On Related-Secret Pseudorandomness

David Goldenberg[1] and Moses Liskov[1]

The College of William and Mary {dcgold, mliskov}@cs.wm.edu *

Abstract. Related-key attacks are attacks against constructions which use a secret key (such as a blockcipher) in which an attacker attempts to exploit known or chosen relationships among keys to circumvent security properties. Security against related-key attacks has been a subject of study in numerous recent cryptographic papers. However, most of these results are attacks on specific constructions, while there has been little positive progress on constructing related-key secure primitives.
In this paper, we attempt to address the question of whether related-key secure blockciphers can be built from traditional cryptographic primitives. We develop a theoretical framework of "related-secret secure" cryptographic primitives, a class of primitives which includes related-key secure blockciphers and PRFs. We show that while a single related-secret pseduorandom bit is sufficient and necessary to create related-key secure blockciphers, hard-core bits with typical proofs are *not* related-secret psuedorandom. Since the pseudorandomness of hard-core bits is the essential technique known to make pseudorandomness from assumptions of simple hardness, this presents a very strong barrier to the development of provably related-key secure blockciphers based on standard hardness assumptions.

1 Introduction

Related-key attacks are attacks against constructions using a secret key (such as a blockcipher) in which an attacker attempts to exploit known or chosen relationships among keys to circumvent security properties. Several related-key attacks on primitives have been developed [1–3], including attacks on AES [4–7]. While the realism of an adversary's ability to directly influence a secret key is questionable, the issue of related-key security has implications beyond such a setting. For instance, weakness in a blockcipher's key scheduling algorithm may result in known likely relationships amongst round keys, which could lead to an attack against the cipher [8]. As another example, blockcipher based hash functions are only proven secure in the ideal cipher model [9]; in this strong model, related-key security is implied [8]. Thus, the use of a real blockcipher for hashing that is not related-key secure is theoretically questionable: in many such constructions, the adversary's ability to choose the message to be hashed implies an ability to launch related-key attacks on the underlying cipher. Indeed, a recent paper by Biryukov et al has made substantial progress on attacking AES-256 in

* Supported by NSF grant CNS-0845662.

Davies-Meyer mode via a strong related-key attack on AES [7]. Finally, there are settings in which related-key security has been put to good use: several papers make use of schemes with one-time related-key security properties in order to make fuzzy extractors robust against adversarial modification [10–12].

Positive results concerning related-key security are few. Bellare and Kohno [8] develop a theoretical framework for defining related-key security, show that some notions of related-key security are inherently impossible, and prove that an ideal cipher is related-key secure for a general class of relations. Lucks [13] shows how to achieve "partial" related-key security (meaning, that only part of the key can be varied), and also gave two proposed constructions of related-key secure pseudorandomness from novel, very strong number theoretic assumptions.

Defining related-secret security. Bellare and Kohno define related-key security as follows. If $\mathsf{F}_K(x)$ is a pseudorandom function (or permutation) then it is related-key secure if an adversary cannot distinguish between (1) an oracle that, on input x and a perturbation δ, returns $\mathsf{F}_{\delta(K)}(x)$ and (2) an oracle that implements a random function (or permutation) on x independently for each distinct δ. In order to study the relationship between such strong primitives and simpler ones, we broaden the concept of related-key attacks to "related-secret" attacks, which extends Bellare and Kohno's notion of related-key security to allow adversarially-specified perturbations on any inputs unknown to the adversary, whether or not these inputs are considered "keys". Simpler related-secret secure primitives are good candidates for intermediate steps between basic primitives and related-key security. We give definitions for many different related-secret secure primitives such as related-secret secure one way functions/permutations, hardcore bits, pseudorandom generators, pseudorandom functions, and pseudorandom permutations.[1]

Our results. We attempt to mirror the development of pseudorandom permutations from one-way functions or permutations in the related-secret setting, without assuming (as Lucks does [13]) that any part of any secret cannot be modified by the adversary. Because of this strong requirement, in most cases, we expect that strong related-secret secure primitives will require simple related-secret secure building blocks.

Most steps in the strengthening of basic primitives into pseudorandom permutations can be translated to the related-secret setting. We show related-secret security for any homomorphic one-way function, for instance, modular exponentiation (under the discrete log assumption). The idea is that homomorphic perturbations can be calculated without knowing the secret. We also show that the critical step is to obtain related-secret pseudorandomness: we show that even a 1-bit related-secret pseudorandom generator is sufficient to build related-secret pseudorandom permutations. Moreover, we show that these "pseudorandom bits" are necessary for related-key blockciphers.

[1] Our definitions of RK-PRF's and RK-PRP's are the same as [8].

In the standard model, a hard-core bit fills this role simply: a hard-core bit is hard to learn, and thus, hard to distinguish from a random bit [14, 15]. However, in the related-secret setting things are different, because of the ability of a related-secret attacker to obtain multiple related values. So we distinguish between a related-secret hard-core bit (in which no bits are shown to the adversary, and one must be learned), and a pseudorandom bit (in which all the bits are shown to the adversary, who must distinguish them from random).

Here, we give a negative result: any "strong" hard-core bit (that is, a hard-core bit with a reduction from learning the bit to inverting the associated function) is *not* a related-secret pseudorandom bit. More specifically, we give a transformation from the reduction between the hard-core bit finder and the function inverter, to an attack against the same hard-core bit as a pseudorandom bit in the related-secret model. This transformation assumes that the reduction is a "black-token" reduction, in other words, that it simply manipulates the function output with known tools while being blind of the actual value of the function output. The notion of a black-token algorithm is akin to the concept of *algebraic reductions* and *generic ring algorithms* [16–19], except that we do not require the same level of algebraic structure.

This leads us to the conclusion that if related-secret pseudorandomness (including related-key blockciphers) are possible, they must be proven either based on other related-secret pseudorandomness assumptions[2], or a dramatically new way of creating pseudorandomness from hardness must be developed.

2 Definitions

If $f : \mathbb{N} \to \mathbb{R}$ is a function, we say that f is *negligible* if $\forall c$, $\exists n_0$ such that for all $n > n_0$, $f(n) < \frac{1}{n^c}$.

We use A to denote an adversary or algorithm. We denote the set of all probabilistic polynomial time adversaries as PPT. If an adversary A takes an oracle O we denote that as A^{O}.

We denote a value x randomly sampled from a set $\mathcal{X}$ as $x \leftarrow \mathcal{X}$. For a function f we denote $\mathcal{D}_f$ as the domain of f and $\mathcal{R}_f$ as the range. We denote the j'th bit of x as x_j and the i'th through the j'th bit of x as $x_{i,\cdots,j}$. For two inputs x, r we denote the inner product $(\sum_{i=0}^{k} x_i r_i)$ of x and r as $\langle x, r \rangle$. If x and r come from a metric space $\mathcal{M}$ we denote the distance between x and r as $||x - r||$.

Definition 1. *A (k, ρ, p) list decodable code is a triple of algorithms $\mathsf{C}, \mathsf{C}^{-1}, \mathsf{R}$ where $\mathsf{C} : \{0,1\}^k \to \mathcal{C} \subset \{0,1\}^n$, $\mathsf{C}^{-1} : \mathcal{C} \to \{0,1\}^k$ and R has the property that if $x \in \{0,1\}^k$ is the message, and y is the corrupted encoding of x, then $Pr[\mathcal{S} \leftarrow \mathsf{R}^{\mathsf{Y}} : \forall x \in \mathcal{S}, ||\mathsf{C}(x) - y|| \leq \rho n] \geq p$ as long as $||\mathsf{C}(x) - y|| \leq \rho n$ where Y is the oracle which on input i returns y_i. and where $|\mathcal{S}|$ is polynomial in k. We say a list decodable code allows for local decoding if R only queries Y a polynomial number of times.*

[2] Such as the related-key pseudorandom constructions of Lucks [13], based on novel assumptions effectively as strong as related-secret pseudorandomness

Definition 2. *A list decodable code* $\mathsf{C}, \mathsf{C}^{-1}, \mathsf{R}$ *is* linear *when* $\mathcal{C}$ *is a linear subspace of* $\{0,1\}^n$.

