Constructing Pseudo-Random Permutations with a Prescribed Structure

Moni Naor[*] Omer Reingold[†]

Abstract

We show how to construct pseudo-random permutations that satisfy a certain cycle restriction, for example that the permutation be cyclic (consisting of one cycle containing all the elements) or an involution (a self-inverse permutation) with no fixed points. The construction can be based on any (unrestricted) pseudo-random permutation. The resulting permutations are defined succinctly and their evaluation at a given point is efficient. Furthermore, they enjoy a *fast forward* property, i.e. it is possible to iterate them at a very small cost.

1 Introduction

A family of permutations $\mathcal{P}_\ell = \{P_k : \{0,1\}^n \mapsto \{0,1\}^n | k \in \{0,1\}^\ell\}$ is called (cryptographic) *pseudo-random* if it satisfies the following:

Succinct Representation: For a permutation $P_k \in \mathcal{P}$, k can be thought of as the key. The length ℓ of k should be small i.e. polynomial in n.

Efficient Computation: Given $k \in \{0,1\}^\ell$ and $x \in \{0,1\}^n$ computing $y = P_k(x)$ can be done efficiently. Similarly, given $y \in \{0,1\}^n$ computing $x = P_k^{-1}(x)$ can be done efficiently. Efficiently means in time polynomial in ℓ.

Indistinguishability: It is computationally infeasible to distinguish whether a given permutation τ is (i) a random member of the family $\mathcal{P}$ or (ii) a truly random permutation on n-bit strings. The access the distinguisher has to the permutation τ is black-box, i.e. it can give x and obtain $\tau(x)$ and give y and obtain $\tau^{-1}(y)$ for x's and y's of its choice[1].

We measure the advantage ε of distinguishing cases (i) and (ii) as a function of m, the number of times the distinguisher gets to call the black-box for τ (in either direction) and t, the running time of the distinguisher. Ideally ε should be a negligible function in ℓ times a polynomial in t and m.

Pseudo-random permutations were defined by Luby and Rackoff [5] as a formalization of block-ciphers. They also showed how to construct such families based on pseudo-random functions,

[*]Dept. of Computer Science and Applied Math, Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: naor@wisdom.weizmann.ac.il. Part of this work was done while visiting Stanford University and the IBM Almaden Research Center.

[†]AT&T Labs - Research. Room A243, 180 Park Avenue, Bldg. 103, Florham Park, NJ, 07932, USA. E-mail: omer@research.att.com.

[1]Sometimes a distinction is made as to whether the inverse permutation is available to the adversary or not, but we always assume that it is available.

as defined by Goldreich Goldwasser and Micali [2]. These permutations and constructions have received a lot of attention since then (cf. [6]).

Suppose now that we are interested in constructing a pseudo-random *cyclic* permutation, i.e. a family $\mathcal{C}$ of cyclic permutations (the cycle type of a cyclic permutation consists of a single cycle that contains all the elements) whose members cannot be distinguished from a random cyclic permutation. Such a question arises, for instance, from the work of Shamir and Tsaban [8] who wanted to succinctly define and generate a *non-repeating* sequence of randomly looking n-bit values $X_1, X_2 \dots$. If one has a random looking cyclic permutation τ, then defining $x_1 = \tau(1)$ and $X_{i+1} = \tau(X_i)$ yields such a sequence.

In this work we show that it is possible to solve this problem and in fact a more general one. For any fixed cycle type, it is possible to construct a family of permutations with the prescribed type that is indistinguishable from a random permutation of this type.

1.1 Definitions

We now formally define the permutations we are trying to construct. A cycle type of a permutation τ is a list stating how many cycles of each size there are in τ. E.g. if $\tau = (164)(57)(238)$, then the cycle type of τ is "one cycle of size 2 and two cycles of size 3" (which can be denoted by $\{(2,1)(3,2)\}$). Let C be a cycle type. We say that a family of permutations $\mathcal{F}_C = \{P_k : \{0,1\}^n \mapsto \{0,1\}^n | k \in \{0,1\}^\ell\}$ is *pseudo-random* of type C if it satisfies the following:

Cycle type: Each $P_k \in \mathcal{F}_C$ has cycle type C.

