The Generalized Randomized Iterate and its Application to New Efficient Constructions of UOWHFs from Regular One-Way Functions

Scott Ames[1], Rosario Gennaro[2], and Muthuramakrishnan Venkitasubramaniam[1]

[1] University of Rochester, Rochester, NY 14611, USA
[2] IBM T.J.Watson Research Center, Hawthore, NY 10532, USA

Abstract. This paper presents the *Generalized Randomized Iterate* of a (regular) one-way function f and show that it can be used to build Universal One-Way Hash Function (UOWHF) families with $O(n^2)$ key length.
We then show that Shoup's technique for UOWHF domain extension can be used to improve the efficiency of the previous construction. We present the *Reusable Generalized Randomized Iterate* which consists of $k \geq n+1$ iterations of a regular one-way function composed at each iteration with a pairwise independent hash function, where we only use $\log k$ such hash functions, and we "schedule" them according to the same scheduling of Shoup's domain extension technique. The end result is a UOWHF construction from regular one-way functions with an $O(n \log n)$ key. These are the first such efficient constructions of UOWHF from regular one-way functions of unknown regularity.
Finally we show that the Shoup's domain extension technique can also be used in lieu of derandomization techniques to improve the efficiency of PRGs and of hardness amplification constructions for regular one-way functions.

1 Introduction

One of the central results in Modern Cryptography is that one-way functions imply digital signatures (as defined in [6]). This result was first established by Naor and Yung in [12] for one-way permutations via the notion of Universal One-Way Hash Functions (UOWHF). Later Rompel in [13] proved that UOWHFs can be built from any one-way function. The notion of UOWHF is interesting on its own, apart from its connection to digital signatures. UOWHFs are *compressing* functions (i.e. the output is shorter than the input) which enjoy a *target collision resistance* property: a function family $\mathcal{G}$ is a UOWHF if no efficient adversary A succeeds in the following game with non-negligible probability:

- A chooses a target input z;
- a randomly chosen function $g \in \mathcal{G}$ is selected;
- A finds a *collision* for $g(z)$, i.e. an input $z' \neq z$ such that $g(z) = g(z')$.

A seemingly weaker notion is *second preimage resistance* where the target input z is randomly chosen (rather than by A). It is however well known how to convert a second preimage resistant function family into a UOWHF.

The security of these constructions is proven by *reductions*: given an adversary A that wins the above UOWHF game, we build an "inverter" I that is able to solve a computationally hard problem, e.g. invert a one-way function. A crucial feature of these reductions is their *efficiency*, i.e. the relationship between the running time of D (or A) and I, and the resulting degradation in the security parameters. For the case of UOWHFs one of the most important efficiency measures is the size of the key needed to run the algorithm.

Unfortunately the construction of UOWHFs based on general one-way functions do not fare very well on that front. If n is the security parameter, the original Rompel construction yielded a key of size $\tilde{O}(n^{12})$ which was later improved to $\tilde{O}(n^7)$ by Haitner *et al.* in [8]. Conversely under the much stronger assumption of one-way *permutations* Naor and Yung in [12] achieve linear key size. Apart from the above works, we are aware of only one work by De Santis and Yung [2] that constructs UOWHFs from *regular* one-way functions (i.e. functions that have constant size preimages). Their construction achieves $O(n \log n)$ key size but is very complicated and more importantly requires knowledge of the regularity parameter.

We go back to investigating the construction of UOWHFs from regular one-way functions. We obtain a very simple construction with $O(n \log n)$ key size, which does *not* require knowledge of the regularity parameter. These are the first such efficient constructions of UOWHF from regular one-way functions of unknown regularity.

Somewhat surprisingly our UOWHF construction is obtained via a simple "tweak" on a well-known algorithm for pseudo-random number generation from regular one-way functions: the *Randomized Iterate* [4, 7]. Another surprising connection established by this paper is that Shoup's domain extension technique [15] can be used to improve the seed size in *both* the PRG and UOWHF.

MOTIVATION. Collision resistant hashing is an ubiquitous tool in Cryptography and in practice a stronger notion of collision resistance is used where the adversary is given as input just $H \in \mathcal{H}$ and must find z, z' that collide (we will refer to this notion as *full collision resistance* as opposed to the target collision-resistance property enjoyed by UOWHFs).

This is problematic because there is strong evidence that this stronger notion cannot be achieved by assuming just OWFs. Simon [16] proves that there is no black-box construction[3] of a fully collision resistant hash function from one-way permutations. While a non black-box construction based on OWFs remains theoretically possible, such construction would probably be very inefficient, since efficient constructions based on general assumptions seem to be black-box ones.

Furthermore the cryptanalysis of practical and widely adopted supposedly collision-resistant functions have reminded us of the importance of construct-

[3] Informally, a black-box construction accesses the underlying OWF only via input queries, without any knowledge of its internal structure.

ing *efficient* candidates for collision-resistant functions which are also *provably secure*, i.e. have a security reduction to a well established computational hard problem. The above explains why researchers and practitioners alike are looking at UOWHFs to replace full collision resistant hashing in practical applications (such as certifications – see for example the work of Halevi and Krawczyk on randomized hashing [10]).

Current efficient candidates for UOWHFs have either no proof of security or make stronger assumptions than the existence of OWPs[4]. Achieving a truly efficient UOWHF construction based on OWFs would offer practitioners a target collision-resistant function which can be used in practice and gives the peace of mind of a strong security guarantee.

In order to achieve this goal, our construction slightly relaxes the assumption to *regular* OWFs, yielding a dramatic improvement to a $O(n \log n)$ key size. We are following the same approach as [7] for pseudo-random generators: looking at the more limited case of regular OWFs not only to improve the efficiency, but also to explore techniques that might benefit constructions in the general case (which is what happened in the PRG case).

1.1 Our Contribution

We present a new algorithm (we call it the *Generalized Randomized Iterate* GRI) which depending on its parameters can be used to build *either* PRGs or UOWHFs starting from *regular* one-way functions.

First proposed in [4] the original Randomized Iterate construction involves composing the regular one-way function with different n-wise (later improved to simply pair-wise independent in [7]) universal hash functions at each iteration. More specifically if f is a regular one-way function, and $h_1, \dots, h_m$ are pairwise independent hash functions all from $\{0,1\}^n$ to $\{0,1\}^n$, the m^{th} randomized iterate of f using the h_i is defined as $f^k = f \circ h_k \circ f \circ h_{k-1} \circ \dots f \circ h_1 \circ f$. In [4, 7] it is shown that this function is hard to invert at each stage and therefore can be used to construct PRGs in conjunction with a generic hard-core predicate (such as the Goldreich-Levin bit [5]).

We generalize the Randomized Iterate to use compressing pair-wise independent hash functions h_i at each stage. Somewhat surprisingly we then show that the resulting family (see Definition 8) is second-preimage resistant.