2.1 Related-secret security

We consider primitives to be *related-secret secure* if they maintain their security even under an adversary which can exploit known or chosen relationships between related secret inputs. We specify the notion of interacting with the primitives using secrets "related to" the secret input $x \in \mathcal{D}$ by allowing the adversary to interact with the primitive under $\delta(x)$, where δ is a perturbation (function) from $\mathcal{D} \to \mathcal{D}$, which is specified by the adversary. For example, a block-cipher is related-key secure if no distinguishing adversary exists, even when we allow the adversary to make queries in which a particular transformation of the secret key is specified. While we might hope to design related-secret primitives for any possible set of perturbations, it has been shown in [8] that related-secret attacks inherently exist when the set of perturbations is "invalid", defined as follows [8]:

Definition 3 (Valid sets). *We say a set of functions* $\Delta : \mathcal{D} \to \mathcal{D}$ *is valid if it satisfies the following two properties:*

- *Output unpredictable:* Δ *is output unpredictable if* $\forall S \subset \Delta$, $\forall X \subset \mathcal{D}$, $Pr[x \leftarrow \mathcal{D}; \{\delta(x) : \delta \in S\} \cap X \neq \emptyset]$ *is negligible as long as* $|X|$ *and* $|S|$ *are polynomial in* $\log|\mathcal{D}|$.
- *Collision resistant:* Δ *is collision resistant if* $\forall S \subset \Delta$, $Pr[x \leftarrow \mathcal{D}; |\{\delta(x) : \delta \in S\}| < |S|]$ *is negligible when* $|S|$ *is polynomial in* $\log|\mathcal{D}|$.

Due to the power of the attack in [8], it is impossible to design cryptographic primitives that remain secure under arbitrary perturbations. As such, we will assume that Δ, the set of "allowable" perturbations, is a valid set. We also require that Δ has a minimal level of additional structure: we assume it is closed under function composition and contains the identity perturbation $\delta_{ident} \in \Delta : \delta_{ident}(x) = x$.[3]

We also say that Δ is *complete* in that $\forall x, y \in \mathcal{D}_f$, $\exists \delta : \delta(x) = y$. We note that related-key or related-secret security against an *incomplete* Δ is a far easier problem [13].

Two standard examples of Δ classes that meet these criteria:

- $\Delta_+ = \{\delta_c : x \mapsto x + c\}$
- $\Delta_\oplus = \{\delta_c : x \mapsto x \oplus c\}$

In both cases we identify the function δ_c with the value c. This sort of perturbation class is the most relevant: in many published related-key attack results, perturbations are from $\Delta_\oplus$ (for example, [7]).

We start by stating the definitions of a related key secure pseudorandom permutation and a related key secure pseudorandom function in the above notation. The definitions are taken from [8].

[3] For any Δ not closed under composition, or not including the identity, we can expand it to its closure under composition, and add the identity.

Definition 4 (Related key secure pseudorandom permutation (RK-PRP)). *An efficiently computable function* $\mathsf{E} : \{0,1\}^{p(k)} \times \{0,1\}^k \to \{0,1\}^{p(k)}$ *where p is a polynomial is considered a related key secure pseudorandom permutation under* Δ *if* $\mathsf{E}(\cdot, K)$ *is a permutation for all* $K \in \{0,1\}^k$ *and if* $\forall \mathsf{A}$, $\exists \nu$ *negligible:* $\forall k \ |Pr[K \leftarrow \{0,1\}^k : \mathsf{A}^{\mathsf{F}_{\mathsf{E}(\cdot,K),K}} = 1] - Pr[K \leftarrow \mathcal{K} : \mathsf{A}^{\mathsf{F}_{\mathcal{P}(\cdot,K),K}} = 1]| \leq \nu(k)$ *where k is the security parameter,* $\mathcal{P} : \{0,1\}^{p(k)} \times \{0,1\}^k \to \{0,1\}^{p(k)}$ *is a family of random permutations indexed by its second parameter, and where* $\mathsf{F}_{f(\cdot,K),K}$ *is the oracle which on input* $x, \delta \in \Delta$ *returns* $f(x, \delta(K))$.

Definition 5 (Related key secure pseudorandom function (RK-PRF)). *An efficiently computable function* $\mathsf{R} : \{0,1\}^{p(k)} \times \{0,1\}^k \to \{0,1\}^{p'(k)}$, *where p and* p' *are polynomials, is considered to be a related-key secure pseudorandom function under* Δ *if* $\forall \mathsf{A}$, $\exists \nu$ *negligible:* $\forall k \ |Pr[K \leftarrow \{0,1\}^k : \mathsf{A}^{\mathsf{F}_{\mathsf{R}(\cdot,K),K}} = 1] - Pr[K \leftarrow \{0,1\}^k : \mathsf{A}^{\mathsf{F}_{\mathcal{F}(\cdot,K),K}} = 1]| \leq \nu(k)$ *where k is the security parameter and where* $\mathcal{F} : \{0,1\}^{p(k)} \times \{0,1\}^k \to \{0,1\}^{p'(k)}$ *is a family of random functions indexed by its second parameter, and where* $\mathsf{F}_{f(\cdot,K),K}$ *is the oracle which on input* $x, \delta \in \Delta$ *returns* $f(x, \delta(K))$.

We can extend the notion of an RK-PRP / PRF to the notion of a related secret pseudorandom generator (RS-PRG). Like an RK-PRP or RK-PRF, an RS-PRG generates pseudorandomness even under an adversary who can affect the "secret" input. However, an RS-PRG only has one input, the secret seed to the generator.

Definition 6 (Related-secret secure pseudorandom generator (RS-PRG)). *Let* $\mathsf{F}_{f,x}$ *be the oracle which on input* $\delta \in \Delta$ *returns* $f(\delta(x))$. *An efficiently computable function* $g(x)$ *which takes n bits to* $l(n)$ *bits is a related-secret secure pseudorandom generator under* Δ *if* $\forall \mathsf{A} \in PPT$, $\exists \nu$ *negligible:* $\forall k$

$$Pr[x \leftarrow \{0,1\}^k; \mathsf{A}^{\mathsf{F}_{g,x}} = 1] - Pr[x \leftarrow \{0,1\}^k; \mathsf{A}^{\mathsf{F}_{\mathcal{O},x}} = 1] \leq \nu(k)$$

where $\mathcal{O}$ *returns a random string from* $\{0,1\}^{l(k)}$ *on input x.*

Note that for plain-model PRGs, it is normally required that $l(n) > n$, as the identity function is a PRG for $l(n) = n$, and because we anticipate using PRGs repeatedly to obtain arbitrary amounts of randomness. Those reasons for requring $l(n) > n$ do not apply to the related-secret setting: (1) $l(n) = n$ does not imply a trivial function here, and (2) we anticipate using ordinary PRGs to stretch randomness, and RS-PRGs to provide related-secret security.

