A Pollard-like pseudorandom number generator over EC

Grzegorz Wojtenko
Wincor Nixdorf EFT Laboratory, Poland
Grzegorz.Wojtenko@wincor-nixdorf.com

Abstract

In this short paper we propose a pseudorandom number generator over *EC* based on Pollard-like method. In contrast to the well known Elliptic Curve Random Number Generator (see e.g. *ANSI* and *NIST* draft standards) the generator is based on a random walk over the group of *EC*-points like in the original Pollard's rho algorithm and only resembles a little bit the linear congruential generator over elliptic curve. Compared to other approaches, the method allows to decrease the cost of generating pseudorandom numbers. This generator could be used in resource constrained devices like smart cards which have already been equipped with *EC*-based tools for other cryptographic purposes.

Key words: random number generation, elliptic curve cryptography, Pollard's rho algorithm, linear congruential generator

Introduction

Use of elliptic curves for generating pseudorandom numbers has been studied for a few years. A good example of this usage can be *ECRGN* which we will use as the reference generator against the proposed one.

ECRGN logically can be divided into two parts: one that generates a point sequence and one that extracts a bit string from the point sequence. The bit extraction based on a truncation is a separate non-trivial problem but is out of the scope of this paper. Our concern is a way of generating the point sequence. We propose to use a Polllard-like method to generate the point sequence as a source of pseudorandom numbers of uniform distribution. An advantage of this method is a low cost of generating pseudorandom bits comparing to *ECRGN*. The method makes use of Teske's findings [1] to make the generator more random.

Actually the method resembles the linear congruential generator (*LCG*). Nevertheless, being more a set of these generators it doesn't suffer from weakness typical to *LCG*s over elliptic curve [7].

Organization of the paper: we shortly recall some basis facts about elliptic curves and existing *ECRNG* and then present the idea of the method. For sake of simplicity we propose to call the method proposed *Pollard-ECRNG* due to the fact of using Pollard's rho-like iterating function.

Elliptic curves

Let *E* be an elliptic curve defined over a prime field *F*. (Note: We use the curve for convenience; the same idea applies to curves over other finite fields.)

$$E(F):\ y^2 = x^3 + ax + b$$

where $a,b \in F$, whose characteristic is *p*.

Let

$\# \ E(F) = nh$,

where *n* is prime and

$h = \# E(F) / n$ is a very small number.

Q, P – any points on E

$Q \in \langle P \rangle$ - where $\langle P \rangle$ is the group of points generated by P of prime order n

Elliptic Curve Random Number Generator (following [2])

Let

$R_i = s_i Q$, where:

- Q is an initialization parameter, actually a point chosen at random
- s_i an integer obtained by taking the bit representation of *x*-coordinate of *P*

$s_{i+1} = x(s_i P)$ where $x(\cdot)$ stands for the transformation of bit representation of *x*-coordinate of *P* into an integer.

The final output of *ECRNG* is a sequence of $r_i = t(x(R_i))$, where $t(\cdot)$ is a function that truncates certain bits from the bit string representation of an elliptic curve point.

As seen above, the most costly for *ECRNG* is calculating R_i which requires on average $\frac{3}{2}\log_2 s$ *EC*-group operations, [11].

Pollard's method

Pollard's method is based on birthday paradox. Its complexity is $O(\sqrt{n})$ and allows to find a solution (e.g. a useful collision when solving discrete logarithm problem on *EC*) in $\sqrt{\pi n / 2}$ steps.

Original Pollard's rho method adapted to *ECDLP* (elliptic curve discrete logarithm problem) is as follows:

Let:

d – an integer: $Q = dP$, $d=?$

T_1, T_2, T_3 – disjoint subsets of $\langle P \rangle$

The iterating function has the following formula:

$$X_{i+1} = \begin{cases} X_i + P, & \text{gdy } X_i \in T_1 \\ 2X_i, & \text{gdy } X_i \in T_2 \\ X_i + Q, & \text{gdy } X_i \in T_3 \end{cases}$$

So, each step of Pollard's rho algorithm requires only one group operation (addition or doubling of a point). The more random the iterating function is the less number of steps must be performed in Pollard's rho method.

Teske presented in [1] that on randomness of Pollard's rho algorithm impact: number of subsets T_i and kind of "walks" (group operations in the iterating function). Teske searched for formulae as close as possible to random mappings. In conclusions Teske suggests to divide $\langle P \rangle$ into about 20 disjoint subsets (at least 6) and to use mixed walks (both additions and doublings).

