

Related-Secret Security: Definitions, Implications, and a Separation

Anonymized for submission

May 28, 2009

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, little work has been done on related-key security for primitives other than blockciphers. In addition, previous work has yet to construct a related key secure blockcipher without resorting to strong assumptions.

In this paper, we develop a theoretical framework for describing a broad class of “related-secret secure” cryptographic primitives, a class of primitives including related-key secure blockciphers and PRFs. We discuss existence implications between these various primitives. We also give a strong argument that *pseudorandom* bits do not exist in the related-secret setting (that is, related-secret hardcore bits are not pseudorandom). As pseudorandom bits have been essential in showing the relationship between “hardness” and pseudorandomness in the normal model, these results cast doubt on the ability of designing a related-key secure blockcipher utilizing similar techniques.

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 [17, 23, 20], including attacks on [15, 4, 24]. 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 [3]. For another, blockcipher based hash functions are only proven secure in the ideal cipher model [5]; in this strong model, related-key security is implied [3]. Thus, the use of a 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.

Positive results concerning related-key security are few. Bellare and Kohno [3] 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 [18] 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 number theoretic assumptions.

In this paper, we are concerned with the relationship between the powerful notion of a related-key secure blockcipher and simpler primitives. In order to study this notion, we broaden the concept of related-key attacks to “related-secret” attacks, which apply to any primitive which takes inputs unknown to the adversary, whether or not these inputs are considered “keys”. We give definitions for many different related-secret secure primitives such as related-secret secure one way functions/permutations (RS-OWF’s/RS-OWP’s), hardcore bits (RS-HCB’s), *pseudorandom bits* (RS-PRB’s), pseudorandom generators (RS-PRG’s), pseudorandom functions (RS-PRF’s), and pseudorandom permutations (RS-PRP’s). (Our definitions of RS-PRF’s and RS-PRP’s are similar to [3]). We also show the following positive results:

1. We construct a RS-OWP and “generic” RS-SHCB.
2. We show that RS-OWPs and related-secret *pseudorandom bits* yield related-secret pseudorandom generators (RS-PRG’s).
3. We show that RS-PRGs can be used to construct RS-PRFs.
4. We show that RS-PRFs can be used to construct RS-PRPs.

This brings us very close to being able to prove that related secret one way functions imply related-secret pseudorandom permutations. A positive step that we are not able to take is showing a related-secret *pseudorandom bit* exists. In the normal attack model, all hardcore bits are pseudorandom. [2, 12] However, in the related-secret attack model we find that related-secret hardcore bits do not imply the existence of related-secret pseudorandom bits and we give evidence that constructing related-secret pseudorandom bits is impossible. We also extend this result to show a connection between related-secret hardcore bits and locally decodable codes. As the idea of a pseudorandom bit is instrumental in extending the notion of one way primitives to pseudorandom primitives in the normal attack model [16, 12] these results demonstrate a significant difficulty in building primitives which are pseudorandom in a related key attack model.

2 Definitions

If $f : \mathbb{N} \rightarrow \mathbb{R}$ 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. Matrices will be denoted by bold upper-case letters \mathbf{X} . We denote the j 'th bit of x as x_j and the i 'th through the j 'th bit of x as $x_{i,\dots,j}$. For a matrix \mathbf{X} we denote the i 'th row of \mathbf{X} as \mathbf{X}_i , the j 'th column as \mathbf{X}^j and the (i, j) 'th element as \mathbf{X}_i^j . For two inputs x, r we denote the inner product $(\sum_{i=0}^k x_i r_i)$ of x and r as $\langle x, r \rangle$.

2.1 Related-secret security

The notion of related-secret security is that primitives maintain their security even under an adversary which can exploit possible relations between the secret input and the output of the primitive. To formalize this idea, we allow adversaries to launch “related-secret attacks”. Informally, a related-secret attack allows an adversary to not only be able to query a function f on a secret x , but to

also perturb the secret x for some δ where $\delta : \mathcal{D}_f \rightarrow \mathcal{D}_f$ and receive $f(\delta(x))$. We denote the set of all perturbations δ we allow the adversary to use as Δ . For a function $f(x)$ that takes secret input x we denote $F_{f,x}$ as the oracle which allows an adversary to launch related-secret attacks. $F_{f,x}$ takes $\delta \in \Delta$ as input and returns $f(\delta(x))$. If a function f is related-secret secure for some set Δ we say that f is “secure under Δ ”.

While we might hope to design related-secret primitives for any possible set of perturbations / relations, it has been shown in [3] that related-secret security is impossible for related-key blockciphers and pseudorandom functions where Δ is an “invalid set” of perturbations.

