Groups-Complexity-Cryptology
Volume 1 (2009), No. 1, 13–31

Generic Case Complexity and One-Way Functions

Alex D. Myasnikov
Department of Mathematical Sciences, Stevens Institute of Technology, Hoboken, NJ 07030, USA
amyasnik@stevens.edu

Received: January 7, 2008

The goal of this paper is to introduce ideas and methodology of the generic case complexity to cryptography community. This relatively new approach allows one to analyze the behavior of an algorithm on "most" inputs in a simple and intuitive fashion which has some practical advantages over classical methods based on averaging.

We present an alternative definition of one-way function using the concepts of generic case complexity and show its equivalence to the standard definition. In addition we demonstrate the convenience of the new approach by giving a short proof that extending adversaries to a larger class of partial algorithms with errors does not change the strength of the security assumption.

1. Introduction

Generic case complexity has originated about a decade ago in combinatorial group theory [10, 2]. This area has long computational traditions with many fundamental problems being algorithmic in nature. It has been shown that most computational problems in infinite group theory are recursively undecidable. However, it was also observed that decision algorithms, sometimes very naive ones, exist for many inputs even if a problem is undecidable in general.

Generic complexity was suggested as a way of analyzing the behavior of undecidable problems. The main question was to describe the complexity of a problem on a *generic* input or on a set which contains most of the inputs. The idea was to separate sets of inputs where algorithms work from the "bad" ones. It happened that quite often the number of inputs on which algorithms fail to provide an answer are small.

In computer science, around 1980s, the same kind of arguments preceded the development of the average case complexity. More recently, heuristic classes of algorithms were introduced [1].

Advocates of generic complexity approach argue (see discussions in [4]) that it is simpler, intuitive and more general then the average case complexity. The connection between the two areas has been studied and it is known that there are

ISSN 1867-1144 / $ 2.50

problems which are hard on average, but generically easy. It turns out however, that if an algorithm is easy on average it is also easy generically.

The relation between generic complexity and heuristic complexity is less explored. It was shown [4] that the class of generic algorithms and errorless heuristic algorithms are equivalent. It seems that generic complexity has some advantage as the area has significantly progressed in recent years. For example the completeness theory for generic complexity has been developed.

Here we list some results in generic complexity. As we mentioned above, the foundations were built in group theory. In particular it has been shown that the famous word and conjugacy problems in finitely presented groups can be decided in linear time on a generic set of inputs, although these problems are undecidable in general [10].

In the scope of the classical complexity results, the most important is the existence of polynomial reductions for generic complexity. Using these reductions it has been shown that there exist generically NP-complete problems, for example bounded versions of the halting and Post correspondence problems are generically NP-complete [4]. Another interesting result shows that the halting problem for a model of a Turing machine with one-way infinite tape is linearly decidable on a generic set of inputs [8]. It is not known whether the result holds for an arbitrary Turing machine, but it was shown that the set on which the problems is decidable cannot be strongly generic [13].

In [11] authors describe a particular procedure which allows one given an undecidable problem to construct a problem undecidable on every generic set of inputs. This generic amplification shows that generically hard (undecidable) problems exist.

It was also suggested that generic complexity might be useful for cryptographic applications, particularly for testing security assumptions of cryptographic primitives. Intuitively, we would like a cryptographic primitive to be hard to break on most inputs which seems like a straightforward application of the ideas of generic complexity. The main goal of this paper is to introduce ideas and methodology of generic complexity to cryptography community. We present alternative definitions of one-way functions based on the concept of generic complexity.

These new definitions allow one to consider, in a natural way, one-way function candidates coming from undecidable problems. We show that any such "generic" one-way function can be used to produce a classical one. Therefore, any new generic one-way function comes along with new classical one. Furthermore, to our opinion these new definitions are more intuitive and are easier to work with. Indeed, the new security assumption is just a more precise formalization of the original notion, due to Diffie and Hellman [3], in a sense, it separates the probability on the inputs from the probability on the oracle choices which makes considerations easier. As an illustration, we give a short proof that extending adversaries to a larger class of partial algorithms with errors does not change the

strength of the security assumption.

In the subsequent paper we are going to discuss some potential generic one-way functions that are related to undecidable problems in algebra.

1.1. Generic complexity notations

In this section we give a brief overview of the basic notions and definitions used in generic complexity. For more detailed introduction to the subject and latest results we refer to [4].

Let I be a set of inputs. In this paper we consider traditional binary representation of inputs and set $I = \{0,1\}^*$. With each input we associate a size function $|\cdot| : I \to \mathbb{N}$ which is the length of a string from I.

First we define a *stratification* of inputs. In general a stratification of the set I is an ascending sequence of subsets whose union is equal to I. In the paper we will use the spherical stratification on strings which we define next.

Definition 1.1 (Spherical Stratification). Let $I = \{0,1\}^*$ be a set of inputs. Define a sphere of radius n by

$$I_n = \{x \mid x \in I, |x| = n\}.$$

Then the sequence $I_0, I_1, I_2, \ldots$ is a spherical stratification of I.

Note that sets I_i are finite and $\cup_{i=0}^{\infty} I_i = I$.

There are other commonly used stratifications available. For example one can stratify set I using balls B_n of inputs of radius n, where B_n is a set of inputs with lengths at most n.

Definition 1.2. Let $I = \{0,1\}^*$ and $I_n \subset I$ be a sphere of radius n. Let μ_n be a probability distribution on the sphere I_n. The collection $\{\mu_0, \mu_1, \mu_2, \ldots\}$ of all distributions is called *an ensemble of spherical distributions over* I and denoted by $\{\mu_n\}$.

In the paper we will be mostly concerned with the ensemble of uniform spherical distributions $\{u_n\}$ over I. For a set $R \subseteq I$ we define

$$u_n(R) = \frac{|R \cap I_n|}{|I_n|},$$

where $|X|$ is the cardinality of a set X.

Next we define an asymptotic density of a set in I.

Definition 1.3 (Asymptotic Density). Let $\mu = \{\mu_n\}$ be an ensemble of spherical distributions over a set I. A set of inputs $R \subseteq I$ is said to have asymptotic density $\rho(R) = \alpha$ if

$$\lim_{n\to\infty} \mu_n(R \cap I_n) = \alpha.$$

A set R is called *generic* with respect to μ if its asymptotic density is 1 and it is called *negligible* if the asymptotic density is 0.

