A Variation on Knellwolf and Meier's Attack on the Knapsack Generator

Florette Martinez
florette.martinez@ens.fr

Abstract

Pseudo-random generators are deterministic algorithms that take in input a random secret seed and output a flow of random-looking numbers. The Knapsack generator, presented by Rueppel and Massey in 1985 is one of the many attempt at designing a pseudo-random generator that is cryptographically secure. It is based on the subset-sum problem, a variant of the Knapsack optimization problem, which is considered computationally hard.

In 2011 Simon Knellwolf et Willi Meier found a way to go around this hard problem and exhibited a weakness of this generator. In addition to be able to distinguish the outputs from the uniform distribution, they designed an algorithm that retrieves a large portion of the secret. We present here an alternate version of the attack, with similar costs, that works on the same range of parameters but retrieves a larger portion of the secret.

1 Introduction

Pseudo-random generators are deterministic algorithms that take in input a random secret seed and output a flow of random-looking numbers. They allow applications to have access to large amount of randomness for a very low computational cost. Cryptosystems are the kind of applications that require a lot of randomness, if only to generate the secrets needed. But to remain secure theses applications need good randomness, computationally indistinguishable from the uniform distribution. The Knapsack Generator, presented by Rueppel and Massey in 1985 [3], is one of the many attempt at designing a pseudo-random generator that is cryptographically secure. It is based on the subset-sum problem, a variant of the Knapsack optimization problem, which is considered computationally hard. One should note that the security of the Knapsack Generator does not reduce to the subset sum problem.

In 2011 Simon Knellwolf et Willi Meier [2] found a way to go around this hard problem and exhibited a weakness that is not related to the subset sum problem. In addition to be able to distinguish the outputs from the uniform distribution, they designed an algorithm that retrieves a large portion of the secret. To obtain the secret seed of the generator, they need a matrix having a small norm. To compute this matrix, they use lattice techniques.

We present here an alternate version of the attack, with similar costs, that works on the same range of parameters but retrieves a larger portion of the secret. We apply the same kind of lattice techniques but to attack more directly the problem. Instead of using lattices to solve a side problem, we highlight how much the lattices are the heart of the attack.

2 Preliminaries

2.1 Norms

In this paper we will talk about vectors "close" to each other and "small" matrices. We need to define theses notions. We will use two norms on the vectors: the 2-norm $\|.\|_2$ and the infinite norm $\|\|_\infty$ defined on $\mathbf{x} = (x_1, \ldots, x_n)$ by:

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2} \text{ and } \|\mathbf{x}\|_\infty = \underset{i\in\{1,\ldots,n\}}{max} |x_i|$$

As $\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq n\|\mathbf{x}\|_\infty$, a "small" vector for the 2-norm is small for the infinite norm and vice versa.

We will also use matrix norms induced by $\|.\|_2$ and $\|\|_\infty$ defined on a matrix $M \in \mathcal{M}_{n\times m}(\mathbb{R})$ by:

$$\|M\|_2 = \max_{\mathbf{x}\in\mathbb{R}^m\setminus 0} \frac{|M\mathbf{x}|_2}{|\mathbf{x}|_2} \text{ and } \|M\|_\infty = \max_{\mathbf{x}\in\mathbb{R}^m\setminus 0} \frac{|M\mathbf{x}|_\infty}{|\mathbf{x}|_\infty}$$

or

$$\|M\|_2 = \max_{\mathbf{x}\in\mathbb{R}^n\setminus 0} \frac{|\mathbf{x}M|_2}{|\mathbf{x}|_2} \text{ and } \|M\|_\infty = \max_{\mathbf{x}\in\mathbb{R}^n\setminus 0} \frac{|\mathbf{x}M|_\infty}{|\mathbf{x}|_\infty}$$

given the context.

2.2 Lattices, Closest Vector Problem and approximate solutions

A lattice $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^m$ of finite rank n. It can be expressed using a base $\mathcal{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ where the $\mathbf{b}_i$ are independent vectors of $\mathbb{R}^m$ as

$$\mathcal{L} = \{\sum_{i=1}^{n} \alpha_i \mathbf{b}_i | (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}^n\}.$$

The basis is not unique.