Notice that in the above applications the universal hash functions h_i are part of the secret key of the resulting algorithm (the seed for the PRG, the index key for the UOWHF). Therefore it is desirable to have constructions in which the number of functions can be minimized.

The Randomized Iterate PRG construction in [7] has an $O(n^2)$ seed, but it was also shown how an $O(n \log n)$ seed could be achieved by using generic

[4] For example Halevi and Krawczyk in [10] propose a mode of operation for typical hash function such as SHA-1 that creates a UOWHF under an assumption on the compression function which is seemingly stronger than OWF, but somewhat weaker than full collision resistance.

de-randomization techniques. First we point out that this approach does not immediately work in the UOWHF case, as in order to reduce the key size, the de-randomization procedure requires an additional property[5].

We then explore another fascinating and somewhat unexpected connection. We observe that instead of using de-randomization techniques, the structure of the Generalized Randomized Iterate can be improved by using Shoup's domain extension technique for UOWHFs [15]. We define the *Reusable Generalized Randomized Iterate* RGRI : Using Shoup's approach we prove that it is possible to "recycle" some of the hash functions in the Generalized Randomized Iterate, to $O(\log m)$ for m iterations (instead of m). The net result is that we achieve a UOWHF with $O(n \log n)$ key size.

Finally we point out that the RGRI also yields an $O(n \log n)$-seed PRG from regular one-way function, and can be also used for hardness amplification of regular one-way functions, obtaining alternative proofs of results already appearing in [7].

1.2 Comparison with Previous Work

We already mention the previous works on UOWHFs based on general assumptions [12, 13, 8, 2] and how they compare to our work.

As discussed above our UOWHF construction uses in a crucial way tools that were developed for the task of pseudo-random generation. In this sense our work follows the path of recent papers on *inaccessible entropy* [9, 8]. Those beautiful works elegantly show that the known constructions of PRGs and UOWHFs can be interpreted as similar manipulation techniques on different forms of computational entropy (pseudo-entropy for PRGs and inaccessible entropy for UOWHFs). While less general, our work shows a more direct and specific connection: a single algorithm (the Generalized Randomized Iterate) which is sufficiently "flexible" to be used either as a PRG or as a UOWHF.

1.3 Paper Organization

We briefly recall the relevant definitions in Section 2. In Section 3 we introduce the Generalized Randomized Iterate and its Reusable variant; we also prove a main technical Lemma that is at the heart of the efficiency claim for our UOWHF construction which appears in Section 4. We present our alternative constructions of a $O(n \log n)$-seed PRG, and the hardness amplification result in Section 5 (the proofs of these constructions will appear in the full-version). We conclude with some discussions and open problems in Section 6.

[5] The actual de-randomization algorithm (the Nisan-Zuckerman PRG for space-bounded computations) used in [7] has this property, but a generic PRG for space-bounded computation might not.

2 Preliminaries and Definitions

2.1 One-Way Functions

Definition 1 *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. f is* one-way *if for every* PPT *machine A, there exists a negligible function $\nu(\cdot)$ such that*

$$\Pr[x \leftarrow \{0,1\}^n; y = f(x) : A(1^n, y) \in f^{-1}(f(x))] \leq \nu(n)$$

Definition 2 (Regular one-way functions) *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a one-way function. f is regular if there exists a function $\alpha : N \to N$ such that for every $n \in N$ and every $x \in \{0,1\}^n$ we have:*

$$|f^{-1}(f(x))| = \alpha(n)$$

We assume that the regularity $\alpha(\cdot)$ of a function f is not known (i.e. not polynomial time computable). Without loss of generality, we assume the one-way function is length preserving i.e. $f(\{0,1\}^n) \subseteq \{0,1\}^n$.

2.2 Hardcore Predicates

Definition 3 *Let $f : \{0,1\}^n \to \{0,1\}^*$ and $b : \{0,1\}^n \to \{0,1\}$ be polynomial-time computable functions. We say b is a hardcore predicate of f, if for every* PPT *machine A, there exists a negligible function $\nu(\cdot)$ such that*

$$\Pr[x \leftarrow \{0,1\}^n; y = f(x) : A(1^n, y) = b(x)] \leq \frac{1}{2} + \nu(n)$$

If f is a one-way function over $\{0,1\}^n$ then Goldreich and Levin in [5] prove that the one-way function f' over $\{0,1\}^{2n}$ defined as $f'(x,r) = (f(x), r)$ admits the following hard-core predicate $b(x,r) =< x, r >= \Sigma x_i r_i \bmod 2$ where x_i, r_i is the i^{th} bit of x, r respectively. In the following we refer to this predicate as the GL bit of f.

2.3 Pseudorandom Generators

Definition 4 *Let $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ be a polynomial time computable function where $l(n) > n$. We say G is a pseudorandom generator, if for every* PPT *machine A, there exists a negligible function $\nu(n)$ such that*

$$\Big| \Pr[x \leftarrow \{0,1\}^n; y \leftarrow G(x) : A(1^n, y) = 1] \\ - \Pr[x \leftarrow \{0,1\}^{l(n)} : A(1^n, y) = 1] \Big| \leq \nu(n)$$

2.4 Universal One-Way Hash Function Families

Definition 5 *Let $\mathcal{G} = \{g_k\}_{k \in \mathcal{K}}$ be a family of functions where each function g_k goes from $\{0,1\}^{n+\ell}$ to $\{0,1\}^n$. We say that $\mathcal{G}$ is a a* Universal One-Way Hash Function Family *if (i) the functions g_k are efficiently computable and (ii) for every efficient adversary A , the probability that A succeeds in the following game is negligible in n:*

- *Let $(x, \sigma) \leftarrow A(1^n)$*
- *Choose $k \leftarrow \mathcal{K}$*
- *Let $x' \leftarrow A(\sigma, k)$*
- *A succeeds if $x \neq x'$ and $g_k(x) \neq g_k(x')$*

Universal One-Way Hash Function Families [12] as defined above enjoy the property of target collision-resistance. Next, we define the seemingly weaker notion of *Second Preimage Resistance* where the adversary cannot find a collision for randomly chosen input and key. It is well-known how to construct UOWHFs from second preimage resistant families.

Definition 6 (Second Preimage Resistance) *Let $\mathcal{G} = \{g_k\}_{k \in \mathcal{K}}$ be a family of functions where each function g_k goes from $\{0,1\}^{n+\ell}$ to $\{0,1\}^n$. We say that $\mathcal{G}$ is a a* Second Preimage Resistant Hash Function Family *if (i) the functions g_k are efficiently computable and (ii) for every efficient adversary A , then the following probability*

$$Pr[z \leftarrow \{0,1\}^{n+\ell}\ ;\ k \leftarrow \mathcal{K}\ ;\ A(z,k) = z'\ :\ z \neq z' \text{ and } g_k(z) = g_k(z')]$$

is negligible in n.