Definition 1.4. Let $R \subseteq I$ and the asymptotic density $\rho(R)$ exists. The function

$$\delta_R(n) = \mu_n(R \cap I_n)$$

is called the *density* function for R.

A practical measure of the "largeness" of a set often corresponds to a rate with which the limit in Definition 1.3 converges. The convergence can be naturally described by obtaining upper bounds on the density function of a set. One particular type of sets of interest are sets which have superpolynomial convergence rates.

Definition 1.5. Let $R \subseteq I$ and $\delta_R(n)$ is the density function of R. We say that R has asymptotic density $\rho(R)$ with superpolynomial convergence if

$$|\rho(R) - \delta_R(n)| < \frac{1}{p(n)}$$

for every polynomial $p(n)$ and all sufficiently large n.

Definition 1.6 (Strongly Generic/Negligible). A generic set with superpolynomial convergence is called *strongly* generic and its complement is called a strongly negligible set.

1.2. One-Way functions

Existence of one-way functions is one of the most basic and important assumptions in cryptography. In fact existence of one-way functions is a minimal assumption required for constructing other cryptographic primitives such as pseudorandom number generators, encryption and signature schemes.

Diffie and Hellman [3] define one-way functions:

> "a function f is a one-way function if, for any argument x in the domain of f, it is easy to compute the corresponding value $f(x)$, yet, for almost all y in the range of f, it is computationally infeasible to solve the equation $y = f(x)$ for any suitable argument x."

There are two key points in the definition above: "for almost all" and "computationally infeasible". A lot of attention is still concentrated on the development and understanding of these two notions and their consequences from the practical point of view.

It is well accepted now that one-way functions cannot be defined using deterministic worst-case complexity classes like **P** and **NP**, and randomized computation is the default model for cryptographic purposes.

A common argument for the necessary conditions for one-way functions to exist proceeds as follows [5]. Suppose we have a cryptographic scheme. Legitimate parties should be able to decode the secret efficiently, which means that there exist a polynomial-time verifiable witness to the decoding and the problem of breaking a cryptographic scheme is in **NP**. For a cryptographic scheme to be considered secure there should be no practical algorithm to break the encryption. Therefore, if a secure cryptographic scheme exists then $\mathbf{NP} \not\subseteq \mathbf{BPP}$. Whether **BPP** contains **NP** is an open problem. Note that $\mathbf{NP} \not\subseteq \mathbf{BPP}$ implies that $\mathbf{P} \neq \mathbf{NP}$.

The $\mathbf{NP} \not\subseteq \mathbf{BPP}$ condition is a necessary, but not sufficient condition for a secure cryptographic scheme to exist. Observe that the probability distribution in the definition of the class **BPP** is taken over the internal states of a probabilistic machine only. The condition which bounds away the probability of an error must hold for all inputs. In this sense **BPP** is analogues to **P** and is still reflects the behavior of a problem on the worst case inputs but with respect to the randomized algorithms.

The positive answer to the problem $\mathbf{NP} \notin \mathbf{BPP}$ may have no practical implications for cryptography, unless there are problems which belong in $\mathbf{NP}\backslash\mathbf{BPP}$ and are hard on a significantly large fraction of inputs. Speaking in terms of generic complexity, a problem may be considered hard if there is no efficient algorithm which solves the problem on any but strongly negligible set of inputs.

In cryptography the existence of many useful primitives like secure symmetric encryption, pseudorandom number generators and digital signature schemes is reduced to the existence of the one-way functions which we define next. In general there are two notions of one-way functions a strong and a weaker one.

Let $\Pr_{(x,\sigma)}$ denote the probability taken uniformly over all pairs $(x, \sigma) \in I_n \times \Sigma$, where I_n is the set of all inputs of length n and $\Sigma = \{0,1\}^{t(n)}$ is the space of internal coin flips of a probabilistic algorithm whose running time is bounded by some polynomial $t(n)$. Similarly we define $\Pr_\sigma$ as the uniform probability taken over Σ only.

One of the most commonly accepted definitions of a one-way function (strong one-way function) is the following.

Definition 1.7 (Strong One-Way function [5]). A function $f : \{0,1\}^* \to \{0,1\}^*$ is called strongly one-way if the following two conditions hold:

1. Easy to compute: there exists a deterministic polynomial-time algorithm $\mathcal{A}'$ such that on an input x algorithm $\mathcal{A}'$ outputs $f(x)$;
2. Hard to invert: For every probabilistic polynomial-time algorithm $\mathcal{A}$, every positive polynomial p, and all sufficiently large n:

$$\Pr_{(x,\sigma)}[A(f(U_n), 1^n) \in f^{-1}(f(U_n))] < \frac{1}{p(n)},$$

where U_n is a random variable uniformly distributed over $\{0,1\}^n$ and the

probability is taken over all input strings from $\{0,1\}^n$ and internal states of $\mathcal{A}$.

Here and in the rest of the article *polynomial-time* algorithm means an algorithm that always halts after a polynomial (in the length of the input) number of steps. Note that in addition to an input in the range of f the algorithm $\mathcal{A}$ is given the auxiliary input 1^n which has the same length as the desired output of $\mathcal{A}$. This is done to protect from the situations when the function f drastically reduces the length of its input (for example $|f(x)| = \log_2(|x|)$). Obviously no algorithm can invert such function f in polynomial number of steps in terms of $|x|$.

2. Generic definitions of one-way functions

2.1. Definition restricted to PPT adversary

In Definition 1.7 the performance of an algorithm $\mathcal{A}$ is averaged over all inputs which results in complicated probability space. We would like to apply ideas of generic complexity and consider the performance of an adversary on each input separately.

Note that a naive random sampling will guess an inverse of a function f on the input of length n with probability $1/2^n$. An algorithm with negligible probability of the correct answer cannot be amplified and, therefore, cannot be considered practical. A reasonable inversion algorithm should have noticeable probability of success. To be more precise the probability that an algorithm $\mathcal{A}$ inverts $f(x)$

$$\Pr[A(f(x), 1^n) \in f^{-1}(f(x))] > \frac{1}{n^c}$$

for any positive constant c. To make a one-way function secure we must limit the number of inputs on which adversary succeeds to a small set. We formalize these arguments in the following definition of a generically strong one-way function.