To compute "small" basis, we will use as a black box the **LLL**-algorithm by , a polynomial algorithm that takes in input a basis $\mathcal{B}$ for $\mathcal{L}$ and outputs a smaller base $\mathcal{B}'$ of the same lattice.

We will first present a classical problem in lattices.

Definition 1 (The Closest Vector Problem)**.** *Given a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a target $\mathbf{t} \in \mathbb{R}^n$ usually not in the lattice,* the Closest Vector Problem (CVP) *is finding $\mathbf{v}$ the closest vector to $\mathbf{t}$ in $\mathcal{L}$. In other words, $\mathbf{v} \in \mathcal{L}$ must satisfies :*

$$\|\mathbf{v} - \mathbf{t}\|_2 = \min_{\mathbf{x}\in\mathcal{L}} \|\mathbf{x} - \mathbf{t}\|_2.$$

In the following, we will not need to solve the CVP, only approximations. To do it we will use the Babai rounding method.

The Babai Rounding method: We consider the lattice $\mathcal{L}$ of full rank and $\mathcal{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ a basis of the lattice. We define M as

$$M = \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{pmatrix} \text{ and } \mathcal{L} = \{\boldsymbol{\alpha} M | \boldsymbol{\alpha} \in \mathbb{Z}^n\}.$$

We also consider $\mathbf{t} \in \mathbb{R}^n$ the target. As M is invertible, there exist $\boldsymbol{\beta} \in \mathbb{R}^n$ such that $\boldsymbol{\beta} M = \mathbf{t}$. If we choose $\mathcal{B}$ small (for example the output of the LLL-algorithm), then $\mathbf{v} = \lceil \boldsymbol{\beta} \rfloor M$ is in the lattice and close to $\mathbf{t}$ where $\lceil \boldsymbol{\beta} \rfloor = (\lceil \beta_1 \rfloor, \dots, \lceil \beta_n \rfloor)$,

$$\|\mathbf{v} - \mathbf{t}\|_\infty \leq \frac{1}{2} \|M\|_\infty.$$

2.3 Fundamental domain and counting points

Let $\mathcal{L}$ be a lattice and $\mathcal{B}$ be one of its basis. We define $\mathcal{D}$ the fundamental domain as the set of points encompassed by the vectors of the basis.

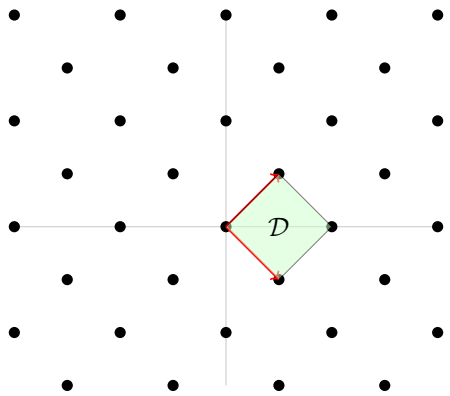

Figure 1: The fundamental domain defined by the basis (1,1),(1,-1)

The fundamental domain, as the basis, is not unique for a given lattice. But its volume is and we define $det(\mathcal{L}) = vol(\mathcal{D})$

We can then center the fundamental domain on a point of the lattice and pave the whole space. Each fundamental domain contains on and only one lattice point in its center. To approximate the number of lattice points in a convex shape C, we compute $\frac{vol(C)}{det(\mathcal{L})}$.

3 The Knapsack Generator and the original attack

3.1 Definition and weakness of the Knapsack Generator

The Knapsack Generator was presented in 1985 by Rueppel and Massey in [3]. We consider three public parameters, n, $\ell \in \mathbb{N}$ and P a retroactive polynomial in $\mathbb{F}_2[X_0, \dots, X_{n-1}]$. Let