Definition 2.1 (Invalid Sets) *Let the length of the secret input to a function be $\{0, 1\}^n$. We say a set of functions Δ is valid if it satisfies the following two properties which we restate from [3].*

- *Output Unpredictable: Δ is output unpredictable if $\forall S \subset \Delta, \forall X \subset \{0, 1\}^n, Pr[x \leftarrow \{0, 1\}^n; \{\delta(x) : \delta \in S\} \cap X \neq \emptyset]$ is negligible as long as $|X|$ and $|S|$ are polynomial in n .*
- *Collision Resistant: Δ is collision resistant if $\forall S \subset \Delta, Pr[x \leftarrow \{0, 1\}^n; |\{\delta(x) : \delta \in S\}| < |S|]$ is negligible when $|S|$ is polynomial in n .*

Since it is impossible to design certain related-secret primitives for unlimited Δ , we assume that Δ is at least a valid set of perturbations. In addition, while we do not require that Δ is a group, we require that Δ has a minimal level of structure: we assume it is closed under function composition and that the identity perturbation $\delta_{ident} \in \Delta$.¹

We say that Δ is *complete* in that $\forall x, y \in \mathcal{D}_f, \exists \delta : \delta(x) = y$. Note that related-key or related-secret security against an *incomplete* Δ is a far easier problem [18].

Two standard examples of Δ classes that meet this definition:

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

In both cases we identify the function δ_c with the value c . We do not require that every Δ be a set of permutations, although this is the case for the standard examples.

We now give the definitions of a related-secret secure one way function, one way permutation and pseudorandom generator. We also give the normal definitions of these primitives to demonstrate the differences between the related-secret attack model and the normal attack model.

Definition 2.2 (One-way function) *An efficiently computable function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is considered a one-way function if for all $A \in PPT$, exists negligible ν such that $\forall k Pr[x \leftarrow \{0, 1\}^k; x' \leftarrow A(f(x)) : f(x') = f(x)] \leq \nu(k)$.*

Definition 2.3 (One-way permutation) *A one way permutation is a function p which is one way, and is a permutation on $\{0, 1\}^k$ for all k .*

Definition 2.4 (Related-secret secure one way function (RS-OWF)) *A function f is considered a related-secret one way function secure under Δ if $\forall A \in PPT, Pr[x \leftarrow \{0, 1\}^*; x' \leftarrow A^{F_{f,x}} : f(x) = f(x')] \leq \nu(k)$ for a negligible function ν , where $|x| = k$.*

¹For 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 2.5 (Related-secret secure one way permutation (RS-OWP)) *The definition of a RS-OWP is identical to the definition of a RS-OWF, except we require that for all k , f is a permutation on $\{0, 1\}^k$.*

Definition 2.6 (Related-secret secure pseudorandom generator (RS-PRG)) *A function $g(x)$ which takes n bits to $l(n)$ bits, $l(n) > n$, is a related-secret secure pseudorandom generator secure under Δ if $\forall A \in PPT$:*

$$\Pr[x \leftarrow \{0, 1\}^n; A^{G_x(\delta)} = 1] - \Pr[x \leftarrow \{0, 1\}^n; A^{\mathcal{O}_x(\delta)} = 1] \leq \nu(n)$$

where G_x returns $g(\delta(x))$ on input δ and \mathcal{O}_x returns a random string from $\{0, 1\}^{l(n)}$ on input δ .

The definitions of a related-secret secure pseudorandom function (RS-PRF) and related-secret secure pseudorandom permutation (RS-PRP) can be found in [3].

2.2 Hard-core bits

A major technique in constructing pseudorandom primitives from hard problems has been the idea of the *hard-core bit*:

Definition 2.7 (Hard-core bit) *A function $B(x) \{0, 1\}^* \rightarrow \{0, 1\}$ is considered hard-core for a function $f \{0, 1\}^* \rightarrow \{0, 1\}^*$ if:*

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

Normally, a function $B(x)$ is considered a hard-core bit for a function f via a reduction from the security of B to some hard problem, usually the one way security of f . It is also possible for a bit to be hard-core in a trivial sense. Consider $f'(x) = f(x_2, \dots, x_k)$ where f is a one way function. It is clear that x_1 is a hard-core bit, however as it is hard-core due to information loss it can never be recovered with probability bounded away from $\frac{1}{2}$. Such a hard-core bit for a function f does not imply that f is a one-way function. Note, however, that a *permutation* with a hard-core bit is necessarily one-way.

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

Definition 2.8 (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 $\exists M \in PPT$: if $\exists A \in PPT$ such that $\forall k \Pr[x \leftarrow \{0, 1\}^k; b \leftarrow A(f(x)) : b = B(x)] \geq \frac{1}{2} + \epsilon(k)$ then, $\Pr[x \leftarrow \{0, 1\}^k; x' \leftarrow M^A(f(x)) : f(x) = f(x')] \geq \epsilon'(k)$ for non-negligible $\epsilon(k), \epsilon'(k)$.*

We will also require that the machine M outputs x with probability 1 when $A(f(x))$ returns $B(x)$ with probability 1. We note that a bit may be a strong hard-core bit for a non one-way function, however as such bits are easy to find, we do not consider them.

