chosen uniformly from a set of size at least 2^{4m}, so it has at least $4m$ bits of min-entropy. Since ext is a strong extractor, then in this case we know that even after fixing r_2, u_1, h, the distribution over $u' = \mathsf{ext}_{r_2}(s)$ is no more than 2^{-m} away from the uniform distribution over $\{0,1\}^m$, except with error probability of 2^{-9m} over the choice of r_2.

We conclude that the statistical distance between the views of A in Game 2 and Game 3 is at most $2^{-m} + 2^{-9m}$. This concludes the proof of Theorem 3. ■

Proof of Lemma 1. The rare collisions between two fixed inputs x, x' are easy to show; the harder problem is showing that there are no collisions between x and $g(s)$ (in principle, since the key to H, u, depends on s then there could be a value x that collides with relatively high probability with $g(s)$). The idea behind this argument is as follows: on one hand, if $g(s)$ has many pre-images then ext will extract an almost uniform hashing key u even given $g(s)$. On the other hand, if there are many values of s for which $g(s)$ has very few pre-images, then $g(s)$ itself has high min-entropy, even given the hashing key $u = \mathsf{ext}_r(s)$. Since for any fixed u there are only small number of values that collide with x, then the chances that $g(s)$ "hits" any of these values is small. To make these arguments formal, we use the following three properties of the hash function H:

- H is "2^{-m}-almost-universal": For every fixed $x \neq x' \in \{0,1\}^{12m}$, $\Pr[H_u(x) \neq H_u(x')] \leq 2^{-m}$, where the probability is over choosing u uniformly at random from $\{0,1\}^m$.

- H is regular: For every fixed key $u \in \{0,1\}^m$ and every output $z \in \{0,1\}^{6m}$, the pre-image size of z under H_u is bounded by $|H_u^{-1}(z)| \leq 2^{12m-6m} = 2^{6m}$.
- H has much shorter keys than outputs: the key-space for H has only 2^m elements.

It is easy to check that the function from Eq. (2) satisfy these conditions.

Denote by $SMALL \subseteq \{0,1\}^{12m}$ the subset of the keys s for which the pre-image of $g(s)$ under g is smaller than 2^{4m},

$$SMALL \stackrel{\text{def}}{=} \left\{ s \in \{0,1\}^{12m} : |g^{-1}(g(s))| < 2^{4m} \right\}$$

Also let $BIG \stackrel{\text{def}}{=} \{0,1\}^{12m} \setminus SMALL$ and $g(BIG) \stackrel{\text{def}}{=} \{g(s) : s \in BIG\}$. For every element $y \in g(BIG)$, consider the uniform distribution on $g^{-1}(y)$: by definition this distribution has at least $4m$ bits of min-entropy, so we hope to be able to use ext to extract from it an almost-uniform m-bit string. A seed r is said to *fail for* y if the distribution $\mathsf{ext}_r(g^{-1}(y))$ is more than 2^{-m} away from the uniform distribution on $\{0,1\}^m$.

Since ext is an $(4m, 2^{-m}, 2^{-9m})$ randomness extractor then for any fixed $y \in g(BIG)$ the probability that a random seed r fails for y is at most 2^{-9m}. Since each $y \in g(BIG)$ has at least 2^{4m} pre-images under g then there could be at most 2^{8m} such elements, $|g(BIG)| \leq 2^{8m}$. It follows from the union bound that the probability that a random seed r fails for any $y \in BIG$ is at most 2^{-m}. If r does not fail for any y then in particular it holds for every $y \in BIG$ and very $x \neq x'$

$$\begin{aligned} &\Pr[s \in_R g^{-1}(y),\ u = \mathsf{ext}_r(s), H_u(x) \neq H_u(x')] \\ &\leq\ \mathsf{dist}(\mathsf{ext}_r(g^{-1}(y)),\ \mathcal{U})\ +\ \Pr[u \in_R \{0,1\}^m, H_u(x) \neq H_u(x')] \quad = \ 2^{-m+1} \end{aligned}$$

From now on, we consider only seeds r that do not fail for any $y \in g(BIG)$.[5] In addition, if the function g has $|SMALL| > 2^{4m}$ then we require also that $\mathsf{ext}_r(SMALL)$ is at most 2^{-m} away from uniform. (Still, a random seed satisfies this with probability at least $1 - 2^{-m}$, since if $|SMALL| > 2^{4m}$ then $|g(BIG)| \leq 2^{8m} - 1$, so again we have at most 2^{8m} different distributions.) For the rest of the proof fix a seed r that satisfies these conditions, and we show that it must be g-good.

Clearly, with this seed r the distribution $\mathsf{ext}_r(\{0,1\}^{12m})$ is at most 2^{-m} away from uniform (since it is a convex combination of distributions that are all at most 2^{-m} away from uniform). It follows that for any two fixed $x, x' \in \{0,1\}^{12m}$ the probability that $H_u(x) = H_u(x')$ when $u = \mathsf{ext}_r(s)$ (for a random $s \in_R \{0,1\}^{12m}$) is at most 2^{-m+1}. It remains to show that the probability of collision between x and $g(s)$ is small.

Consider now the following experiment: we choose $s^* \in_R \{0,1\}^{12m}$ and compute $y = g(s)$. If $s^* \in BIG$ then we re-choose $s \in_R g^{-1}(y)$, and otherwise we re-choose $s \in_R SMALL$. Clearly, this experiment induces the same distribution on s. Now fix some $x \in \{0,1\}^{12m}$ and we show that the probability of a collision $H_u(x) = H_u(g(s))$ where $u = \mathsf{ext}_r(s)$ is at most 2^{-m+1}. We start by proving that the collision probability *conditioned on* $s^* \in BIG$ is small. This is quite clear, since after we choose s^* then the value of $y = g(s^*)$ is already fixed (since we re-choose $s \in_R g^{-1}(y)$).

[5] It is sufficient to consider r's that may fail for a negligible fraction of y's, but the argument is simplified when one considers r values that never fail.