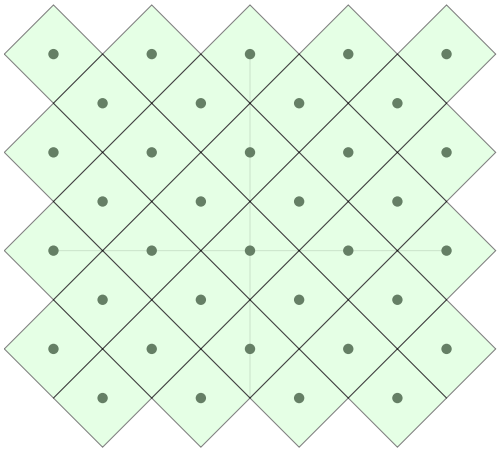

Figure 2: The space paved with fundamental domains

$\mathbf{u} = (u_0, \ldots, u_{n-1}) \in \{0,1\}^n$ be a secret seed and $\boldsymbol{\omega} = (\omega_0, \ldots \omega_{n-1}) \in \{0, \ldots, 2^n - 1\}^n$ be a vector of n secret weights.

At step j the knapsack generator produce a new bit out of an LFSR given by

$$u_{n+j} = P(u_j, \ldots, u_{j+n-1})$$

and an internal state v_i given by

$$v_j = \sum_{i=0}^{n-1} u_{i+j}\omega_i \bmod 2^n$$

The output s_j is given by the leading $n - \ell$ bits of v_j:

$$s_j = v_j // 2^\ell$$

where // is the Euclidean division. We denote by δ_j the truncated value: $v_j = 2^\ell s_j + \delta_j$

In 2011, S. Knellwolf and W. Meier found the main weakness of this generator and presented a first attack [2].

Weakness: The secret of the generator is made of the seed $\mathbf{u}$ –n bits– and the vector of weights $\boldsymbol{\omega}$ – n^2 bits –. It is *unbalanced*: if we decide to work with a secret key of 1024 bits it means the seed $\mathbf{u}$ will only be made of 32 bits. We can construct an attack based on an exhaustive search on $\mathbf{u}$.

The main part of the attack will be to describe an algorithm ApproxWeights that, given the secret seed $\mathbf{u}$ and some outputs of the generator, can compute an approximation of the vector $\boldsymbol{\omega}$. As the subset-sum step is linear in $\boldsymbol{\omega}$, the outputs given by the generator applied on $\mathbf{u}$ and an approximation of $\boldsymbol{\omega}$ will be close to the outputs of the generator applied on $\mathbf{u}$ and $\boldsymbol{\omega}$.

Let m be the number of outputs need in ApproxWeights and ϵ the number of additional outputs need for checking the consistance. We denote $\mathbf{s} = (s_0, \ldots, s_{m-1})$ and $\hat{\mathbf{s}} = (s_0, \ldots, s_{m-1+\epsilon})$.

The layout of the attack is the following:

Algorithm 1 General Attack

```
1: procedure Attack(ŝ)
2:     for u′ ∈ {0,1}^n do
3:         ω′ ←ApproxWeights(u, s)
4:         t ← KnapsackGenerator(u, ω′)
5:         if t is close to ŝ then
6:             return u, ω′
```

3.2 ApproxWeights by Knellwolf and Meier

We present in this subsection the algorithm by Knellwolf and Meier to highlight difference between their version and our version.

Let $\mathbf{v} = (v_0, \ldots, v_{m-1})$ be the vector of m consecutive internal states, with $m > n$, then

$$\boldsymbol{\omega} \times U \equiv \mathbf{v} \bmod 2^n$$

where the matrix $U \in \mathcal{M}_{n\times m}$ is given by

$$U = \begin{pmatrix} u_0 & u_1 & \ldots & u_{m-1} \\ u_1 & u_2 & \ldots & u_m \\ & & \vdots & \\ u_{n-1} & u_n & \ldots & u_{n+m-2} \end{pmatrix}.$$

As U is full rank over 2^n [4] and $\mathbf{u}$ is supposed to be known, we can construct with basic arithmetical operations $T \in \mathcal{M}_{m\times n}(\mathbb{Z})$ such that

$$UT = I_n \bmod 2^n$$

If we consider $\boldsymbol{\delta} = (\delta_0, \ldots, \delta_{m-1})$, we can try to decompose $\boldsymbol{\omega}$ as

$$\begin{aligned} \boldsymbol{\omega} &\equiv vT \bmod 2^n & (1) \\ &\equiv 2^\ell \mathbf{s}T + \boldsymbol{\delta}T \bmod 2^n & (2) \end{aligned}$$

The only unknown left is $\boldsymbol{\delta}T$ where $\boldsymbol{\delta}$ is small. If we can control the size of T we can obtain the higher bits of $\boldsymbol{\omega}$ as the higher bits of $2^\ell \mathbf{s}T$.