Succinct Representation: The length ℓ of k (the key of P_k) should be small (polynomial in n).

Efficient Computation: Given $k \in \{0,1\}^\ell$ and $x \in \{0,1\}^n$ computing $y = P_k(x)$ can be done efficiently. Similarly, given $y \in \{0,1\}^n$ computing $x = P_k^{-1}(x)$ can be done efficiently (in time polynomial in ℓ.)

Indistinguishability: It is computationally infeasible to distinguish whether a given permutation τ is (i) a random member of the family $\mathcal{F}_C$ or (ii) a truly random permutation of cycle type C. The access the distinguisher has to the permutation τ is black-box, i.e. it can give x and obtain $\tau(x)$ and give y and obtain $\tau^{-1}(y)$ for x's and y's of its choice.

We measure the advantage ε of distinguishing cases (i) and (ii) as a function of m, the number of times the distinguisher gets to call the black-box for τ (in either direction) and t, the running time of the distinguisher.

2 The construction

Let C be a cycle type of permutations on $N = 2^n$ elements. Let σ be some fixed permutation on n-bit strings whose cycle type is C. We assume that it is easy given x to find $\sigma(x)$ as well as $\sigma^{-1}(x)$. For instance, if we are interested in cyclic permutations, then σ can be $(0, 1, 2, \dots 2^n - 1))$. Let $\mathcal{P}_\ell = \{P_k : \{0,1\}^n \mapsto \{0,1\}^n | k \in \{0,1\}^\ell\}$ be a family of pseudo-random permutations. Then $\mathcal{F}_C$, the family of pseudo-random permutations of cycle type C is defined as

$$\mathcal{F}_C = \{F_k = P_k \circ \sigma \circ P_k^{-1} | k \in \{0,1\}^\ell\}.$$

In other words, a permutation in $\mathcal{F}_C$ is determined by an ℓ-bit key that define a permutation $P_k \in \mathcal{P}_\ell$. To evaluate $F_k(x)$ one computes $P_k^{-1}(\sigma(P_k(x)))$. In order to evaluate $F_k^{-1}(y)$ one computes $P_k^{-1}(\sigma^{-1}(P_k(y)))$. Both directions require two invocations of the original pseudo-random permutation and a single evaluation of σ or σ^{-1}.

Why does it work? The fact that the members of $\mathcal{F}_C$ have the desired cycle type follows from a well known theorem in elementary group theory that states that the cycle structures of the permutations σ and $\pi \circ \sigma \circ \pi^{-1}$ are the same[2]. We can show an even stronger statement:

Theorem 1 *Let σ be a some permutation with cycle type C and let π be a random permutation, then the permutation $\pi \circ \sigma \circ \pi^{-1}$ is distributed uniformly among the permutations with the same cycle type C as σ.*

Proof: Let σ, τ and τ' be three permutations with cycle type C. Define two sets of permutations $\Pi = \{\pi | \ \tau = \pi \circ \sigma \circ \pi^{-1}\}$ and $\Pi' = \{\pi | \ \tau' = \pi \circ \sigma \circ \pi^{-1}\}$. It is enough to show that Π and Π' have the same size. The main observation is that there exists a permutation P such that $\tau' = P \circ \tau \circ P^{-1}$. Given this claim we get a 1-to-1 and onto mapping between Π and Π': every permutation $\pi \in \Pi$ is mapped to $\pi' = P \circ \pi$. It remains to prove the claim. Let $(i_0, i_1, \ldots i_c)$ be a cycle in τ and $(i'_0, i'_1, \ldots i'_c)$ be a cycle of the same length in τ'. It is easy to see that $(P^{-1}(i_0), P^{-1}(i_1), \ldots P^{-1}(i_c))$ is a cycle in $P \circ \tau \circ P^{-1}$. Therefore, if we define $P(i'_j) = i_j$ for $j = 1 \ldots m$, we get that the cycle in τ is mapped to a corresponding cycle in τ'. To define P such that $\tau' = P \circ \tau \circ P^{-1}$, we can just continue mapping all of the cycles in τ to a unique corresponding (same length) cycle in τ'. Note that this (arbitrary) correspondence between cycles is possible since τ and τ' have the same cycle type. $\Box$

