CSE 206A: Lattice Algorithms and Applications Fall 2017

Cryptanalysis of Pseudorandom Generators

Instructor: *Daniele Micciancio* UCSD CSE

As a motivating application for the study of lattice in cryptography we consider the construction of pseudorandom generators. We recall that a pseudorandom generator is a program $G(x)$ (computable in deterministic polynomial time) that maps bitstrings $x \in \{0,1\}^n$ to longer strings $G(x) \in \{0,1\}^m$ such that, if x is chosen uniformly at random and kept secret, then the output $G(x)$ will "look" random to any efficient observer or adversary. We will formally define secure pseudorandom generators later on. But, for now, we will use the minimal security requirement that given $G(x)$, it should be computationally hard to recover the secret seed x.

We consider two popular types of generators: subset-sum generators and linear congruential generators. For simplicity, in both cases, we consider a generalized definition of generator where the input x and output $G(x)$ are not necessarily bitstrings, but elements of some arbitrary set. We will assume that the secret seed x is chosen uniformly at random from a set of size approximately 2^n, so that mounting an exhaustive search attack on the seed would take exponential time. The task of the generator is to stretch this relatively short random seed into a polynomially longer string, e.g., an element from a set of size roughly 2^{n^2}.

Subset-Sum Generator Let p be a prime of size $\log_2 p \approx n^2$, and $a_1, \ldots, a_n \in \mathbb{Z}_p$ be chosen uniformly at random. The parameters $p, a_1, \ldots, a_n$ are made public, and known to everybody. You can think of these parameters as part of the program that computes G.

- The input to the generator is a sequence of n bits $(x_1, \ldots, x_n) \in \{0,1\}^n$.
- The output is $G(\mathbf{x}) = \sum_i a_i \cdot x_i \pmod p$.

Notice that the output is represented by $\log_2 p \approx n^2$ bits. So, the generator stretches n-bits to roughly n^2 bits.

You can think of the output $G(\mathbf{x})$ as the sum (modulo p) of a random subset $I = \{i : x_i = 1\}$ of the values $a_1, \ldots, a_n$, hence the name "subset-sum". The subset-sum problem asks, given $a_1, \ldots, a_n$ and target value b, recover a subset I (or its indicator vector $\mathbf{x}$) that adds up to $G(\mathbf{x}) = b$. Subsetsum is a famous NP-hard problem, so one may hope that recovering $\mathbf{x}$ from b is indeed computationally hard, and cannot be solved much faster than an exponential time exhaustive search.

Truncated Linear Congruential Generator (LCG) Here we consider another popular type of generators based on linear recurrence relations. The popularity of the generator is due to its simplicity, and variants of this generator were used for some time as the standard pseudorandom generator for the Unix operating system.

As before, we first choose random public parameters. Let p be a prime with $\log_2 p \approx n$ bits, and choose $a, b \in \mathbb{Z}_p$ uniformly at random. The values p, a, b are made public and hardwired in the program computing the generator. The input to the generator is a value $x \in \mathbb{Z}_p$, representable with $\log_2 p \approx n$ bits. This value defines a sequence according to the recurrence relation

1. $x_0 = x$
2. $x_{i+1} = a \cdot x_i + b \pmod{p}$ for $i = 1, 2, \ldots$.

Of course, producing $x_0, \ldots, x_{k-1}$ as the output of G would be insecure, because it would reveal the secret x_0. (Omitting x_0, and starting the output with x_1 would hardly make any difference because the secret seed can still be recovered as $x = (x_1 - b)/a \pmod{p}$.) In order to protect the seed x, the generator G will output only a portion of the bits from each x_i. For example, G may output only the $n/10$ least significant bits $y_i = x_i \bmod 2^{n/10}$. (Here we are implicitly assuming n is a multiple of 10. Alternatively, one can round $n/10$ to an integer.) In order to stretch the n input bits x to a sequence of n^2 bits, the generator computes a sequence of length $k = 10n$, and outputs $G(x) = (y_0, \ldots, y_{k-1})$. Since each y_i is $n/10$ bits, the output of G has length $k * (n/10) = n^2$.