But the value of u is still nearly uniform, so if $y \neq x$ then the probability of a collision (over the re-choosing s and computing $u = \mathsf{ext}_r(s)$) is at most 2^{-m+1}. Hence

$$\begin{aligned}&\Pr_s[s \in BIG \text{ and } x \neq g(s) \text{ and } H_u(g(s)) = H_u(x)]\\ &\leq \Pr_s[x \neq g(s) \text{ and } H_u(g(s)) = H_u(x) \mid s \in BIG] \leq 2^{-m+1}\end{aligned}$$

We complete the proof by bounding the collision-probability also for the case $s \in SMALL$. (This is the only place where we use the last two conditions from above on the hash function H.) Indeed, we have

$$\begin{aligned}&Pr_s[s \in SMALL \text{ and } H_{\mathsf{ext}_r(s)}(g(s)) = H_{\mathsf{ext}_r(s)}(x)]\\ &= \sum_u Pr[s \in SMALL \text{ and } u = \mathsf{ext}_r(s) \text{ and } H_u(g(s)) = H_u(x)]\\ &\leq \sum_u Pr[s \in SMALL \text{ and } H_u(g(s)) = H_u(x)] \quad (*)\end{aligned}$$

Recall that for any fixed hashing key u and output $z = H_u(x)$, the pre-image of z under H_u is of size 2^{6m}, so there are at most 2^{6m} values of $g(s)$ that can collide with x under H_u. Moreover, since $s \in SMALL$ then each such value of $g(s)$ has at most 2^{4m} pre-images. Hence for every fixed hashing key u there are at most 2^{10m} values $s \in SMALL$ for which $H_u(g(s)) = H_u(x)$. If we now use the fact that there are at most 2^m different keys u then we can continue the inequality $(*)$ as

$$(*) \quad \leq \quad \sum_u \frac{2^{6m} \cdot 2^{4m}}{2^{12m}} \quad = \quad 2^{m+6m+4m-12m} \quad = \quad 2^{-m}$$

We therefore conclude that

$$\begin{aligned}&\Pr_s[x \neq g(s) \text{ and } H_u(g(s)) = H_u(x)]\\ &\leq \Pr_s[s \in BIG \text{ and } x \neq g(s) \text{ and } H_u(g(s)) = H_u(x)] \; + \; \Pr_s[s \in SMALL \text{ and } H_u(g(s)) = H_u(x)]\\ &\leq \; 2^{-m+1} + 2^{-m} \; \leq \; 2^{-m+2}\end{aligned}$$

■

Extension to larger families. One can ask whether the result from Theorem 3 can be extended to provide KDI-security against larger families, not just a singleton $\{g\}$. One hurdle in proving such extensions is that, as we saw in Section 2, when the family of functions g is too large then no construction can be KDI secure against it. So we must restrict the function family $\mathcal{C}$ so as to exclude a binary-search on the key. This can be formalized in a manner similar to the way it was done in [3]. Moreover, by extending $\mathcal{C}$ we get more distributions from which we may want to extract entropy (roughly 2^{8m} for every function $g \in \mathcal{C}$), and we need to extend Lemma 1 to prove that even collisions between $H_u(g_1(s))$ and $H_u(g_2(s))$ are rare, for any $g_1, g_2 \in \mathcal{C}$. This can be done by using even stronger extractors and even smaller hashing keys. Using these modifications, we can show that the construction from Eq. (3) (with somewhat stronger extractors and smaller hashing keys) is KDI-secure with respect to every class $\mathcal{C}$ with $2^{poly(n)}$ functions that "does not allow binary search for the key".

4.3 On g-dependent deterministic KDI-secure schemes

The construction from Section 4.2 implies a *non-constructive deterministic* g-dependent KDI-secure scheme for each function g (i.e., for each g, there exist values r_g, r'_g for which F^{r_g,r'_g} is KDI-secure with respect to g). In this section we attempt a constructive proof of the same fact, namely, a deterministic transformation that takes a circuit that computes g and produces a PRF that is KDI-secure with respect to function g. We succeed to do so only for functions g that are "well spread" (i.e., for each s, $g^{-1}(g(s))$ is not too big).

First try. We begin with a simple construction that at first glance looks as if it should work. Let $f = \{f_s\}$ be a pseudo-random function with $(n+1)$-bit inputs and n-bit outputs and keys. Fix some function $g : \{0,1\}^n \to \{0,1\}^n$, and define a family $F^{(g)} = \{F_s^{(g)}\}$ with n-bit inputs, outputs, and keys:

$$F_s^{(g)}(x) = \begin{cases} f_s(1|x) & \text{if } x \neq g(s) \\ f_s(0) & \text{if } x = g(s) \end{cases} \tag{4}$$

Perhaps surprisingly, we show that this construction does not always work:

Negative Example 2 *There are functions g s.t. $F^{(g)}$ from Eq. (4) is not KDI-secure with respect to $\{g\}$.*

Proof Define $g(s) = f_s(0)$. We show a KDI-attacker $\mathcal{A}$ that distinguishes between $F^{(g)}$ and a random function. The attacker $\mathcal{A}$ has input $a = \phi'(g) = \phi(g(s))$, it queries the oracle ϕ on a to get $b = \phi(a)$, and outputs 1 if $a = b$ and 0 otherwise.

When the ϕ-oracle is a real one (i.e., instantiated with $F_s^{(g)}$) then we have $a = F_s^{(g)}(g(s)) = f_s(0) = g(s)$, and therefore also $b = F_s^{(g)}(a) = F_s^{(g)}(g(s)) = f_s(0)$, thus $\mathcal{A}$ outputs 1 with probability 1. On the other hand, if the ϕ oracle is a random function then a is a random value and $b = \phi(a)$ is an independent random value, hence $a = b$ happens only with small probability. ■

The reason for this counter-example is that the attacker is able to compute $g(s)$ given access to $F^{(g)}$. Indeed, it is not hard to show that when $g(s)$ is unpredictable even given access to $F^{(g)}$ then the construction from above is secure (a proof is implicit in the proof of Lemma 2 below). Thus we try to construct a PRF $F^{(g)}$ such that $g(s)$ is unpredictable given access to $F_s^{(g)}(\cdot)$ and then use it as above. (Clearly, this is not possible if there is a single value of $g(s)$ that occurs with noticeable probability, but it does work as long as $g(s)$ has sufficient min-entropy.)