We can extend the notion of related secret security to even simpler primitives such as one way functions.

Definition 7 (Related-secret secure one way function family (RS-OWFF)). *An indexed family of functions* $\{\mathcal{F}_k\}$, *where each function* $f_s \in \mathcal{F}_k$ *goes from* $\mathcal{D}_s \to \mathcal{R}_s$ *is considered a related secret secure one way function family under* Δ_s *if* $\forall \mathsf{A} \in PPT$, $\exists \nu : \forall k$, $Pr[f_s \leftarrow \{\mathcal{F}\}_k; x \leftarrow \mathcal{D}_{f_s}; x' \leftarrow \mathsf{A}^{\mathsf{F}_{f_s,x}}(f_s) : f_s(x) = f_s(x')] \leq \nu(k)$ *where* ν *is negligible and where* $\mathsf{F}_{f_s,x}$ *on input* $\delta \in \Delta_s$ *returns* $f_s(\delta(x))$.

Definition 8 (Related-secret secure one way permutation family (RS-OWPF)). *A related-secret secure one-way permutation family is a related-secret secure one-way function family in which each individual f_s is a permutation.*

2.2 Hard-core bits

The technique that has been used to create pseudorandomness from the existence of hard problems has been the idea of the *hard core bit*.

Definition 9 (Hard-core bit). *A function $B(x)$: $\{0,1\}^k \rightarrow \{0,1\}$ is considered* hard-core *for a function f if:*

1. *$B(x)$ is polynomial time computable.*
2. *$\forall \mathsf{A} \in PPT$, $\exists \nu$ negligible: $\forall k$, $Pr[x \leftarrow \{0,1\}^k; b \leftarrow \mathsf{A}(f(x)) : B(x) = b] \leq \frac{1}{2} + \nu(k)$.*

A hardcore bit $B(x)$ for a function f can easily be shown to be pseudorandom even given the value $f(x)$. This does not imply however, that the pseudorandomness of the bit is related to the hardness of any particular problem. Consider $f'(x) = f(x_{2,\cdots,k})$. It is clear that x_1 is a hard-core bit for f however it is hard-core due to information loss. As such, no matter what the properties of the function f are, $B(x)$ can never be recovered with probability bounded away from $\frac{1}{2}$. [4]

With this in mind, we define the notion of a *strong hard-core bit*, a bit whose security is directly related to the one way security of the function f.

Definition 10 (Strong hard-core bit). *A function $B(x)$: $\{0,1\}^k \rightarrow \{0,1\}$ is considered a* strong hard-core bit *for a function f if for any $\mathsf{A} \in PPT$ such that $\exists$ non-negligible ϵ where $\forall k$ $Pr[x \leftarrow \{0,1\}^k; b \leftarrow \mathsf{A}(f(x)) : b = B(x)] \geq \frac{1}{2} + \epsilon(k)$ then $\exists$ $\mathsf{A}' \in PPT$ and ϵ' non-negligible such that $\forall k, Pr[x \leftarrow \{0,1\}^k; x' \leftarrow \mathsf{A}'(f(x)) : f(x) = f(x')] \geq \epsilon'(k)$.*

In addition we note that all known non-trivial hard-core bits are in fact strong hard core bits, as their security is proven via a reduction between the ability to predict $B(x)$ and the one-way security of f.

We can extend these definitions to the ideas of a related-secret secure hard-core bit and a related-secret secure strong hard-core bit:

Definition 11 (Related-secret secure hard core bit (RS-HCB)). *A function $B(x)$ is a related-secret secure hard core bit for a function f secure under Δ, if $\forall A \in PPT$, $\exists \nu$ negligible: $\forall k$, $Pr[x \leftarrow \{0,1\}^k; b \leftarrow \mathsf{A}^{\mathsf{F}_{f,x}} : b = B(x)] \leq \frac{1}{2} + \nu(k)$ where $\mathsf{F}_{f,x}$ returns $f(\delta(x))$ on input δ.*

Definition 12 (Related-secret secure strong hard core bit (RS-SHCB)). *A function $B(x)$ is considered a related-secret secure strong hard core bit for a function f secure under Δ, if $\forall \mathsf{A} \in PPT$ such that if $\epsilon(k)$ non-negligible: where $\forall k$ $Pr[x \leftarrow \{0,1\}^k; b \leftarrow \mathsf{A}^{\mathsf{F}_{f,x}} : b = B(x)] \geq \frac{1}{2} + \epsilon(k)$ $\exists \mathsf{A}'$ and $\epsilon'(k)$ non-negligible : $Pr[x \leftarrow \{0,1\}^k; x' \leftarrow \mathsf{A}'^{\mathsf{F}_{f,x}}(f(x)) : f(x) = f(x')] \geq \epsilon'(k)$ for non-negligible ϵ' where $\mathsf{F}_{f,x}$ returns $f(\delta(x))$ on input δ.*

[4] Note, however, that a *permutation* with a hard-core bit is necessarily one-way.

We finally introduce the idea of a related-secret secure pseudorandom bit or RS-PRB. As noted before, normal hardcore bits are inherently pseudorandom. As we will see in the next section however, when the adversary is allowed to see $f(\delta(x))$ and $B(\delta(x))$ for adversarially chosen δ, $B(\delta(x))$ is no longer necessarily indistinguishable from random. With this in mind, we define the notion of a related-secret secure pseudorandom bit as a bit which is pseudorandom even when the adversary gets to see adaptively perturbed bits $B(\delta(x))$.

Definition 13 (Related-secret secure pseudorandom bit (RS-PRB)). *A function $B(x)$ is considered a related-secret secure pseudorandom bit for a function f, under Δ, if $\forall \mathsf{A} \in PPT$, $\exists \nu$ negligible: $\forall k$*

$$Pr[x \leftarrow \{0,1\}^k; \mathsf{A}^{\mathsf{F}_{f||B,x}} = 1] - Pr[x \leftarrow \{0,1\}^k; \mathsf{A}^{\mathsf{F}_{f||\mathcal{R},x}} = 1] \leq \nu(k)$$

where $\mathsf{F}_{g,x}$ is an oracle that on input $\delta \in \Delta$, returns $g(\delta(x))$, and where $f||g$ denotes the function $f||g : x \mapsto f(x)||g(x)$, and where $\mathcal{R}(x)$ is a random function from $\{0,1\}^k$ to $\{0,1\}$.

2.3 Black token algorithms

In this paper we introduce the idea of a *black token* algorithm. Informally, an algorithm is black token if it operates, or could equivalently operate, oblivious to the value of its input, but rather uses a set of allowed operations to manipulate that value.

Explicitly, in the black token model of computation, an algorithm A_{BT} works with two types of values: public values, which are known fully, and private values, which A_{BT} must work with while ignorant of the actual value. For every private value x, A_{BT} sees only a pseudonym for x, id_x.

When A_{BT} receives a private input, it is given only id_x rather than the actual value x; similarly, when A_{BT} makes a private output, the output is taken to be the value for which A_{BT} specified the pseudonym. That is, if A_{BT} outputs id_y, this is interpreted as outputting y. If A_{BT} makes a private output of a pseudonym that has not been determined externally to A_{BT}, we interpret this as outputting a special error symbol $\perp$. The input and output wires of A (both its initial inputs and final outputs, and its means of communicating with its oracles) are each classified as either public or private and always treated in this manner. As such, A_{BT} cannot send a pseudonym down a public channel, or vice versa. Note that A is not inherently given a way to get pseudonyms for values it knows (or chooses) completely. All pseudonyms A receives, it receives from some outside source (as input, or as the output from some oracle).