2.5 Universal Hash Function Families

Definition 7 *Let $\mathcal{H}$ be a family of functions where each function $h \in \mathcal{H}$ goes from $\{0,1\}^{n+\ell}$ to $\{0,1\}^n$. We say that $\mathcal{H}$ is a an efficient family of pairwise independent hash functions if (i) the functions $h \in \mathcal{H}$ can be described with a polynomial (in n) number of bits; (ii) there is a polynomial (in n) time algorithm to compute $h \in \mathcal{H}$; (iii) for all $x \neq x' \in \{0,1\}^{n+\ell}$ and for all $y, y' \in \{0,1\}^n$*

$$Pr_{h \in \mathcal{H}}[h(x) = y \text{ and } h(x') = y'] = 2^{-2n}$$

3 The Generalized Randomized Iterate

A well known fact about one-way functions is that if you iterate them, you may not end up with a function that is difficult to invert. Indeed while a permutation f, when iterated $f^{(i)} = f \circ \ldots \circ f$ (i.e. f composed with itself i times) remains one-way, this is not true for general one-way functions as a single application could concentrate the outputs on a very small fraction of the inputs of f, where f might even be easy to invert.

Goldreich, Krawczyk and Luby in [4] introduced the *Randomized Iterate* construction where a randomization step is added between two application of f, in its iteration. As shown in [7] when using pair-wise independent hashing to implement this randomization step, the Randomized Iterate is hard to invert.

We introduce the Generalized Randomized Iterate (GRI) and we show how it can be used to construct *both* pseudo-random generators *and* target collision-resistant hashing. We then show a randomness efficient form of the (generalized) Randomized Iterate, where some of the hash functions are "recycled" during the iteration. This *Reusable Generalized Randomized Iterate* is the core of our efficient construction of UOWHFs.

Definition 8 *Let $f : \{0,1\}^n \to \{0,1\}^n$ and let $\mathcal{H}$ be an efficient family of pairwise-independent hash functions from $\{0,1\}^{n+\ell}$ to $\{0,1\}^n$. For input $x \in \{0,1\}^n$, $z \in \{0,1\}^{\ell k}$, $h_1, \ldots, h_m \in \mathcal{H}$ and $m \geq k$, define the k^{th} Generalized Randomized Iterate $g^k : \{0,1\}^n \times \{0,1\}^{\ell k} \times \mathcal{H}^m \to \{0,1\}^n$ recursively as:*

$$g^k(x, z, h_1, \ldots, h_m) = h_k(f(g^{k-1}(x, z, h_1, \ldots, h_m))||z_{[(k-1)\ell+1 \ldots k\ell]})$$

where $g^0(x, z, h_1, \ldots, h_m) = x$, $||$ denotes concatenation and $z_{[a \ldots b]}$ is the substring of z from position a to position b.

In other words at each iteration of the Generalized Randomized Iterate, first f is applied to the output of the previous iteration, then a block of ℓ bits from z are appended to the output, and then a pair-wise independent hash function is applied. Note that at each iteration a new hash function is used.

While we are defining GRI for any value of ℓ, we are going to be interested to two cases:

- $\ell = 0$ in which case z is the empty string, and the pair-wise independent hash functions map n bits to n bits. This case is equivalent to the Randomized Iterate from [4, 7] and as shown there it can be used to build PRGs;
- $\ell = 1$ in which case z is k-bits long, and the hash functions compress one bit. We will show in Section 4 that this function is a second preimage resistant function (from which a UOWHF can be easily built).

3.1 The Reusable Generalized Randomized Iterate

We now introduce the *Reusable Generalized Randomized Iterate* (RGRI) which is a version of the Randomized Generalized Iterate that uses fewer hash functions. While the GRI described in the previous Section use new distinct hash functions at each iteration, we "recycle" some of this hash functions during the process. More specifically we sample m hash functions $h_1, \ldots, h_m$ from $\mathcal{H}$ and then in the i^{th} iteration of the RGRI we use the function $h_{\phi(i)}$ where $\phi(i)$ is the function that on input i, outputs the highest power of 2 that divides i. It is not hard to see that if we have k iterations it is sufficient to set $m = \lceil \log k \rceil + 1$. This "scheduling" of the hash functions is identical to the way Shoup recycles random masks in his construction of a domain extender for TCR functions [15].

Definition 9 *Let $f : \{0,1\}^n \to \{0,1\}^n$ and let $\mathcal{H}$ be an efficient family of pairwise-independent hash functions from $\{0,1\}^{n+\ell}$ to $\{0,1\}^n$. For input $x \in \{0,1\}^n$, $z \in \{0,1\}^{\ell k}$, $h_1, \ldots, h_m \in \mathcal{H}$ and $m \geq \lceil \log k \rceil + 1$, define the k^{th} Reusable Generalized Randomized Iterate $\widetilde{g^k} : \{0,1\}^n \times \{0,1\}^{\ell k} \times \mathcal{H}^m \to \{0,1\}^n$ recursively as:*

$$\widetilde{g^k}(x, z, h_1, \ldots, h_m) = \begin{cases} h_{\phi(k)}(f(\widetilde{g^{k-1}}(x, z, h_1, \ldots, h_m))||z_{[(k-1)\ell+1 \ldots k\ell]}) & k > 0 \\ x & otherwise \end{cases}$$

where $\phi(n)$ is one greater than the highest power of 2 that divides n.

3.2 A Technical Lemma

We now prove a preliminary Lemma which is crucial in allowing us to achieve logarithmic key size for our UOWHF construction. This Lemma abstracts the property of the "Shoup domain extension" technique we use to construct the RGRI : intuitively the Lemma proves a preliminary result that will allows us later to claim that the distribution induced by the RGRI is not that far from the distribution induced by the GRI with distinct (i.e. non-reused hash functions).

The goal of the Lemma is to count how many input pairs lead to two specific values a_0, a_1 as outputs of the RGRI.

Lemma 1 *Fix two arbitrary values $a_0, a_1 \in \{0,1\}^n$ and an integer i. The number of pairs $[(x_0, z_0, h_1, \ldots, h_m), (x_1, z_1, h_1, \ldots, h_m)]$ such that*

$$\widetilde{g^i}(x_0, z_0, h_1, \ldots, h_m) = a_0 \quad and \quad \widetilde{g^i}(x_1, z_1, h_1, \ldots, h_m) = a_1$$

is bounded by $2^{2\ell k} \cdot |\mathcal{H}|^m$.

Note that in the Lemma we are counting the pairs with possibly distinct inputs x, z but same hash functions h_i.