From this theorem we can deduce the security of the construction:

Theorem 2 *Suppose that we have an adversary D that can distinguish with advantage at least ε whether a given permutation is (i) a random member of $\mathcal{F}_C$ or (ii) a random permutation with cycle type C, while making m calls to the permutation and running in time t. Then there is a distinguisher D' for the family $\mathcal{P}$ that runs in time $O(t)$ and makes $2m$ calls to the input permutation and has ε advantage of distinguishing a member of $\mathcal{P}$ from a truly random permutation.*

Proof: The theorem follows from a simulation argument: given π as a black-box, D' simulates D on the permutation $\tau = \pi \circ \sigma \circ \pi^{-1}$: whenever D requests to evaluate its input permutation τ on a point x, D' requests for $\pi(x)$ and then requests for π^{-1} on $\sigma(\pi(X))$; it then feeds D with the result (a similar process is done when D requests the inverse of x. D' outputs the same guess ('random'/'pseudo-random') as D.

Clearly if D makes m calls to the input permutation, then D' makes $2m$ calls. Let $D[\tau]$ denote the output of D when the input permutation is τ. From Theorem 1 we can conclude that

$$\Pr[D[\tau] = \text{'random'} \ |\tau \text{ is random of cycle type } C] = \Pr[D'[\pi] = \text{'random'} \ |\pi \text{ is random}]$$

and by the definition of the construction

$$\Pr[D[\tau] = \text{'pseudo-random'} \ |\tau \in \mathcal{F}_C] = \Pr[D'[\pi] = \text{'pseudo-random'} \ |\pi \in \mathcal{P}].$$

Hence if D distinguishes with advantage ε, so does D'. The number of calls to the input permutation D' makes is twice that of D, The running time of D' is similar to that of D, except that D' needs to call σ for each operation. Assuming that this can be done in $O(1)$ time, we have the desired result. $\Box$

[2]Körner [3] says that some may label this fact as a candidate for the dullest theorem, but it turns out to have played an important role in cryptography (and world history), in the breaking of The Enigma, the Second World War German encryption machine.

3 Applications and Properties

Involutions An interesting family of permutations this method allows us to construct are pseudo-random involutions. An involution is a permutation that is the inverse of itself. The advantage of using such permutations for encryption is that the encryption operation and decryption operation are identical (this is not necessarily a good property for an encryption scheme, but it may be useful in some situations.) The encryption done by the Enigma was an involution.

The construction is exactly as in Section 2. Fix σ to be the involution mapping even i's to $i+1$ and odd i's to $i-1$. Then the resulting family $\mathcal{F}_I = \{F_k = P_k \circ \sigma \circ P_k^{-1} | k \in \{0,1\}^\ell\}$ is a family of pseudo-random involutions with no fixed-points.

t-wise independent permutations: The combinatorial counterpart to cryptographic pseudo-randomness is (almost) t-wise independence. While there are no known good constructions of exact t-wise independent permutations for $t > 3$, various approximations are possible (see [6] for a discussion). Suppose that we are interested in a family of t-wise independent permutations that has cycle type C, i.e. considering the permutation at any t values has the same distribution as a random permutation with cycle type C. If we have a family H of $2t$-wise independent permutations, then $H_C = \{h \circ \sigma \circ h^{-1} | h \in H\}$ is a t-wise independent family with cycle type C. This follows from appealing to Theorem 2. Similarly, if H is an *approximation* to a $2t$-wise independent permutation, then H_C is a related approximation to a t-wise independent family with cycle type C.