In summary, for some parameter $\delta \in [0, 1]$:

- The input to the generator is a value $x \in \mathbb{Z}_p$, representable with $\log_2 p \approx n$ bits.
- The output of the generator is $G(\mathbf{x}) = (y_0, \ldots, y_{k-1})$ where $y_i = x_i \bmod 2^{\lceil \delta n \rceil}$ is obtained by truncating the values produced by the linear congruential recurrence $x_0 = x$, $x_{i+1} = ax_i + b \bmod p$.

1 Lattices

Neither generator (LCG or subsetsum) is trivially insecure. In fact, both types of generators, for some values of the parameters, have been (or may still be) used in practice. However, we will see that both generators can be completely broken using lattices. This may be surprising at first because neither generator makes explicit use of lattices, and demonstrates how lattices are powerful combinatorial objects that can be used to attack a variety of seemingly unrelated problems.

So, the first question to answer is: where is the hidden lattice, and what lattice problem is relevant to the cryptanalysis problem? We will see that in both cases the relevant lattice problem is the Closest Vector Problem.

Definition 1 *The closest vector problem (CVP), given a lattice basis* $\mathbf{B}$ *and a target vector* $\mathbf{t}$*, asks to find the lattice point* $\mathbf{Bx}$ *closest to the target, i.e., find an integer vector* $\mathbf{x}$ *such that* $\|\mathbf{Bx} - \mathbf{t}\|$ *is minimized.*

In general, as we will see, CVP is an NP-hard problem. So, we have little hope to efficiently solve this problem in its full generality. However, in practice one often needs to solve only a special variant of this problem, where the target is known to be close to the lattice.

Definition 2 *The Bounded Distance Decoding problem (α-BDD), given a lattice basis* $\mathbf{B}$ *and a target vector* $\mathbf{t}$ *such that dist*$(\mathbf{t}, \mathcal{L}(\mathbf{B})) \leq \alpha\lambda_1(\mathcal{L}(\mathbf{B}))$*, asks to find a lattice point* $\mathbf{Bx}$ *such that dist*$(\mathbf{t}, \mathbf{Bx}) \leq \alpha\lambda_1(\mathcal{L}(\mathbf{B}))$.

In the above definition $\text{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) = \min\{\|\mathbf{t} - \mathbf{Bx}\| : \mathbf{x} \in \mathbb{Z}^n\}$ is the distance between the target and the closest lattice point, and $\lambda_1(\mathcal{L}(\mathbf{B})) = \min\{\|\mathbf{Bx}\| : \mathbf{x} \in \mathbb{Z}\backslash\{\mathbf{0}\}\}$ is the minimum distance of the lattice. Typically, α is very close to 0, making the target very close to the lattice, compared to the distance between lattice points. Notice than as soon as $\alpha < 1/2$, the solution to BDD is necessarily unique, as for any two solutions $\mathbf{x}, \mathbf{y}$, their difference $\mathbf{x} - \mathbf{y}$ is a lattice vector of length at most

$$\|\mathbf{x} - \mathbf{y}\| \leq \|\mathbf{x} - \mathbf{t}\| + \|\mathbf{t} - \mathbf{y}\| \leq 2\alpha\lambda_1 < \lambda_1.$$

So, by definition of λ_1 it must be $\mathbf{x} - \mathbf{y} = \mathbf{0}$, i.e., $\mathbf{x} = \mathbf{y}$ are the same solution.

Later in the course we will show that the LLL basis reduction algorithm allows to solve α-BDD in polynomial time when $\alpha = 2^{-(n-1)/2}$ is exponentially small in the lattice dimension n.