Proof: To prove the Lemma we use a "key-reconstruction" strategy introduced by Shoup in [15]. The algorithm in Figure 1 on input $i \in [0..k]$, $z_0, z_1 \in \{0,1\}^{\ell k}$ and $a_0, a_1 \in \{0,1\}^n$ generates a pair of inputs $(x_0, \overline{h})$ and $(x_1, \overline{h})$ such that the output of the i^{th} iterate is a_0 and a_1, i.e.

$$\widetilde{g^i}(x_0, z_0, h_1, \ldots, h_m) = a_0 \quad \text{and} \quad \widetilde{g^i}(x_1, z_1, h_1, \ldots, h_m) = a_1$$

We prove that this algorithm outputs all possible input pairs $(x_0, \overline{h})$ and $(x_1, \overline{h})$ with some probability. To complete the proof of the claim we show that the total number of distinct outputs by the algorithm is $|\mathcal{H}|^m$ (the Lemma follows since there are $2^{2\ell k}$ possible values of z_0, z_1).

The high-level idea of the Shoup reconstruction strategy described in Figure 1 is the following. Consider the simple case of the randomized iterate function g^k (where a different hash function is used after each iterate). Since, we use different hash functions at every iterate, we choose all the hash functions

$h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_m$ arbitrarily, except the one in the i^{th} iterate (i.e. h_i). Using $x_0, z_0, h_1, \ldots, h_{i-1}$ and $x_1, z_1, h_1, \ldots, h_{i-1}$ we compute y_0, y_1 as the outputs of $f \circ g^{i-1}$. We then choose h_i so that $h_i(y_0||z_{0,[(i-1)\ell+1 \ldots i\ell]}) = a_0$ and $h_i(y_1||z_{1,[(i-1)\ell+1 \ldots i\ell]}) = a_1$ simultaneously holds. This is possible since $\mathcal{H}$ is a pairwise-independent family. Furthermore, the number of such functions h_i is equal to $|\mathcal{H}|/2^{2n}$. Observe that, every input pair satisfying the conditions is output by the strategy for some random choices and every random choice yields different outputs satisfying the conditions. Therefore, the total number of pairs satisfying the conditions equals the total number of random choices made by the strategy and that is $2^{2n}2^{2\ell i}|\mathcal{H}|^{m-1} \times |\mathcal{H}|/2^{2n} = 2^{2\ell i}|\mathcal{H}|^m$.

However this procedure does not work for the reusable randomized iterate since the hash functions are recycled. Instead, we consider segments and perform a "right to left" sweep from the i^{th} iterate to the first iterate, ensuring that each segment is locally consistent. More precisely, in each segment, for a particular a, the algorithm selects hash functions and string x such that if x is fed as input to the j^{th} iterate, then the output of the computation at the i^{th} iterate $(i > j)$ is a. For the segments to compose, we need to ensure that the hash functions selected by different segments do no conflict with each other and that is the technical part of the proof. To extend the algorithm to achieve consistency for two inputs it suffices to observe that for all $x_0 \neq x_1$ and arbitrary values a_0, a_1, there exists an h such that $h(x_0) = a_0$ and $h(x_1) = a_1$. The formal description of the algorithm is presented in Figure 1.
First, we prove correctness and then compute the number of colliding pairs.

Sub-Claim 1 *If the algorithm in Figure 1 outputs* $(x_0, \overline{h}), (x_1, \overline{h})$*, then it holds that* $\widetilde{g^i}(x_0, z_0, \overline{h}) = a_0$ *and* $\widetilde{g^i}(x_1, z_1, \overline{h}) = a_1$.

Proof: Every iteration of the algorithm, considers the segment from the j^{th} iterate to the i^{th} iterate and achieves the following: if x_0^j (and x_1^j) is fixed as the partial input to the j^{th} iterate then a_0 (and a_1) is the output of the i^{th} iterate. This follows from the fact that, $h_{\phi(i)}$ is assigned a value at step 2(d) after knowing what the output of the $i-1^{st}$ iterate is computed. It only remains to show that two iterations do not assign values to the same hash function. The algorithm assigns value to a hash function in steps 2(b), 2(d) and 4. By construction step 2(b) and 4 only assign values to hash functions that have not been defined yet (indicated by the flag being false). It suffices to ensure that there are no conflicts in the assignment made at step 2(d). This is ensured by maintaining the invariant that $h_{\phi(i)}$ is undefined before executing 2(d) in any iteration. Observe that, in every iteration, $\phi(j) > \phi(i)$ and for all c such that $j < c < i$, $\phi(c) < \phi(i)$. Hence, before step 2(d) is reached in any iteration, the only hash-functions that are defined are those with indices c such that $\phi(c) < \phi(j)$. $\square$

Sub-Claim 2 *The number of distinct pairs output by the Shoup Reconstruction algorithm is bounded by* $|\mathcal{H}|^m$.

Proof: From Sub-Claim 1, we know that every pair output of the algorithm satisfies the condition that a_0 and a_1 are the output of the i^{th} iterate. Furthermore, every pair that satisfies the condition occurs as an output for some choice

```
Input: $i, z_0, z_1, a_0, a_1$
1. Set Flags $F_0, \ldots, F_{m-1}$ to $false$ // Flags indicate which hash-functions are assigned
2. while $i \neq 0$
   (a) $j \longleftarrow (i - 2^{\phi(i)})$        // The new condition will be at position j
   (b) Randomly choose $x_0^j$, $x_1^j$ from $\{0,1\}^n$. For all $j < c < i$, if $F_{\phi(c)} = false$,
       randomly choose $h_{\phi(c)}$ from $\mathcal{H}$ and set $F_{\phi(c)} \longleftarrow true$.
   (c) Compute
       $x_0^j \xrightarrow{f} \xrightarrow{||z_{0,[j\ell+1\ldots(j+1)\ell]}} \xrightarrow{h_{\phi(j+1)}} \xrightarrow{f} \cdots \xrightarrow{h_{\phi(i-1)}} \xrightarrow{f} y_0$
       $x_1^j \xrightarrow{f} \xrightarrow{||z_{1,[j\ell+1\ldots(j+1)\ell]}} \xrightarrow{h_{\phi(j+1)}} \xrightarrow{f} \cdots \xrightarrow{h_{\phi(i-1)}} \xrightarrow{f} y_1$
   (d) Randomly choose $h \in \mathcal{H}$ conditioned on
       $h(y_0||z_{0,[(i-1)\ell+1\ldots i\ell]}) = a_0$ and $h(y_1||z_{1,[(i-1)\ell+1\ldots i\ell]}) = a_1$
       Set $h_{\phi(i)} \longleftarrow h, F_{\phi(i)} \longleftarrow true$.
   (e) $i \longleftarrow j$, $a_0 \longleftarrow x_0^j$, $a_1 \longleftarrow x_1^j$
3. endwhile
4. For all $c$, if $F_{\phi(c)} = false$, pick $h_{\phi(c)}$ uniformly from $\mathcal{H}$ and set $F_{\phi(c)}$ to $true$.
5. output $(x_0 = x_0^1, h_1, \ldots, h_m), (x_1 = x_1^1, h_1, \ldots, h_m)$
```