Fast Forward Property: The construction of $\mathcal{F}_C$ has the appealing property that it is possible to iterate F_k on itself at 'zero' cost. To compute $F_k^{(m)}(x)$ for any m, x and $F_k \in \mathcal{F}_C$ note that

$$F_k^{(m)}(x) = P_k^{-1}(\sigma(P_k(P_k^{-1}(\sigma \cdots)))) = P_k^{-1}(\sigma^{(m)}(P_k(x))).$$

Therefore, assuming σ is such where fast forward is possible, then computing $F_k^{(m)}(x)$ has of the same complexity as $F_k(x)$. For instance, in the case of the cyclic permutation, performing m iterations amounts to computing $P_k^{-1}(P_k(x) + m \bmod 2^n)$.

We can therefore allow the adversary queries of the form (x, m) that will be answered by $F_k^{(m)}(x)$. Here, again, a simple adaptation of Theorem 2 implies that such queries cannot enable distinguishing F_k from a random permutation of the given cycle type, unless $\mathcal{P}$ is weak as well.

Finally, another operation that can be performed efficiently and is relevant when the cycle type contains cycles of medium length is to test whether two elements x_1 and x_2 are in the same cycle of the given permutation F_k. If x_1 and x_2 are in the same cycle, then there exists an m such that $x_2 = F_k^{(m)}(x_1)$ and therefore $P_k(x_2) = \sigma^{(m)}(P_k(x_1))$. Therefore x_1 and x_2 are in the same cycle iff $P_k(x_2)$ and $P_k(x_2)$ are in the same cycle in σ (which we assume can be determined easily).

4 Open Problems

We showed how to construct a family of pseudo-random permutations where it is possible to quickly iterate the permutation. However, this works only for a fixed cycle type. The question is whether it is possible to construct a family of permutations such that (i) The cycle type distribution is close to that of a random permutation (ii) It is possible to iterate a member of the family very quickly. (iii) The family is indistinguishable from truly random permutations even with the fast iteration queries.

Using the approach of this paper it is sufficient to construct a family of permutations $\mathcal{C}$ that satisfies (i) and (ii). Composing it with a (regular) pseudo-random permutation will yield property (iii) as well. Note that a pseudo-random cyclic permutation satisfies properties (ii) and (iii) (but not (i)), since it is not easy to distinguish such a permutation from a random one (it should require roughly $2^{n/2}$ evaluations).

Another interesting question is whether it is possible to construct pseudo-random *functions* that can be iterated. The need for such functions (or the k-wise independent version of them) arises in algorithmic applications such as Pollard's rho method [7] and Hellman's time-space tradeoff for inverting function ([4], see also [1]). Here, again, the approach of this paper tells us that it is sufficient to come up with a family $\mathcal{F}$ of functions that has the correct distribution on the "tree structure" as well as the ability to compute iterations, but is not necessarily pseudo-random. Then for any $\mathcal{P}$ that is a (regular) family of pseudo-random permutations, the family $\{P^{-1} \circ F \circ P | F \in \mathcal{F}, P \in \mathcal{P}\}$ has all the desired properties.

References

[1] A. Fiat, M. Naor, Rigorous Time/Space Trade-offs for Inverting Functions, *SIAM J. Comput.* 29, 1999, pp. 790–803.

[2] O. Goldreich, S. Goldwasser and S. Micali, How to construct random functions, *J. of the ACM.*, vol. 33, 1986, pp. 792-807.

[3] T. W. Körner, **The Pleasure of Counting**, Cambridge Univ. Press, 1997.

[4] M. E. Hellman, A cryptanalytic time-memory tradeoff, *IEEE Transactions on Information Theory* 26, 1980, pp. 289–294.

[5] M. Luby and C. Rackoff, How to construct pseudorandom permutations and pseudorandom functions, *SIAM J. Comput.* 17, 1988, pp. 373-386.

[6] M. Naor and O. Reingold, On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited. *Journal of Cryptology* 12(1), 1999, pp. 29–66.

[7] J. M. Pollard, A Monte Carlo method for factorization, *BIT* 15, 1975, pp. 331–334.

[8] A. Shamir and B. Tsaban, Guaranteeing the diversity of Pseudorandom Generators, Manuscript.