We now define the notions of a related-secret secure hard-core bit and a related-secret secure strong hard-core bit:

Definition 2.9 (Related-secret secure hard core bit (RS-HCB)) *A function $B(x)$ is a related-secret secure hard core bit for a function f , under Δ , if $\forall A \in PPT, \forall k, \Pr[x \leftarrow \{0, 1\}^k; b \leftarrow A^{F_{f,x}} : b = B(x)] \geq \frac{1}{2} + \epsilon(k)$.*

Definition 2.10 (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 , under Δ , if $\exists M \in PPT$ such that if $\exists A$ where $\forall k \Pr[x \leftarrow \{0, 1\}^k; b \leftarrow A^{F_{f,x}} : b = B(x)] \geq \frac{1}{2} + \epsilon(k)$ then $\Pr[x \leftarrow \{0, 1\}^k; x' \leftarrow M^{A, F_{f,x}} : f(x) = f(x')] \geq \epsilon'(k)$ for non-negligible ϵ, ϵ' .

We finally introduce the idea of a related-secret secure pseudorandom bit or RS-PRB. This is function $B(x)$ where maintains the pseudorandomness of its output, even given a related-secret attack.

Definition 2.11 (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 $\forall A \in PPT, \forall k$

$$\Pr[x \leftarrow \{0, 1\}^k; A^{B_x, F_{f,x}} = 1] - \Pr[x \leftarrow \{0, 1\}^k; A^{\mathcal{R}_x, F_{f,x}}(f(x)) = 1] \leq \nu(k)$$

where B_x on input δ returns $B(\delta(x))$, \mathcal{R}_x on input δ returns a random bit which it associates with that δ .

It is easy to show that all RS-HCB's are HCB's, and that all RS-PRB's must be RS-HCB's. It is also trivial to show that all HCB's are pseudorandom. In addition we note that most known HCB's are proven via a reduction between their security and the one-way security of a function f , thus making them hard-core bits. In this paper we will demonstrate that a wide class of SHCB's are not RS-PRB's. As all RS-PRB's must be HCB's this demonstrates a severe difficulty in constructing RS-PRB's.

2.3 Black token algorithms

In this paper we introduce the idea of a *black token* algorithm/ the *black token* model of computation. The black token model of computation is based off of the ideas of *algebraic reductions* and *generic ring algorithms* [22, 7, 1, 19]. Informally, a reduction R is algebraic/generic for a group/ring \mathbb{G} if:

- $\forall g \in \mathbb{G}$, R only performs group/ring operations on g and computes certain predetermined predicates.
- For all any $g \in \mathbb{G}$ that is produced by R , we have a way of finding how R arrives at g from the inputs to R and R 's random tape.

The main differences between algebraic reductions and the black token model of computation is that in the black token model we do not require that each piece of data forms a ring or group, and we allow R to perform non-ring operations on its data.

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).

There may be 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 receives two inputs, a pseudonym id_x and a function δ , and returns an output 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, $(T^{A_{BT}})^O(x)$ is the machine that runs A_{BT}^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 .

Certain algebraic reductions may be thought of as black-token ones, if the actual values of any group elements are ignored. Effectively, we consider all group elements to be private values, and provide access to the group operation through P . We are unaware of any algebraic reductions that cannot be made to be black token reductions. Black token reductions, although they seem to have strong restrictions, are in fact the norm: most reductions, if examined carefully, manipulate a class of private values in only basic ways, blind to the actual value. As such the reduction can be thought of as black-token.

3 Positive results

In this section we state some positive results regarding the relationships between related-secret secure primitives. We construct a RS-OWP secure under Δ_+ , a RS-HCB for any function secure under Δ_{\oplus} , we construct an RS-PRG from a RS-PRB, and we finally show that RS-PRG’s imply both RS-PRF’s and RS-PRP’s. This gets us very close to an explicit construction of an RS-PRP. The major gap in this is the construction of an RS-PRB, which we address in the next section.

Theorem 3.1 *Let f be defined as $f(x) = g^x \pmod p$ for prime p and where g is the generator for \mathbb{Z}_p^* . Under the discrete log assumption f is a RS-OWP secure under Δ_+ .*

Proof. We know that $g^x \pmod p$ is a one way permutation under the discrete log assumption [11]. To show that f is a RS-OWP we give a reduction from the related-secret security of f to the one way security of f . Let A be the adversary capable of breaking the related-secret security of f and let A' be the adversary capable of breaking the one way security of f . Since $f(x + x') = g^{x+x'} \pmod p = g^x g^{x'} \pmod p = f(x)f(x')$, A' is capable of answering all queries $F_{f,x}(\delta_c)$ made by A by returning $f(x)f(c)$ when A queries $F_{f,x}(\delta_c)$ for $\delta_c \in \Delta_+$. Since A' provides a perfect simulation of $F_{f,x}$, A' succeeds whenever $A^{F_{f,x}}$ succeeds. \square

Before giving a construction of an RS-SHCB for RS-OWF secure under Δ_{\oplus} we state the following theorem from Goldreich and Levin [12].