Definition 2.1 (Generically Strong One-Way function). Let $u = \{u_n\}$ be an ensemble of uniform spherical distributions over $\{0,1\}^*$.

A function $f : \{0,1\}^* \to \{0,1\}^*$ is called *generically strong one-way* if the following two conditions hold:

1. Easy to compute: there exists a deterministic polynomial-time algorithm $\mathcal{A}'$ such that on input x algorithm $\mathcal{A}'$ outputs $f(x)$;
2. Hard to invert almost all inputs: For every probabilistic polynomial-time algorithm $\mathcal{A}$, all constants $c > 0$, every positive polynomial p and all sufficiently large n:

$$u_n\left(\{x \in I_n \mid \Pr[A(f(x), 1^n) \in f^{-1}(f(x))] > n^{-c}\}\right) < \frac{1}{p(n)},$$

where the probability is taken over internal states of the algorithm A.

Similarly we can define a generically weak one-way function.

Definition 2.2 (Generically Weak One-Way function). Let $u = \{u_n\}$ be an ensemble of uniform spherical distributions over $\{0,1\}^*$.

A function $f : \{0,1\}^* \to \{0,1\}^*$ is called *generically weak one-way* if the following two conditions hold:

1. Easy to compute: there exists a deterministic polynomial-time algorithm $\mathcal{A}'$ such that on input x algorithm $\mathcal{A}'$ outputs $f(x)$;
2. Hard to invert on a large enough set of inputs: For every probabilistic polynomial-time algorithm $\mathcal{A}$, every constant $c > 0$ there exists a polynomial $p(n)$ such that for all sufficiently large n:

$$u_n\left(\{x \in I_n \mid \Pr[A(f(x), 1^n) \in f^{-1}(f(x))] < n^{-c}\}\right) \geq \frac{1}{p(n)},$$

where the probability is taken over internal states of the algorithm A.

The following lemmas show that Definitions 2.1 and 1.7 are equivalent. We give equivalence results for strong one-way functions. Similar results hold for the weak notion as well (see Appendix for the detailed proof). We use standard reduction argument which proceeds by showing that if there exists an algorithm which violates the conditions of the first definition then we can construct an algorithm which will violate conditions of the second one.

Lemma 2.3. *Let $f : \{0,1\}^* \to \{0,1\}^*$ and suppose there is a probabilistic polynomial time algorithm $\mathcal{A}$ such that for some constants $c > 0$ and $d > 0$ and infinitely many n*

$$u_n\left(\{x \in I_n \mid \Pr_\sigma[A(f(x), 1^n) \in f^{-1}(f(x))] > n^{-c}\}\right) > \frac{1}{n^d}.$$

Then there exists a probabilistic polynomial-time algorithm $\mathcal{A}'$ such that for infinitely many n

$$\Pr_{(x,\sigma)}[A'(f(U_n), 1^n) \in f^{-1}(f(U_n))] > \frac{1}{n^{d+1}}.$$

Proof. First of all observe that since we can compute f, we can also check whether an algorithm indeed returns an inverse of $f(x)$ or not. By definition, $f^{-1}(y) = \{x \mid y = f(x)\}$ therefore if $f(\mathcal{A}(f(x))) = f(x)$ then $\mathcal{A}(f(x))$ is an inverse of $f(x)$.

Now construct an algorithm $\mathcal{A}'$ as follows. Repeat algorithm $\mathcal{A}$ on a given input x until a witness for the inverse problem (i.e. the inverse itself) is obtained. Let

$$S_n = \left\{x \in I_n \mid \Pr_\sigma[\mathcal{A}(f(x)) \in f^{-1}(f(x))] \geq n^{-c}\right\}.$$

For the algorithm $\mathcal{A}'$ to be practical on the set S_n we need to show that for every $x \in S_n$ we can obtain an inverse with high probability using only polynomially many repetitions of $\mathcal{A}$, i.e.

$$\Pr_\sigma[\mathcal{A}'_k(f(x)) \in f^{-1}(f(x))] \geq 1 - \epsilon, \tag{1}$$

where $k = p(n)$ and $\epsilon < \frac{1}{n^m}$ for any $m > 0$.

Let y_i be the output of the ith run of the algorithm $\mathcal{A}$ on an input $x \in S_n$ and let X_i, $i = 1, \ldots, k$ be random variables such that $X_i = 1$ if $y_i \in f^{-1}(f(x))$ and $X_i = 0$ otherwise. X_i are mutually independent and $\mathrm{E}[X_i] = \Pr[X_i = 1] \geq \frac{1}{n^c}$. We also define X_i^o, $i = 1, \ldots, k$ to be random variables such that $X_i^o = 0$ if $y_i \in f^{-1}(f(x))$ and $X_i^o = 1$ if ith run of $\mathcal{A}$ fails. X_i^o are also mutually independent and $\mathrm{E}[X_i^o] = 1 - \Pr[X_i = 1] \geq 1 - \frac{1}{n^c}$.

Note for $\mathcal{A}'$ to produce an answer only one of y_is needs to be a witness, therefore to show (1) we need to show that

$$\Pr\left[\sum_{i=1}^{k} X_i \geq 1\right] = \Pr\left[\sum_{i=1}^{k} X_i^o \leq k - 1\right] \geq 1 - \epsilon$$

which is equivalent to showing

$$\Pr\left[\sum_{i=1}^{k} X_i^o > k - 1\right] \leq \epsilon.$$

Using Chernoff bound we have

$$\Pr\left[\sum_{i=1}^{k} X_i^o - k \cdot \left(1 - \frac{1}{n^c}\right) \geq \delta \cdot k \cdot \left(1 - \frac{1}{n^c}\right)\right] \tag{2}$$

$$= \Pr\left[\sum_{i=1}^{k} X_i^o \geq k \cdot \left(1 - \frac{1}{n^c}\right) \cdot (\delta + 1)\right] \leq 2^{-\frac{\delta^2}{2}k}. \tag{3}$$