2 Cryptanalysis of LCG

First thing to do is to set up an appropriate lattice. Let p, a, b be the publicly known parameters, and, for any seed x, define the sequence $x_0, \ldots, x_{k-1}$ where $x_0 = x$ and $x_i = ax_{i-1} + b \pmod{p}$ for $i = 1, \ldots, k-1$. By solving the recurrence we see that $x_i = a^i x + b_i \pmod{p}$ for $b_i = b\sum_{j<i} a^j \pmod{p}$. (The exact expression for b_i is not really relevant. All that matters is that each b_i can be easily computed from the public parameters a, b, p.) Write each x_i as $x_i = z_i 2^u + y_i$, where u is the number of bits output by the generator from each x_i, $z_i = \lfloor x_i/2^u \rfloor < p/2^u$ and $y_i = x_i \bmod 2^u$. The output of the generator can be written as $\mathbf{y} = y_0, \ldots, y_{k-1}$. In vector notation

$$\mathbf{a}x + \mathbf{b} = 2^u\mathbf{z} + \mathbf{y} \pmod{p}$$

where $\mathbf{a} = (1, a, a^2, \ldots, a^{k-1}) \pmod{p}$, $\mathbf{b} = (b_0, \ldots, b_{k-1})$, and $\mathbf{y}$ are known, and $\mathbf{z}$ is an unknown vector with relatively small entries $z_i \in [0, p/2^u)$.

To fix some parameters, let $n = \log_2 p$ be the bitsize of the modulus p, and assume $u = \delta \cdot n$ for some (small) constant $0 < \delta < 1$.

Let k be the number of outputs the attacker takes from the generator. At the end of the proof, we discuss appropriate choices of k for which the attacker can recover the seed with high probability.

Consider the integer lattice Λ_a generated by the basis

$$\mathbf{B} = \begin{bmatrix} 1 & & & & \\ a & p & & & \\ a^2 & & p & & \\ \vdots & & & \ddots & \\ a^{k-1} & & & & p \end{bmatrix} \qquad \mathbf{t} = 2^{-u}(\mathbf{y} - \mathbf{b}) \pmod{p}$$

and the target vector $\mathbf{t}$. We remark that in the definition of $\mathbf{t}$, the value 2^{-u} should be interpreted as the inverse of 2^u in $\mathbb{Z}_p$, so that the target vector has integer coordinates. Note that k, the number of generator outputs, becomes the dimension of the lattice. First, we claim that the target $\mathbf{t}$ is close to the lattice. In fact, if we multiply the first basis vector by by $2^{-u}x \pmod{p}$ and using the other basis vectors for reduction modulo p gives

$$\mathbf{a}(2^{-u}x) = 2^{-u}(2^u\mathbf{z} + (\mathbf{y} - \mathbf{b})) = \mathbf{z} + \mathbf{t} \pmod{p}$$

which is at distance $\|\mathbf{z}\| \leq \sqrt{k}(p/2^u) = \mu$ from the target $\mathbf{t}$. If the minimum distance of the lattice Λ is much larger than μ, then this is an instance of the BDD problem, and can be efficiently solved by lattice approximation algorithms.

We conclude the analysis by showing that if a is chosen uniformly at random, then $\lambda(\Lambda)$ is much larger than μ (say, by a factor $\gamma > 1$) with very high probability. We will see that BDD can be solved in polynomial time when γ is exponential in the dimension of the lattice. Since $\mathbf{B}$ defines a k-dimensional lattice, we may set $\gamma = 2^{k-1}$. All vectors in Λ_a equal $c \cdot \mathbf{a} \pmod{p}$ for some integer c. So, the shortest nonzero vector in Λ_a can be obtained by picking an integer $c \neq 0 \pmod{p}$, and then bringing all coordinates of $c \cdot \mathbf{a}$ in the range $(-p/2, p/2)$ by reduction modulo p. Let A be the set of all a's such that there is a $c \in \mathbb{Z}_p^*$ for which $|c \cdot a^i \pmod{p}| \leq \gamma\mu$ for all $i = 0, \ldots, k-1$. (When taking the absolute value of a number modulo p, we always assume that the number has been reduced to the interval $(-p/2, p/2)$.) Note that $a \in A$ is a necessary but insufficient condition for $\lambda(\Lambda) < \gamma\mu$. Certainly, if any component of the shortest vector satisfies $|c \cdot a^i \pmod{p}| > \gamma\mu$, then the length of the vector will be greater than $\gamma\mu$. So $a \notin A$ implies $\lambda(\Lambda) > \gamma\mu$. But even if no component is longer than $\gamma\mu$, it may turn out that the vector is still longer than this value. We will show that satisfying even this relaxed condition is improbable by bounding the size of A. To this end, we claim that all $a \in A$ are the root of a nonzero polynomial $m(a) = 0 \pmod{p}$ with small coefficients.

