On Constructing 1-1 One-Way Functions

Oded Goldreich, Leonid A Levin, and Noam Nisan

Abstract. We show how to construct length-preserving 1-1 one-way functions based on popular intractability assumptions (e.g., RSA, DLP). Such 1-1 functions should not be confused with (infinite) families of (finite) one-way permutations. What we want and obtain is a single (infinite) 1-1 one-way function.

Keywords: One-Way Functions, RSA, Discrete Logarithm Problem.

This work was conducted in the summer of 1994. An early version of it appeared as TR95-029 of *ECCC*. Section 4 has been revised and improved by relying on subsequent advances regarding primality testing [1]. Specifically, we replace the randomized primality tester of [3] (which builds upon [13, 17]) by the deterministic primality tester of Agrawal, Kayal, and Saxena [1]. Various footnotes indicate these (as well as other significant) deviations from the aforementioned early version.

1 Introduction

Given any one-way permutation (i.e., a length preserving 1-1 one-way function), one can easily construct an efficient pseudorandom generator. The construction follows the scheme given by Blum and Micali [4], using the fact that every one-way function has a hard-core bit [8]. Specifically, assume that f is such a function and let b be a hard core-bit for it (e.g., starting with a function f', we may define $f(x, r) \stackrel{\text{def}}{=} (f'(x), r)$ and $b(x, r)$ as the inner-product mod 2 of the strings x and r when viewed as binary vectors of length $|x| = |r|$). Then, on input a seed s, the pseudorandom generator outputs the sequence $b(s), b(f(s)), b(f(f(s))), b(f^3(s)), ...$

Pseudorandom generators can be constructed also based on arbitrary one-way functions [12]; yet, the known construction is very complex and inefficient.[1] In fact, it is of no practical value. The construction in [9], which uses arbitrary *regular* one-way functions is more attractive in these respects, yet it is far less attractive than the simple construction outlined above. A similar situation occurs with respect to the construction of digital signature schemes (cf., [14] vs [19]). In general, 1-1 one-way functions currently offer simpler and more practical constructions (of more complex primitives) than offered by general one-way functions.

[1] The same applies also to subsequent improvements, currently culminating in [11].

These facts were our initial motivation for trying to construct length-preserving 1-1 one-way functions. Such functions should not be confused with what is commonly referred to (especially in the "Crypto Community") as "one-way permutations", which are actually infinite sets of finite functions – see definitions below. What we want is a single infinite function that is both length-preserving and 1-1 (and needless to say one-way). We show how to construct such 1-1 one-way functions based on popular intractability assumptions such as the intractability of DLP and inverting RSA.

Indeed, some (but not all) of the constructions that use length-preserving 1-1 one-way functions can be modified such that families of one-way permutations can be used instead. Still the question of whether the former (i.e., length-preserving 1-1 one-way functions) exists is of both theoretical and practical importance.

2 One-Way Functions and Families

We start by recalling the standard definitions.

Definition 2.1 (one-way functions): *Let $f:\{0,1\}^* \to \{0,1\}^*$ be a* length preserving *function that is polynomial-time computable.*

– (strongly one-way): *f is called* (strongly) one-way *if for any probabilistic polynomial-time algorithm A, any positive polynomial p and all sufficiently large n, it holds that*

$$\text{Prob}[A(f(x)) \in f^{-1}f(x)] < \frac{1}{p(n)}$$

where the probability is taken uniformly over $x \in \{0,1\}^n$, and the internal coin tosses of algorithm A.

– (weakly one-way): *f is called* weakly one-way *if there exists a positive polynomial p such that for any probabilistic polynomial-time algorithm A and all sufficiently large n, it holds that*

$$\text{Prob}[A(f(x)) \notin f^{-1}f(x)] > \frac{1}{p(n)}$$

where the probability is as above.

Recall that $f:\{0,1\}^* \to \{0,1\}^*$ is 1-1 if $f(x) \neq f(y)$ for all $x \neq y$. In the case that $f(x) \neq f(y)$ for all but a negligible fraction of the pairs (x,y) we say that f is almost 1-1. Namely, an almost 1-1 function f satisfies, for every positive polynomial p and all sufficiently large n, it holds that

$$\text{Prob}[f(x) = f(y)] < \frac{1}{p(n)}$$

where the probability is taken uniformly and independently over all $x, y \in \{0,1\}^n$.