If there is a class of allowable operations that are polynomially time computable given the actual values of the pseudonyms, we allow A to perform these functions by providing A_{BT} with a "private operation oracle" P, which can be used to perform such operations without revealing the actual values of the inputs to A. P returns outputs which may be public or private.

If such a machine A_{BT} exists in a black token model, we can create a black token algorithm in the standard model by creating a machine T to act as a tokenizer. T has oracle access to A_{BT}, as well as any oracle A_{BT} might possess. As such, $(\mathsf{T}^{\mathsf{A}_{BT}})^{\mathsf{O}}(x)$ is the machine that runs $\mathsf{A}_{BT}^{\mathsf{O}}$ where T translates things to and from pseudonyms as appropriate as it gives input to A, passes messages back and forth between A and O, and receives final output from A.

For example, $B(x)$, defined to be the parity of x, is a strong hard-core bit for RSA. The reduction from a hard-core bit finder to inverting RSA is blac-token: the x^e can be treated as a pseudonym, and the algorithm requires only the ability to calculate $(-x)^e$ and $(x \cdot 2^{-1})^e$, both of which can be done via private operations with private outputs. For details, see Theorem 11 in the appendix.

Effectively, the assumption that an algorithm is black-token is akin to assuming that the algorithm "knows" how to derive the ultimate output from the input using the allowed private operations. This is a stronger assumption than "knowledge of exponent assumptions," which do not restrict the allowable private operations, and it is a stronger than the claim that the algorithm is *algebraic* or a *generic ring algorithm* [16–19], which do restrict allowable operations but do not explicitly require this sort of knowledge. As such, assuming that any *adversary* is a black-token algorithm is an uncomfortably strong assumption. However, unlike prior results that use assumptions of this type, we never assume this of an adversary.

3 The importance of RS-PRBs

In this section we attempt to mirror the construction of a pseudorandom permutation from simpler primitives in the related-secret security model. Since we address related-secret security for *complete* perturbation sets Δ, we cannot expect to build any related-secret secure constructions without an underlying component with related-secret security. As such, we show that homomorphic one-way functions or permutations are related-secret one-way under a Δ compatible with the homomorphism.

Theorem 1. *Let f be one-way and homomorphic, so that there exist efficiently computable binary operations $\odot$ and $\star$, such that for all x, y, $f(x \odot y) = f(x) \star f(y)$. Then f is related-secret one-way under $\Delta_{\odot}$.*

Proof. We use the homomorphic property of f to simulate related-secret queries $f(\delta_c(x)) = f(c \odot x)$ by making queries $f(c)$, $f(x)$ and outputting $f(c) \star f(x)$.

For instance, if f is defined as $f(x) = g^x \bmod p$ for prime p and where g is a generator for $\mathbb{Z}_p^*$, then f is homomorphic where $\odot$ is addition and $\star$ is multiplication. So if f is a one-way permutation, f is also a RS-OWP.

In the standard model, the next step would be to find hard-core bits of hard functions. We give a general construction of related-secret strong hard-core bits. Our construction will use similiar techniques to the ones found in [20].

Theorem 2. *Let* $\mathsf{C}, \mathsf{C}^{-1}, \mathsf{R}$ *be a* (k, ρ, p) *linear list decodable code over* $\mathbb{F}_2^n$*. Define* $B(x,r)$ *as* $\mathsf{C}(x)_r$*, the* r^{th} *bit of* $\mathsf{C}(x)$*. Let* f *be an RS-OWF secure under* $\Delta_\oplus$*. Define* $f'(x,r)$ *as* $f(x), r$*.* $B(x,r)$ *is an RS-SHCB for* f'*.*

Proof. To prove that $B(x,r)$ is an RS-SHCB for f' we will create a black-box reduction $\mathsf{M}^{\mathsf{F}_{f',(x,r)},\mathsf{A}}$ which will invert $f(x)$ with non-negligible probability as long as $\mathsf{A}^{\mathsf{F}_{f',(x,r)}}$ returns $B(x,r)$ with probability non-negligibly better than $1/2$. We will build $\mathsf{M}^{\mathsf{F}_{f',(x,r)}}$ to use R to reconstruct x. As such, M needs to simulate oracle access to Y for a corrupted codeword y such that $||C(x) - y|| \leq \rho n$ where $|C(x)| = n$. M has access to an oracle A which returns $B(x,r) = \mathsf{C}(x)_r$ with probability $\frac{1}{2} + \epsilon$ for non-negligible ϵ. A difficulty in this proof is that $1 - \rho$, the probability that R requires $B(x,r) = \mathsf{C}(x)_r$ to be correct, will often be much larger than $\frac{1}{2} + \epsilon$, the probability that $\mathsf{A}^{\mathsf{F}_{f',(x,r)}}$ returns $B(x,r)$ correctly. As such, we cannot directly use the answers returned by A.

To amplify the success probability of A, M will use the fact that Δ is closed under composition. As such, M given $\mathsf{F}_{f',(x,r)}$ can simulate $\mathsf{F}_{f',(\delta_c(x),r)}$ for random δ_c as $\mathsf{F}_{f',(\delta_c(x),r)}(\delta, \delta_{c'}) = \mathsf{F}_{f',(x,r)}(\delta\delta_c, \delta_{c'})$ When $\mathsf{A}^{\mathsf{F}_{f',(\delta_c(x),r)}}$ returns its guess at g at $B(\delta_c(x), r)$, M can compute $B(x,r) = g \oplus \mathsf{C}(c)_r$ which is correct as long as $g = B(\delta_c(x), r)$ as the code is linear. For random $\delta_c \in \Delta_\oplus$ $\delta_c(x)$ is random. This allows $\mathsf{M}^{\mathsf{F}_x,\mathsf{A}}$ to find many independent votes for $B(x,r)$, where each individual vote is correct with non-negligible probability. As such, M can find $\mathsf{C}(x)_r = B(x,r)$ with high enough probability to simulate Y and thus run the reconstruction program R. When R returns $\mathcal{S}$, M computes $f(x_i)$ $\forall x_i \in \mathcal{S}$ until he finds $x_* :, f(x_*) = f(x)$.

This can be seen as a generalization of the Goldreich-Levin bit to the related-secret case, and expanded to capture the use of other list-decodable codes. As most known list decodable codes are linear, this suggests that list decodable codes in general imply a RS-HCB for any p secure under $\Delta_\oplus$.[5] In Appendix B we prove a partial converse to this theorem, showing that certain well behaved strong hard core bits can be used to create list decodable codes.

In the standard model, the next step towards a pseudorandom permutation would be to show that hard-core bits are pseudorandom. Unfortunately, this does not hold in general in the related-secret case. In particular, we have shown the Goldreich-Levin hardcore bit $\langle x, r\rangle$ to be an RS-SHCB, but it is trivially seen to not be a RS-PRB under $\oplus$ for $f'(x,r) = f(x)||r$.

Theorem 3. *The Goldreich Levin hardcore bit* $\langle x, r\rangle$ *for the function* $f'(x,r) = f(x), r$ *is not an RS-PRB for* f' *under* $\Delta_\oplus$*.*

Proof. Just query the oracle under $(\delta_{ident}, \delta_c)$, $(\delta_{ident}, \delta_{c'})$, and $(\delta_{ident}, \delta_{c\oplus c'})$ receiving either $b_1 = \langle x, r \oplus c\rangle$, $b_2 = \langle x, r \oplus c'\rangle$ and $b_3 = \langle x, r \oplus c \oplus c'\rangle$ or random bits b_1, b_2, b_3. Output 1 if $b_1 \oplus b_2 = b_3$. If the bits are the inner products (outputs

[5] Also note that if we have a linear list decodable code that takes words from $\mathbb{F}_q^k$ to $\mathbb{F}_2^n$, this gives us an RS-SHCB for any RS-OWF secure under vector addition, where the vectors are in $\mathbb{F}_q^k$.

of $B(x, r)$) then this equation will hold with probability 1. If the bits are random, the equation will hold with probability $\frac{1}{2}$.