Theorem 3.2 (Goldreich Levin Hard-Core Bit) *Let $f(x)$ be a one-way function. Let $f'(x, r) = (f(x), r)$. Then $B(x, r) = \langle x, r \rangle$ is a hard core bit for $f'(x, r)$.*

To prove Theorem 3.2, Goldreich and Levin prove the following theorem which demonstrates that the function $B(x, r)$ as defined in Theorem 3.2 is a strong hard-core bit in accordance with Definition 2.8:

Theorem 3.3 *Let $A_x()$ be an adversary which outputs a single bit where for some fixed $x \leftarrow \{0, 1\}^k$ and for any $R \leftarrow \{0, 1\}^k$ we have that*

$$|Pr[A_x(R) = \langle x, R \rangle] - Pr[A_x(R) \neq \langle x, R \rangle]| \geq \epsilon(k)$$

Then there is a machine M such that $M^{A_x} \rightarrow x$ with non-negligible probability.

We can now give a construction of a RS-SHCB for any RS-OWF secure under Δ_{\oplus} .

Theorem 3.4 *Let f be any function. Let $f'(x, r) = f(x), r$. Then $B(x, r) = \langle x, r \rangle$ is a RS-SHCB for f under Δ_{\oplus} .*

Proof. We describe the reduction from recovering $B(x)$ under a related-secret attack to inverting f under a related-secret attack.

Assume there exists an adversary A such that $A^{F^{f', (x, r)}}$ returns $\langle x, r \rangle$ with probability non-negligibly bounded away from $\frac{1}{2}$. As such, for any x we can build the machine $R^{F^{f, x}}$ which will attempt to break the related-secret one way security of f . Dividing all $\delta \in \Delta$ into the strings which apply to x and r , $\delta_{x, r} = \delta_x || \delta_r$, $R^{F^{f, x}}$ can simulate the machine $F_{f', (x, r_i)}$ for any r_i as $F_{f', (x, r_i)}(\delta_{x, r}) = F_{f, x}(\delta_x), \delta_r(r_i)$. As such, $R^{F^{f, x}}$ can run $A^{F^{f', (x, r_i)}}$ and as such find $B(x, r_i) = \langle x, r_i \rangle$ with probability bounded away from $\frac{1}{2}$ for any value r_i . R can then use the machine M defined in Theorem 3.3 to find x with non-negligible probability. \square

We now have a RS-SHCB secure under Δ_{\oplus} and a RS-OWF secure under Δ_+ . We next give implicit constructions of RS-PRG's, RS-PRF's and RS-PRP's. We begin by demonstrating that RS-OWF's and RS-PRB's secure under Δ , imply RS-PRG's secure under Δ^n .

Theorem 3.5 *Let p be a n -bit RS-OWF secure under a valid set Δ , let $g'(x)$ an n bit to $n^2 + 1$ bit pseudorandom generator and let $B(x)$ be a RS-PRB for p . We define $g: \{0, 1\}^{n^2} \rightarrow \{0, 1\}^{n^2+1}$ as the function which takes an input x , parses x as n n -bit blocks x_1, \dots, x_n , computes $y = B(x_1), \dots, B(x_n)$ and outputs $g'(y)$. Then g is an RS-PRG for Δ^n , where $(\delta_1, \dots, \delta_n)(x_1 || \dots || x_n) = \delta_1(x_1) || \dots || \delta_n(x_n)$.*

Proof. 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. \square

This proof illustrates an important point, namely that sometimes we only need to use a related-secret secure primitive at the beginning of a computation, and then can utilize a normal primitive while maintaining security.

We state the next lemma, whose proof comes directly from the definition of an RS-PRG. The proof is obvious.

Lemma 3.6 *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 $\Delta \rightarrow \{0, 1\}^{p(n)}$.*

We now show that RS-PRG's imply RS-PRF's, and RS-PRF's imply RS-PRP's.

Theorem 3.7 *Let g' be an n bit to $2n$ bit function. Define $f_y(x)$ for n bit string y as the function where:*

- $f_y(\epsilon) = y$ for empty string ϵ .
- $f_y(s0) = g'(f_y(s))_{1, \dots, n}$ for any string s .
- $f_y(s1) = g'(f_y(s))_{n+1, \dots, 2n}$ for any string s .

Let F_K be the family of all functions f_K . Let g' be a n bit to $2n$ bit pseudorandom generator. Let $g()$ be a l bit to n bit RS-PRG. Then $F_{g(K)}(x)$ is a pseudorandom function.