Lemma 2 *If PRF families exist, then there exist a transformation $\mathcal{T}$ that takes any constant $\alpha > 0$ and a description of an (efficiently computable) function g, and outputs a description of a PRF family $F^{(g)}$, such that if the distribution $g(s)$ has min-entropy at least $\alpha n + \omega(\log n)$ then $F^{(g)}$ is KDI-secure PRF with respect to the singleton class $\{g\}$.*

Proof Fix some $\alpha > 0$, and let $f : \{0,1\}^{\alpha n} \times \{0,1\}^n \to \{0,1\}^n$ be a standard pseudo-random function family with αn-bit keys and n-bit inputs and outputs, and denote $f_s(\cdot) = f(s,\cdot)$. Denote $\mathsf{prfix}(s)$ the first αn bits of s, and we define

$$F_s^{(g)}(x) \stackrel{\text{def}}{=} \begin{cases} f_{\mathsf{prfix}(s)}(1|x) & \text{if } x \neq g(s) \\ f_{\mathsf{prfix}(s)}(0) & \text{if } x = g(s) \end{cases} \tag{5}$$

We show that if the distribution on $g(s)$ for a random s has min-entropy more than $\alpha n + \omega(\log n)$ then the function $F_s^{(g)}(x)$ from above is KDI-secure with respect to g. Roughly, the reason is that since $\mathsf{prfix}(s)$ has only αn bits then $g(s)$ still has min-entropy $\omega(\log n)$ even conditioned on $\mathsf{prfix}(s)$. As the view of A when interacting with $F_s^{(g)}$ depends on s only via $\mathsf{prfix}(s)$, then $g(s)$ is unpredictable to A, and therefore so $F_s^{(g)}$ should be secure.

Formally we show how to transform a KDI attacker $\mathcal{A}^{(g)}$ against $F^{(g)}$ into a standard PRF attacker A against the underlying PRF family f, where both attackers have the same distinguishing advantage except for a negligible difference.

Attacker A interacts with a "box" (oracle) ϕ that is implemented via either a random function or a function from the family f_s. It invokes $\mathcal{A}^{(g)}$, and since $\mathcal{A}^{(g)}$ expect as input the value $F_s^{(g)}(g(s))$ then A gives it the input value $\phi(0)$. When $\mathcal{A}^{(g)}$ queries a value x_i, then A responds with $\phi(1|x_i)$. When $\mathcal{A}^{(g)}$ halts then A halts as well with the same output bit. It is clear that in both the "real" and "random" cases, the view of $\mathcal{A}^{(g)}$ during an attack on $F^{(g)}$ (as per Definition 1) is identical to its view when interacting with A, unless one of the x_i's that it queries happens to be equal to $g(s)$. (In the "Real" case this is true since $s' = \mathsf{prfix}(s)$ is random when s is). We now show that in both the "real" and "random" cases, this event only occurs with a negligible probability.

"Random" case. In this case, as long as $\mathcal{A}^{(g)}$ does not query $x_i = g(s)$, the answers that it gets are only random nit-strings, and they do not carry any information on the value s (or on $g(s)$). Hence the probability that $\mathcal{A}^{(g)}$ will query any $x_i = g(s)$ is at most as the most likely value of $g(s)$ times the number of queries that $\mathcal{A}^{(g)}$ makes. By assumption, $\mathcal{A}^{(g)}$ makes only polynomially many queries, and the min-entropy of $g(s)$ is at least $\alpha n + \omega(\log n)$, so the probability of having any $x_i = g(s)$ is at most $poly(n) \cdot 2^{-\alpha n + \omega(\log n)} = \mathsf{negl}(n)$.

"Real" case. In this case, as long as $\mathcal{A}^{(g)}$ did not query $x_i = g(s)$ explicitly, all responses depend on s only via $s' = \mathsf{prfix}(s)$. Therefore, for every query i the probability of $x_i = g(s)$ is no more than the most likely value of $g(s)$ *given* $s' = \mathsf{prfix}(s)$. But since the min-entropy of $g(s)$ is $\alpha n + \omega(\log n)$ and $s' = \mathsf{prfix}(s)$ only has αn bits, then the probability of this most-likely input can be bounded by

$$\begin{aligned}
&\sum_{s'} \Pr_s[\mathsf{prfix}(s) = s'] \cdot \max_y \left\{ \Pr_s \left[g(s) = y \mid \mathsf{prfix}(s) = s' \right] \right\} \\
&\leq \sum_{s'} \Pr_s[\mathsf{prfix}(s) = s'] \cdot \max_y \left\{ \frac{\Pr_s[g(s) = y]}{\Pr_s[\mathsf{prfix}(s) = s']} \right\} \\
&= \sum_{s'} \max_y \left\{ \Pr_s[g(s) = y] \right\} \leq 2^{\alpha n} \cdot 2^{-(\alpha n + \omega(\log n))} = \mathsf{negl}(n)
\end{aligned}$$

Hence in this case too the probability of getting any query with $x_i = g(s)$ is negligible. ■

***Randomized* extension to *any* function g.** The construction from Eq. (5) can be extended (via randomization!) to deal also with "low entropy" functions g by replacing $\mathsf{prfix}(s)$ with a randomness extractor $\mathsf{ext}_r(s)$, i.e.,

$$F_s^{(g,r)}(x) = \begin{cases} f_{ext_r(s)}(1|x) & \text{if } x \neq g(s) \\ f_{ext_r(s)}(0) & \text{if } x = g(s) \end{cases} \tag{6}$$

where f accepts $(n+1)$-bit inputs and short keys (if needed, such keys can be expanded into longer ones using a pseudorandom generator applied to $\mathsf{ext}_r(s)$).

As in the case of the scheme from Eq. (3), the intuition is that if g is "low entropy" then $\mathsf{ext}_r(s)$ is close to uniform even given $g(s)$, and for "high-entropy" $g(s)$ the reasoning from the proof above still holds when replacing $\mathsf{prfix}(s)$ with $\mathsf{ext}_r(s)$. These arguments can be extended to arbitrary functions g as stated in Lemma 3 below. Note that the resulting scheme is both randomized and "tailored" for each g, and hence inferior to the construction in Section 4.2. On the other hand, scheme (6) preserves invertibility, namely, if f is a family of invertible pseudorandom permutations (over $n+1$ bits) then $F^{(g,r)}$ is a KDI-secure invertible pseudorandom family mapping n bits into $n+1$ bits. This fact is used in an essential way in Section 6 to claim KDI-secure encryption (cf. Theorem 4).