Thus we see a potential separation between the difficulty in predicting the bit $B(x)$ and the bit being pseudorandom in the related secret attack setting, a separation that does not exist with regards to normal hardcore bits.

We go on to show that related-secret pseudorandom bits can be used to construct related-key secure blockciphers. First, we show how to construct an RS-PRG from an RS-PRB.

Theorem 4. *Let f be a function from $\{0,1\}^k$ to $\{0,1\}^{p(k)}$, such that there is a B that is a RS-PRB for f under Δ. Then there exists a g that is an RS-PRG from k^2 to k^2 bits.*

Proof. Since f has an RS-PRB, f must be an RS-OWF: if an adversary were able to invert f with probability ϵ, we could attack the PRB by inverting f, and, if successful, checking the outputs of B. Since f is an RS-OWF, f must in particular be a OWF.

Let $g'(x)$ be a k-bit to k^2 bit PRG; we know g' exists because we have shown that OWFs exist. Define $g : \{0,1\}^{k^2} \rightarrow \{0,1\}^{k^2}$ as follows. On input x, parse x as k k-bit blocks $x_1, \ldots, x_k$, compute $y = B(x_1)||\ldots||B(x_k)$, and output $g'(y)$.

Then g is an RS-PRG for Δ^k where $(\delta_1, \ldots, \delta_k)(x_1||\ldots||x_k) = (\delta_1(x_1)|| \ldots || \delta_k(x_k))$.

This proof comes easily from the idea that $y_i = B(\delta_i(x_i))$ is indistinguishable from random for all δ_i selected by A, due to the fact that $B()$ is an RS-PRB and x_i is random. As each x_i is random and independent of any other x_j, and each δ_i is independent from the other δ_j, we can consider each bit $B(\delta_i(x_i))$ to be indistinguishable from random, even given the other bits $B(\delta(x_j))$. The *normal* PRG expands the pseudorandom string to the correct length, finishing the proof.

The proof of the following corollary is obvious from the proof of the previous theorem.

Corollary 1. *Let $g()$ be a RS-PRG that takes n bits to $p(n)$ bits. Then $f_x(\delta) = g(\delta(x))$ is a PRF from $\Delta \rightarrow \{0,1\}^{p(n)}$.*

This proof illustrates an important trick, namely, that if related-key pseudorandomness is applied directly to a random secret, we can achieve security using traditional techniques afterwards.

We now give two proofs, together showing that the existence of an RS-PRG implies the existence of an RK-PRP and RK-PRF. The proofs illustrate an easy way to gain related-key security from a related-secret secure pseudorandom generator. We may use a related-secret pseudorandom generator to eliminate any advantage an adversary may gain from a related-secret attack on a standard construction. The adversary's choice of δ has no effect as it gets translated to a key that looks random and independent of other keys.

Theorem 5. *If $RS-PRGs$ exist under Δ, $RK-PRFs$ exist under Δ.*

Proof. Say g is an RS-PRG that takes k bits to $l(k)$ bits. If g is an RS-PRG, g must be a one-way function. If not, an adversary could distinguish the output of g from random by inverting g and checking if the result is correct and consistent with queries to g. From [21] and [22], we know that if one-way functions exist, so do (standard) pseudorandom functions. Let f be a pseudorandom function taking a $p(k)$ bit input and an $l(k)$-bit seed to a $p'(k)$-bit output. Let $f'(x, K) = f(x, g(K))$. Then f' is a related-key secure pseudorandom function under Δ.

We prove this by a simple hybrid argument. If A can distinguish between $\mathsf{F}_{f',K}$ and $\mathsf{F}_{\mathcal{F},K}$ for random K, then either A can distinguish between $\mathsf{F}_{f'(\cdot,K),K}$ and $\mathsf{F}_{f''(\cdot,K),K}$ or between $\mathsf{F}_{f''(\cdot,K),K}$ and $\mathsf{F}_{\mathcal{F}(\cdot,K),K}$, where $f''(x, K) = f(x, \mathcal{R}(K))$ where $\mathcal{R}$ is a random function from k bits to $l(k)$ bits. If the former, then A breaks the related-key security of g. If the latter, then A distinguishes between a random function and f with many independent random seeds.

If $|Pr[K \leftarrow \{0,1\}^k : \mathsf{A}^{\mathsf{F}_{f'(\cdot,K),K}} = 1] - Pr[K \leftarrow \{0,1\}^k : \mathsf{A}^{\mathsf{F}_{f''(\cdot,K),K}} = 1]|$ is non-negligible, then we can use A to break g. Given an oracle O, we run A in its attack with oracle $\mathsf{F}_{\mathsf{f}'_{\mathsf{O}},\mathsf{K}}$, where $f'_{\mathsf{O}}(x, K) = f(x, \mathsf{O}(K))$.

If $|Pr[K \leftarrow \{0,1\}^k : \mathsf{A}^{\mathsf{F}_{f''(\cdot,K),K}} = 1] - Pr[\mathsf{A}^{\mathsf{F}_{\mathcal{F}(\cdot,K),K}} = 1]| \geq \epsilon(k)$ non-negligible, then we can use A to break f. Simply put, $\mathsf{F}_{\mathcal{F}(\cdot,K),K}$ differs from $\mathsf{F}_{f''(\cdot,K),K}$ only in that f'' runs f on a random seed for each distinct $\delta(K)$, while $\mathcal{F}$ is a random function for each distinct $\delta(K)$. By a simple hybrid reduction, we obtain a probability of at least $\epsilon(k)/T(\mathsf{A})$, where $T(\mathsf{A})$ is the running time of A, which is polynomial.

We state the next theorem, showing that RS-PRG's imply RK-PRP's, as a corollary. We omit the proof, but it is essentially identical to the proof of Theorem 5.

Corollary 2. *Let $\mathsf{E}(x, K)$ be a pseudorandom permutation family taking a $p(k)$-bit input and an $l(k)$-bit seed to a $p(k)$-bit output. Let g be a RS-PRG taking a k-bit input to an $l(k)$-bit output. Then $\mathsf{E}(x, g(K))$ is a RK-PRP.*

Remark 1. A related-secret PRG effectively allows us to "harden" any secret-key-based construction to be secure against related-key attacks, by using $g(K)$ in place of K.

This should apply to any construction X for which security implies security when an attacker may query X with many independent random secret keys. For such constructions (such as PRFs, as in the proof of 5), related-key security follows because no adversary can distinguish between the modified construction and querying the original consturction on many independent random secret keys, one for each perturbation.

We have shown that RS-PRBs are sufficient to construct PK-PRPs. We end this section by showing that RS-PRBs are "necessary" for the existence of related key secure pseudorandom functions and permutations. We show that RK-PRP's imply RS-PRG's, and RS-PRG's imply a one way function f and an RS-PRB $B()$ for f.