Proof. From [13] we know that $F_K(x)$ is a PRF for random K and PRG g' . As such, $F_K(x)$ is indistinguishable from a random family of functions, \mathcal{F} . Since g is an RS-PRG, each $g(\delta^i(K))$ for adversarially chosen δ^i and random K is an independent pseudorandom value. Thus $F_{g(\delta^i(K))}(x)$ is indistinguishable from $F_{K_i}(x)$ for a randomly selected K_i , even for adversarially selected δ^i . By a simple hybrid argument any adversary successfully attacking $F_{g(K)}(x)$ can be reduced to one attacking $F_{K_i}(x)$ for a random, independent K_i which by [13] is shown to be hard. As such, $F_{g(K)}(x)$ is indistinguishable from a family of random functions, even under related-secret attacks. \square

This proof illustrates an easy way to gain related key security for many primitives. We use a related-secret pseudorandom generator to eliminate any advantage an adversary may gain from a related-secret attack as $g(\delta^i(K))$ is indistinguishable from random, even given other $g(\delta^j(K))$. As such, the adversary's choice of δ has no effect as it gets translated to a key that looks random and independent of other keys. This leads us to the following corollary:

Corollary 3.8 *Let $E(x, K)$ be a pseudorandom permutation family. Let g be a RS-PRG. Then $E(x, g(K))$ is a RS-PRP.*

In addition to Corollary 3.8, similar statements can be made with regards to related key MAC's and other similar primitives.

We finally we give a theorem showing that the notion(s) of an RS-PRF and RS-PRP imply an RS-PRG.

Theorem 3.9 *Let $F_K(x)$ be an RS-PRF under Δ , and let $E(x, K)$ be an RS-PRP under Δ . Let $g(x)$ be the function which parses x as x', K and outputs $F_K(x), F_K(x+1), F_K(x+2)$. Let $g'(x)$ be the function which parses x as x', K and outputs $E(x, K), E(x+1, K), E(x+2, K)$. Then for any Δ' , $g(x)$ and $g'(x)$ are RS-PRG's under $\Delta' \times \Delta$.*

Proof. The proof follows from the fact that $g(x)$ and $g'(x)$ are both simulatable given access to oracles F_X and $E(x, K)$ respectively. As such, if an adversary A can distinguish between g or g' and random values, we can build an adversary A' which picks a random x , queries its F or 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 function / permutation then A simulates g, g' otherwise A' simply returns a random value for each δ_{12} . As such, A' can use A to distinguish between the two cases and thus distinguish between the pseudorandom function / permutation and random oracle. \square

4 Impossibility results

In the past section we gave many constructions of related-secret secure primitives, however we did not give a construction of an RS-PRB. In this section we give a surprising result concerning possible constructions of RS-PRB's, in that we show that most black token SHCB's, (SHCB's where the machine M works in the black token model), for a one way permutation cannot be an RS-PRB.

Theorem 4.1 *Let $B(x)$ be a SHCB for a one way function f where the reduction M is a black token algorithm with private operation oracle P such that $P(id_{f(x)}, \delta) = id_{f(\delta(x))}$ for $\delta \in \Delta$ and $P(id_{f(\delta(x))}, g) = g(\delta(x))$ where $g(\delta(x))$ is polynomial time computable given $f(\delta(x))$. Then $B(x)$ is not an RS-PRB under any $\Delta' \supset \Delta$.*

Proof. If $B(x)$ is a black token SHCB for a one-way permutation f , then there exists a black token algorithm M such that $M^{P,A}(f(x))$ finds x with non-negligible probability as long as $A(f(x))$ 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' . A' has access to two oracles, O and $F_{f,x}$. First A' queries $F_{f,x}(\delta_{ident})$ to obtain $f(x)$, which it tokenizes and then uses as input to M . A' runs M , tokenizing its values, using O to answer A queries, using $F_{f,x}$ to answer P queries as well as finding $f(\delta(x))$ to compute $g(\delta(x))$. When M returns x' we check if $f(x') = f(x)$; if so, we output 1, otherwise we output 0.

Since M is black token it only obtains 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 $y = f(\delta(x))$. A' keeps track of this association, and when asked for $A(id_{f(\delta(x))})$, queries $O(\delta)$. Queries to P of the form $id_{f(\delta(x))}, \delta'$ are answered by querying $F_{f,x}(\delta' \circ \delta)$, and returning the pseudonym of the result. Queries to P of the form $id_{f(\delta(x))}, g$ are answered by querying $F_{f,x}(\delta)$, obtaining $f(\delta(x))$ and computing $g(\delta(x))$, as we require that $g(\delta(x))$ be polynomial time computable given $f(\delta(x))$.

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

We have shown that a black token SHCB cannot be an RS-PRB. We further show that all RS-PRBs are HCBs:

Theorem 4.2 *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 $A(f(x))$ returns $B(x)$ with probability $1/2 + \epsilon$. Then we can attack B as an RS-PRB by obtaining $f(x)$ and $O(x)$ and checking whether $O(x) = A(f(x))$; 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. \square

4.1 Discussion

Put together, these two theorems *nearly* represent a cycle of contradiction that would disprove the existence of RS-PRBs altogether. Because the conditions do not match exactly, a construction of an RS-PRB may be possible. The key question is, if B is an RS-PRB, is B an SHCB?

- If B is an SHCB, then it must be an SHCB that does not meet the conditions of Theorem 4.1. That is, there must *exist* no black-token reductions of the type specified in the theorem.