Definition 2.2 (family of one-way permutations – simplified version): *An infinite set of finite permutations,* $\mathbf{F} = \{f_i : D_i \overset{\text{1-1}}{\to} D_i\}_{i \in I}$*, is called a* family of one-way permutations *if the following conditions hold*

- (efficient evaluation): *There exists a polynomial-time algorithm that on input an index* (of a permutation) $i \in I$ *and a domain element* $x \in D_i$ *returns* $f_i(x)$.
- (efficient index selection): *There exists a probabilistic algorithm* S *that on input* n*, runs for* $\text{poly}(n)$ *time and returns a uniformly distributed index of length* n (i.e., an i uniformly distributed in $I \cap \{0,1\}^n$).
- (efficient domain sampling): *There exists a probabilistic polynomial-time algorithm* D *that on input an index* $i \in I$*, returns a uniformly distributed element of* D_i.
- (one-wayness): *For any probabilistic polynomial-time algorithm* A*, any positive polynomial* p *and all sufficiently large* n*, it holds that*

$$\text{Prob}[A(i, f_i(x)) = x] < \frac{1}{p(n)}$$

  where the probability is taken uniformly over $i \in I \cap \{0,1\}^n$, $x \in D_i$*, and the internal coin tosses of algorithm* A.

In the non-simplified version, both the aforementioned probabilistic algorithms (i.e., S and D) are allowed to produce output with only noticeable probability (i.e., probability at least $1/\text{poly}(n)$). Furthermore, given these algorithms have produced an output, the output is allowed to be wrong (i.e., out of the target set or non-uniformly distributed) with negligible probability (e.g., with probability at most 2^{-n}). Our transformations will take advantage of the first relaxation, but not of the second.[2]

Analogously to Definition 2.1, families of permutations can be defined to be *weakly* one-way, rather than (strongly) one-way.

3 Transforming One-Way Families into Functions

Clearly, any family of one-way permutations can be converted into a single one-way function; namely, $f(r,s) \overset{\text{def}}{=} f_i(x)$, where $i = S(n,r)$ is the index selected using coin-tosses r and $x = D(i,s) \in D_i$ is the element selected on input i and coin-tosses s. (Padding can be applied, if necessary, to make f length preserving.) However, this procedure does not necessarily yield a 1-1 function; furthermore, for most natural examples such as RSA, DLP, etc., the resulting function will be many-to-one.

[2] In the earlier version of this work, we also took advantage of the second relaxation. This was done in order to account for the probability of error that was present in the probabilistic primality tests that we used. The need to accommodate error is currently eliminated by using the deterministic primality test of [1].

An alternative construction, which does yield a 1-1 one-way function, is possible under some additional conditions, as demonstrated below. In fact, the conditions are defined to make this natural construction work and the thrust of this paper is in demonstrating that these conditions can be met under reasonable and popular assumptions (see next section).

3.1 The Conditions

Let **F** be a family of one-way permutations and that let $q(n)$ denote the number of coins flipped by the index-selection algorithm S on input n. We consider the following conditions that **F** may satisfy.

Definition 3.1 (additional conditions)

– Augmented one-wayness:[3] *For any probabilistic polynomial-time algorithm A, any positive polynomial p and all sufficiently large n, it holds that*

$$\text{Prob}[A(r, f_{S(n,r)}(x)) = x] < \frac{1}{p(n)}$$

where the probability is taken uniformly over $r \in \{0,1\}^{q(n)}$, $x \in D_{S(n,r)}$, and the internal coin tosses of algorithm A.
(Namely, the permutations are hard to invert even when the inverting algorithm is given the random coins used to generate the index of the permutation.)

– Canonical domain sampling:[4] *The domain-sampling algorithm may consist of uniformly selecting a string of specific* (easy to determine) *length and testing whether the string resides in the domain. In other words, we require*
  - (recognizable domain): *There exists a polynomial-time algorithm that on input an index $i \in I$ and a string x decides if $x \in D_i$.*
  - (noticeable domain): *There exists a polynomial-time computable function $l : \mathbb{N} \to \mathbb{N}$ and a positive polynomial $p(\cdot)$ so that $D_i \subseteq \{0,1\}^{l(n)}$ and $|D_i| > \frac{1}{p(n)} \cdot 2^{l(n)}$*

3.2 The Construction

Given a family of one-way permutations that satisfies the additional conditions, we explicitly construct a 1-1 one-way function as follows.