Let $\mathcal{L}_1 = \{\ x \in \mathbb{Z}^m \mid Ux \equiv 0 \bmod 2^n\}$. We will construct the matrix $\hat{T}$ column by column.

For the i-th column we consider a vector T_i such that $UTi \equiv \mathbf{e}_i \bmod 2^n$ where $\mathbf{e}_i$ is the vector with a one on the i-th position and zeros elsewhere. Now, using lattice techniques, we find $\mathbf{x}_i$ a close vector to T_i in $\mathcal{L}_1$. Then the vector $\hat{T}_i = T_i - \mathbf{x}_i$ is small and satisfies $U\hat{T}_i \equiv \mathbf{e}_i \bmod 2^n$: it becomes the i-th column of $\hat{T}$.

The infinite norm $|\boldsymbol{\omega} - 2^\ell \mathbf{s}\hat{T}|_\infty$ is bounded by $|\boldsymbol{\delta}|_\infty \|\hat{T}\|_\infty$. Bounding heuristically $\|\hat{T}\|_\infty$ appears to be challenging.

4 A more directly lattice-based original attack

In this section we present our version of the ApproxWeight algorithm that retrieve more information.

We still consider $\mathbf{u}$ to be known. Instead of using lattice technique to shrink the matrix T, we can use it to directly solve the problem.

As $\boldsymbol{\omega}U \equiv \mathbf{v} \bmod 2^n$, $\mathbf{v}$ is in the lattice $\mathcal{L}_2 = \{\ y \in \mathbb{Z}^m \mid xU \equiv y \bmod 2^n \text{ for } x \in \mathbb{Z}^n\}$ and by construction $\mathbf{v}$ is close to the public vector $2^\ell\mathbf{s}$. We compute $\mathbf{v}'$ the closest vector to $2^\ell\mathbf{s}$ in $\mathcal{L}_2$. In practice, this vector is never $\mathbf{v}$ so it does not give us $\boldsymbol{\omega}$. Let $\boldsymbol{\omega}'$ be a pre image of $\mathbf{v}'$ such that

$$\omega' U \equiv \mathbf{v}' \bmod 2^n.$$

Is $\boldsymbol{\omega}'$ a good approximation of $\boldsymbol{\omega}$? Yes !

Why that it works ?

4.1 Counting lattice points again

We would like to have something in the line of :

$$|\mathbf{v} - \mathbf{v}'|_\infty \text{ small } \Rightarrow |\boldsymbol{\omega} - \boldsymbol{\omega}'|_\infty \text{ small}$$

We will now denote $\mathbf{x}$ the small vectors in $\mathbb{Z}^n$ and $\mathbf{y}$ the small vectors in $\mathcal{L}_2$ with the relation $\mathbf{y} = \mathbf{x}U \bmod 2^n$

We already know the following given by the definition of the induced norm

$$|\mathbf{x}|_\infty < \frac{2^k}{\|U\|_\infty} \Rightarrow |\mathbf{y}|_\infty < 2^k. \tag{3}$$

We consider two sets :

$$\mathcal{A} = \{\mathbf{x} \in \mathbb{Z}^n \text{ s.t. } |\mathbf{x}|_\infty \leq \frac{2^k}{\|U\|_\infty}\}$$

and

$$\mathcal{B} = \{\mathbf{y} \in \mathcal{L}_2 \text{ s.t. } |\mathbf{y}|_\infty \leq 2^k\}$$

With the notation $\mathcal{A} \times U := \{xU \text{ s.t. } x \in \mathcal{A}\}$, we already know that $\mathcal{A} \times U \subset \mathcal{B}$ thanks to equation 3 and as long as $2^k\|U\|_\infty < 2^n$ we have $|\mathcal{A} \times U \bmod 2^n| = |\mathcal{A}|$.