We only make the most general restrictions on the types of reductions in Theorem 4.1, the main limitations being that we require that M only ask for the hardcore bit of a $\delta(x)$ where we know δ and that M operates in a black token manner. All known examples of strong hard-core bits have black-token reductions of the type necessary for our impossibility proof. See Appendix A for some specific discussion of examples; see also [7, 22] for similar claims regarding the different but related notion of algebraic reductions.

At a higher level, (1) since no assumptions can be made about observable properties of $f(x)$, these values are generally ignored in any proof involving OWF's or OWP's. (2) "Private operations" in these proofs are kept to a strict minimum because they must be efficiently computable without help. (3) The format restriction is adequate for known examples; it captures any one-input functional transformations of f outputs and any information about x that can be computed from $f(x)$. (4) Even if the requirements of a black token reduction and the format restrictions are objectionable, note that our impossibility proof only requires that the reduction be successful when A is successful with probability 1. (5) 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.

- If B is not an SHCB, it is still a HCB, and if there is a proof that B is a RS-PRB, then there is a proof reducing from an adversary that breaks B as a HCB to some assumption, which must be *other* than the hardness of inverting f .

We note that our impossibility proof can easily be extended to any SHCB-like property where the problem solved (rather than inverting f) can be checked with access only to $F_{f,x}$, and which is assumed to be hard. Thus, such a RS-PRB would have to be based on a non-checkable assumption (for instance, predicting a small portion of bits of x , or solving DDH when the hardness of f is based on discrete log). The key issue is, what assumption can be used if the hardness of inverting f cannot?

Furthermore, we are at a loss as to how to construct higher forms of related-secret pseudorandomness, except via pseudorandom bits. As an example, we show that RS-PRP's and RS-PRF's are equivalent to the notion of an RS-PRG, and known constructions of RS-PRG's and indeed most known construction of PRG's in general rely on the existence of pseudorandom bits. The only explicit related-secret secure constructions of pseudorandom primitives we are aware of are those of Lucks, who admits to relying on non-standard assumptions, in particular, assumptions that inherently provide related-value pseudorandomness.

5 Relationship to coding theory

We begin this section by describing the notions of error correcting, locally decodable, and strong locally decodable [21, 10] codes over $\{0, 1\}^*$. Let \mathcal{M} be a metric space, and let $\|x - y\|$ be the distance between $x, y \in \mathcal{M}$.

An *error-correcting code* is a code $\mathcal{C} \subset \mathcal{M}$ along with a triple of algorithms (C, C^{-1}, D) where [8, 9]:

1. C is the encoding function which takes elements of a domain of size 2^k to \mathcal{C}
2. C^{-1} is the inverse of C
3. $\exists t : \forall m \in \mathcal{C}, \forall m' : \|m' - m\| \leq t, D(m') = m$.

We refer to the tuple $(\mathcal{C}, C, C^{-1}, D)$ as an n, k, t error correcting code.

A $[n, k, d]$ linear code is a code where \mathcal{C} is a linear subspace of $\mathcal{M} = \mathbb{F}_p^n$ which has dimension $k < n$, and minimum distance d . As such $C(x) = \mathbf{C}x$ for a *generator matrix* \mathbf{C} .

Definition 5.1 An (l, ρ, p) locally decodable code is a triple of algorithms C, C^{-1}, R where $C\{0, 1\}^k \rightarrow \mathcal{C} \subset \{0, 1\}^n$, $C^{-1} \mathcal{C} \rightarrow \{0, 1\}^k$ and R has the property that if $x \in \{0, 1\}^k$ is the message, and y is the corrupted encoding of x , then $\Pr[R^Y(b) = x_b] \geq p$ as long as $\|C(x) - y\| \leq \rho n$ where Y is the oracle which on input i returns y_i and R only queries Y l times.

Definition 5.2 An (l, ρ, p) strong locally decodable code is a triple of algorithms C, C^{-1}, R where $C\{0, 1\}^k \rightarrow \mathcal{C} \subset \{0, 1\}^n$, $C^{-1} \mathcal{C} \rightarrow \{0, 1\}^k$ and R has the property that if $x \in \{0, 1\}^k$ is the message, and y is the corrupted encoding of x , then $\Pr[R^Y = x] \geq p$ as long as $\|C(x) - y\| \leq \rho n$ where Y is the oracle which on input i returns y_i and R only queries Y l times.

The machine R defined above is very similar to the reduction M defined for SHCB's. In fact, the machine M defined in Theorem 3.3 is basically the reconstruction algorithm R for the Hadamard code, which is a strong locally decodable code. In this section we extend this notion, demonstrating a relationship between generic RS-SHCB's and strong locally decodable codes. We show that linear strong locally decodable codes imply RS-SHCB's, and certain black token RS-SHCB's imply the existence of strong locally decodable codes.

Theorem 5.3 Let C, C^{-1}, R be a (l, ρ, p) linear strong locally decodable code over \mathbb{F}_2^n . Define $B(x, r)$ as $C(x)_r$, the r 'th bit of $C(x)$. Let p be an RS-OWP secure under Δ_{\oplus} . Define $p'(x, r)$ as $p(x), r$. $B(x, r)$ is an RS-SHCB for p' .