Fig. 1. Shoup Reconstruction Algorithm

made by the algorithm and each choice made by the algorithm yields distinct outputs. Therefore, it suffices to compute the total number of choices made by the algorithm. To compute the number of pairs, observe that, for every choice made for x_0^j and x_1^j (such that $x_0^j \neq x_1^j$) in step (b), the number of hash functions h such that

$$h(y_0||z_{0,[(i-1)\ell+1\ldots i\ell]}) = a_0 \text{ and } h(y_1||z_{1,[(i-1)\ell+1\ldots i\ell]}) = a_1$$

is $\frac{|\mathcal{H}|}{2^{2n}}$, by the pairwise independence property. We treat the choices made for x_0^j, x_1^j as a choice made for $h_{\phi(i)}$ set in step 2(d). Thus, the number of choices for the hash function in step 2(d) is at most $2^{2n} \times \frac{|\mathcal{H}|}{2^{2n}} = |\mathcal{H}|$. The only other choices are the hash functions picked in step 2(b) and 4. Since they can take any value, they have $|\mathcal{H}|$ many choices. Hence, corresponding to every hash function the algorithm makes $|\mathcal{H}|$ many choices. Thus, the total number of pairs is bounded by $|\mathcal{H}|^m$. □

This concludes the proof of Lemma 1. □

The following Corollary is proven by using the same counting argument and the same "reconstruction strategy" of Lemma 1 (intuitively, the bound results from the fact that you can choose x in 2^n ways, z in 2^{lk} ways, $m-1$ hash functions uniformly at random in $\mathcal{H}$, and the hash function h_i via pairwise independence among $|\mathcal{H}|/2^{2n}$ possible candidates).

Corollary 1 *Fix arbitrary values $a_0, a_1 \in \{0,1\}^n$, $y \in \{0,1\}^{n+\ell}$ and an integer i. The number of inputs $(x, z, h_1, \ldots, h_m)$ such that*

$$\widetilde{g^i}(x, z, h_1, \ldots, h_m) = a_0 \quad \textit{and} \quad h_i(y) = a_1$$

is bounded by $2^{\ell k - n} \cdot |\mathcal{H}|^m$. Moreover there exists a polynomial time algorithm that samples such an input uniformly at random.

Remark: We point out that the "reconstruction" property outlined in Lemma 1 is exactly what is needed in order to prove the security of our UOWHF with $O(n \log n)$ key based on the RGRI.

This is in contrast to the case of PRG [7] where any PRG for space-bounded computation would work to "de-randomize" the seed from n^2 to $n \log n$. We can show that the particular space-bounded PRG used in [7] satisfies a Lemma similar to Lemma 1, and therefore could be used to reduce the size of the key of our UOWHF. For simplicity we just show the construction based on Shoup's technique.

4 Constructions of Universal One-Way Hash Functions

In this section, we show how to construct second preimage resistant functions from regular one-way functions. We start with a simple construction (that already improves the efficiency from previous work) of quadratic key size. We then provide a more efficient and essentially optimal solution with $O(n \log n)$ key size. Note that our functions compress a single bit (higher compression can be achieved by standard modes of iteration). Note also that UOWHFs can be easily built from second preimage resistant families.

4.1 A Construction with Linear Key Size

Definition 10 *Let $f : \{0,1\}^n \to \{0,1\}^n$ and let $\mathcal{K} = \{0,1\}^n \times \mathcal{H}^{n+1}$ where $\mathcal{H}$ is an efficient family of pairwise-independent hash functions from $\{0,1\}^{n+1}$ to $\{0,1\}^n$. Define the function $g(z, k)$ with input space $z \in \{0,1\}^{n+1}$ and key-space $k = (x, h_1, \ldots, h_{n+1}) \in \mathcal{K}$ as follows:*

$$g(z, (x, h_1, \ldots, h_{n+1})) = g^{n+1}(x, z, h_1, \ldots, h_{n+1})$$

where g^i is the Generalized Randomized Iterate with $\ell = 1$.

Theorem 1 *Suppose f is a 2^r-regular one-way function. Then g defined according to Definition 10 is a second preimage resistant function family.*

Proof Overview: To understand how our construction works, let us assume (as a simplifying assumption) that we can uniformly sample pairs (a_1, a_2) such that $f(a_1) = f(a_2)$. Let us refer to such pairs as *siblings* for f.

Given such a pair it is possible to set up the hash functions in the above construction so that if the adversary finds a collision, then we invert the one-way function on a point y. Intuitively this is done as follows: given a random input z for the UOWHF, we choose the hash functions (i.e. the key k) so that $g^i(z,k) = a_1$ and $h_i(y||b) = a_2$ for a random index i and a random bit b. We then run the adversary on z, k and if the adversary finds a collision z', with non-negligible probability the collision "goes through" a_2 at index i, i.e. $g^i(z', k) = a_2$ allowing us to find a preimage of y.

The intuition here is that given any input z, and key k, at each iterate the input going into the one-way function has most 2^r collisions w.r.t f. For a collision to occur at a particular iterate, it must be the case that some range element y of the one-way function f must occur at the previous iterate and the hash function takes y and an input bit into one of the 2^r collisions in the next iterate. Since there are at most 2^{n-r} range elements, in expectation over hash functions, the number of possible inputs at the previous iterate that are mapped into the 2^r collisions are small, in fact $O(1)$. Thus the hash functions selected above will succeed with high probability.

But how do we get to sample a_1, a_2, i.e. siblings for f in the first place? For this we use the adversary again. Indeed when an adversary finds a collision to input z (say z'), it must be that at some iterate, the inputs into the intermediate hash functions are different and the outputs to the next iterate are strings a_1 and a_2 such that $f(a_1) = f(a_2)$, i.e. siblings for f. It remains to argue that sampling a_1 and a_2 by first querying the adversary is good enough, and this is established using a collision-probability-type analysis. We now proceed to a formal proof.

Proof: Assume for contradiction, there exists an adversary A and polynomial $p(\cdot)$ such that for infinitely many lengths n, the probability with which A finds a collision on a random input $z \in \{0,1\}^n$ and key $k = (x, \overline{h}) \in \mathcal{K}$ is at least $\epsilon \geq \frac{1}{p(n)}$. We assume for simplicity that A is deterministic. Fix a particular n for which this happens. Using A, we construct a machine M that inverts f with probability that is polynomially related to ϵ and thus arrive at a contradiction.