[3] Note that this condition is different from the notion of *enhanced* one-wayness as defined in [6, Apdx. C.1]. Specifically, here the inverting algorithm gets the coins that were used by the *index selection* algorithm, whereas in [6, Apdx. C.1] the inverting algorithm gets the coins that are used by the *domain sampling* algorithm.

[4] Interestingly, this condition was rediscovered in [10] as an alternative to the enhanced one-wayness condition of [6, Apdx. C.1]. Actually, Haitner [10] only requires noticeable domains (and refers to collections of permutations that satisfy this condition by the term *collections having dense domains*).

Construction 3.1 (simple version): *Let* $\mathbf{F}$ *be a family of permutations with an index-selection algorithm* S *that uses* $q(\cdot)$ *coins and having domains* D_i*'s that are subsets of* $\{0,1\}^{l(|i|)}$*, for some function* $l(\cdot)$*. We construct the function* f *as follows*

$$f(r,s) \stackrel{\rm def}{=} \begin{cases} (r, f_i(s)) \text{ if } s \in D_i, \text{ where } i \stackrel{\rm def}{=} S(n,r) \\ (r,s) \qquad \text{otherwise} \end{cases}$$

where $r \in \{0,1\}^{q(n)}$ *and* $s \in \{0,1\}^{l(n)}$.

Proposition 3.1 (analysis of Construction 3.1): *The function* f *is 1-1 and length preserving. If* $\mathbf{F}$ *is a family of one-way permutations satisfying the additional conditions of Definition 3.1, then* f *is weakly one-way. The latter holds even if* $\mathbf{F}$ *is only weakly one-way* (as long as it satisfies the additional conditions).

Proof: By definition f is length-preserving. Let G_n be the set of pairs $(r,s) \in \{0,1\}^{q(n)} \times \{0,1\}^{l(n)}$ such that $s \in D_{S(n,r)}$ holds and let B_n be the set of the other pairs (i.e., $B_n = (\{0,1\}^{q(n)} \times \{0,1\}^{l(n)}) \setminus G_n$). The key observation is that if $(r,s) \in G_n$, then for $i = S(n,r)$ it holds that $s \in D_i$. Furthermore, in that case, $f_i(s) \in D_i$ and $f(r,s) = (r, f_i(s)) \in G_n$ follows. On the other hand, if $(r,s) \in B_n$, then $f(r,s) = (r,s) \in B_n$. Thus, f maps G_n (resp., B_n) to itself and furthermore the mapping induced on G_n (rep., B_n) is 1-1. It follows that f is 1-1.

The function f is polynomial-time computable by virtue of the first two efficiency conditions of $\mathbf{F}$ and the additional 'recognizable domain' condition. By the additional 'noticeable domain' condition we know that G_n forms a noticeable fraction of $G_n \cup B_n$ and by the 'augmented one-wayness' condition we infer that f is hard to invert on G_n. Thus, we conclude that f is weakly one-way. In fact, the latter conclusion remain valid even if the family of permutations $\mathbf{F}$ is only weakly one-way. $\square$

Remark: The function f (constructed above) may be only weakly one-way, since it equals the identity transformation for a part of its domain (and this part may have a noticeable measure). To get a (strongly) one-way function, one may apply the transformation in [7] (cf. [5, Sec. 2.6]) to the function f. (In fact, degenerate versions of the transformation in [7] suffice for this purpose; see Section 5.)

Handling the non-simplified version of Definition 2.2. The above construction is stated with respect to the simplified definition of a family of one-way permutations. Recall that in the non-simplified version, the index-selecting algorithm, S, is only required to have an output with noticeable probability (i.e., the probability is at least $1/p(n)$, where p is some fixed positive polynomial). Furthermore, S is allowed to err (i.e., have output not in I) with a negligible probability. For the general case, we redefine the function f as follows.