Substituting $\delta = (k - n^c)/(k(n^c - 1))$ into (3) we obtain

$$\Pr\left[\sum_{i=1}^{k} X_i^o \geq k - 1\right] \leq 2^{-\frac{1}{2}\cdot\left(\frac{k-n^c}{k(n^c-1)}\right)^2 k} = 2^{-\frac{(k-n^c)^2}{2k(n^c-1)^2}}.$$

Let $k = n^{3c}$, then

$$2^{-\frac{(k-n^c)^2}{2k(n^c-1)^2}} < 2^{-\frac{1}{2}(n+2)}$$

and we have

$$\Pr\left[\sum_{i=1}^{k} X_i^o \geq k - 1\right] < 2^{-\frac{1}{2}(n+2)}.$$

Therefore we obtained

$$\Pr_\sigma[\mathcal{A}'_k(f(x)) \in f^{-1}(f(x))] \geq 1 - \epsilon,$$

where $\epsilon = 2^{-\frac{1}{2}(n+2)}$. Note that a similar result can be obtained without using the Chernoff bound, however, it allows us to obtain a tighter bound on the number of repetitions of the algorithm $\mathcal{A}$.

Taking the sum over all $x \in S_n$ we obtain

$$\sum_{x \in S_n} \Pr_\sigma[\mathcal{A}'(f(x)) \in f^{-1}(f(x))] \geq \sum_{x \in S_n} (1 - \epsilon) = |S_n|(1 - \epsilon).$$

Note that

$$u_n(S_n) = \frac{|S_n|}{|I_n|} \geq \frac{1}{n^d}.$$

Therefore

$$|S_n| \geq \frac{|I_n|}{n^d} = \frac{2^n}{n^d}.$$

It follows

$$\sum_{x \in S_n} \Pr_\sigma[\mathcal{A}'(f(x)) \in f^{-1}(f(x))] \geq |S_n|(1 - \epsilon) \geq \frac{2^n}{n^d}(1 - \epsilon). \tag{4}$$

Next we show that $\Pr_{(x,\sigma)}[\mathcal{A}'(f(U_n), 1^n) \in f^{-1}(f(U_n))] \geq \frac{1}{n^d} - \epsilon$.

Define $A'(x, \sigma) = 1$ if the computation of $\mathcal{A}'$ corresponding to oracle σ inverts $f(x)$ and $A'(x, \sigma) = 0$ otherwise.

Now we have

$$\Pr_{(x,\sigma)}[\mathcal{A}'(f(U_n), 1^n) \in f^{-1}(f(U_n))] = \sum_{\forall (x,\sigma)} A'(x, \sigma)p(x, \sigma),$$

where $p(x, \sigma)$ is the joint probability mass function.

Note that x and σ are independent from each other, therefore

$$\begin{aligned}
\sum_{\forall (x,\sigma)} A'(x, \sigma)p(x, \sigma) &= \sum_{x \in I_n} \sum_{\sigma \in \{0,1\}^{t(n)}} A'(x, \sigma)p(x)p(\sigma) \\
&= \frac{1}{2^n} \sum_{x \in I_n} \sum_{\sigma \in \{0,1\}^{t(n)}} A'(x, \sigma)p(\sigma) \\
&= \frac{1}{2^n} \sum_{x \in I_n} \Pr_\sigma[\mathcal{A}'(f(x)) \in f^{-1}(f(x))].
\end{aligned}$$

From (4) and the equation above we have

$$\begin{aligned}\Pr_{(x,\sigma)}[\mathcal{A}'(f(U_n),1^n)\in f^{-1}(f(U_n))] &= \frac{1}{2^n}\sum_{x\in I_n}\Pr_\sigma[\mathcal{A}'(f(x))\in f^{-1}(f(x))]\\ &\geq \frac{1}{2^n}\sum_{x\in S_n}\Pr_\sigma[\mathcal{A}'(f(x))\in f^{-1}(f(x))]\\ &= \frac{1}{n^d}(1-\epsilon).\end{aligned}$$

Now let $d'=d+1$. It is easy to see that $1/n^d(1-\epsilon)>1/n^{d'}$ for $n\geq 2$. Therefore we have

$$\Pr_{(x,\sigma)}[\mathcal{A}'(f(U_n),1^n)\in f^{-1}(f(U_n))]\geq \frac{1}{n^d}(1-\epsilon)>\frac{1}{n^{d+1}}.$$

□

The implication holds in the the opposite direction as well.

Lemma 2.4. *Let $f:\{0,1\}^*\to\{0,1\}^*$ and suppose there is a probabilistic polynomial time algorithm $\mathcal{A}$ such that for some polynomial $p(n)$ and infinitely many n*

$$\Pr_{(x,\sigma)}[A(f(U_n),1^n)\in f^{-1}(f(U_n))]\geq\frac{1}{p(n)}.$$

Then there exists a probabilistic polynomial-time algorithm $\mathcal{A}'$ such that for every $c>0$ and infinitely many n

$$u_n\left(\{x\in I_n\mid \Pr_\sigma[A'(f(x))\in f^{-1}(f(x))]>n^{-c}\}\right)\geq\frac{1}{2p(n)}.$$

Proof. First we show that

$$u_n\left(\{x\in I_n\mid \Pr_\sigma[A(f(x))\in f^{-1}(f(x))]>1/2p(n)\}\right)\geq\frac{1}{2p(n)}. \tag{5}$$

The proof follows directly from the following averaging argument:

Claim 2.5. *Let $a_1,\ldots,a_N\in[0,1]$ and $\rho\geq 0$ such that $\frac{1}{N}\sum_{i=1}^N a_i\geq\rho$ and let $k=\#\{a_i\mid a_i>\rho/2\}$. Then*

$$\frac{k}{N}\geq\frac{\rho}{2}.$$

Observe that

$$\Pr_{(x,\sigma)}[A(f(U_n),1^n)\in f^{-1}(f(U_n))]=\frac{1}{2^n}\sum_{x\in I_n}\Pr_\sigma\left[A(f(x))\in f^{-1}(f(x))\right]\geq\frac{1}{p(n)}.$$

