Math-Net.Ru

Общероссийский математический портал

V. O. Popov, S. V. Smyshlyaev, Построение генераторов случайных чисел с помощью вероятностных автоматов и “одно-сторонних” функций, *Матем. вопр. криптогр.*, 2014, том 5, выпуск 2, 109–115

DOI: <https://doi.org/10.4213/mvk123>

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением
<http://www.mathnet.ru/rus/agreement>

Параметры загрузки:

IP: 54.166.219.16

26 апреля 2023 г., 12:22:13

Construction of RNG using random automata and “one-way” functions

V. O. Popov, S. V. Smyshlyaev

LLC "CRYPTO-PRO", Moscow

Received 25.09.2013

We consider a number of practical issues related to requirements for pseudorandom number generators used for cryptographic software needs. They extend further than general requirements of (practical) indistinguishability of output sequence from the sequence of independent uniformly distributed random variables. We formulate these additional requirements and present a general construction of RNG for usage in cryptographic software that is proposed to meet all of them.

Keywords: random number generators, key generation, cryptoproviders, random automata.

Построение генераторов случайных чисел с помощью вероятностных автоматов и "односторонних" функций

В. О. Попов, С. В. Смышляев

ООО «КРИПТО-ПРО», Москва

Аннотация. Рассматривается ряд практических вопросов, связанных с требованиями к генераторам псевдослучайных чисел, используемых в криптографическом программном обеспечении. Они жестче общих требований (практической) неотличимости выходной последовательности от последовательности независимых равномерно распределенных случайных величин. Формулируются эти дополнительные условия, описана общая конструкция генератора случайных чисел для криптографического программного обеспечения, предположительно удовлетворяющая всем этим условиям.

Ключевые слова: генераторы случайных чисел, генерация ключей, криптопровайдеры, случайные автоматы.

1 Introduction

Functioning of software and hardware cryptoproviders is impossible without using random number generators (RNGs) for their needs: generating key material, initialization vectors (IVs), seeds, elements of input user key material (UKMs) etc. Such RNGs have to meet a large number of strong requirements regarding output values distribution. Nevertheless, such RNGs have to provide enough performance, thus they have to be flexible to gain superior properties of the output sequence in certain cases (e.g. generation of key material) and excellent performance in other cases (e.g. generation of IVs). Another necessary property that must be met by RNG is preserving unpredictability of output symbols in case of certain hardware failures.

In the current paper we propose a construction of RNG based on random automaton with state-transition function with property of “one-wayness” (e.g. discrete exponential, prime numbers multiplication, Rabin function). Since the state transition is made with optional random input, balance between performance and security of such an RNG may be obtained for any class of objectives.

Details of our construction are described and statements about most important properties of the proposed construction are presented.

2 Requirements for RNGs used in cryptographic software

All general requirements for RNGs and PRNGs (pseudorandom number generators) are based on indistinguishability (under certain assumptions) of its output from a sequence of independent uniformly distributed random variables. Required properties of RNG that follow from this general requirement include: hardness of guessing the preceding output tuple knowing a certain output tuple; hardness of guessing the following output tuple knowing a certain output tuple; absence of period during large enough length; negligible correlation between output tuples on fixed distances; entropy being close to maximum possible, etc. All these requirements are formulated in a certain computational model under some assumptions about adversary. A thorough review of definitions connected with PRNG security may be found in [1]. In the current paper we will mostly consider practical assumptions about adversary based on adversary models for cryptographic software.

Nevertheless RNGs, that are used for needs of software implementations of cryptographic algorithms and protocols, e. g. in cryptographic providers, have to meet particular requirements connected with specificity of environment, outer requirements for the whole system, diversity of usage of produced output tuples.

These additional requirements include following.

- Ability of flexible control of RNG for the purpose of generating random values for different purposes. For example, generation of initialization vectors does not require hardness of backward guessing and, in practice, has much weaker requirements for

statistical properties of output values. However, initialization vectors have to be generated more efficiently.

- In all variants of usage, output symbols must retain unpredictability in case of hardware failures.
- RNG must be able to work in multiple sessions without decrease of required properties.

3 Related work

Since the proposed general framework is based on RNG/PRNG construction using functions with conjectured one-wayness (for the purpose of simplification of the paper from now we shall call them as 'one-way functions' and be assume that such functions are truly one-way), we start with a short review of known PRNG constructions of this type.

In 1982 Blum and Micali [3] proposed a cryptographically secure generator that was based on discrete exponentiation modulo p . Its output was one bit per one exponentiation operation: this bit was known to be hard-core predicate for discrete exponentiation. Security of the Blum-Micali generator is based on intractability of the discrete logarithm problem in $\mathbb{Z}_p^*$. Other PRNGs based on "one-way" functions include Long and Widgerson PRNG ([10], discrete exponentiation in $\mathbb{Z}_p^*$, $O(\log \log p)$ bits per one exponentiation), Kaliski PRNG ([9], multiplication on elliptic curves), Goldwasser PRNG ([7], RSA (Rivest, Shamir, Adleman) function, one bit for one RSA public-key encryption). It is known [8] that a provably secure pseudorandom generator may be constructed using any one-way function.