Construction 3.2 (general version): *Let* $\mathbf{F} = \{f_i : D_i \stackrel{1\text{-}1}{\rightarrow} D_i\}_{i \in I}$ *be a family of permutations with an index-selecting algorithm,* S*, that produces output with noticeable probability and errs with negligible probability. We construct the function*

f as follows

$$f(r,s) \stackrel{\text{def}}{=} \begin{cases} (r, f_i(s)) & \textit{if } i \stackrel{\text{def}}{=} S(n,r) \neq \perp \textit{ and } s \in D_i \\ (r,s) & \textit{otherwise} \end{cases}$$

where the convention is that if on input n and coin tosses $r \in \{0,1\}^{q(n)}$ the algorithm S halts with no output, then $S(n,r) \stackrel{\text{def}}{=} \perp \notin \{0,1\}^*$.

Proposition 3.2 (analysis of Construction 3.2): *The function f is length preserving and almost 1-1. Furthermore, f is 1-1 if S never errs. If $\mathbf{F}$ is a family of one-way permutations satisfying the additional conditions of Definition 3.1, then f is weakly one-way. The latter holds even if $\mathbf{F}$ is only weakly one-way* (and satisfies the additional conditions).

Proof: In case algorithm S never errs, the proof is similar to the proof of Proposition 3.1 (i.e., G_n is redefined as the set of all pairs (r,s) such that $i \stackrel{\text{def}}{=} S(n,r) \neq \perp$ and $s \in D_i$). Otherwise, we observe that the collision probability of f is bounded above by the probability that S errs (and outputs a string not in I). Since this happens with negligible probability, we are done. □

4 Applying the Transformation

Using the transformation specified in the previous section, we show how to construct a 1-1 one-way function based on one of several popular intractability assumptions. To this end, we use these intractability assumptions in order to construct families of one-way permutations satisfying the additional conditions of Definition 3.1. Before presenting these constructions, we wish to stress an important aspect regarding them; namely, their (quantified) level of "security" (see next).

Security

The security of a one-way function f is a function, $s : \mathbb{N} \to \mathbb{N}$, specifying the amount of "work" required to invert f on inputs of given length. The work of an algorithm is defined as the product of the running-time (of the inverting algorithm) and the inverse of its success probability; namely, $w_A(n) \stackrel{\text{def}}{=} t_A(n) \cdot \frac{1}{p_A(n)}$, where $t_A(n)$ is the running time of A on f-images of length n and $p_A(n) \stackrel{\text{def}}{=} \text{Prob}_{x \in \{0,1\}^n}[A(f(x)) \in f^{-1}f(x)]$ is its success probability.

Typical cryptographic constructions, and in particular our constructions, transform one object (in our case, a family of one-way permutations) of security $s(\cdot)$ into another object (in our case, a single 1-1 one-way function) of related security $s'(\cdot)$. The relation between s and s' is of key importance. A weak relation, which is usually easier to obtain, is that $s'(\text{poly}(n)) > s(n)/\text{poly}(n)$. Although this relation translates any super-polynomial security s into a super-polynomial security s', it is of limited practical value. In order to use the resulting object of security s' one may needs to use very big instances. For example, if $s'(n^5) = s(n)$ and the original object is "secure in reality" for instance size 100 (bits), then the resulting object (of security s') will be "secure in reality" only for instances of size 10^{10} (and is thus unlikely to be of practical value). Thus, stronger relation between the security s of the original object and the security s' of the resulting object are of more value. In particular, it is desirable to have $s'(O(n)) > s(n)/\text{poly}(n)$, in which case we say that the transformation preserves the security.

Getting back to the constructions of the previous section, we note that the security of the resulting one-way 1-1 function f, on f-images of length $q(n) + l(n)$, is closely related to the security of the family of one-way permutations on f_i-images of length $l(n)$. (Recall, n denotes the length of the index of the permutation, $l(n)$ the length of the description of elements in the domain of the permutation, and $q(n)$ the randomness complexity of the index-selecting algorithm.) Thus, $s'(q(n)+l(n)) > s(l(n))/\text{poly}(n)$, where s denotes the security of the family $\mathbf{F}$ and s' the security of the function f. Therefore, *the smaller the polynomial $q(\cdot)$ is, the better security one gets.* It is particularly desirable to keep $q(n)$ linear in $l(n)$. All the constructions presented below achieve this goal. Consequently, *the one-way functions constructed below preserve the security of the intractability assumption on which they are based.* We remark that the (weak to strong one-way) transformation of [7] (mentioned in the Remark in Section 3) preserves security too.