Lemma 3 *If the family f is a secure PRF and* ext *is a strong randomness extractor, then for every (efficiently computable) function g, the family $F^{(g,r)}$ from Eq. (6) is a KDI-secure PRF with respect to the singleton class $\{g\}$, with high probability over the choice of the salt r.*

4.4 KDI security relative to the identity function

In the case of the identity function, $g(s) = s$, one can use the simple scheme from Eq. (4). Here we describe a construction that is more involved but somewhat more "natural" (or "less tailored"), which is KDI-secure with respect to the identity function.[6] Assume that we have a pseudo-random generator $G : \{0,1\}^n \to \{0,1\}^{2m}$, and denote the first m bits of output of G on seed s by $G_1(s)$ and the last m bits by $G_2(s)$. Assume further that G has the extra property that it is hard to find two different seeds s_1, s_2 such that $G_2(s_1) = G_2(s_2)$.

We call a generator with this extra property a ***collision-resistant generator***. For example, one can verify that such a pseudo-random generator (with $m = n$) can be constructed from any one-way permutation ρ over m bits using the Blum-Micali construction [6]:

$$G(s) = b(s)\ b(\rho(s))\ b(\rho^2(s))\ \ldots\ b(\rho^{m-1}(s)) \mid \rho^m(s) \tag{7}$$

where $b(\cdot)$ is a hard-core predicate for $\rho(\cdot)$. It is also plausible that ad-hoc constructions such as $G(s) = AES_s(0)|AES_s(1)|AES_s(2)\cdots$ have this property, since we can make m sufficiently larger than n. Given a collision-resistant pseudo-random generator $G(\cdot)$ and a pseudo-random function $f_s(\cdot)$, we construct another pseudo-random function $F_s(\cdot)$ by setting

$$F_s(x) = f_{G_1(s)}\left(G_2(x)\right). \tag{8}$$

Lemma 4 *The construction F is KDI-secure with respect to the singleton class containing the identity function $\mathcal{C} = \{ID\}$, assuming that f is a pseudo-random function and G is a collision-resistant pseudo-random generator.*

[6] This construction is inspired by an unpublished "symmetric PRF" construction due to Barak [2]. (A "symmetric PRF" is a function $f(x,y)$, which is a PRF both when x is viewed as the key and y as the input, as well as when y is viewed as the key and x as the input.)

Proof Sketch We first note that because of the collision-resistance of G it is unlikely that the attacker will ever query F on two inputs x, x' such that $G_2(x) = G_2(x')$. Assuming that this does not happen, we consider a "hybrid game" where the attacker's functional query (i.e., its query of $F_s(s)$) is answered by $f_{G_1(s)}(r)$ for a random r instead of by $f_{G_1(s)}(G_2(s))$. Then we show that the advantage of the attacker in the hybrid game is close to its advantage in the real game (since G is a PRG), and at the same time this advantage must be negligible (since f is a PRF). ■

We remark that although the identity function does not appear to be "hard wired" in the definition of F from above, this construction is not secure in general against any other function. For example, consider the function $g(s) = s + 1$, and we show a PRF f and a collision-resistant PRG G such that the resulting F is not KDI-secure with respect to $\mathcal{C} = \{g\}$.

Assume that we have a PRG $G(s) = G_1(s)|G_2(s)$ which is collision resistant with respect to both G_1 and G_2. (Again, it is plausible that an ad-hoc construction such as $G(s) = AES_s(0)|AES_s(1)|AES_s(2)\cdots$ has this property, since we can have $|G_1(s)| = |G_2(s)| \gg |s|$.) We then define a new PRG

$$\tilde{G}(s|b) = \tilde{G}_1(s|b) \mid \tilde{G}_2(s|b)$$

where

$$\begin{array}{l} \text{if } b = 0 \text{ then } \tilde{G}_1(s|b) = G_1(s),\ \tilde{G}_2(s|b) = G_2(s)|b \\ \text{if } b = 1 \text{ then } \tilde{G}_1(s|b) = G_2(s),\ \tilde{G}_2(s|b) = G_1(s)|b \end{array}$$

It is clear that $\tilde{G}$ is a collision-resistant PRG. Also, let f be a PRF with input which is one-bit longer than the key, and we modify it as follows:

$$f'_s(x|b) = \begin{cases} 0 & \text{if } x = s \\ f_s(x|b) & \text{otherwise} \end{cases}$$

Again, clearly f' is still a PRF. However the construction F from Eq. (8) is NOT KDI-secure with respect to $g(s) = s + 1$, since for any key whose last bit is zero, $s' = s|0$, we get

$$F_{s'}(s' + 1) = F_{s|0}(s|1) = f'_{\tilde{G}_1(s|0)}(\tilde{G}_2(s|1)) = f'_{G_1(s)}(G_1(s)|1) = 0$$

5 Tweakable Pseudo-random Permutations

Recall that a *tweakable* cipher (or tweakable pseudo-random permutation) [18] has a key and two inputs: a plaintext and a tweak. Below we use the convention that for security parameter n, both the plaintext and the tweak are n-bit strings, and that the cipher key is of length $\{0,1\}^{\ell(n)}$ where ℓ is some polynomially bounded function (we often use $\ell(n) = n$ or $\ell(n) = 2n$). Formally a family of tweakable pseudo-random permutations is an ensemble

$$\mathcal{P} = \left\{ p_{s,t} \in \mathcal{S}\left(\{0,1\}^n\right) \;\middle|\; s \in \{0,1\}^{\ell(n)}, t \in \{0,1\}^n \right\}_{n \in \mathcal{N}}$$

where $\mathcal{S}$ denotes the symmetric group, and we require that there are efficient evaluation and inversion procedures that given any $s, t \in \{0,1\}^n$ and any $x \in \{0,1\}^{\ell(n)}$ compute $y = p_{s,t}(x)$ and $z = p^{-1}_{s,t}(x)$, respectively.