Also there is a number of practically oriented constructions: Dual Elliptic Curve PRNG (DEC PRNG [2], 240 bits per one multiplication on elliptic curve over the prime field $\mathbb{F}_p$ with 256-bit length p), Sidorenko PRNG ([1], multiplication on elliptic curves).

4 Proposed general RNG construction

Though for DEC PRNG there were a number of results showing its positive statistical properties (e.g. [4]), known distinguishing attacks on this generator exist ([1]). These attacks are based on the fact that each output value of the generator contains rough elliptic curve point coordinate bits.

Thus one lesson that has to be learned is that output values of generator should not contain explicit information about internal PRNG state. Another argument for this is importance of security of generator against forward attacks: finding unknown tuples of output bits if preceding tuples are known.

A natural solution of this problem is applying "one-way" function to the PRNG state data. A plausible conjecture here is that a better choice for this output computation

function is “one-way” function distinct from the state transition function to exclude the existence of threats connected with algebraic properties of the functions having common algebraic structure.

To meet the previously formulated requirements, the proposed construction has the following elements. Though it is based on state transition function with the property of “one-wayness”, every state transition also depends on random input value that determines the input value for the underlying “one-way” function when it is applied to the current state.

Our construction is based on the following elements.

1. S is the set of all possible states (e. g. $E(\mathbb{F}_p)$, elliptic curve over $\mathbb{F}_p$).
2. R is some finite set.
3. $f(s, c)$, $f : S \times R \rightarrow S$ is “one-way” function with respect to c for any fixed s .
4. Function $compress(s, r)$, $compress : S \times R \rightarrow R$ is such that $f(s, compress(s, r))$ is “one-way” with respect to s for any fixed r .
5. $b \in \mathbb{N}$, the number of output bits per one state transition.
6. $h : S \rightarrow \{0, 1\}^b$ is “one-way” function (e. g. cryptographic hash function).

The general description of functioning of RNG follows:

$$s(1) = s_0;$$

$$r(i) \in_R R, s(i+1) = f(s(i), compress(s(i), r(i))), i = 1, 2, \dots;$$

$$out(i) = h(s(i)), i = 1, 2, \dots$$

Random choice of elements of the set R is made (denoted as “ $\in_R$ ”) using some external mechanism if such is available. The choice of such a mechanism, an external entropy source, is a separate task, it will be briefly discussed in the following section. But the important property here is that in certain cases a large number of positive properties may be proved even if such an entropy source is not available or if it becomes broken (in this case $r(i) = \text{const}$ for every i). Hardness of backward guessing (guessing $out(i-1)$ knowing $out(i)$) is based on “one-wayness” of functions h and $f(s, compress(s, r))$ with respect to s ; hardness of forward guessing (guessing $out(i+1)$ knowing $out(i)$) is based on “one-wayness” of functions h and f with respect to r . Large period may be guaranteed by the properties of functions f and h . The s_0 value for initialization of each session of RNG is taken equal to output of the root RNG, which is operating only for initialization of sessions. The s_0 value for root RNG is taken from any external entropy source.

One important additional requirement for f and $compress$ is that the sequence $s(i), i = 1, 2, \dots$, must have large period for any $r(i), i = 1, 2, \dots$.

Let $S = E(\mathbb{F}_p)$ and the order q of elliptic curve is prime, $f(P, c) = P + cG$, where G is a generator of $E(\mathbb{F}_p)$, $x(\cdot)$ is a function that returns the x -coordinate of a point of the curve, $compress(P, r) = 1 + x(P) + r \bmod \lfloor \sqrt{q} \rfloor$. Function f is evidently "one-way" with respect to c for fixed P , $f(s, compress(s, r))$ is believed to be "one-way" with respect to s for any fixed r . One can choose any $r(i), i = 1, 2, \dots$, and be sure that the period of the sequence $s(1), s(2), \dots$ will not be less than $\sqrt{q}$ for any (even degenerate) sequence $r(1), r(2), \dots$.

As it was shown in [6], for $S = E(\mathbb{F}_p)$ for practical needs one can choose b equal to $\frac{\log_2 p}{2}$. For example, to obtain output $out(i)$ one can compute hash value of binary representation of $s(i)$ and then truncate it up to the first $\frac{\log_2 p}{2}$ bits.