The machine M on input $y \in \{0,1\}^n$ internally incorporates the code of A and proceeds as follows:

1. Sample a random input z and key $k = (x, \overline{h})$. Internally run A on input (z, k). If A fails to return a collision, halt outputting $\perp$. Otherwise, let z' be the output of A.
2. Let i be the smallest index such that $f(g^{i-1}(z,k))||z_i \neq f(g^{i-1}(z',k))||z'_i$ and $f(g^i(z,k)) = f(g^i(z',k))$ (since $g(z,k) = g(z',k)$ such an i must exists). Let $a_1 = g^i(z,k)$ and $a_2 = g^i(z',k)$. It follows now that $f(a_1) = f(a_2)$. For any two colliding inputs such as z and z' with key k, we call this i the colliding-index.
3. Choose $z^*, k^* = (x^*, h_1^*, \ldots, h_{n+1}^*)$ and a random bit b such that $g^i(z^*, k^*) = a_1$ and $h_i^*(y||b) = a_2$. This can be done using the pairwise independence

property of $\mathcal{H}$. More precisely, choose z^*, x^* and all the hash functions except h_i^* at random and set h_i^* so that both the conditions hold. Run A on input (z^*, k^*). If A fails to return a collision or such a hash function h_i can not be sampled,[6] halt outputting $\perp$. Otherwise, let z'' be the output of A.

4. If $f(g^{i-1}(z'', k^*)) \neq y$, halt outputting $\perp$. Otherwise, output $g^{i-1}(z'', k^*)$.

It follows from the construction that if M outputs w, then $f(w) = y$. We now proceed to compute the success probability of M. But first, we require the following definition. Define sets $N(i, a_1, a_2)$ to contain all input-key pairs (z, k) such that the following hold true: $f(a_1) = f(a_2)$ and $g^i(z, k) = a_1$, and A on input (z, k) returns z' such that $g^i(z', k) = a_2$ and i is the colliding-index. We first express the success probability of M using these sets.

Claim 1 *The probability with which M succeeds in inverting f is*

$$2^{n+r-1} \sum_{i,a_1,a_2} \frac{|N(i, a_1, a_2)|^2}{(2^{2n+1}|\mathcal{H}|^{n+1})^2}$$

Proof: Given a tuple (z^*, k^*, i, a_1, a_2) such that $(z^*, k^*) \in N(i, a_1, a_2)$, define the following events:

Event $E1$: The randomly chosen input-key pair (z, k) by M in Step 1 is in $N(i, a_1, a_2)$.
Since the input and key are chosen uniformly at random, it holds that $\Pr[E_1] = 1/2^{n+1} \times 1/2^n \times 1/|\mathcal{H}|^{n+1} \times |N(i, a_1, a_2)| = |N(i, a_1, a_2)|/2^{2n+1}|\mathcal{H}|^{n+1}$
Event $E2$: If A on input (z^*, k^*) returns z'—where $k^* = (x, h_1, \ldots, h_n)$—this event denotes that M's random choice $b = z'_i$ and M's input is y such that $g^{i-1}(z', k^*) = y$. Therefore, $h_i(y||b) = h_i(y||z'_i) = a_2$.
The probability that $b = z'_i$ is $1/2$. Therefore, since f is a 2^r-regular OWF, $\Pr[E_2] = 1/2 \cdot 2^r/2^n = 2^{r-1}/2^n$.
Event $E3$: M chooses z^*, k^* in Step 3.
From the pairwise-independence property of $\mathcal{H}$, it follows that[7] $\Pr[E_3] = 1/(2^{2n+1}\frac{|\mathcal{H}|^{n+1}}{2^{2n}}) = 1/2|\mathcal{H}|^{n+1}$

It follows from the description that for any tuple (z^*, k^*, i, a_1, a_2) such that $(z^*, k^*) \in N(i, a_1, a_2)$, if E_1, E_2 and E_3 occurs, M inverts y. Note that E_1, E_2 and E_3 are independent. Therefore, for a fixed tuple (z^*, k^*, i, a_1, a_2) such that $(z^*, k^*) \in N(i, a_1, a_2)$ the probability that E_1, E_2 and E_3 occurs is

$$\frac{|N(i, a_1, a_2)|}{2^{2n+1}|\mathcal{H}|^{n+1}} \times \frac{2^{r-1}}{2^n} \times \frac{1}{2|\mathcal{H}|^{n+1}}$$

[6] This occurs when $a_1 \neq a_2$ and $f(g^{i-1}(z^*, k^*)) = y$ and $z_i = b$

[7] z, x and all the hash functions except h_i are randomly chosen. There are $2^{2n+1}|\mathcal{H}|^{m-1}$ such tuples. h_i is chosen so that two of its values are fixed. Since $\mathcal{H}$ is a pairwise-independent family of hash functions, there are exactly $\frac{|\mathcal{H}|}{2^{2n}}$ such functions. Finally, one of these tuples are chosen uniformly at random.

It follows from the definition of the sets $N(\cdot,\cdot,\cdot)$, that for every (z,k) there exists at most one tuple (i,a_1,a_2) such that $(z,k) \in N(i,a_1,a_2)$. Therefore, the success probability of M can be expressed as the sum of the success probability of M on each tuple (z^*,k^*,i,a_1,a_2) such that $(z^*,k^*) \in N(i,a_1,a_2)$. More precisely, the success probability of M is,

$$\sum_{i,a_1,a_2} \sum_{(z^*,k^*)\in N(i,a_1,a_2)} \frac{|N(i,a_1,a_2)|}{2^{2n+1}|\mathcal{H}|^{n+1}} \times \frac{2^{r-1}}{2^n} \times \frac{1}{2|\mathcal{H}|^{n+1}}$$

$$= \sum_{i,a_1,a_2} \frac{|N(i,a_1,a_2)|^2}{2^{2n+1}|\mathcal{H}|^{n+1}} \times \frac{2^{r-1}}{2^n} \times \frac{1}{2|\mathcal{H}|^{n+1}}$$

$$= 2^{n+r-1} \sum_{i,a_1,a_2} \frac{|N(i,a_1,a_2)|^2}{\left(2^{2n+1}|\mathcal{H}|^{n+1}\right)^2}$$

□

We now relate this expression to the success probability of A.