The standard security definition for strong tweakable pseudo-random permutations as defined in [19, 18] asserts that no feasible attacker $\mathcal{A}^{\pi,\pi'}(1^n)$ (with oracle access to π, π^{-1}) can distinguish with any non-negligible advantage the case where for a random $s \in_R \{0,1\}^n$ we set $\pi(t,x) \equiv p_{s,t}(x)$ and $\pi^{-1}(t,x) \equiv p_{s,t}^{-1}(x)$, from the case where for every $t \in \{0,1\}^n$, $\pi(t,\cdot)$ is chosen as a random permutation over $\{0,1\}^{\ell(n)}$ and $\pi^{-1}(t,\cdot)$ is set to the inverse permutation.

Adding KDI-security to this definition requires some choices. The attacker in this model has two oracles, each with two inputs (namely $\pi(\cdot,\cdot)$ and $\pi^{-1}(\cdot,\cdot)$) and we need to decide what input to what oracle can depend on the key. In this work we only consider the variant where the plaintext/ciphertext inputs to both π and π^{-1} can depend on the key, but not the tweaks. The reason that we do not consider key-dependent tweaks is that tweaks typically represent some context information or label that comes from a higher layer (e.g., in the storage application that motivated this paper the tweak represents the physical position where the data is to be stored), and so it may be reasonable to assume that it does not depend on the key.

With these choices, we modify the standard definition of tweakable PRPs by giving the adversary access to two additional oracles ψ, ψ^{-1} that take as input a tweak t and a description of a function g and output $\pi(t, g(s))$ and $\pi^{-1}(t, g(s))$, respectively.

Definition 2 (KDI-secure tweakable strong PRPs) *A family $\mathcal{P}$ of tweakable pseudo-random permutations is* KDI-secure *with respect to a class $\mathcal{C}$ of circuits if no feasible attacker $\mathcal{A}^{\pi,\pi^{-1},\psi,\psi^{-1}}(1^n)$ can distinguish with non-negligible advantage between the following two cases:*

1. *The key $s \in_R \{0,1\}^n$ is chosen at random, and for any t, x, g the oracles are set as, $\pi(t,x) \equiv p_{s,t}(x)$, $\pi^{-1}(t,x) \equiv p_{s,t}^{-1}(x)$, $\psi(t,g) \equiv p_{s,t}(g(s))$, and $\psi^{-1}(t,g) \equiv p_{s,t}^{-1}(g(s))$;*
2. *The key $s \in_R \{0,1\}^n$ is chosen at random, for every $t \in \{0,1\}^n$ we set $\pi(t,\cdot)$ to a random permutation over $\{0,1\}^\ell$ and $\pi^{-1}(t,\cdot)$ to its inverse, and then $\psi(t,g) \equiv \pi(t,g(s))$, and $\psi^{-1}(t,g) \equiv \pi^{-1}(t,g(s))$.*

As before, Definition 2 is adapted to the "ideal cipher model" by giving oracle access to the ideal cipher Π, Π^{-1} to the construction itself, the attacker $\mathcal{A}$, and potentially also the circuits in $\mathcal{C}$.

Below we demonstrate that some constructions of tweakable ciphers in the literature are KDI insecure against simple functions of the key, while others can be proven secure in the ideal cipher model.

KDI-insecurity of the LRW constructions. Consider the following instantiation of the second construction of Liskov et al. from [18]. This instantiation has two keys, denoted s_1, s_2, where s_1 is used as a key for an underlying block cipher E and s_2 is treated as an element of $GF(2^n)$ with n the block-size of E. This construction then defines the following tweakable cipher, with both the block size and the tweak size equals to n bits:

$$\tilde{E}_{s_1,s_2}(t,x) = E_{s_1}((t \cdot s_2) \oplus x) \oplus (t \cdot s_2) \tag{9}$$

where $t \cdot s_2$ is a multiplication in $GF(2^n)$). In [18] it is shown that the generic construction $E_{s_1}(h_{s_2}(t) \oplus x) \oplus h_{s_2}(t)$ is a secure tweakable cipher when E is a secure cipher and h is a "xor-universal" hash function, which implies the security of Eq. (9) since $h_{s_2}(t) = t \cdot s_2$ is indeed xor-universal.

However, as pointed out when this construction was considered for the IEEE 1619 standard, this construction is not KDI-secure with respect to the function $g(s_1, s_2) = s_2$ (i.e., when "encrypting" the element s_2 from the secret key). The attacker can query $\psi(0, g)$ (i.e., using tweak value 0 and "plaintext" s_2) and also $\pi(1, 0)$ (i.e., tweak value 1 and "plaintext" 0), thus getting

$$c_1 = \tilde{E}_{s_1,s_2}(0, s_2) = E_{s_1}(s_2) \quad \text{and} \quad c_2 = \tilde{E}_{s_1,s_2}(1, 0) = E_{s_1}(s_2) \oplus s_2$$

Next the attacker can compute $s_2 = c_1 \oplus c_2$ and then verify this value (e.g., by asking to "decrypt" the value of $c_1 \oplus 2s_2$ with respect to the tweak value 2).

Negative Example 3 *The construction from Eq. (9) is not KDI-secure with respect to the function* $g(s_1, s_2) = s_2$. ■

KDI-security of the "trivial construction". Alternatively, consider the "trivial" construction of tweakable SPRP from a block cipher

$$\tilde{E}_s(t, x) = E_{E_s(t)}(x) \tag{10}$$

It is easy to see that if E is a secure cipher then this construction is a secure tweakable cipher. Although there are functions g for which this construction is not KDI-secure (for example the function $g(s) = E^{-1}_{E_s(t)}(0)$, we can show, however, that $\tilde{E}$ from Eq. (10) is KDI-secure in the "ideal cipher model" with respect to any function that *does not depend on the cipher itself.*

Lemma 5 *Let* g *be any Boolean circuit* with no Π-gates or Π^{-1}-gates. *Then the construction* $\tilde{E}$ *from Eq. (10) is a KDI-secure tweakable strong pseudo-random permutation in the "ideal cipher model" with respect to the singleton class* $\mathcal{C} = \{g\}$.