Theorem 6. *Let* $\mathsf{E}(x, K)$ *be an RK-PRP under* Δ*. Let* $g(x')$ *be the function which parses* x' *as* (x, K) *and outputs* $\mathsf{E}(x, K), \mathsf{E}(x+1, K), \mathsf{E}(x+2, K)$*. Then for any valid set of perturbations* Δ' *on* $\{0,1\}^{p(k)}$*,* $g(x)$ *is an RS-PRG under* $\Delta' \times \Delta$*.*

Proof. The proof follows from the fact that $g(x)$ is simulatable given access to oracle access to $\mathsf{E}(x, K)$. As such, if an adversary A can distinguish between g and random values, we can build an adversary A' which picks a random x, queries its E oracle on $(\delta_1(x), \delta_2)$, $(\delta_1(x+1), \delta_2)$, $(\delta_1(x+2), \delta_2)$ when A queries on $\delta_{12} = (\delta_1 || \delta_2)$. If A's oracle is the oracle to the pseudorandom permutation then A simulates g, otherwise A' simply returns a random value for each distinct $\delta_{12}(x, K)$. As such, A' can use A to distinguish between the two cases.

We now demonstrate that RS-PRG's give us a one way function $f(x)$ and an RS-PRB $B(x)$ for that one way function $B()$.

Theorem 7. *Let* $g(x)$ *be a* k *bit to* $l(k) > k$ *bit RS-PRG. Let* $f(x)$ *be the first* $l(k)-1$ *bits of* $g(x)$ *and let* $B(x)$ *be the last bit of* $g(x)$*. Then* $f(x)$ *is an RS-OWF, and* $B(x)$ *is an RS-PRB for* f*.*

Proof. If f is not an RS-OWF, then it is easy to show that g is not an RS-PRG. If A attacks the one way security of f, A' takes the δ queries made by A queries it's oracle, chops off the last bit of the result and returns it to A. If A returns a value x, A' can check the outputs of its oracle to see if they are equal to $g(\delta(x))$.

The fact that $B(x)$ is an RS-PRB for $f()$ comes from the fact that $g(x) = f(x)||B(x)$ is an RS-PRG.

4 Difficulties in constructing an RS-PRB

In the previous section we demonstrated that related-secret pseudorandom bits were both necessary and sufficient for the existence of provably secure related-key pseudorandom permutations. While we discussed related-secret one-wayness and constructed related-secret strong hard-core bits we were unable to give any construction of an RS-PRB. In fact, the homomorphic properties of known hardcore bits mean that they cannot be pseudorandom.

In this section we give a surprising result concerning possible constructions of RS-PRB's, in that we show that black token, black box strong hard-core bits cannot be related-secret pseudorandom bits.

Definition 14. *If* M *is a black-token algorithm, its private operation oracle* P *is said to be perturbation-private if all operations performed by* P *with private outputs are of the form* $(id_{f(x)}, \delta) \mapsto id_{f(\delta(x))}$ *for* δ *in a class* Δ*, for some* f*.*

Theorem 8. *Let* $B(x)$ *be a strong hard-core bit for a one way function* f *that has a black token, black-box reduction* M *where the private operation oracle* P *is efficiently computable, and perturbation-private for a class* Δ *that is closed under composition. Then* $B(x)$ *is not an RS-PRB under any* $\Delta' \supset \Delta$*.*

For example, the reduction that proves that parity is a hard-core bit for RSA is a black-token, perturbation-private reduction, where the class $\Delta = \{\delta_r : x \mapsto x \cdot r\}$.

Proof. If $B(x)$ is a black token, black box SHCB for a one-way function f, then there exists a black token algorithm M such that $\mathsf{M}^{\mathsf{P},\mathsf{A}}(id_{f(x)})$ finds x with non-negligible probability as long as $\mathsf{A}(f(x))$ outputs $B(x)$ with probability $1/2 + \epsilon$ where ϵ is non-negligible.

We construct A' to attack B as an RS-PRB under Δ'. We can view A' as having access to two oracles, O and $\mathsf{F}_{f,x}$, where O either returns $B(x)$ or a random bit. First A' queries $\mathsf{F}_{f,x}(\delta_{ident})$ to obtain $f(x)$, creates a token $id_{f(x)}$ and then uses as $id_{f(x)}$ as input to M. A' then runs M, acting as the tokenizer T. A' uses O to answer M's queries to A, and uses $\mathsf{F}_{f,x}$ to answer M's queries to P, when necessary. When M returns x', A' checks if $f(x') = f(x)$; if so, it outputs 1, otherwise it outputs 0.

Since M is black token it can only obtain new pseudonyms from P. Since Δ is closed under composition we can always associate every pseudonym id_y M has with a specific $\delta \in \Delta$ such that $y = f(\delta(x))$. Since A' keeps track of this association, when M asks for $\mathsf{A}(id_{f(\delta(x))})$, A' can query $\mathsf{O}(\delta)$ and return the result as the answer. A' answers queries to P of the form $id_{f(\delta(x))}, \delta'$ by querying $\mathsf{F}_{f,x}(\delta' \circ \delta)$, and returning a pseudonym of the result. Other queries to P may be possible, but always produce "public" results. For such queries, the answers are computable given the actual values of all inputs. Thus, A' need only translate pseudonyms to real values and then perform the computation.

When $\mathsf{O} = \mathsf{B}$, A' provides a faithful simulation of A and P. As such, the probability that A' will output 1 is non-negligible, since the probability that M outputs the correct x is non-negligible (since B is always right, it has advantage $\epsilon = 1/2$). On the other hand, if $\mathsf{O} = \mathcal{R}_x$, the bits given to M are random. If $\mathsf{M}^{\mathsf{P},\mathsf{A}}(f(x))$ could output the correct value x with non-negligible probability where A returns only random bits, then $\mathsf{M}^{\mathsf{P},\mathsf{S}}(id_{f(x)})$ can output x with non-negligible probability where S just outputs random bits. As such M can be used by itself to break the one-wayness of $f(x)$. Thus, with all but negligible probability, if $\mathsf{O} = \mathcal{R}_x$, A' will receive x' which is not a preimage of $f(x)$, and will thus return 0. As such, A' is a successful distinguisher.

Note that we make only the most general restriction on the types of acceptable reduction in Theorem 8, the main limitation being that we require that M only ask A for the hardcore bit of a $\delta(x)$ where we know the δ. All known examples of strong hard-core bits have reductions that can be seen as black-token and perturbation-private. Note that the conditions of Theorem 8 only require such a reduction for A that is correct with probability 1: a very useful observation, since the probability-1 reductions are often far simpler. As an example, consider the Goldreich-Levin hard-core bit, and its reduction assuming A is always correct:

Theorem 9. *Let $f(x)$ be a one way function. Let $f'(x,r) = f(x), r$. Let $B(x,r) = \langle x, r\rangle$. $B(x,r)$ is a black token SHCB for f' with $\{I\} \times \Delta_{\oplus}$*

Proof. The machine M is simple to construct. M begins by receiving $id_{f(x),r}$. M makes use of P only to compute $P(id_{f(x),r}) = r$ and to compute $P(id_{f(x),\delta(r)}, \delta') = id_{f(x),\delta'(\delta(r))}$. It aims to query A on $id_{f(x),r_i}$ for different r_i until it can obtain x via standard linear algebra. In order to do this, it learns r, and calculates $id_{f(x),r_i}$ by querying P on $id_{f(x),r}, \delta_{r_i \oplus r}$.

This reduction can easily extend to an A whose success probability is less than 1. The proof of Goldreich and Levin involves querying $f(x), r_i$ for many random pairs of r_i with specified differences. This allows for us to determine $\langle x, r_i \rangle$ with high probability. Since the $f(x)$ value is left untouched, the same argument applies; the general reduction is also black token.