Preliminaries: Selecting prime numbers

Prime numbers play a key role in all our constructions, and so efficient algorithms for selecting such numbers are of key importance to us. We will use two algorithms, the first being being the celebrated deterministic polynomial-time primality tester of Agrawal, Kayal, and Saxena [1],

Theorem 4.1 (primes are in $\mathcal{P}$): *There exists a deterministic polynomial-time primality tester; that is, an algorithm that decides whether a given integer is a prime number.*

The second algorithm is Bach's probabilistic polynomial-time algorithm that on input 1^n uniformly generates an n-bit long composite number along with its factorization [2]. A straightforward implementation of Bach's algorithm requires a super-linear number of coin tosses (i.e., a number of coin tosses that is super-linear in the length of the composite being generated). Here we claim an approximate version that uses a linear number of coin tosses. We say that a distribution X on n-bit long strings is almost uniformly distributed over $S \subseteq \{0,1\}^n$ if the variation distance between X and the uniform distribution over S is negligible (as a function of n).

Theorem 4.2 (randomness efficient generation of integers with known prime factorization): *There exists a probabilistic polynomial-time algorithm that, on* *input 1^n, uses $O(n)$ coin tosses to select a random number N almost uniformly in the interval $[2^{n-1}, 2^n - 1]$, and outputs the prime factorization of N.*

Proof: While it is possible to present a direct implementation of an approximate version of Bach's algorithm that uses only a linear number of coin tosses, the details are quite tedious. Hence, we prefer to invoke a general result of Nisan and Zuckerman [15] that asserts that *any probabilistic polynomial-time algorithm that uses linear space has an approximated version that uses a linear number of coin tosses*, where approximation means that the output distribution of the new algorithm (on any fixed input) is statistically close to the output distribution of the original algorithm (i.e., the variation distance is negligible). It is easy to see that Bach's algorithm utilizes linear space, and the theorem follows. $\square$

4.1 A construction based on RSA

The standard presentation of RSA [18] yields a family of permutations, which is believed to be one-way, but is certainly *not* one-way in the augmented sense of Definition 3.1. Here we refer to a family in which the indices are pairs (N, e), where N is the product of two primes of equal length and e is relatively prime to $\phi(N)$. The index is generated by randomly selecting these two primes, multiplying them and next selecting a proper e. Thus, giving these random choices away compromises the security of RSA, since given the prime factors it is easy to invert the function.

We consider, instead, the following family of weak one-way permutations. The indices in this family are pairs of integers (N, P) such that P is a prime and $|P| = |N|$. For each such pair we define a permutation over $\mathbb{Z}_N^*$, the multiplicative group modulo N; specifically, $f_{N,P}(x) \stackrel{\text{def}}{=} x^P \bmod N$. Note that we do not insist that N is a product of two primes of the same length. Yet, a noticeable fraction of the possible N's will have this form. Thus, if the standard RSA family is strongly one-way (for random exponent) then it is also (strongly) one-way for a prime exponent, and consequently the foregoing (non-standard) family of functions will be weakly one-way (due to the noticeable fraction of composites of the standard form). Since P is relatively-prime to $\phi(N)$, the functions in this family are in fact permutations over $\mathbb{Z}_N^*$. (Note that the index-selecting algorithm does not know $\phi(N)$, and so relative-primality of P and $\phi(N)$ is imposed by requiring that P is prime.)

We now show that the foregoing family satisfies the non-simplified requirements (from a family of one-way permutations) as well as the additional conditions in Definition 3.1. Among the efficiency conditions of Definition 2.2 only the one referring to the index selection is problematic, yet it does hold when only requiring that output is produced merely with noticeable probability; specifically, we select two n-bit integers at random and check whether the second is prime, producing an output only if the answer is affirmative. Furthermore, $\mathbb{Z}_N^*$ is easily recognizable and has noticeable density with respect to $\{0,1\}^{|N|}$. This family is one-way in the augmented sense (under the "RSA assumption"), since the modulus is generated via an identity transformation from the coins of the index-selecting algorithm (and thus these coins add no knowledge to the inverter). It follows that we can apply Proposition 3.2 and derive a length-preserving 1-1 one-way function.