If we set $a_i = \Pr_\sigma [A(f(x_i)) \in f^{-1}(f(x_i))]$, $x_i \in I_n$, $N = 2^n$, $\rho = 1/p(n)$ and $k = \#\{x \in I_n \mid \Pr_\sigma[A(f(x)) \in f^{-1}(f(x))] > 1/2p(n)\}$ then it follows from the claim above that

$$\frac{k}{2^n} \geq \frac{1}{2p(n)}$$

and

$$u_n\left(\{x \in I_n \mid \Pr_\sigma[A(f(x)) \in f^{-1}(f(x))] > 1/2p(n)\}\right) \geq \frac{1}{2p(n)}.$$

Now observe that for any $c > 0$ there exists a probabilistic polynomial-time algorithm $\mathcal{A}'$ such that

$$\#\{x \in I_n \mid \Pr_\sigma[A'(f(x)) \in f^{-1}(f(x))] > n^{-c}\} \geq k. \tag{6}$$

Indeed, in the case when $n^{-c} \geq 1/2p(n)$ the claim follows directly. In the second case when $n^{-c} < 1/2p(n)$ we can use the probabilistic error reduction and construct an algorithm $\mathcal{A}'$ such that (6) holds. Therefore there exists a polynomial-time algorithm $\mathcal{A}'$ such that

$$u_n\left(\{x \in I_n \mid \Pr_\sigma[A'(f(x)) \in f^{-1}(f(x))] > n^{-c}\}\right) \geq \frac{1}{2p(n)}.$$

□

The following result demonstrates the connection between the security assumption and asymptotic properties of the input sets.

Proposition 2.6. *A polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is strongly one way if and only if every probabilistic polynomial-time algorithm $\mathcal{A}$ fails to invert f on all but strongly negligible sets of inputs with respect to an ensemble of uniform spherical distributions over $\{0,1\}^*$.*

Proof. Suppose f is strongly one-way and suppose there exists an algorithm $\mathcal{A}$ which inverts f on a set S which is not strongly negligible. Then there exists a polynomial $p(n)$ such that

$$u_n(\{x \in I_n \mid \Pr_\sigma[\mathcal{A}(f(x)) \in f^{-1}(f(x))] > n^{-c}\}) = u_n(S \cap I_n) = \delta_s(n) > \frac{1}{p(n)}.$$

Therefore f is not strongly one-way by Definition 2.1.

Now, suppose f is not one-way. Then there exists an algorithm $\mathcal{A}$ such that

$$u_n(\{x \in I_n \mid \Pr_\sigma[A(f(x)) \in f^{-1}(f(x))] > n^{-c}\}) > \frac{1}{p(n)}$$

for some polynomial p, which contradicts the proposition assumption. □

2.2. Generic definition with a more general adversary

The most interesting question is whether the generic approach may give us new, more general security assumptions. Note that the polynomial bound on the adversary is not necessary. The only condition that a successful adversary needs to satisfy is to have an algorithm which terminates in polynomial time and with correct answer on a non-negligible set of inputs. Suppose we would like to make a security statement which holds against a much stronger adversary, i.e. a partial probabilistic heuristic algorithm which may output incorrect answers. Although an adversary algorithm may not terminate on some inputs, it would still be a threat if it succeeds on a relatively large set of inputs.

Definition 2.7 (Partial algorithm with errors). Let I be the set of inputs. We say that an algorithm $\mathcal{A}$ is a partial algorithm with errors if it is correct on a subset $X \subseteq I$ of inputs and on the set $I - X$ it either does not stop or stops with an incorrect answer.

To make a formal statement we need a notion of achievement ratio of an adversary which is similar to the notions given in [6, 9].

Definition 2.8 (Achievement ratio). Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function and let $\mathcal{A}$ be a partial probabilistic algorithm with errors. The achievement ratio of $\mathcal{A}$ on an instance $f(x)$ is defined as

$$\mathcal{R}_{\mathcal{A},f}(x) = T_{\mathcal{A},f}(x)/\delta_{\mathcal{A},f}(x),$$

where $T_{\mathcal{A},f}(x)$ is the time required for $\mathcal{A}$ to terminate on the input $f(x)$ and

$$\delta_{\mathcal{A},f}(x) = \Pr_\sigma[\mathcal{A}(f(x), 1^n) \in f^{-1}(f(x), 1^n)].$$

Achievement ratio allows one to consider a larger class of algorithms whose running time may not be bounded by a polynomial. In order for an adversary to have a polynomial achievement ratio on a given input x, it has to have both: the polynomial running time and a noticeable probability of inverting $f(x)$.

The following definition is an attempt to give an intuitive notion of a generalized practical security assumption for a one-way function.

Definition 2.9. Let $u = \{u_n\}$ be an ensemble of uniform spherical distributions over $\{0,1\}^*$.

A function $f : \{0,1\}^* \to \{0,1\}^*$ is called strongly one-way if the following two conditions hold:

1. Easy to compute: there exists a deterministic polynomial-time algorithm $\mathcal{A}'$ such that on input x algorithm $\mathcal{A}'$ outputs $f(x)$;
2. Hard to invert: For every partial probabilistic algorithm with errors $\mathcal{A}$, all constants $c > 0$, every positive polynomial p and all sufficiently large n:

$$u_n\left(\{x \in I_n \mid \mathcal{R}_{\mathcal{A},f}(x) \leq n^c\}\right) < \frac{1}{p(n)}.$$

The question is whether or not this definition gives us any advantage over the definitions given earlier. The following argument says that if we allow only a polynomial number of steps for an adversary on a success then, in fact, this definition is equivalent to the one which is limited to the PPT adversary.

The main idea is that since the *success* of an adversary on an input x means that it has to terminate in polynomial number of steps, then we do not really care if adversary is a partial algorithm or not. If we have a successful partial algorithm then we can construct a PPT algorithm by allowing it to run for polynomial number of steps and this polynomial-time algorithm will be as successful as the partial one.

Let GSPPT and GSPART be the classes of one way functions which satisfy conditions of Definition 2.1 and Definition 2.9 respectively.