See Appendix A for discussion of some other hard-core bit reductions.

We have shown that black token strong hard-core bits cannot be related-secret pseudorandom. We further show that all related-secret pseudroandom bits are hard-core bits:

Theorem 10. *If B is an RS-PRB for a function f under Δ then B is a hard-core bit for f.*

Proof. Suppose there exists an A such that $\mathsf{A}(f(x))$ returns $B(x)$ with probability $1/2 + \epsilon$. Then we can attack B as an RS-PRB by obtaining $f(x)$ and $\mathsf{O}(x)$ and checking whether $\mathsf{O}(x) = \mathsf{A}(f(x))$; if so, we return 1, otherwise, we return 0. Then the difference in probabilities is ϵ, which for non-negligible ϵ is enough to break $B(x)$ as an RS-PRB.

5 Discussion

We know of only two ways to construct pseudorandom primitives: directly from hard-core bits in the standard model, or from other standard-model pseudorandom primitives. Our impossibility result shows that related-secret pseudorandomness based off of hard-core bits is unlikely. Nor is it plausible to construct a related-secret pseudorandom bit directly from standard-model pseudorandomness: all such primitives have a secret seed or key, and thus, in any construction, the adversary (because of the completeness we require of Δ) must be able to make queries that require modifications of those secrets. In other words, either the construction will fail, or the pseudorandomness we are using must already be related-secret secure.

Since any RS-PRB is inherently a hard-core bit, Theorem 8 leaves open two potential ways in which an RS-PRB might yet be possible. The RS-PRB might be a hard-core bit, but not a strong one, or it might be a strong hard-core bit but not one with a reduction that is black-token and perturbation-private. Normally, one proves that a bit is hard-core by providing a reduction to the hardness of inverting the associated function: in other words, normally, hard-core bits are always strong hard-core bits. This is natural, since the associated function must be one-way in any case, and thus any proof not requiring extra assumptions would reduce to its one-wayness.

If B is a strong hard-core bit, then its reduction must not meet the conditions of Theorem 8. As we discussed, all known examples of strong hard-core bits have black-token reductions of the type necessary for our impossibility proof. Note also, that it is difficult to see what useful information M can receive by running A on input that is not a valid $f(\delta(x))$ value for some δ, or when δ is unknown to M.

At a higher level, our restrictions on the type of reductions used in the proof are reasonable ones. Since no assumptions can be made about observable properties of $f(x)$, these values are mostly ignored in any proof involving generic one-way functions or permutations. As such, any "private operations" in those proofs are kept to a strict minimum because they must be efficiently computable given only $f(x)$. Also, in the case of bits that are *generic* - that is, secure for a variety of functions - it is hard to imagine a proof of their security that is not black-token and black-box. Finally note that the proof applies if there *exists* a reduction of the specified type. As such, even if there is a very unusual reduction that does not meet the conditions of Theorem 8, if other more usual reductions exist, the theorem still applies.

The major open problem in related-secret security is whether or not related-key secure blockciphers exist. We have shown that related-secret pseudorandom bits are necessary and sufficient for higher forms of related-secret pseudorandomness. However, related-secret pseudorandom bits cannot be constructed using traditional techniques. This leaves a significant open problem: are related-secret pseudorandom bits possible under only basic assumptions? Or alternatively, can fundamentally new techniques be found to create related-secret pseudorandom bits?

References

1. Kelsey, J., Schneier, B., Wagner, D.: Related-key cryptanalysis of 3-way, biham-des, cast, des-x, newdes, rc2, and tea. In: ICICS '97: Proceedings of the First International Conference on Information and Communication Security, London, UK, Springer-Verlag (1997) 233–246
2. Poorvi L. Vora, D.J.M.: Related-key linear cryptanalysis. In: 2006 IEEE International Symposium on Information Theory. (2006) 1609 – 1613
3. Mir, D.J., Vora, P.L.: Related-key statistical cryptanalysis. Cryptology ePrint Archive, Report 2007/227 (2007) http://eprint.iacr.org/.
4. Gorski, M., Lucks, S.: New related-key boomerang attacks on AES. In Chowdhury, D.R., Rijmen, V., Das, A., eds.: INDOCRYPT. Volume 5365 of Lecture Notes in Computer Science., Springer (2008) 266–278
5. Biham, E.: New types of cryptanalytic attacks using related keys. Journal of Cryptology **7**(4) (Fall 1994) 229–246 Also available at: `citeseer.nj.nec.com/biham94new.html`.
6. Zhang, W., Zhang, L., Wu, W., Feng, D.: Related-key differential-linear attacks on reduced aes-192. In: Progress in Cryptology - INDOCRYPT 2007. (2007)
7. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and related-key attack on the full AES-256. In Halevi, S., ed.: Advances in Cryptology – CRYPTO 2009. Volume 5677 of Lecture Notes in Computer Science., Springer-Verlag (August 2009)

8. Bellare, M., Kohno, T.: A Theoretical Treatment of Related-Key Attacks: PKA-PRPs, RKA-PRFs, and Applications. In Biham, E., ed.: Advances in Cryptology – EUROCRYPT '03. Volume 2656 of LNCS. (2003) 491–506
9. Black, J., Cochran, M., Shrimpton, T.: On The Impossibility of Highly-Efficient Blockcipher-Based Hash Functions. In: Advances in Cryptology – Eurocrypt 2005. Volume 3494 of LNCS., Springer Verlag (May 2005) 526–541
10. Dodis, Y., Katz, J., Reyzin, L., Smith, A.: Robust fuzzy extractors and authenticated key agreement from close secrets. In: Advances in Cryptography - CRYPTO. (2006) 232–250
11. Cramer, R., Dodis, Y., Fehr, S., Padro, C., Wichs, D.: Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In: Advances in Cryptology - EUROCRYPT. (April 2008) 471–488
12. Kanukurthi, B., Reyzin, L.: An improved robust fuzzy extractor. In: SCN. (2008) 156–171
13. Lucks, S.: Ciphers secure against related-key attacks. In Roy, B.K., Meier, W., eds.: FSE. Volume 3017 of Lecture Notes in Computer Science., Springer (2004) 359–370
14. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In Reingold, O., ed.: TCC. Volume 5444 of Lecture Notes in Computer Science., Springer (2009) 474–495
15. Goldreich, O., Levin, L.: A hard-core predicate for all one-way functions. In: Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing, Seattle, Washington (15–17 May 1989) 25–32
16. Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log. In: Advances in Cryptology - ASIACRYPT 2005. (2005) 1–20
17. Boneh, D., Venkatesan, R.: Breaking RSA may not be equivalent to factoring. In: Advances in Cryptology - EUROCRYPT 1998. (1998)
18. Aggarwal, D., Maurer, U.: Breaking rsa generically is equivalent to factoring. Cryptology ePrint Archive, Report 2008/260 (2008) http://eprint.iacr.org/.
19. Maurer, U.: Abstract models of computation in cryptography. In Smart, N., ed.: Cryptography and Coding 2005. Volume 3796 of Lecture Notes in Computer Science., Springer-Verlag (December 2005) 1–12
20. Akavia, A., Goldwasser, S., Safra, S.: Proving hard-core predicates using list decoding. In: FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, Washington, DC, USA, IEEE Computer Society (2003) 146
21. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM **33**(4) (October 1986) 792–807
22. Håstad, J., Impagliazzo, R., Levin, L., Luby, M.: Construction of pseudorandom generator from any one-way function. SIAM Journal on Computing **28**(4) (1999) 1364–1396
23. Goldwasser, S., Micali, S., Tong, P.: Why and how to establish a private code on a public network. In: SFCS '82: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, Washington, DC, USA, IEEE Computer Society (1982) 134–144
24. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo-random bits. SIAM Journal on Computing **13**(4) (November 1984) 850–863

A Black token reductions for known hardcore bits

In this section we examine two other well-known hardcore bits, the hardcore bits for RSA, discrete log, (the generic hardcore bit of Goldreich and Levin was examined in the body of the paper). For each one we demonstrate that these hardcore bits are black token hardcore bits that fit the requirements of Theorem 8. Many of these proofs will simply be restatements of previous reductions, or slightly modified versions to emphasize the fact that they are black token. For each proof, we only formally show that the necessary reduction exists and is black token for an adversary which always returns the correct hardcore bit perfectly, however we also informally state how the proof is changed to adapt an imperfect adversary and how this is also black token.