Lemma 3 *For any $a \in A$ there is a nonzero polynomial $q \in \mathbb{Z}_p[x]$ of degree less than k with coefficients $|q_i| \leq \beta = (k\gamma\mu)^{1/(k-1)}$ such that $q(a) = 0$.*

Proof. Fix an $a \in A$ and let $c \neq 0 \pmod{p}$ be an integer such that $|c \cdot a^i \pmod{p}| \leq \gamma\mu$ for all $i < k$. Let M be the set of all polynomials $m(x) \in \mathbb{Z}_p[x]$ with coefficients $|m_i| \leq \beta/2$. Notice that for any $m(x) \in M$ we have

$$|c \cdot m(a) \pmod{p}| = |\sum_i c \cdot m_i \cdot a^i \pmod{p}| \leq \sum_i |m_i \pmod{p}| \cdot |c \cdot a^i \pmod{p}| \leq \frac{\beta}{2} k\gamma\mu.$$

So, $cm(a) \pmod p$ ranges over a set $\{-\beta k\gamma\mu/2, \ldots, \beta k\gamma\mu/2\}$ of size at most $\beta k\gamma\mu + 1$. Since M has size $(\beta+1)^k > \beta^k + 1 = (k\gamma\mu)^{k/(k-1)} + 1 = \beta k\gamma\mu + 1$, by the pigeonhole principle, there must be two distinct polynomials $m(x), \hat{m}(x) \in M$ such that $c \cdot m(a) = c \cdot \hat{m}(a) \pmod p$. Dividing by c and subtracting gives a nonzero polynomial $q(x) = (m - \hat{m})(x)$ with coefficients $|q_i \pmod p| \leq \beta$, such that $(m - \hat{m})(a) = 0 \pmod p$. □

The number of nonzero polynomials of degree $< k$ with coefficients bounded by β is $(2\beta+1)^k - 1$, and each one of them has at most $k-1$ roots. Therefore, the set A has size at most

$$|A| \leq (k-1)(2\beta+1)^k < k(3\beta)^k.$$

So, the probability that Λ_a has a nonzero vector of length bounded by $\gamma\mu$ is at most

$$\Pr_a\{\lambda(\Lambda_a) \leq \gamma\mu\} \leq \frac{|A|}{p} \leq \frac{k(3\beta)^k}{p} = \frac{k3^k(k\gamma\mu)^{k/(k-1)}}{p} = \frac{k3^k(k2^{k-1}\sqrt{k}(p/2^u))^{k/(k-1)}}{p}$$

$$= \frac{k3^k(k^{\frac{3}{2}}(2^{k-1})(2^{-u})p)^{k/(k-1)}}{p} = \frac{k3^k k^{\frac{3}{2}\frac{k}{k-1}} 2^{(k-1)(\frac{k}{k-1})} p^{\frac{k}{(k-1)}}}{p2^{u\frac{k}{k-1}}} = \frac{6^k k^{(\frac{3}{2}\frac{k}{k-1})+1} p^{\frac{1}{(k-1)}}}{2^{\delta n\frac{k}{k-1}}}$$

This expression, though complicated, informs the attacker of how to choose k. Setting $k = 1$ results in an expression with no sensible interpretation, so $k \geq 2$, in which case we can simplify the inequality:

$$\Pr_a\{\lambda(\Lambda_a) \leq \gamma\mu\} \leq \frac{6^k k^{(\frac{3}{2}\frac{k}{k-1})+1} p^{\frac{1}{(k-1)}}}{2^{\delta n\frac{k}{k-1}}} \leq \frac{6^k k^4 p^{\frac{1}{(k-1)}}}{2^{\delta n\frac{k}{k-1}}}$$