Proposition 2.10. *A function $f \in$ GSPPT if and only if $f \in$ GSPART.*

Proof. First we show that $f \in$ GSPART implies $f \in$ GSPPT. The proof is by contradiction. Let $f : \{0,1\}^* \to \{0,1\}^*$ and assume that $f \in GSPART$, but $f \notin GSPPT$, then there exists a PPT algorithm A, a constant $c > 0$, a polynomial $p(n)$ such that for infinitely many n

$$u_n(\{x \mid \delta_{A,f}(x) > n^{-c}\}) \geq \frac{1}{p(n)}.$$

Note that a PPT algorithm A is also a partial probabilistic algorithm such that $T_{A,f}(x) \leq q(n)$, for some positive polynomial q for all x. Therefore,

$$\begin{aligned}
u_n(\{x \mid \delta_{A,f}(x) > n^{-c}\}) &\geq \frac{1}{p(n)} \\
u_n(\{x \mid \delta_{A,f}(x)/T_{A,f}(x) > n^{-c}/T_{A,f}(x)\}) &\geq \frac{1}{p(n)} \\
u_n(\{x \mid T_{A,f}(x)/\delta_{A,f}(x) < n^c T_{A,f}(x)\}) &\geq \frac{1}{p(n)} \\
u_n(\{x \mid \mathcal{R}_{A,f}(x) < n^c T_{A,f}(x)\}) &\geq \frac{1}{p(n)} \\
u_n(\{x \mid \mathcal{R}_{A,f}(x) \leq n^d\}) &\geq \frac{1}{p(n)},
\end{aligned}$$

where d is chosen such that $n^d \geq q(n) \cdot n^c$. This is a contradiction to the condition $f \in GSPART$.

The proof in the opposite direction uses a similar argument. Suppose that $f \in$ GSPPT but $f \notin$ GSPART. In other words we suppose there exists a partial probabilistic algorithm $\mathcal{B}$ such that for some polynomial $p(n)$ and infinitely many n

$$u_n\left(\{x \in I_n \mid \mathcal{R}_{\mathcal{B},f}(x) \leq n^c\}\right) \geq \frac{1}{p(n)}.$$

Define $\mathcal{A}$ to be an algorithm which on a given input $x \in I_n$ runs $\mathcal{B}$ for n^c steps. Let $S = \{x \mid \mathcal{R}_{\mathcal{B},f}(x) \leq n^c\}$. First observe that by the conjecture for all $x \in S$

$$\delta_{\mathcal{B},f}(x) \geq \frac{T_{\mathcal{B},f}(x)}{n^c} \geq \frac{1}{n^c}.$$

Obviously, $\delta_{\mathcal{A},f}(x) = \delta_{\mathcal{B},f}(x)$ for all x such that $T_{\mathcal{B},f}(x) \leq n^c$. Therefore, since $\delta_{\mathcal{B},f}(x) \in [0,1]$ we have

$$\delta_{\mathcal{A},f}(x) = \delta_{\mathcal{B},f}(x)$$

for all x such that $T_{\mathcal{B},f}(x) \leq \delta_{\mathcal{B},f}(x) \cdot n^c$, i.e. for all $x \in S$.

Hence we have $\delta_{\mathcal{A},f}(x) \geq \frac{1}{n^c}$ for all $x \in S$ and

$$u_n\left(\{x \in I_n \mid \delta_{\mathcal{A},f}(x) \geq n^{-c}\}\right) \geq u_n(S) \geq \frac{1}{p(n)}.$$

Therefore, a probabilistic polynomial time algorithm $\mathcal{A}$ inverts f on a not strongly negligible set which contradicts our assumption that f is one-way with respect to Definition 2.1. □

Note that the proof is simple and quite compact. Using the equivalence Lemmas 2.3 and 2.4 we can conclude that the Definition 2.9 is equivalent to Definition 1.7 which is based on the averaging argument. It seems that obtaining the same result would be a more difficult task when working with the average type definitions directly.

Similarly one can define a weaker variation of a one-way function with a partial adversary.

Definition 2.11. Let $u = \{u_n\}$ be an ensemble of uniform spherical distributions over $\{0,1\}^*$.

A function $f : \{0,1\}^* \to \{0,1\}^*$ is called weakly one-way if the following two conditions hold:

1. Easy to compute: there exists a deterministic polynomial-time algorithm $\mathcal{A}'$ such that on input x algorithm $\mathcal{A}'$ outputs $f(x)$;
2. Hard to invert on non-negligible set: For every partial algorithm $\mathcal{A}$ and every constant $c > 0$, there exists a polynomial $p(x)$ such that for all sufficiently large n

$$u_n\left(\{x \in I_n \mid \mathcal{R}_{\mathcal{A},f}(x) > n^c\}\right) \geq \frac{1}{p(n)}.$$

The equivalence result for weak one-way functions holds as well. Let GWPPT be the class of generically weak one-way functions and GWPART be the class of one way functions satisfying Definition 2.11.

Proposition 2.12. *A function* $f \in$ GWPPT *if and only if* $f \in$ GWPART.

Proof. The proof is similar to the proof of Proposition 2.10. Suppose that $f \in GWPART$ but $f \notin GWPPT$. Then there exists a PPT algorithm $\mathcal{B}$ and constant $c > 0$ such that for all polynomials $p(n)$

$$u_n(\{x \mid \delta_{\mathcal{B},f}(x) < n^{-c}\}) < \frac{1}{p(n)}$$

The probabilistic polynomial time algorithm $\mathcal{B}$ is a probabilistic partial algorithm such that its time $T_{\mathcal{B},f}(x) \leq q(n)$ for some positive polynomial q and all x.

Therefore, there exists a probabilistic partial algorithm $\mathcal{B}$ such that for all positive polynomials p:

$$\begin{aligned}
\frac{1}{p(n)} &> u_n(\{x \mid \delta_{\mathcal{B},f}(x) < n^{-c}\}) \\
&= u_n(\{x \mid T_{\mathcal{B},f}(x)\delta_{\mathcal{B},f}(x) < T_{\mathcal{B},f}(x)n^{-c}\}) \\
&= u_n(\{x \mid T_{\mathcal{B},f}(x)/\delta_{\mathcal{B},f}(x) \geq T_{\mathcal{B},f}(x)n^{c}\}) \\
&= u_n(\{x \mid R_{\mathcal{B},f}(x) \geq T_{\mathcal{B},f}(x)n^{c}\}) \\
&\geq u_n(\{x \mid R_{\mathcal{B},f}(x) \geq n^{d}, \forall d > 0\})
\end{aligned}$$

Which contradicts the assumption that $f \in GWPART$.