Following Teske we get:

$v : P \rightarrow \{1, ..., r + q\}$, where $r, q \in N$
$T_1, ..., T_r$ and $V_1, ... V_q$ – pairwise disjoint subsets of similar number of points
$m, n \in \{1,2, ..., |P|\}$

$A_s = m_s P + n_s Q$
$m_s, n_s \in \{1,2, ..., |P|\}, \quad 1 \le s \le r$

$$X_{i+1} = \begin{cases} X_i + A_s & X_i \in T_s \quad 1 \le s \le r \\ 2X_i & X_i \in V_u \quad 1 \le u \le q \end{cases}$$

Teske recommends that an average ratio between number of additions and doublings should be between ¼ and ½.

Pollard-ECRNG

Having in mind Teske's studies one can assume that there is some very random mapping of Pollard type. The mapping allows to generate at random pseudorandom numbers of uniform distribution. This generator doesn't suffer from weakness typical to the linear congruetial generators due to being a set of *LCG*s.

Comparing to *ECRNG*, *Pollard-ECRNG* can also be divided into two parts: generating randomly points and extracting pseudorandom bit sequences. So, the final output of Pollard-*ECRNG* is also a sequence of $r_i = t(x(R_i))$ and the truncating function $t(\cdot)$ remains unchanged. The only difference is that R_i comes from the Pollard-like algorithm instead of the formula $R_i = s_i Q$. The most crucial advantage of the method is that it costs only one group operation to generate next pseudorandom sequence.

Sketch of the Pollard-ECRNG algorithm

1. choose at random $Q \in \langle P \rangle$ and generate s_i like in *ECRGN*

2. find R_i from Teske's formula (as above in the paper)

3. find $r_i = t(x(R_i))$ where $x(\cdot)$ and $t(\cdot)$ like in ECRNG

4. repeat recursively steps 2 and 3

Summary

This short paper presents, according to the author’s knowledge, the new concept of using elliptic curves for generating pseudorandom numbers. In comparison to the standard *ECRNG* only point generation function is changed so all studies concerning *ECRNG*’s security aspects applies to the method in a straightforward way. As the new method is almost cost free it seems to be very suitable for constrained systems like smart cards or mobile phones.

References

[1] Teske E. „Better random walks for pollard’s rho method”
Research Report CORR 98-52, Department of Combinatorics and Optimization,
University of Waterloo, Waterloo, Ontario, Canada. November, 1998.

[2] Brown Daniel, „Conjectured Security of the ANSI-NIST Elliptic Curve RNG”,
http:// www.cacr.math.uwaterloo.ca/techreports/2006/cacr2006-11.pdf

[3] Kristian Gjøsteen, “Comments on dual-ec-drbg/nist sp 800-90 draft december 2005”,
http: //www.math.ntnu.no/_kristiag/drafts/dual-ec-drbg-comments.pdf, March 2006.

[4] Nicolas Guerel, Extracting bits from coordinates of a point of an elliptic curve, Cryptology ePrint Archive, Report 2005/324, 2005, http://eprint.iacr.org/.

[5] Edwin El Mahassni and Igor Shparlinksi, On the uniformity of distribution of congruential generators over elliptic curves, International Conference on Sequences and Their Applications, SETA ’01, 2002, pp. 257–264.

[6] Guang Gong, Thomas A. Berson, Douglas R. Stinson, “Elliptic Curve Pseudorandom Sequence Generators” http:// www.anagram.com/berson/ecpsg99.pdf

[7] Hallgren S., “Linear congruential generators over elliptic curves”
http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=865027

[8] Mascagni M. ,” Parallel linear congruential generators with prime moduli”, http://
www.ima.umn.edu/preprints/MARCH1997/1470.pdf

[9] Brent R., „Uniform Random Number Generators for
Vector and Parallel Computers”, wwwmaths.anu.edu.au/~brent/pd/rpb132tr.pdf

[10] Chung-Chih Li, Bo Sun „Using Linear Congruential Generators for ryptographic Purposes”, http://libra.msra.cn/authordetail.aspx?id=1792495

[11] Guajard J., Paar Ch. „Efficient Algorithms for Elliptic Curve Cryptosystems”
http://citeseer.ist.psu.edu/guajardo97efficient.html