Using public-key cryptosystem PolyDragon to create a PRNG based on random covers for finite groups

Viliam Hromada, Milan Vojvoda

Institute of Computer Science and Mathematics
Slovak University of Technology
Bratislava, Slovakia

13.05.2012, ISCAMI 2012

Generating random bit sequences

- Generating random bit sequences is an important problem in cryptography
- There are two main concepts: RNG and PRNG
- It is impossible to prove whether PRNG is truly „random“, therefore we need statistical testing
- One is always looking for new and better methods of generating random sequences

Cornerstone - MSTg

- In 2011 a new PRNG was proposed (Marquadt, P, et.al) based on random covers for finite groups - MSTg

Definition

Let G be a finite abstract group. Suppose that $\alpha = [A_1, A_2, \ldots, A_s]$ is an ordered collection of subsets $A_i \subseteq G$. We call α a *cover* for G if each element $g \in G$ can be expressed in at least one way as a product of the form

$$g = g_1 \cdot g_2 \cdot \ldots \cdot g_s \tag{1}$$

for $g_i \in A_i$.

Induced mappings

Definition

Let $\alpha = [A_1, A_2, ..., A_s]$ be a cover of a finite group G. Then the vector $(r_1, r_2, ..., r_s)$ where $r_i = |A_i|$ is called the *type* of α.

Definition

Let G be an abstract finite group and let $\alpha = [A_1, A_2, ..., A_s]$ be its random cover. Let $m = \prod_{i=1}^{s} |A_i|$. Then we can define the surjective mapping

$$\breve{\alpha} : \mathbb{Z}_m \to G$$

$$\breve{\alpha}(x) := a_{1,j_1} \cdot a_{2,j_2} \cdot ... \cdot a_{s,j_s}$$

which, we say, is the mapping $\breve{\alpha}$ induced by random cover α.

MSTg - Function F

Definition

Let G_1 and G_2 be two chosen finite groups, with $|G_1| = n$ and $|G_2| = m$. Let α be a random cover of G_1 and γ be a random cover of G_2. Let $\breve{\alpha}, \breve{\gamma}$ be their respective induced mappings. Let ℓ be an integer such that $\ell \geq n$. We define a funcion

$$F : \mathbb{Z}_\ell \rightarrow \mathbb{Z}_m$$

as a composition of mappings:

$$F : \mathbb{Z}_\ell \xrightarrow{\breve{\alpha}} G_1 \xrightarrow{f_1} \mathbb{Z}_n \xrightarrow{\breve{\gamma}} G_2 \xrightarrow{f_2} \mathbb{Z}_m$$

MSTg algorithm

- **Input:** Integers ℓ, m, function $F : \mathbb{Z}_\ell \to \mathbb{Z}_m$ as defined before and random and secret seed $s_0 \in \mathbb{Z}_\ell$
- **Output:** t pseudorandom numbers $z_1, z_2, \ldots, z_t \in \mathbb{Z}_m$

1. For i from 1 to t do following:
   1. $s_i = (s_{i-1} + 1) \pmod{\ell}$
   2. $z_i = F(s_i)$
2. Return $(z_1, z_2, \ldots, z_t)$.

Ok, and what next?

- The security and the properties of the output sequences of the MSTg generator relies heavily on the generated random covers (i.e. on the function F).
- The design of MSTg allows great flexibility in generating the random covers
- We have decided to generate the covers by using the public-key cryptosystem PolyDragon

Public-key cryptography

- You use it without knowing it :)
- Imagine a regular metal letterbox
- Who can throw letters into it?
- Who can open it?
- Analogy with public-key cryptography

PolyDragon

- PolyDragon is a multivariate public-key encryption scheme proposed in 2009 (Singh, R. P., et.al.)
- Its security is based on the MQ-problem
- Its design is based on permutation polynomials over a finite field $\mathbb{F}_{2^n}$.

PolyDragon - Public-key

Let $\mathbb{F}_{2^n}$ be a finite field. Then for each element $u \in \mathbb{F}_{2^n}$ there exists exactly one element $v \in \mathbb{F}_{2^n}$ such that the equation

$$(u^{2^m} + u + \alpha)^{2^m}(v + v^2 + \gamma) + u(u^{2^m} + u + \alpha)(v + v^2 + \gamma) + \\ +(u^{2^m} + u + \alpha)(v + v^2 + \gamma)^{2^m} + Tr(v)(u^{2^m} + u + \alpha)(v + v^2 + \gamma) = 0$$

holds, where $\alpha, \gamma \in \mathbb{F}_{2^n}$

PolyDragon - Public-key

We can easily identify the field $\mathbb{F}_{2^n}$ with the field $\mathbb{F}_2^n$. Therefore we can substitute:

- u by a vector $x = (x_1, x_2, \dots, x_n)$
- v by a vector $y = (y_1, y_2, \dots, y_n)$
- $Tr(v) = \zeta_y$

By doing this, we obtain the final public-key, which is used in the cryptosystem.

$$\begin{aligned}&\sum a_{lijk} x_i x_j y_k + \sum b_{lij} x_i x_j + \sum (c_{lij} + \zeta_y) x_i y_j \\ &+ \sum (d_{lk} + \zeta_y) y_k + \sum (e_{lk} + \zeta_y) x_k + f_l = 0,\end{aligned}$$

where $1 \leq l \leq n$ and $a_{lijk}, b_{lij}, c_{lk}, d_{lk}, e_{lk} \in \mathbb{F}_2$.

Generating random covers

- We have used PolyDragon cryptosystem in CTR mode to generate random covers
- Each time we generated a random cover we initialized the PolyDragon with different values of α, γ.
- A random seed was chosen which then was used as a nonce in the CTR mode

Testing approach

Our testing proceeded as follows:

- We have used 3 different test suites: NIST, Diehard, AIS-31
- We tested 3 configurations of MSTg:
  - 257/129, 2 covers
  - 129/65, 3 covers
  - 129/65, 4 covers
- For each configuration we generated 50 sequences of length 1 MB
- For each configuration we generated 5 sequences of length 10 MB
- Software for statistical testing - Paranoya 2011 (c) UIM FEI STU

First construction

- $n = 257, m = 129$
- Random covers α, γ
- 10 MB sequences:
  - AIS-31: 60% success rate
  - DIEHARD: 60% success rate
- 1 MB sequences:
  - NIST: 78% success rate
  - 2 tests have not been passed at all by all sequences!
  - Generator has not passed the test

Second construction

- $n = 129, m = 65$
- Random covers α, α_1, γ
- 10 MB sequences:
  - AIS-31: 100% success rate
  - DIEHARD: 60% success rate
- 1 MB sequences:
  - NIST: 96% success rate
  - The highest fail rate was 8%
  - Generator has not passed the test

Third construction

- $n = 129, m = 65$
- Random covers $\alpha, \alpha_1, \alpha_2, \gamma$
- 10 MB sequences:
  - AIS-31: 100% success rate
  - DIEHARD: 80% success rate
- 1 MB sequences:
  - NIST: 98% success rate
  - The highest fail rate was 8%
  - Generator has not passed the test

Conclusion

- We have constructed a PRNG by combining two interesting designs
- Unfortunately, the tests rejected the hypothesis that the generator is a „good“ PRNG
- However, there's still hope...

Bibliography I

Singh, R.P., et.al
PolyDragon: An Efficient Multivariate Public-key Cryptosystem.
Journal of Mathematical Cryptology, 4:365–373,2009.

Marquardt, P., et. al.
Pseudorandom number generators based on random covers for finite groups.
Designs, Codes and Cryptography, 0925-1022:1–12, 2011. 2000.