Proof Sketch Again, the proof is straightforward. Recall that the adversary in this game has six oracles: $\Pi(\cdot,\cdot)$ and $\Pi^{-1}(\cdot,\cdot)$ that represent the ideal cipher, $\tilde{E}(\cdot,\cdot)$ and $\tilde{E}^{-1}(\cdot,\cdot)$ that represent either the construction with a fixed random key s or a random tweakable permutation (independent of Π), and $\psi(\cdot)$ and $\psi^{-1}(\cdot)$ that allow key-dependent queries[7] with $\psi(t) = \tilde{E}(t, g(s))$ and $\psi^{-1}(t) = \tilde{E}^{-1}(t, g(s))$ (where s is the secret key in case one and just a random string in case two).

Similarly to the proof of Theorem 1, the proof goes by arguing that the attacker is very unlikely to ever query its Π or Π^{-1} oracles on the right "key" s, and without such queries the view of the attacker is the same in both cases. ■

Other constructions. We comment that a similar lemma can be proven also for Rogaway's XEX construction from [23], where on input x and tweak (i, j) one computes:

$$\tilde{E}_s((i, j), x) = E_s((2^i \cdot E_s(j)) \oplus x) \oplus (2^i \cdot E_s(j)) \tag{11}$$

[7]Compared to Definition 2 we slightly simplify notations here by having ψ, ψ^{-1} as single-input oracles. We can do this because the function g is always the same, since we are interested in the singleton class $\mathcal{C} = \{g\}$.

Namely, this construction too can be proven KDI-secure in the "ideal cipher model" with respect to any function g that does not depend on the ideal cipher. The proof itself is very similar to Rogaway's proof of security for XEX [23]. The key-dependent queries are handled using the fact that in the "ideal cipher model" the quantity $E_s(j)$ is independent of s for all j, and therefore the attacker is unlikely to be able to issue two queries for which $(2^i \cdot E_s(j)) \oplus x = (2^{i'} \cdot E_s(j')) \oplus x'$ (even if x, x' can be set as functions of the secret key s).

6 Symmetric Encryption

Being randomized, encryption schemes are potentially easier to make KDI secure than non-randomized primitives such as PRFs and ciphers. Black, Rogaway, and Shrimpton studied in [5] the question of KDI-security for symmetric encryption (under the name KDM-security). They presented a definition of security (using much of the same rationale as in Section 2), and proved that it can be easily met in the random-oracle model. However, we do not know of a construction that achieves similar level of KDI security in the standard model (not even with respect to all singletons $\{g\}$). Yet, we provide two significant results in this section: The first shows that a natural "textbook" PRF-based encryption scheme that is KDI-secure in the random oracle model (with respect to all functions g) is *not* KDI-secure in the standard model, not even with respect to the identity function, not even when the PRF itself is KDI secure, and not even with respect to "practical instantiations" of the pseudorandom function. The second result shows that for every function g we can build an encryption scheme that is KDI-secure with respect to $\{g\}$ (based on any invertible pseudorandom function, such as any block cipher).

6.1 Definitions

Recall that a (symmetric) encryption scheme consists of algorithms for key-generation, encryption and decryption, $\mathcal{E} = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, with Gen, Enc randomized and where both Enc and Dec use the secret key that is generated by Gen. Below we assume that for security parameter n, the messages that are encrypted are of length n. The standard definitions of CPA-security for symmetric encryption (cf. [11, 16]) asserts that no feasible attacker with oracle access to the encryption algorithm (with the secret key hard-wired) can produce two equal-length messages m, m' for which it can distinguish with non-negligible advantage a random encryption of m from a random encryption of m'. The standard definition for CCA-security is similar (cf. [22, 16]) except that the attacker is also given access to a decryption oracle, but is not allowed to query that oracle on the ciphertext for which it needs to decide if it came from m or m'.

As usual, we incorporate KDI-security by providing another oracle to the attacker that on input a function g outputs the encryption of $g(s)$ under the secret key s.[8]

Definition 3 (KDI-secure encryption) *Let $\mathcal{C}$ be a class of functions (circuits). A symmetric encryption scheme $\mathcal{E} = (Gen, Enc, Dec)$ is* CPA-KDI-secure *with respect to $\mathcal{C}$ if no feasible attacker $\mathcal{A}^{e,e'}(1^n)$ has non-negligible advantage in guessing the value of the bit b in the following game:*

[8] The output size of the functions g considered here is assumed to match the size of inputs to the encryption function Enc.

1. *The key-generation algorithm is run to get* $s \leftarrow Gen(1^n)$.

2. *The attacker* $\mathcal{A}^{e,e'}(1^n)$ *is given the security parameter and access to two oracles* $e(\cdot)$, $e'(\cdot)$, *which are defined by* $e(m) \equiv Enc(s;m)$ *for* $m \in \{0,1\}^n$, *and* $e'(g) \equiv Enc(s;g(s))$ *for* $g \in \mathcal{C}$.

   The attacker $\mathcal{A}$ *interacts with these oracles and then outputs two queries* q_1, q_2 *for these oracles (each* q_i *can be a query to either* $e(\cdot)$ *or* $e'(\cdot)$*).*

3. *A bit* $b \in_R \{0,1\}$ *is chosen at random and the attacker is given* c^*, *the ciphertext resulting from encrypting* q_b *under the corresponding oracle. The attacker* $\mathcal{A}$ *can continue interacting with its oracles, and then it outputs a guess* b' *for the bit* b.

We similarly say that $\mathcal{E}$ *is* CCA-KDI-secure *with respect to* $\mathcal{C}$ *if no no feasible attacker has non-negligible advantage in guessing the value of the bit* b *in the modified game where* $\mathcal{A}^{e,e',d}(1^n)$ *is also given access to an oracle* $d(c) \equiv Dec(s;c)$, *but cannot query the oracle* $d(\cdot)$ *on the challenge ciphertext* c^* *produced in step 3 or in any output from oracle* e'.

It is important to observe that we do not allow key-dependent queries to the decryption oracle. In particular, note that if we allowed such queries then one could query the decryption oracle with the function $g(s) = \mathrm{Enc}_s(s)$ and obtain s (this is possible even if one restricts the functions in $\mathcal{C}$ to be deterministic, as we could define the function $g(s)$ to be the encryption of s with the random coins set to a fixed string, say all zeros). Finally, note that Definition 3 can be adapted to the "ideal cipher model" by providing $\mathcal{A}$ (and potentially also the circuits in $\mathcal{C}$) with oracle access to Π and Π^{-1}.