If $compress(\cdot, \cdot)$ is such that $0 \leq compress(\cdot, \cdot) \leq \sqrt{q}$, then the minimum possible period of output values of one session is $\sqrt{q}$, and the probability of collision of any pair of outputs in two distinct sessions may be estimated as $1 - (1 - \frac{1}{\sqrt{q/2}})^N \approx \frac{N}{\sqrt{q/2}}$, where N is the total number of keys generated during one session. This estimate is via considering the probability of coinciding $compress(s(i), r(i))$, hence obtaining the estimate of higher bound of probability of collision of $s(i)$ for the case of two sessions starting from equivalent initial states. Assuming that the distribution of $compress(s(i), r(i))$ is close to uniform on $\{1, \dots, \lfloor \sqrt{q} \rfloor - 1\}$ the probability of collision on each step is not larger than $\frac{1}{\sqrt{q/2}}$ with the total number of steps being equal to N . If the total number of sessions is T , then the probability of collision of any pair of outputs in any pair of sessions can be estimated (analogously, using birthday paradox, with N experiments of T sessions uniformly choosing $compress(s(i), r(i)) \in \{1, \dots, \lfloor \sqrt{q} \rfloor - 1\}$) as $1 - e^{-\frac{T^2 N}{\sqrt{q}}}$. For $T = 10^{14}$ sessions (10^7 systems, 10^3 sessions per system every day for 20 years), $N = 10^2$ keys per session and $q \approx 2^{256}$ the latter probability is less than 10^{-8} .

5 External entropy sources

Though most important properties of proposed RNG construction may be proved with zero external entropy ($r(i) = const$ for every i), the main purpose of it is the usage as RNG in cryptographic software with available external entropy sources. In the current section we will say some words about possible options here in the case of absence of hardware RNG.

The first option is usage of bio-RNG, based on delays connected with keyboard and mouse activity. It asks the user to type random text and move mouse for a certain period of time. This option is usually used when generating static keypairs for long-time usage (for one year and more).

The second option is background collecting of entropy connected with computer system operation. For example, in Linux-based systems (see [5]) accumulation of random data buffer is being made during the whole functioning of the system; random data is collected

from hardware disk drive events, system interrupts, routine keyboard activity. This option is often used to generate session keys.

The third option is usage of immediate system entropy collectors. These mechanisms usually make use of information from system timers and counters in the exact moment of key generation. Such techniques prevent attacks that are possible in multi-user systems, where processes in simultaneous sessions can collect the same routine random data, but cannot obtain immediately collected system entropy. This option is faster and more user-friendly than the first one, more secure and more reliable than the second one and when used with dynamical entropy control, it may be exploited for generation of session keys, ephemeral keypairs and initialization vectors.

6 Conclusion

The proposed RNG construction, based on well-known ideology of constructing RNG with “one-way” functions, also meets all mentioned specific requirements for RNGs used in cryptographic software. It is flexible enough: each state transition depends on additional random input, thus the corresponding choice of extra entropy source for this random input can provide best efficiency in some cases (e.g. IVs, seeds, UKMs) when using fastest available external entropy source; best possible statistical properties in other cases (e.g. static keypairs) when using bio-RNGs, for example; balanced solutions for routine cryptographic material (e.g. session keys, ephemeral keys) when using standard external entropy sources, for example, immediate system entropy collectors with usage of previously collected (in background mode) random data. As it was shown before, it keeps all necessary properties of the output data even in the case of hardware failures that make external entropy input degenerate. It is applicable for usage in a large number of simultaneous RNG sessions in one system due to generating initial state of each session RNG from the unique root RNG.

References

- [1] *Sidorenko A.* Design and analysis of provably secure pseudorandom generators // Ph.D. Thesis. — Eindhoven, 2007.
- [2] *Barker E., Kelsey J.* Recommendation for random number generation using deterministic random bit generators, December 2005, NIST Special Publication (SP) 800-90.
- [3] *Blum M., Micali S.* How to generate cryptographically strong sequences of pseudo random bits // Symp. Found. Comput. Sci. — 1982. — P. 112-117.
- [4] *Brown D.* Conjectured security of the ANSI-NIST elliptic curve RNG // Cryptology ePrint Archive, Report 2006/117, 2006, <http://eprint.iacr.org/>

-
- [5] *Guterman Z., Pinkas B., Reinman T.* Analysis of the Linux random number generator // In IEEE Symp. Security and Privacy. — Oakland, CA, USA, May 2006.
 - [6] *Farashahi R.R., Pellikaan R., Sidorenko A.* Extractors for binary elliptic curves // Designs, Codes, and Cryptography. — 2008. — V. 49.
 - [7] *Goldwasser S., Micali S., Tong P.* Why and how to establish a private code on a public network // Symp. Found. Comput. Sci. — 1982. — P. 134-144.
 - [8] *Hstad J., Impagliazzo R., Levin L. A., Luby M.* Construction of a pseudo-random generator from any one-way function // SIAM J. Comput. — 1999. — V. 28. — P. 1364-1396.
 - [9] *Kaliski B. S.* Elliptic curves and cryptography: A pseudorandom bit generator and other tools // Ph.D. thesis. — MIT, Cambridge, MA, USA, 1988.
 - [10] *Long D. L., Wigderson A.* How discreet is the discrete log // ACM Symp. Theory of Computing. — 1983. — P. 413-420.