Claim 2 *If A succeeds with probability ϵ then* $\sum_{(i,a_1,a_2)} \frac{|N(i,a_1,a_2)|^2}{\left(2^{2n+1}|\mathcal{H}|^{n+1}\right)^2} \geq \frac{\epsilon^2}{n2^{n+r}}$

Proof: Since for every pair (z,k), there exists at most one tuple (i,a_1,a_2) such that $(z,k) \in N(i,a_1,a_2)$ and by definition if $(z,k) \in N(i,a_1,a_2)$ then A succeeds on input (z,k), we have that the success probability of A is

$$1/\left(2^{2n+1}|\mathcal{H}|^{n+1}\right) \times \sum_{(i,a_1,a_2)} |N(i,a_1,a_2)| = \epsilon$$

Let us consider the sum in the left-hand side and use the Cauchy-Schwartz inequality to obtain a bound on the sum of the squares of each term. It suffices to consider the sum over all tuples (i,a_1,a_2) such that $N(i,a_1,a_2)$ is not empty. In particular, they are not empty only if $f(a_1) = f(a_2)$. Therefore, the total number of such tuples is at most $n2^{n+r}$. Using the Cauchy-Schwartz inequality, we have that

$$\sum_{(i,a_1,a_2)} \frac{|N(i,a_1,a_2)|^2}{\left(2^{2n+1}|\mathcal{H}|^{n+1}\right)^2} \geq \frac{\epsilon^2}{n2^{n+r}}$$

□

Now, we conclude the proof of the theorem. Applying Claim 2 to Claim 1, we obtain that the success probability of M is at least $2^{n+r-1} \times \frac{\epsilon^2}{n2^{n+r}} = \frac{\epsilon^2}{2n}$ which is non-negligible. Therefore, M inverts f with non-negligible probability and we arrive at a contradiction. □

4.2 A Construction with Logarithmic Key Size

We now show how to construct a more efficient second preimage resistant family from regular one-way functions, by showing that if f is a regular OWF then the Reusable Generalized Randomized Iterate is second preimage resistant.

Definition 11 *Let $f : \{0,1\}^n \to \{0,1\}^n$ and let $\mathcal{K} = \{0,1\}^n \times \mathcal{H}^m$ where $\mathcal{H}$ is an efficient family of pairwise-independent hash functions from $\{0,1\}^{n+1}$ to $\{0,1\}^n$ and $m = O(\log n)$. Define the function $g(z,k)$ with input space $z \in \{0,1\}^{n+1}$ and key-space $k = (x, h_1, \dots, h_m) \in \mathcal{K}$ as follows:*

$$g(z, (x, h_1, \dots, h_m)) = \widetilde{g^{n+1}}(x, z, h_1, \dots, h_m)$$

where $\widetilde{g}^i$ is the Reusable Generalized Randomized Iterate with $\ell = 1$.

Theorem 2 *Suppose f is a 2^r-regular one-way function. Then g defined according to Definition 11 is a second preimage resistant function family.*

Proof: Assume for contradiction, there exists an adversary A and polynomial $p(\cdot)$ such that for infinitely many lengths n, the probability with which A finds a collision on a random input $z \in \{0,1\}^n$ and key $k = (x, \overline{h}) \in \mathcal{K}$ is $\epsilon \geq \frac{1}{p(n)}$. As before, we assume for simplicity that A is deterministic.

Fix a particular n for which this happens. Using A, we construct a machine M that inverts f with probability that is polynomially related to ϵ and thus arrive at a contradiction. The machine M on input $y \in \{0,1\}^n$ internally incorporates the code of A and proceeds as follows:

1. Sample a random input z and key $k = (x, \overline{h})$. Internally run A on input (z,k). If A fails to return a collision, halt outputting $\perp$. Otherwise, let z' be the output of A.
2. Let i be the colliding-index. Let $a_1 = g^i(z,k)$ and $a_2 = g^i(z',k)$
3. Choose $z^*, k^* = (z^*, h_1^*, \dots, h_m^*)$ and a random bit b such that $g^i(z^*, k^*) = a_1$ and $h^*_{\phi(i)}(y||b) = a_2$. This can be done in polynomial time following Corollary 1. Internally run A on input (z^*, k^*). If A fails to return a collision, halt outputting $\perp$. Otherwise, let z'' be A's output.
4. If $f(g^{i-1}(z", k^*)) \neq y$, halt outputting $\perp$. Otherwise, output $g^{i-1}(z'', k^*)$.

As before, we define sets $N(i, a_1, a_2)$ that satisfy the same condition with the exception that we rely on $\widetilde{g}^i$ instead of g^i. The next claim relates these sets to the success probability of M.

Claim 3 *The probability with which M succeeds in inverting the one-way function f is $2^{n+r-1} \sum_{i,a_1,a_2} |N(i,a_1,a_2)|^2 / \left(2^{2n+1}|\mathcal{H}|^m\right)^2$*

Proof: Consider the events E1,E2 and E3 exactly as before. We now have that given a tuple (z, k, i, a_1, a_2),

- Probability that E_1 occurs is $1/2^{n+1} \times 1/2^n \times 1/|\mathcal{H}|^m \times |N(i,a_1,a_2)| = |N(i,a_1,a_2)|/2^{2n+1}|\mathcal{H}|^m$.
- Probability that E_2 occurs is $2^{r-1}/2^n$ as before.
- Probability that E_3 occurs given E_1 and E_2 occurs is $1/(2^{2n+1}\frac{|\mathcal{H}|^m}{2^{2n}}) = 1/2|\mathcal{H}|^m$. This follows from Corollary 1 for $\ell = 1$ and $k = n+1$ (which are the parameters used in this construction).

Again, we have that every (z, k) belongs to at most one set $N(i, a_1, a_2)$. Therefore, the success probability of M is

$$\sum_{i,a_1,a_2} \sum_{(z,k)\in N(i,a_1,a_2)} \frac{|N(i,a_1,a_2)|}{2^{2n+1}|\mathcal{H}|^m} \times \frac{2^{r-1}}{2^n} \times \frac{1}{2|\mathcal{H}|^m} = 2^{n+r-1} \sum_{i,a_1,a_2} \frac{|N(i,a_1,a_2)|^2}{(2^{2n+1}|\mathcal{H}|^m)^2}$$

$\square$

The next claim follows identically to Claim 2.

Claim 4 *If A succeeds with probability ϵ then* $\sum_{(i,a_1,a_2)} \frac{|N(i,a_1,a_2)|^2}{(2^{2n+1}|\mathcal{H}|^m)^2} \geq \frac{\epsilon^2}{n2^{n+r}}$

As before applying Claim 3 to Claim 4, we obtain that the success probability of M is at least $\frac{\epsilon^2}{2n}$ and thus we arrive at a contradiction. $\square$

5 PRG Construction and Hardness Amplification

The idea of iterating a one-way permutation f on itself to obtain a PRG originates from the work of Blum, Micali and Yao [1, 17]. Since f is a permutation, the function $f^{(i)} = f \circ \ldots \circ f$ (f iterated on itself i times) is also one-way. This means that the hardcore bit of every intermediate step is unpredictable. Iterating $n + 1$ times on a random input of length n and outputting all the hardcore bits would then yield a PRG that stretches by 1 bit. We refer to this as the BMY construction[8].

This approach, unfortunately does not work for general one-way functions. For the special case of regular one-way functions, Goldreich, Krawczyk and Luby [4], showed how to extend the BMY construction by adding a randomization step using an n-wise independent hash-function between every two applications of f. Haitner, et. al [7] simplified the construction to use just pair-wise hashing and further derandomized the construction by showing how to generate the n hash-functions required at the randomization steps using just $n \log n$ bits thus obtaining a PRG of seed length $O(n \log n)$.