Proof. To prove that $B(x, r)$ is an RS-SHCB for p' we need to create a machine $M^{F_{p', (x, r)}, A}$ which will find x as long as $A^{F_{p', (x, r)}}$ returns $B(x, r)$. We will build $M^{F_{p', (x, r)}}$ 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 $B(x, r) = C(x)_r$ with probability $\frac{1}{2} + \epsilon$ for non-negligible ϵ . A difficulty in this proof is that $1 - \rho$, the probability that R requires $B(x, r) = C(x)_r$ to be correct, will often be much larger than $\frac{1}{2} + \epsilon$, the probability that $A^{F_{p', (x, r)}}$ 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 $F_{p', (x, r)}$ can simulate $F_{p', (\delta_c(x), r)}$ for random δ_c as $F_{p', (\delta_c(x), r)}(\delta, \delta_{c'}) = F_{p', (x, r)}(\delta\delta_c, \delta_{c'})$ When $A^{F_{p', (\delta_c(x), r)}}$ returns its guess at g at $B(\delta_c(x), r)$, M can compute $B(x, r) = g \oplus C(c)_r$ 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 $M^{F_x, A}$ to find many independent votes for $B(x, r)$, where each individual vote is correct with non-negligible probability. As such, M can find $C(x)_r = B(x, r)$ with high enough probability to simulate Y and thus run the reconstruction program R and recover x . \square

As most known strong locally decodable codes are linear, this suggests that strong locally decodable codes in general imply a RS-HCB for any p secure under Δ_{\oplus} .

We now show that black token RS-SHCB's can be used to construct a strong locally decodable code.

Theorem 5.4 *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 $P(id_{f(\delta(x))}, \delta')$. Establish an ordering on Δ , $\delta_1, \delta_2, \dots, \delta_{|\Delta|}$. Define $C(x)_i$ as $B(\delta_i(x))$. Then we can create a C^{-1} and a R such that C, C^{-1}, R is a (l, ρ, p) strong locally 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 $P, F_{f,x}$ and A . The simulations of $F_{f,x}$ 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 $F_{f,x}(\delta)$ or $P(id_{f(\delta(x))}, \delta')$, R returns a randomly generated token which it associates with $id_{f(\delta(x))} / id_{f(\delta'(x))}$. When M makes a query $A(id_{f(\delta_i(x))})$, R^Y simulates A by querying Y for $B(\delta_i(x)) = C(x)_i$, 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 $M^{A, F_{f,x}}$ will output x with probability p .

The construction of C^{-1} is identical as we require that M output x with probability 1 when A is always correct. Since $C^{-1}(C(x))$ can simulate such a perfect A , we can use the previous construction to find x with probability 1. \square

6 Conclusion

In this paper we extend the idea of related key security to the notion of related secret secure primitives. We analyze the relationships between these primitives. In attempting to translate the techniques in the normal model into the related secret setting, we observe an important difference between hardcore bits (which are hard to *learn*) and pseudorandom bits (which, when seen, are hard to *verify*). There is no distinction between these notions in the normal setting, but in the related secret setting, we show a troubling separation result. Specifically, we show that hard-core bits with well-behaved “black token” proofs cannot be pseudorandom bits in the related secret setting, yet pseudorandom bits are always hard-core bits. Since RS-PRGs are equivalent to RS-PRFs and RS-PRPs, and since the only way we know to construct PRGs is from PRBs, this presents a substantial obstacle to the creation of related-secret pseudorandomness from basic cryptographic primitives.

References

- [1] Divesh Aggarwal and Ueli Maurer. Breaking rsa generically is equivalent to factoring. Cryptology ePrint Archive, Report 2008/260, 2008. <http://eprint.iacr.org/>.

- [2] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.
- [3] M. Bellare and T. Kohno. A Theoretical Treatment of Related-Key Attacks: PKA-PRPs, RKA-PRFs, and Applications. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT ’03*, volume 2656 of LNCS, pages 491–506, 2003.
- [4] Eli Biham. New types of cryptanalytic attacks using related keys. *Journal of Cryptology*, 7(4):229–246, Fall 1994. Also available at: citeseer.nj.nec.com/biham94new.html.
- [5] J. Black, M. Cochran, and T. Shrimpton. On The Impossibility of Highly-Efficient Blockcipher-Based Hash Functions. In *Advances in Cryptology – Eurocrypt 2005*, volume 3494 of LNCS, pages 526–541. Springer Verlag, May 2005.
- [6] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–863, November 1984.
- [7] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In *Advances in Cryptology - EUROCRYPT 1998*, 1998.
- [8] Xavier Boyen. Reusable cryptographic fuzzy extractors. In *ACM Conference on Computer and Communications Security—CCS 2004*, pages 82–91. New-York: ACM Press, 2004.
- [9] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *SIAM Journal on Computing*, volume 38, pages 523–540, 2008.
- [10] Zeev Dvir and Amir Shpilka. Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. In *STOC ’05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 592–601, New York, NY, USA, 2005. ACM.
- [11] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, Cambridge, the United Kingdom, 2001.
- [12] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, 15–17 May 1989.
- [13] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.
- [14] Shafi Goldwasser, Silvio Micali, and Po Tong. 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*, pages 134–144, Washington, DC, USA, 1982. IEEE Computer Society.
- [15] Michael Gorski and Stefan Lucks. New related-key boomerang attacks on AES. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT*, volume 5365 of *Lecture Notes in Computer Science*, pages 266–278. Springer, 2008.