Definition 4.1 (standard RSA Assumption): *We say that* inverting RSA is intractable with security $s(\cdot)$ *if any algorithm for the* inverting task *uses work greater than* $s(\cdot)$*. The inverting task consists of finding* x *such that* $y = x^e \bmod N$*, when given* N*,* e *and* y*, where* N *is uniformly selected among all composites that are the product of two* $(n/2)$*-bit long primes,* e *is uniformly selected among the elements of the multiplicative group modulo* $\phi(N)$*, and* y *is uniformly selected among the elements of the multiplicative group modulo* N*.*

To justify our claim that the security (of the RSA Assumption) is preserved, we note that pairs (N, P) as required can be selected using $O(|(N, P)|)$ random bits.[5] Thus, we get

Corollary 4.1 (a length-preserving 1-1 one-way function based on RSA): *Suppose that inverting RSA is intractable with security* $s(n)$*. Then, there exists a length-preserving 1-1 one-way function with security* $s'(O(n)) \stackrel{\rm def}{=} s(n)/\mathrm{poly}(n)$*.*

4.2 A construction based on a restricted DLP

Here we rely on the assumption that the Discrete Logarithm Problem (DLP) in the multiplicative group modulo P is hard also for the special case of primes of the form $P = 2Q + 1$, where Q is a prime. We also use the assumption that such primes form a noticeable fraction of the integers of the same length. Based on these assumptions, the following family of permutations is one-way. The indices in the family are pairs (P, g), where P is a prime of the above form and g is a primitive element modulo P. The index is selected by first selecting a prime of the above form and next using the known factorization of $\phi(P) = 2Q$ to test candidates for primitivity (see details below). For each index, (P, g), we define a permutation over $\mathbb{Z}_P^*$, the multiplicative group modulo P; specifically, $f_{P,g}(x) \stackrel{\rm def}{=} g^x \bmod P$. Noting that $\mathbb{Z}_P^*$ is both 'noticeable' and easy to recognize, we can apply Proposition 3.2, provided that the index-selection process satisfies the augmented one-way condition.

To address the last concern as well as justify our claim that the resulting 1-1 one-way function preserves the security of the family, we need to specify the way in which the pairs (P, g) are selected. On input n we uniformly select an $(n-1)$-bit integer, Q, and test Q and $P = 2Q + 1$ for primality. In case

[5] Currently, the random bits are merely used to select (N, P) uniformly among all pairs of n-bit long integers. Indeed, checking primality is done by using the deterministic algorithm guaranteed in Theorem 4.1 (whereas in the earlier versions Bach's randomness-efficient algorithm [3] was used for that purpose, which only resulted in an almost 1-1 function).

we are successful, we uniformly select $g \in \mathbb{Z}_P^*$ and test if it is primitive (mod P) by computing $g^{P-1} \bmod P$, $g^Q \bmod P$ and $g^2 \bmod P$. (If the first expression evaluates to 1 whereas the other two do not, then g is a primitive element modulo P.)[6] Thus, we use $|(P,g)|$ random bits to generate pairs (P,g), and these coins are identical to the pairs themselves. Combining this with the foregoing assumption regarding the density of primes of the desired form and the fact that in this case approximately half the elements of $\mathbb{Z}_P^*$ are primitive, we get

Corollary 4.2 (a length-preserving 1-1 one-way function based on restricted DLP): *Suppose that the restricted DLP is intractable with security* $s(n)$ (as in Definition 4.2)*, and that the set of n-bit primes, P, for which $\phi(P)/2$ is prime, constitute a* $1/\text{poly}(n)$ *fraction of the n-bit long integers. Then, there exists a length-preserving 1-1 one-way function with security* $s'(O(n)) \stackrel{\text{def}}{=} s(n)/\text{poly}(n)$.

Definition 4.2 (restricted DLP Assumption): *We say that* the restricted DLP is intractable with security $s(\cdot)$ *if any algorithm for the following* inverting task *uses work greater than $s(\cdot)$. The inverting task consists of finding x such that $y = g^x \bmod P$, when given P, g and y, where P is uniformly selected among all n-bit primes for which $\phi(P)/2$ is prime, g is uniformly selected among the primitive elements modulo P, and y is uniformly selected among the elements of the multiplicative group modulo P.*

4.3 A construction based on the general DLP

Here we rely on a alternative assumption concerning the DLP. Specifically, we assume that the Discrete Logarithm Problem (DLP) in the multiplicative group modulo a prime P is hard also when given the factorization of $\phi(P)$. (Note that for primes of the special form $P = 2Q+1$ the factorization of $\phi(P) = 2 \cdot Q$ is always known.) Furthermore, we shall assume that this DLP problem remains hard when given any $O(|P|)$ bits of information regarding P; that is, we assume that there are no trapdoors (of linear length) for the DLP in the multiplicative group modulo a prime. Making this assumption, we can waive the assumption made in the previous subsection concerning the density of primes of special form $P = 2Q+1$, where Q is a prime.