In [7], Haitner et. al, showed that the same randomized iterate can also be used for hardness amplification to obtain strong one-way function from any regular weakly one-way function with unknown regularity. They also showed that similar derandomization yielded corresponding efficiency gains.

Using the Reusable Generalized Randomized Iterate, we obtain analogous PRG constructions and hardness-amplification with same efficiency. More precisely, we obtain the following results.

Theorem 3 *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a regular one-way function and $\mathcal{H}$ be an efficient family of pairwise-independent length preserving hash functions. Define $G : \{0,1\}^{2n} \times \mathcal{H}^m \to \{0,1\}^{2n+1} \times \mathcal{H}^m$ as*

$$G(x, r, \overline{h}) = (b(\widetilde{f}^0(x, \overline{h}), r), \ldots, b(\widetilde{f}^n(x, \overline{h}), r), r, \overline{h})$$

[8] If f is a permutation over n-bit strings a more efficient construction is to set the generator G as $G(x) = f(x).b(x)$. However this uses in a crucial way the property that f is a permutation (since if x is uniform then $f(x)$ is also uniform).

where $\widetilde{f^k}(x, h_1, \ldots, h_m) = f(\widetilde{g^k}(x, h_1, \ldots, h_m))$ *and* $\widetilde{g^k}$ *is the* RGRI *defined by* $x, h_1, \ldots, h_m$ *with* $\ell = 0$ *and* b *is the Goldreich-Levin hardcore predicate. Then* G *is a pseudorandom generator.*

Theorem 4 *Let* f *be a* $\frac{1}{p(n)}$*-weak one-way function for some polynomial* $p(\cdot)$.[9] *Let* $k = 4np(n)$ *and* $m = \lceil \log k \rceil$. *For input* $x \in \{0,1\}^n$, $\overline{h} = [h_1, \ldots, h_m] \in \mathcal{H}^m$, *define* $g(x, \overline{h}) = (\widetilde{f^k}(x, \overline{h}), \overline{h})$ *where* $\widetilde{f^k}$ *is the Reusable Randomized Iterate of* f. *Then,* g *is a (strong) one-way function.*

The proofs of both these theorems appear in the full-version of the paper and on a high-level follow the proofs presented in [7].

6 Discussion and Conclusions

This paper presented the Reusable Generalized Randomized Iterate, and its application to new efficient constructions of Universal One-Way Hash Functions based on regular one-way functions. These are the first such efficient constructions of UOWHF from regular one-way functions of unknown regularity.

We also showed that the Reusable Generalized Randomized Iterate can be used to construct PRGs based on regular-one way functions, obtaining an alternative proof of a result by [7].

An interesting question raised by our work is the following: can we replace Shoup's technique for TCR domain extension with any appropriate log-space derandomizer? This is not immediately clear, since the reconstruction algorithm of Lemma 1 plays a crucial role in our construction and such a property does not follow from the definition of derandomizers (although, the current derandomizers indeed have that property).

A more conceptual contribution of this paper is to show that by combining techniques from the collision-resistant hashing and PRG toolboxes we can improve efficiency in both areas. Following [9, 8] we believe that exploring the interplay between the two fields, and the possibility to apply techniques from one field to the other can lead to new and interesting discoveries.

The works in [9, 8] highlight a inherent "black-box duality" between PRGs and UOWHFs. Starting from the PRG constructions based on OWPs [1] and OWFs [11], one can obtain the UOWHF constructions based on OWPs [12] and OWFs [13, 8] using the following "parallelism". If there is "unpredictable" entropy in an input to an application of the one-way function in the PRG construction from which pseudo-entropy can be extracted, then there exists a symmetric TCR construction with the same structure where the output of the application of the one-way function has "inaccessible" entropy and can be compressed.

Our Generalized Randomized Iterate justifies this observation for the case of regular one-way functions in a more direct way, by showing a *single* algorithm that yields either a PRG or a UOWHF depending on the parameters.

[9] A function f is an ϵ-weak one-way function, if no adversary can succeed in inverting f with probability better than $1 - \epsilon$.

The approaches in [9, 8] and in this paper can hopefully help in addressing the following interesting open problem. Is there a transformation that takes any PRG construction from a primitive $\mathcal{P}$ with ℓ-bit-expansion to a TCR construction from the same primitive $\mathcal{P}$ with ℓ-bit-compression and vice-versa. For example, given a OWP with a large ℓ-bit hard-core function, we know how to build a PRG that expands by ℓ bits per invocation of the OWP: is it possible to obtain a TCR which compresses by ℓ bits per invocation of the OWP? Conversely, an answer to the above general question would allow us to achieve more efficient PRG constructions from stronger primitives such as collision-resistant hash-functions.

References

1. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo random bits. In *Siam Journal of Computing*, pages 112–117, 1982.
2. A. De Santis, M. Yung. On the Design of Provably Secure Cryptographic Hash Functions. In *EUROCRYPT 1990*, pp.412–431.
3. R. Gennaro and L. Trevisan. Lower Bounds on the Efficiency of Generic Cryptographic Constructions. *FOCS 2000*: pp.305-313.
4. O. Goldreich and H. Krawczyk and M. Luby. On the existence of pseudorandom generators. In *Siam Journal of Computing*, Vol 22(6), pages 1163–1175, 1993.
5. O. Goldreich and L.A. Levin. A hard-core predicate for all one-way functions. In *STOC 1989*, pages 25–32, 1989.
6. S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17:281–308, 1988.
7. I. Haitner, D. Harnik, and O. Reingold. On the Power of Randomized Iterate. In *CRYPTO 2006*, pages 22–40, 2006.
8. I. Haitner, T. Holenstein, O. Reingold, S.P. Vadhan, H. Wee. Universal One-Way Hash Functions via Inaccessible Entropy. In *EUROCRYPT 2010*, pp.616–637.
9. I. Haitner, O. Reingold, S. Vadhan and H. Wee. Inaccessible Entropy. In *STOC'09*, pp.611-620.
10. S. Halevi and H. Krawczyk. Strengthening Digital Signatures Via Randomized Hashing. In *CRYPTO'06*, pp.41–59.
11. J. Hastad, R. Impagliazzo, L.A. Levin and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computing*, 28(4):1364–1396, 1989.
12. M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Applications. In *STOC'89*, pp. 33–43.
13. J. Rompel. One-Way Functions are Necessary and Sufficient for Secure Signatures. In *STOC'90*, pp.387–394.
14. P. Sarkar. Masking-based domain extenders for UOWHFs: bounds and constructions. *IEEE Transactions on Information Theory* 51(12):4299-4311 (2005)
15. V. Shoup. A composition theorem for Universal One-Way Hash Functions. In *Eurocrypt 2000*, pages 445–452.
16. D.R. Simon. Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In *EUROCRYPT'98*, pp.334–345
17. A. Yao. Theory and applications of trapdoor functions. In *FOCS*, pages 80–91, 1982.