We next address the hardcore bit for the RSA function. The reduction is taken from [23].

Theorem 11. *Let* $\mathsf{RSA}_{N,e}$ *be the RSA function, that is* $\mathsf{RSA}_{N,e}(x) = x^e \bmod N$. *Define* $B(x)$ *as the parity of* x. *Then* $B(x)$ *is a black token SHCB for* $\mathsf{RSA}_{N,e}$

Proof. P allows two transformations, $\delta_{\frac{1}{2}}(x) = x(2^{-1}) \bmod N$ and $\delta_{-1}(x) = -x \bmod N$. These can be viewed as two specific transformation among a more general class of multiplicative transformations $\delta_r(x) = xr \bmod N$. In general, to calculate $(rx)^e$ given x^e, we need only multiply x^e by r^e.

M asks A for the parity (LSB) of id_{x^e}. If 0, then M runs P on $(id_{x^e}, \delta_{\frac{1}{2}})$ to obtain $id_{(x/2)^e}$. If 1, then M runs P on $(id_{x^e}, \delta_{\frac{1}{2}} \circ \delta_{-1})$ to obtain $id_{(-x/2)^e}$.

Since $-x$ has the opposite parity of x (since N is odd), we always divide an *even* residue by 2, thus effectively shifting the unknown bits down by one. We collect one bit of x at a time, keeping track of the number of times we have applied δ_{-1}, as these reverse our results.

For A with probability of success less than 1, the reduction is far more complicated, but still can be viewed as a sequence of applications of multiplicative transformations of x by manipulating x^e.

We next address the hardcore bit for the discrete log function. The reduction is taken from [24]

Theorem 12. *Let* $\mathsf{f}_{g,p}$ *be the modular exponentiation function, where* g *is a generator of the group* $\mathbb{Z}_p^*$. *Let* $B_p(x)$ *be the function that outputs 1 if* $x \leq \frac{p-1}{2}$, *0 otherwise.* $B_p(x)$ *is a black-token SHCB for* $\mathsf{f}_{g,p}$.

Proof. P computes several transformations, $\delta_{-1}(x) = x - 1$, $\delta_{\frac{1}{2}}(x)$ which returns either $\frac{x}{2}$ or $\frac{x}{2} + \frac{p-1}{2}$, $\delta_{+\frac{p-1}{2}}$ which returns $x + \frac{p-1}{2}$, and $p(x)$, a predicate which returns the least significant bit of x. These can all be seen as computable using multiplicative and/or additive transformations on x, which are efficiently computable. $g^{\delta_{+r}(x)} = g^{x+r} = g^x g^r$, and $g^{\delta_{*r}(x)} = g^{xr} = (g^x)^r$. Take Δ to be the resulting class, closed under composition.

M proceeds as follows. It first obtains id_y for $y = g^x \bmod p$. It then queries $p(id_y)$ and obtains a bit. If 1, M queries P on (id_y, δ_{-1}) obtaining a pseudonym

for $y' = g^{x-1}$. If 0, $\mathcal{M}$ considers $y' = y$ and $id_y = id_{y'}$. M then makes a query to $\mathsf{P}(id_{y'}, \delta_{\frac{1}{2}})$ and $\mathsf{P}(\cdot, \delta_{+\frac{p-1}{2}})$, obtaining pseudonyms for the two square roots, g^s and $g^{s+\frac{p-1}{2}}$ where $g^{2s} = y'$, in unknown order. M then sends both pseudonyms to A which enables him to find the pseudonym for $g^{\frac{x'}{2}}$ where $x' = x$ or $x - 1$.

This allows M to obtain the least significant bit of x, then shift the bits of x one to the right. By repeating this process we may obtain all the bits of x.

We note that dealing with imperfect A is done by simply computing multiplication mod p by known quadratic residues. As such, the full proof still remains black token.

A.1 Other hardcore bits

There are too many examples of hardcore bits to analyze all known proofs. Hardcore bits for specific functions tend to work via homomorphic properties of $f(x)$; the RSA and discrete log examples show how these can be viewed as black token. Generalized hardcore bits are extensions of Goldreich-Levin, and, as such, ignore the value of $f(x)$ completely. Note that virtually any algorithm can be viewed as a black-token algorithm of this sort, for the appropriate class Δ. One question may be whether Δ is "valid," but actually the answer is irrelevant: all perturbatinos in the Δ that arise are in fact secret perturbation we can calculate efficiently.

B Black Token RS-SHCB's to Codes

In this section we prove a corollary to Theorem 2, demonstrating that certain well behaved strong hard core bits can be viewed as error correcting codes.

Theorem 13. *Let $B(x)$ be a black token RS-SHCB for a function f secure under Δ where Δ is of finite size and where M only makes queries to P of the form $\mathsf{P}(id_{f(\delta(x))}, \delta')$. Establish an ordering on Δ, $\delta_1, \delta_2, \cdots \delta_{|\Delta|}$. Define $\mathsf{C}(x)_i$ as $B(\delta_i(x))$. Then we can create a C^{-1} and a R such that $\mathsf{C}, \mathsf{C}^{-1}, \mathsf{R}$ is a (k, ρ, p) list decodable code where $1 - \rho = \frac{1}{2} + \epsilon$, where p is the success probability of M and where l is the number of queries M makes to A.*

Proof. The machine R will use the machine M so it needs to be able to simulate P, $\mathsf{F}_{f,x}$ and A. The simulations of $\mathsf{F}_{f,x}$ and P will be relatively easy as their outputs in the black token model are random pseudonyms $id_{f(\delta(x))}$. The simulation of A is accomplished by using the oracle Y.

R^{Y} first creates a random value $id_{f(x)}$ as the "token" for $f(x)$ (which it does not know) and passes $id_{f(x)}$ to M. When M makes a query $\mathsf{F}_{f,x}(\delta)$ or $\mathsf{P}(id_{f(\delta(x))}, \delta')$, R returns a randomly generated token which it associates with $id_{f(\delta(x))}$ / $id_{f(\delta'\delta(x))}$. When M makes a query $\mathsf{A}(id_{f(\delta_i(x))})$, R^{Y} simulates A by querying Y for $B(\delta_i(x)) = \mathsf{C}(x)_i$, which is correct with probability $\frac{1}{2} + \epsilon$. Since M operates in the black token model, and receives only $id_{f(x)}$ as input, and only queries P on $id_{f(\delta(x))}, \delta'$, R can simulate A perfectly and as such $\mathsf{M}^{\mathsf{A},\mathsf{F}_{f,x}}$ will output x with probability p.