Now note that if f is not weakly one-way in terms of Definition 2.11 then there exists a partial algorithm $\mathcal{B}$ such that for some constant $c > 0$ and every polynomial $poly(n)$

$$u_n\left(\{x \in I_n \mid \mathcal{R}_{\mathcal{B},f}(x) \leq n^c\}\right) \geq 1 - \frac{1}{poly(n)}.$$

Define a probabilistic polynomial-time algorithm $\mathcal{A}$ which runs $\mathcal{B}$ for n^c steps. Using the equalities from Proposition 2.10 we obtain

$$u_n\left(\{x \in I_n \mid \delta_{\mathcal{A},f}(x) \geq n^{-c}\}\right) \geq u_n\left(\{x \in I_n \mid \mathcal{R}_{\mathcal{B},f}(x) \leq n^c\}\right) \geq 1 - \frac{1}{poly(n)}.$$

Therefore,

$$u_n\left(\{x \in I_n \mid \delta_{\mathcal{A},f}(x) < n^{-c}\}\right) < \frac{1}{poly(n)}$$

for any polynomial $poly(n)$. Therefore, f is not weakly one way with respect to a PPT algorithm $\mathcal{A}$. □

One of the important results about one-way functions is the so-called amplification theorem which states that having a weak one-way function we can always construct a strong one. Equivalences shown above allow us to make a similar statement for generic one-way function.

Theorem 2.13 (Amplification). *Generically weak one-way functions exist if and only if generically strong one-way functions exist.*

Proof. The proof is a corollary of the equivalence Lemmas 2.3, 2.4, 2.10, 2.12 and the classical amplification theorem. □

3. Conclusion

The definition based on generic case complexity methodology has significant advantage in the fact that the probabilities over inputs and internal states of the algorithm are taken separately. The definition is very intuitive and easy to understand. In fact it may be seen as a direct formalization of the definition by Diffie and Hellman which we quote in the introduction.

Operating with simpler probability spaces and considering inputs separately may have some practical implications. The work in this direction started very recently and the potential of generic approach has been little realized. It would be interesting to see if generic complexity can be used to simplify definitions of cryptographic primitives and reducibility arguments. Applications of generic case complexity analysis of the security of particular one-way function candidates is also could be of great interest.

A. Proof of equivalence for the definitions of the Weak One-Way functions

The following is the classical definition of a weak one-way function.

Definition A.1 (Weak One-Way function). A function $f : \{0,1\}^* \to \{0,1\}^*$ is called weakly one-way if the following two conditions hold:

1. Easy to compute: there exists a deterministic polynomial-time algorithm $\mathcal{A}'$ such that on an input x algorithm $\mathcal{A}'$ outputs $f(x)$;
2. Slightly hard to invert: There exists a polynomial p such that for every PPT $\mathcal{A}$ and all sufficiently large n:

$$\Pr_{(x,\sigma)}[\mathcal{A}(f(U_n), 1^n) \notin f^{-1}(f(U_n))] \geq \frac{1}{p(n)},$$

where U_n is a random variable uniformly distributed over $\{0,1\}^n$ and the probability is taken over all input strings from $\{0,1\}^n$ and internal states of $\mathcal{A}$.

Proposition A.2. *Definitions* A.1 *and* 2.2 *are equivalent.*

The following two lemmas give the proof. Denote

$$\delta_{\mathcal{A},f}(x) = \Pr[A(f(x), 1^n) \in f^{-1}(f(x))]$$

and

$$\bar{\delta}_{\mathcal{A},f}(x) = \Pr[A(f(x), 1^n) \notin f^{-1}(f(x))].$$

Obviously

$$\delta_{\mathcal{A},f}(x) = 1 - \bar{\delta}_{\mathcal{A},f}(x).$$

Lemma A.3 (Generic implies Classic). *Suppose there exists a PPT algorithm $\mathcal{A}$ such that for some (equivalently all) $c > 0$, all polynomials p and infinitely many n*

$$u_n\left(\{x \in I_n \mid \delta_{\mathcal{A},f}(x) < n^{-c}\}\right) < \frac{1}{p(n)}$$

then there exists a PPT algorithm $\mathcal{A}'$ such that for all polynomials $q(n)$ and infinitely many n

$$\Pr_{(x,\sigma)}[\mathcal{A}(f(U_n), 1^n) \notin f^{-1}(f(U_n))] < \frac{1}{q(n)}.$$

Proof. Observe that

$$u_n(\{x \mid \delta_{\mathcal{A},f}(x) \geq n^{-c}\}) \geq 1 - \frac{1}{p(n)}.$$

Let

$$S_n = \{x \mid \delta_{\mathcal{A},f}(x) \geq n^{-c}\}$$

Then

$$u_n(S_n) = \frac{|S_n|}{2^n} \geq 1 - \frac{1}{p(n)}$$

However,

$$\sum_{x \in S_n} \delta_{\mathcal{A},f}(x) \geq \sum_{x \in S_n} n^{-c} = |S_n| n^{-c}$$

and we obtain

$$\frac{1}{2^n} \sum_{x \in S_n} \delta_{\mathcal{A},f}(x) \geq \frac{|S_n|}{2^n} n^{-c} \geq n^{-c}\left(1 - \frac{1}{p(n)}\right) = \frac{p(n)-1}{n^c p(n)} > \frac{1}{n^c p(n)}$$

From the proof of the equivalence for the case of strong one way functions we know that

$$\Pr_{(x,\sigma)}[\mathcal{A}(f(U_n), 1^n) \in f^{-1}(f(U_n))] \geq \frac{1}{2^n} \sum_{x \in S_n} \delta_{\mathcal{A},f}(x).$$

Therefore

$$\Pr_{(x,\sigma)}[\mathcal{A}(f(U_n), 1^n) \in f^{-1}(f(U_n))] \geq \frac{1}{n^c p(n)}$$