We search for k such that $|\mathcal{B}| \leq |\mathcal{A}|$ to ensure $\mathcal{B} \subset \mathcal{A} \times U$

How to compute $|\mathcal{A}|$: We can exactly compute the number of points in $\mathcal{A}$, it is given by

$$|\mathcal{A}| = (2 \times \lceil \frac{2^k}{\|U\|_\infty} \rfloor - 1)^n$$

How to approach $|\mathcal{B}|$: The set $\mathcal{B}$ can be rewritten as an intersetcion: $\mathcal{B} = \mathcal{L}_2 \bigcup B_{m,\infty}(2^k)$. So the number of point in $\mathcal{B}$ is roughtly $\frac{vol(B_{m,\infty}(2^k))}{|det(\mathcal{L}_2)|}$

The lattice $\mathcal{L}_2$ is a 2^n-ary lattice or rank n and $\mathbb{Z}^m$ so its determinant satisfies $det(\mathcal{L}_2) > 2^{n(m-n)}$ [1].

Hence

$$|\mathcal{B}| \leq \frac{(2 \times 2^k - 1)^m}{2^{n(m-n)}}$$

For $n = 32$ and $m = 40$ we obtain $|\mathcal{B}| < |\mathcal{A}|$ for $k \leq 15$ (we approximate $\|U\|_\infty$ by $\frac{n}{2}$).

To end the explanation: We fix $y = \mathbf{v} - \mathbf{v}'$. Because both $\mathbf{v}$ and $\mathbf{v}'$ are close to $2^\ell \mathbf{s}$, we have $|y|_\infty < 2^k$ with $k = \ell + 1$. If k is such that $|\mathcal{B}| < |\mathcal{A}|$ then there exists $x \in \mathbb{Z}^n$ such that $|x|_\infty < \frac{2^k}{|U\|_\infty}$ and $xU = y$.

If we consider $\alpha = \boldsymbol{\omega} - \boldsymbol{\omega}' \bmod 2^n$ then $\alpha U \equiv y \bmod 2^n$. As U has a pseudo inverse $\bmod 2^n$ then $\alpha \equiv x \bmod 2^n$ and if the two already live in $\{-2^{n-1}, \ldots, 2^{n-1}\}^n$, we obtain $\alpha = x$ and thus $|\boldsymbol{\omega} - \boldsymbol{\omega}'| < \frac{2^{\ell+1}}{\|U\|_\infty}$.

4.2 Experimental Results

For $n = 32$ and $m = 40$ we obtain the following results for the two algorithms ApproxWeigths.

ℓ	5	10	15	20	25
$log_2(\|\boldsymbol{\omega} - 2^\ell \mathbf{s}\hat{T}\|_\infty)$	9.9	14.9	19.8	24.7	~~31~~
$log_2(\|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_\infty)$	3.6	8.7	13.6	18.7	~~31~~

About the range of the attack: This new attack was supposed to work at least up to $\ell = 15$, we can see it works better in practice. But it does not seem to work on a larger range than the original attack by Knellwolf and Meier.

About the information retrieved: On the set of parameters where both attacks work, our retrieve much more information.

About the complexity: The bottleneck of theses algorithms is the use of the LLL-algorithm. In our version we run one LLL by call to our ApprowWeight function. In their version, they would call n LLL by call to their ApprowWeight function but they could reuse some of the results for other calls to their ApprowWeight function. In the end we both call the LLL-algorithm around 2^n times for the full attack.

References

[1] Daniel Dadush. Mastermath, Lecture Notes: Intro to Lattice Algorithms and Cryptography , 2018. URL: `https://homepages.cwi.nl/~dadush/teaching/lattices-2018/notes/lecture-9.pdf`.

[2] Simon Knellwolf and Willi Meier. Cryptanalysis of the knapsack generator. In Antoine Joux, editor, *Fast Software Encryption – FSE 2011*, volume 6733 of *Lecture Notes in Computer Science*, pages 188–198. Springer, Heidelberg, February 2011.

[3] RA Rueppel and JL Massey. Knapsack as nonlinear function, ieee intern. symp. of inform. theory, 46, 1985.

[4] Joachim von zur Gathen and Igor Shparlinski. Predicting subset sum pseudorandom generators. In Helena Handschuh and Anwar Hasan, editors, *SAC 2004: 11th Annual International Workshop on Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 241–251. Springer, Heidelberg, August 2004.