Based on the foregoing intractability assumption, the following family of permutations is one-way. The indices in the family are pairs (P,g), where P is a prime and g is a primitive element modulo P. The index is chosen by first generating a random prime P with known factorization of $\phi(P)$ (see details below), and next using this factorization to test candidates for primitivity. For each index, (P,g), we define a permutation over $\mathbb{Z}_P^*$ as before (i.e., $f_{P,g}(x) \stackrel{\text{def}}{=} g^x \bmod P$). Again, we shall apply Proposition 3.2 to the current family.

We have postponed the discussion of how to randomly generate primes P with known factorization of $\phi(P)$. Here is where we use Theorem 4.2, which

[6] Again, checking primality is done by using the deterministic algorithm guaranteed in Theorem 4.1. Indeed, this allows to obtain a 1-1 function (rather than an almost 1-1 function, as in earlier versions).

asserts the existence of an adequate algorithm and furthermore one that uses a linear number of coin tosses. This yields an index-selection algorithm that selects pairs (P, g) using $O(|(P, g)|)$ random bits, which is instrumental to our claim that the resulting 1-1 one-way function preserves the security of the family. The fact that the coins used by this index-selection algorithm provide additional information on P is "covered" by the assumption formulated in Definition 4.3. Thus, we get:

Corollary 4.3 (a length-preserving 1-1 one-way function based on general DLP): *Suppose that DLP is intractable with security $s(n)$, even when the factorization of the order of the group is given* (as in Definition 4.3). *Then, there exists a length-preserving 1-1 one-way function with security $s'(O(n)) \overset{\text{def}}{=} s(n)/\text{poly}(n)$.*

Definition 4.3 (DLP Assumption): *We say that* the DLP is intractable with security $s(\cdot)$ *if, for every randomized mapping Π such that $|\Pi(z)| = O(|z|)$, any algorithm for the following* inverting task *uses work greater than $s(\cdot)$. The inverting task consists of finding x such that $y = g^x \bmod P$, when given $\Pi(P)$ and a pair (g, y), where P is uniformly selected among all n-bit primes, g is uniformly selected among the primitive elements modulo P, and y is uniformly selected among the elements of the multiplicative group modulo P.*

The randomized mapping Π captures possible trapdoor information that may assist in inverting $f_{P,g}$, and the assumption asserts that the inverting task remains hard also in the presence of such information.[7] In particular, the randomized mapping Π may yield the coins used by the factored-number generating algorithm of Theorem 4.2. Thus, inverting $f_{P,g}$ is hard also in the augmented sense of Definition 3.1.

5 Conclusions and Open Problems

We have presented a method for constructing (strongly) one-way permutations. The method consists of three steps.

Step (1): Using well-known intractability assumptions to construct families of one-way permutations satisfying the additional properties specified in Definition 3.1.
Step (2): Using such a family to construct a weak one-way function.
Step (3): Transforming the resulting function into a strongly one-way function.

We consider the identification of the conditions in Definition 3.1 and the construction of families of one-way permutations satisfying these conditions to be the most important contributions of the current paper. Thus, most of the paper

[7] Indeed, the fact that Π is applied only to P and that $|\Pi(P)| = O(|P|)$ makes the assumption potentially weaker. On the other hand, the fact that $\Pi(P)$ may contain P allows us to omit P from the list of inputs to the inverting task.

is dedicated to the implementation of Step (1), whereas Step (2) is obtained by Construction 3.2, and Step (3) is obtained by referring to [7].

Regarding Step (3), we remark that applying the general ("weak to strong") transformation of [7] seems an over-kill, since in our case the weakly one-way function f has a special structure (e.g., it is hard to invert almost on all points on which it is not the identity transformation). However, in our attempts to avoid using [7], we were not able to avoid using random walks on expander graphs (for the repeated attempts to generate a valid index and/or a sample in the corresponding domain). Since expander graphs are the only non-elementary component of [7], we see no point in presenting these alternatives here. Certainly, it will be better to avoid the use of expander graphs and perform Step (3) in a more efficient manner.