Again, from the proof of the strong version we know that by repeating the algorithm $\mathcal{A}$ polynomially many times we can obtain an algorithm $\mathcal{A}'$ such that

$$\Pr_{(x,\sigma)}[\mathcal{A}'(f(U_n), 1^n) \in f^{-1}(f(U_n))] \geq 1 - \epsilon$$

where $\epsilon < 1/q(n)$ for any positive polynomial $q(n)$. Then

$$\Pr_{(x,\sigma)}[\mathcal{A}'(f(U_n), 1^n) \notin f^{-1}(f(U_n))] < 1 - (1 - \epsilon) = \epsilon < \frac{1}{q(n)}$$

for all polynomials $q(n)$. □

Lemma A.4 (Classic implies Generic). *Suppose there exists a PPT algorithm $\mathcal{A}$ such that for all polynomials p and infinitely many n*

$$\Pr_{(x,\sigma)}[\mathcal{A}(f(U_n), 1^n) \notin f^{-1}(f(U_n))] < \frac{1}{p(n)}.$$

then there exists a PPT algorithm $\mathcal{A}'$ such that for some (equivalently all) $c > 0$, all polynomials $p(n)$ and infinitely many n

$$u_n\left(\{x \in I_n \mid \delta_{\mathcal{A}',f}(x) < n^{-c}\}\right) < \frac{1}{p(n)}$$

Proof. Let

$$S_n = \{x \mid \bar{\delta}_{\mathcal{A},f}(x) \geq n^{-d}\}$$

Observe that

$$n^d \cdot \Pr_{(x,\sigma)}[\mathcal{A}(f(U_n), 1^n) \notin f^{-1}(f(U_n))] < \frac{1}{p(n)}$$

for all positive polynomials $p(n)$.

Proof. Suppose that it is not. Then there exists a polynomial $p'(n)$ such that

$$n^d \cdot \Pr_{(x,\sigma)}[\mathcal{A}(f(U_n), 1^n) \notin f^{-1}(f(U_n))] \geq \frac{1}{p'(n)}$$

and

$$\Pr_{(x,\sigma)}[\mathcal{A}(f(U_n), 1^n) \notin f^{-1}(f(U_n))] \geq \frac{1}{p'(n)n^d}$$

which contradicts the condition of the lemma.

Now using the same argument as in the previous proofs we can show that

$$\Pr_{(x,\sigma)}[\mathcal{A}(f(U_n), 1^n) \notin f^{-1}(f(U_n))] = \frac{1}{2^n}\sum_{x \in I_n} \bar{\delta}_{\mathcal{A},f}(x) \geq \frac{1}{2^n}\sum_{x \in S_n} \bar{\delta}_{\mathcal{A},f}(x)$$

Therefore, for every $p(n)$

$$\begin{aligned}
\frac{1}{p(n)} &> n^d \cdot \Pr_{(x,\sigma)}[\mathcal{A}(f(U_n), 1^n) \notin f^{-1}(f(U_n))] \\
&\geq \frac{n^d}{2^n}\sum_{x \in S_n} \bar{\delta}_{\mathcal{A},f}(x) \\
&\geq \frac{n^d}{2^n}\sum_{x \in S_n} n^{-d} \\
&= n^d \cdot \frac{|S_n|}{2^n} \cdot n^{-d} \\
&= u_n(S_n).
\end{aligned}$$

Note that

$$S_n = \{x \mid 1 - \bar{\delta}_{\mathcal{A},f}(x) < 1 - n^{-d}\}\} = \{x \mid \delta_{\mathcal{A},f}(x) < 1 - n^{-d}\}\}.$$

Using amplification we can construct a PPT algorithm $\mathcal{A}'$ which repeats $\mathcal{A}$ polynomially many times and such that

$$S_n = \left\{x \mid \delta_{\mathcal{A}',f}(x) < \frac{1}{2^n}\right\}$$

Therefore, there exists a PPT algorithm $\mathcal{A}'$ such that for every polynomial $p(n)$

$$u_n\left(\left\{x \mid \delta_{\mathcal{A}',f}(x) < \frac{1}{2^n}\right\}\right) < \frac{1}{p(n)}.$$

□

References

[1] A. Bogdanov, L. Trevisan: Average-Case Complexity, Now, Boston (2006).

[2] A. V. Borovik, A. G. Miasnikov, V. N. Remeslennikov: Multiplicative measures on free groups, Int. J. Algebra Comput. 13 (6) (2003) 705–731.

[3] W. Diffie, M. Hellman: New Directions in Cryptography, IEEE Trans. Inf. Theory 22(6) (1976) 644–654.

[4] R. Gilman, A. G. Miasnikov, A. D. Myasnikov, A. Ushakov: Generic complexity of algorithmic problems, preprint (2007).

[5] O. Goldreich: Foundations of Cryptography: Volume 1, Basic Tools, Cambridge University Press, Cambridge (2001).

[6] O. Goldreich, L. Levin: A hard-core predicate for all one-way functions, in: ACM Symposium on Theory of Computing (STOC, Seattle, 1989), ACM, New York (1989) 25–32.

[7] S. Goldwasser, S. Micali: Probabilistic encryption, J. Comput. Syst. Sci. 28 (1984) 270–299.

[8] J. D. Hamkins, A. G. Miasnikov: The halting problem is decidable on a set of asymptotic probability one, Notre Dame J. Formal Logic 47(4) (2006) 515–524.

[9] J. Hastad, R. Impagliazzo, L. Levin, M. Luby: Construction of pseudorandom generator from any one-way function, manuscript (1993).

[10] I. Kapovich, A. G. Miasnikov, P. Schupp, V. Shpilrain: Generic-case complexity, decision problems in group theory and random walks, J. Algebra 264 (2003) 665–694.

[11] A. Miasnikov, A. Rybalov: On generically undecidable problems, preprint (2007).

[12] C. Papadimitriou: Computational Complexity, Addison-Wesley, Amsterdam (1994).

[13] A. Rybalov: On the strongly generic undecidability of the halting problem, Theor. Comput. Sci. 377(1-3) (2007) 268–270.