- [16] J. Håstad, R. Impagliazzo, L.A. Levin, and M. Luby. Construction of pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
- [17] John Kelsey, Bruce Schneier, and David Wagner. Related-key cryptanalysis of 3-way, bihamdes, cast, des-x, newdes, rc2, and tea. In *ICICS '97: Proceedings of the First International Conference on Information and Communication Security*, pages 233–246, London, UK, 1997. Springer-Verlag.
- [18] Stefan Lucks. Ciphers secure against related-key attacks. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 359–370. Springer, 2004.
- [19] Ueli Maurer. Abstract models of computation in cryptography. In Nigel Smart, editor, *Cryptography and Coding 2005*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, December 2005.
- [20] Darakhshan J. Mir and Poorvi L. Vora. Related-key statistical cryptanalysis. *Cryptology ePrint Archive*, Report 2007/227, 2007. <http://eprint.iacr.org/>.
- [21] Rafail Ostrovsky, Omkant Pandey, and Amit Sahai. Private locally decodable codes. In *Lecture Notes in Computer Science: Automata, Languages and Programming*, pages 387 – 398, 2007.
- [22] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *Advances in Cryptology - ASIACRYPT 2005*, pages 1–20, 2005.
- [23] Darakhshan J. Mir Poorvi L. Vora. Related-key linear cryptanalysis. In *2006 IEEE International Symposium on Information Theory*, pages 1609 – 1613, 2006.
- [24] Wentao Zhang, Lei Zhang, Wenling Wu, and Dengguo Feng. Related-key differential-linear attacks on reduced aes-192. In *Progress in Cryptology - INDOCRYPT 2007*, 2007.

A Black token reductions for known hardcore bits

In this section we examine the three well-known hardcore bits, the hardcore bits for RSA, discrete log, and the generic hardcore bit of Goldreich and Levin. For each one we demonstrate that these hardcore bits are black token hardcore bits that fit the requirements of Theorem 4.1. 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 begin with the generic hardcore bit of Goldreich and Levin.

Theorem A.1 *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' , where f outputs are the private values.*

Proof. The machine M is simple to construct. M begins by receiving $id_{f(x)}, r$. M never makes use of P ; it simply queries A on $id_{f(x)}, r_i$ for different r_i until it can obtain x via standard linear algebra.

□

Note that this reduction, which is not black token overall, is black token enough to be applicable in the argument of Theorem 4.1.

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.

We next address the hardcore bit for the RSA function. The reduction is taken from [14].

Theorem A.2 *Let $\text{RSA}_{N,e}$ be the RSA function, that is $\text{RSA}_{N,e}(x) = x^e \pmod N$. Define $B(x)$ as the parity of x . Then $B(x)$ is a black token SHCB for $\text{RSA}_{N,e}$*

Proof. P allows two transformations, $\delta_{\frac{1}{2}}(x) = x2^{-1} \pmod N$ and $\delta_{-1}(x) = -x \pmod N$. These can be viewed as two specific transformation among a more general class of multiplicative transformations. 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. \square

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 finally address the hardcore bit for the discrete log function. The reduction is taken from [6]

Theorem A.3 *Let $\text{Log}_{g,p}$ be the discrete log 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 hardcore bit for $\text{Log}_{g,p}$.*

Proof. P computes several transformations, $\delta_{+1}(x) = x + 1$, $\delta_{\frac{1}{2}}(x)$ which returns $\frac{x}{2}$ and $\frac{x}{2} + \frac{p-1}{2}$ and $p(x)$ a predicate which returns the least significant bit of x . Each transformation is polynomial time computable given g^x values as $g^{\delta_{+1}(x)} = gg^x$, $g^{\delta_{\frac{1}{2}}(x)}$ is the square root of g^x , and $p(x)$ is testing to see if g^x is a quadratic residue mod p .

M proceeds as follows. It first obtains id_y for $y = g^x \pmod p$. 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, M considers $y' = y$ and $id_y = id_{y'}$. M then makes a query to $P(id_{y'}, \delta_{\frac{1}{2}})$ obtaining pseudonyms for the two square roots, g^s and $g^{s+\frac{p-1}{2}}$ where $g^{2s} = y'$. M then sends both pseudonyms to A which enables him to find the pseudonym for g^s .

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 . \square

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 homomorphism; 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.