Another obvious open problem is to construct length-preserving 1-1 one-way functions based on the conjectured intractability of factoring.[8] To achieve this goal using our method one will need to construct a family of one-way permutations satisfying the additional properties specified in Definition 3.1. (The standard construction of a family of one-way permutations based on factoring [16] does not satisfy the augmented one-wayness condition.)

Acknowledgments

We would like to thank Eric Bach and Hugo Krawczyk for helpful discussions and comments.

References

1. M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. *Annals of Mathematics*, Vol. 160 (2), pages 781–793, 2004.
2. E. Bach, *Analytic Methods in the Analysis and Design of Number-Theoretic Algorithms* (ACM Distinguished Dissertation 1984), MIT Press, Cambridge MA, 1985.
3. E. Bach, "Realistic Analysis of some Randomized Algorithms", *19th STOC*, pages 453–461, 1987.
4. M. Blum and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM J. on Computing*, Vol. 13, pages 850–864, 984.
5. O. Goldreich. *Foundation of Cryptography: Basic Tools*. Cambridge University Press, 2001.
6. O. Goldreich. *Foundation of Cryptography: Basic Applications*. Cambridge University Press, 2004.
7. O. Goldreich, R. Impagliazzo, L. Levin, R. Venkatesan and D. Zuckerman, "Security Preserving Amplification of Hardness", *31st FOCS*, pages 318–326, 1990.

[8] We comment that, using Theorem 4.1, one can obtain a 1-1 function based on factoring, alas this function is not length preserving. Specifically, consider the function that maps a pair of integers of the form (x, y) to their multiple if $x < y$ and both numbers are prime, and maps it to (x, y) otherwise. Using an adequate encoding that distinguishes the two cases, this mapping is 1-1, but not length preserving.

8. O. Goldreich and L. Levin, "A Hard-Core Predicate for any One-way Function", *21st STOC*, pages 25–32, 1989.
9. O. Goldreich, H. Krawczyk and M. Luby, "On the Existence of Pseudorandom Generators", *SIAM J. on Computing*, Vol. 22, pages 1163–1175, 1993.
10. I. Haitner. Implementing Oblivious Transfer Using a Collection of Dense Trapdoor Permutations. Master thesis, Weizmann Institute of Science, January 2004. An extended abstarct appeared in the *1st TCC*, Springer, LNCS Vol. 2951, pages 394–409, 2004.
11. I. Haitner, O. Reingold, and S. Vadhan. Efficiency Improvements in Constructing Pseudorandom Generator from any One-way Function. In *42nd STOC*, pages 437–446, 2010.
12. J. Håstad, R. Impagliazzo, L.A. Levin and M. Luby. A Pseudorandom Generator from any One-way Function. *SICOMP*, Volume 28, Number 4, pages 1364–1396, 1999. Combines papers of Impagliazzo, Levin, and Luby (*21st STOC*, 1989) and J. Håstad (*22nd STOC*, 1990).
13. G.L. Miller, "Riemann's Hypothesis and tests for primality", *JCSS*, Vol. 13, pages 300–317, 1976.
14. M. Naor and M. Yung, "Universal Hash Functions and their Cryptographic Applications", *21st STOC*, pages 33–43, 1989.
15. N. Nisan and D. Zuckerman. Randomness is Linear in Space. *JCSS*, Vol. 52 (1), pages 43–52, 1996. Preliminary version in *25th STOC*, 1993.
16. M.O. Rabin, "Digitalized Signatures and Public Key Functions as Intractable as Factoring", MIT/LCS/TR-212, 1979.
17. M.O. Rabin, "Probabilistic algorithm for testing primality", *Jour. of Number Theory*, Vol. 12, pages 128–138, 1980.
18. R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *CACM*, Vol. 21, pages 120–126, 1978.
19. J. Rompel, "One-way Functions are Necessary and Sufficient for Secure Signatures", *22nd STOC*, pages 387–394, 1990.
20. R. Solovay and V. Strassen, "A fast Monte-Carlo test for primality", *SIAM Jour. on Computing*, Vol. 6, pages 84–85, 1977.