6.2 Insecurity of a textbook randomized scheme

Below we consider a very natural PRF-based symmetric encryption scheme, which is essentially the scheme that was proven secure in the random-oracle model by Black et al. We show that not only this construction fails to be KDI secure in the standard model, but *this failure is manifested even for a natural instantiations of the PRF*. Given a PRF $f_s(\cdot)$ we define

$$\mathrm{Enc}_s(x) = (r, f_s(r) \oplus x) \tag{12}$$

where r is chosen at random with each encryption. This encryption scheme is CPA-secure (up to the birthday bound on $|r|$) if $f_s(\cdot)$ is a secure PRF, and, intuitively, it appears that it "should" also be KDI-secure. In particular, it was shown in [5, Thm 5.1] that when f_s is implemented as $f_s(x) = H(s|x)$ and H is a random oracle, then the scheme in Eq. (12) is KDI-secure with respect to *all functions of the key* s.

We demonstrate, however, that this construction is not KDI-secure in general, and perhaps more surprising it even fails for practical PRFs. Specifically, we show that when the underlying PRF is implemented from a block cipher via the Davies-Meyer construction, the resulting encryption scheme is not KDI-secure, even with respect to the identity function *(and even if the block cipher itself is an ideal cipher!).* Recall the Davies-Meyer construction

$$f_s(x) = E_x(s) \oplus s \tag{13}$$

This construction was meant as a component of a collision-resistant hash function, but for contemporary block ciphers one can expect it to also be a good PRF. (For example, the assumption that the Davies-Meyer compression function keyed via its IV is a PRF was used in the analysis of HMAC.) Moreover, it is easy to prove that when E is an ideal cipher, then *the Davies-Meyer construction is a KDI-secure PRF* with respect to any function g that does not depend on the ideal cipher itself

Plugging the Davies-Meyer construction in Eq. (12) we obtain the encryption scheme $\text{Enc}_s(x) = (r, (E_r(s) \oplus s) \oplus x)$. An attacker that asks to encrypt the secret key will get $\text{Enc}_s(s) = (r, (E_r(s) \oplus s) \oplus s) = (r, E_r(s))$, from which it can recover s (using the decryption routine E^{-1} with r as a key). We note that this construction fails even if E is an ideal cipher (and also when augmented with a MAC function to provide CCA security).

Negative Example 4 *The symmetric encryption scheme $Enc_s(x) = (r, x \oplus f_s(r))$, with f_s implemented via the Davies-Meyer construction $f_s(x) = E_x(s) \oplus s$, is KDI-insecure with respect to the identity function, even when E is an ideal cipher.* ■

6.3 KDI-secure symmetric encryption

On the positive side, we show that an *invertible* KDI-secure PRF can be used to obtain KDI-secure symmetric encryption, using a slight variation of the insecure "textbook construction" from above. (We call a PRF f_s invertible if given s and $f_s(x)$ one can efficiently recover x.) Specifically, consider the encryption scheme

$$\text{Enc}_s(x) = (r, f_s(r) \oplus f_s(x)), \quad \text{Dec}_s(r, y) = f_s^{-1}(y \oplus f_s(r)) \tag{14}$$

Lemma 6 *If the family f_s has input size $\omega(\log n)$ (with n the security parameter), and if it is a KDI-secure (invertible) PRF with respect to a class $\mathcal{C}$, then the encryption scheme from Eq. (14) is CPA-KDI secure with respect to the same class $\mathcal{C}$.*

Proof Fix a class $\mathcal{C}$ and assume toward contradiction that the encryption scheme from Eq. (14) is not CPA-KDI secure with respect to $\mathcal{C}$, and we show that the underlying family f_s cannot be KDI-secure PRF with respect to $\mathcal{C}$. Let A_E be an attacker that demonstrates the KDI-insecurity of the encryption scheme, and we use it to describe an attacker A_f that demonstrates the KDI-insecurity of the PRF family.

A_f is provided with oracle access to ϕ, ϕ', as described in Definition 1. It activates the attacker A_E (on the same security parameter). When A_E asks to encrypt an explicit string x then A_f chooses a random string r, queries $s = \phi(r)$ and $t = \phi(x)$ and returns the ciphertext $(r, s \oplus t)$ to A_E. When A_E asks to encrypt $g(s)$ for some $g \in \mathcal{C}$ then A_f chooses again a random string r queries $s = \phi(r)$ and $t = \phi'(g)$ $(= \phi(g(s)))$, and returns the ciphertext $(r, s \oplus t)$ to A_E. When A_E outputs q_0, q_1 then A_f chooses a random bit b and returns an encryption of q_b using the same procedure as above. Finally A_f outputs whatever A_E does.

It follows by definition that when $\phi = f_s$ then the view of A_E is identical to its view when interacting with the real encryption scheme. On the other hand, when ϕ is a random function then the view of A_E is nearly independent of q_0, q_1 (except for the case where the random string r that A_f chooses for the encryption of q_b collides with an earlier input to ϕ or ϕ', which happens with

negligible probability if $|r|$ is large enough). Hence the advantage of A_f is negligibly close to half the advantage of A_E. ■

Lemma 6 tells us that we can get KDI-secure encryption from invertible KDI-secure PRFs, so it is natural to ask if the KDI-secure PRF schemes studied in this paper can be made invertible. Clearly, the trivial construction $f_s(x) = E_S(x)$ from Section 3.1 is invertible, and it is KDI-secure in the ideal cipher model (but not in the standard model). On the other hand, the construction from Eq. (3) in Section 4.2, i.e., $F_s(x) = f_{\mathsf{ext}'(s)}(H_{\mathsf{ext}(s)}(x))$, seems inherently non-invertible; in fact the security proof for it relies in an essential manner on $H(x)$ having many fewer bits than x itself. Fortunately, the g-dependent construction (6) from Section 4.3 is invertible if the underlying PRF f is invertible (e.g., when f is a block cipher) and it is KDI secure wrt $\{g\}$ in the standard model. Therefore, if we instantiate the encryption scheme defined in Eq. (14) with the scheme $F^{(g,r)}$ of Eq. (6) (and assuming the function f underlying $F^{(g,r)}$ is invertible) we obtain an encryption scheme $Enc^{(g)}$ that is, according to Lemma 6, CPA-KDI-secure. Note that the random salt r used in $F^{(g,r)}$ (which is in addition to, and independent from, the randomness r in Eq. (14)) can be chosen by the encryptor with each encryption or can be chosen at random and be fixed as a parameter of $Enc^{(g)}$. From this discussion and Lemma 3 we obtain the following result.

Theorem 4 *If secure PRF families exist then for every efficiently computable function g there is a symmetric encryption scheme that depends on g and is CPA-KDI-secure with respect to the singleton class $\{g\}$.*

References

[1] P. Adão, G. Bana, J. Herzog, and A. Scedrov. Soundness of Formal Encryption in the Presence of Key-Cycles. In *10th European Symposium on Research in Computer Security - ESORICS 2005*, volume 3679 of *Lecture Notes in Computer Science*, pages 374–396. Springer, 2005.

[2] B. Barak. Symmetric PRFs. Personal communications, 2001.

[3] M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In *Advances in Cryptology – EUROCRYPT '03*, volume 2656 of *LNCS*, pages 491–506. Springer, 2003.

[4] M. Bellare, T. Krovetz, and P. Rogaway. Luby-rackoff backwards: Increasing security by making block ciphers non-invertible. In *Advances in Cryptology - EUROCRYPT'87*, volume 1403 of *Lecture Notes in Computer Science*, pages 266–280. Springer, 1998.

[5] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 62–75. Springer, 2002.

[6] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[7] J. Camenisch and A. Lysyanskaya, An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *Advances in Cryptology - EUROCRYPT'01*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118 Springer, 2001.

[8] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003. Preliminary version in Crypto'98.

[9] Y. Dodis, A. Sahai, and A. Smith. On Perfect and Adaptive Security in Exposure-Resilient Cryptography. In *Advances in Cryptology - EUROCRYPT'01*, volume 2045 of *Lecture Notes in Computer Science*, pages 301–324. Springer, 2001.

[10] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):210–217, 1986.

[11] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[12] C. Hall, D. Wagner, J. Kelsey, and B. Schneier. Building PRFs from PRPs. In *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 370–389. Springer, 1998.

[13] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[14] IEEE P1619.* email archive. `http://grouper.ieee.org/groups/1619/email/`.

[15] IEEE P1619. Standard for cryptographic protection of data on block-oriented storage devices. Draft standard, available temporarily from `http://ieee-p1619.wetpaint.com/page/IEEE+Project+1619+Home`, 2007.

[16] J. Katz and M. Yung. Characterization of security notions for probabilistic private-key encryption. *Journal of Cryptology*, 19(1):67–95, January 2006. Earlier version in STOC'00, pages 245-254.

[17] P. Laud and R. Corin. Sound Computational Interpretation of Formal Encryption with Composed Keys. In *6th International Conference on Information Security and Cryptology - ICISC 2003*, volume 2971 of *Lecture Notes in Computer Science*, pages 55–66. Springer, 2003.

[18] M. Liskov, R. L. Rivest, and D. Wagner. Tweakable block ciphers. In *Advances in Cryptology – CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2002.

[19] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal of Computing*, 17(2), Apr. 1988.

[20] S. Lucks. The sum of PRPs is a secure PRF. In *Advances in Cryptology - EUROCRYPT'00*, volume 1807, pages 470–484. Springer, 2000.

[21] N. Nisan and D. Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[22] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1991.

[23] P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In *Advances in Cryptology - ASIACRYPT'04*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.

[24] L. Trevisan and S. P. Vadhan. Extracting Randomness from Sampleable Distributions. In *41st Symposium on Foundations of Computer Science - FOCS'00*, pages 32–42. IEEE Computer Society, 2000.

A Randomness extractors

Min-entropy and statistical distance. We use the terms *probability distributions* and *random variables* pretty much interchangeably. For a discrete random variable X, the min-entropy of X is related to the most probable value that X can assume, specifically min-entropy$(X) = -\log_2(\max_x \Pr[X = x])$. In other words, if min-entropy$(X) = \ell$ then X does not assume any value with probability of more than $2^{-\ell}$. In this case we also say that "X has ℓ bits of min-entropy." The statistical distance between two random variables X, Y is defined as

$$\mathsf{dist}(X,Y) \stackrel{\text{def}}{=} \sum_x |\Pr[X = x] - \Pr[Y = x]|$$

where the summation is taken over the union of the supports of both X and Y. If $\mathsf{dist}(X,Y) = \epsilon$ then we say that X is ϵ-close to Y.

Randomness extractors. A (strong) randomness extractor [21] is a function $\mathsf{ext}_r(x)$ that takes a random seed r and a "somewhat random" input x, and produces an output y (shorter than x) which should be almost uniform, even given r. Specifically, a function $\mathsf{ext} : \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^m$ is a strong (k,ϵ) extractor if for any distribution $\mathcal{D}$ over $\{0,1\}^n$ with more than k bits of min-entropy, the statistical distance between the two distributions

$$\left\{\langle r, y\rangle : r \stackrel{\$}{\leftarrow} \{0,1\}^t,\ x \stackrel{\$}{\leftarrow} \mathcal{D},\ y \leftarrow \mathsf{ext}_r(x)\right\} \text{ and } \left\{\langle r, y\rangle : r \stackrel{\$}{\leftarrow} \{0,1\}^t,\ y \stackrel{\$}{\leftarrow} \{0,1\}^m\right\}$$

is at most ϵ. For any $k > m$, there are known constructions (e.g., using the leftover hash lemma [13]) that are strong (k,ϵ) extractors with $\epsilon = 2^{-(k-m)/2}$.

We will also need a slightly stronger notion of a (k,ϵ,δ) extractor, where with probability $1-\delta$ over the seed r, the distribution $\mathsf{ext}_r(\mathcal{D})$ is ϵ-close to uniform. An extension of the leftover hash lemma [24, 9] says that we can get (k,ϵ,δ) extractors with $\delta = 2^{-\ell}$ and $\epsilon = 2^{-(k-m-O(\log \ell))/2}$ using $O(\ell)$-wise independent hash functions.