Note that so the quantity $p^{1/(k-1)}$ has less than $\frac{n}{k-1} + 1$ bits. Note also that as k increases, the denominator approaches $2^{\delta n}$, which has $\delta n + 1$ bits. Thus, the attacker should pick k so that $\frac{n}{k-1} + 1 < \delta n + 1$, or $k - 1 > \frac{1}{\delta}$, to ensure that the numerator has fewer bits than the denominator. In fact, k needs to be a bit larger than this to offset the additional contributions of 6^k and k^4, and then a bit more to ensure the probability is low. Setting $k = 2/\delta$ is sufficient to result in a failure probability that is exponentially small in the security parameter n, in which case the attack is almost certain to succeed.

3 Cryptanalysis of the Subset-Sum generator

Let p be a prime of length $\log_2 p = n^2$, and $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ be chosen uniformly at random. These values are public and known to the attacker. The Subset-Sum generator, on input a uniformly random string $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$, outputs $b = G(\mathbf{x}) = \sum_i a_i x_i \pmod p \in \mathbb{Z}_p$.

Given the public parameters $p, \mathbf{a}$ and output b we set up the following lattice and target

vector:

$$\mathbf{B} = \begin{bmatrix} Cp & Ca_1 & \dots & Ca_n \\ & 2 & & \\ & & \ddots & \\ & & & 2 \end{bmatrix} \qquad \mathbf{t} = \begin{bmatrix} Cb \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

where C is a large constant to be determined.

Clearly, there is a lattice vector $\mathbf{B}(c, \mathbf{x}) = (Cb, 2\mathbf{x})$ within distance $\sqrt{n}$ from the target. We will prove that the minimum distance of the lattice $\Lambda_{\mathbf{a}}$ generated by $\mathbf{B}$ is much larger than that (say, by an exponential factor $\gamma = 2^{n/2}$,) so, this is an instance of the BDD problem, and it can be easily solved using lattice reduction algorithms.

We set $C = \gamma\sqrt{n}$, so that all lattice vectors of length bounded by $C = \gamma\sqrt{n}$ must have the form $(0, \mathbf{z})$ for some $\|\mathbf{z}\| \leq C$. We consider the set Z of all nonzero integer vectors $\mathbf{z}$ of length $\|\mathbf{z}\| \leq C$. For each $\mathbf{z} \in Z$, $(0, \mathbf{z})$ is a lattice vector if and only if $\sum_i a_i z_i = 0 \pmod{p}$. But, when the a_i are chosen uniformly at random and $\mathbf{z} \neq 0$, the value $\sum_i a_i z_i$ is uniformly distributed in $\mathbb{Z}_p$. So, $(0, \mathbf{z})$ is a lattice vector with probability $1/p$. By union bound, the probability (over the choice of $\mathbf{a}$) that there is a short vector is

$$\Pr_{\mathbf{a}}\{\lambda(\Lambda_{\mathbf{a}}) \leq C\} \leq \frac{|Z|}{p} \leq \frac{(2C+1)^n}{p} \approx \frac{1}{2^{n/2}}.$$

So, the failure probability is exponentially small.

4 Conclusion

We have seen how the cryptanalysis of different types of pseudorandom generators reduces to the bounded distance decoding (BDD) problem, a variant of the closest vector problem (CVP) on point lattices. In the case of the truncated LCG generators, we have seen that even stretching the seed by a small constant factor allows to recover the seed by solving a lattice problem in constant dimension $k = O(1/\delta)$. In the case of subset-sum, we used an n-dimensional lattice, where n is the security parameter of the pseudorandom generator. The attack works when the generator outputs more than n^2 bits. Performing a similar attack on subset-sum generators with smaller output (say, $O(n)$ bits), is closely related to finding better solutions to lattice problems. In fact, we will see that subset-sum can be quite a fine problem to build cryptographic functions if used properly.

Both attacks illustrate a common feature of lattice algorithms when applied in cryptography: both for the linear congruential generator, and the subset-sum generator, the attack we described is not guaranteed to work for any input. Rather, it works with high probability when run on a randomly selected problem instance. This makes perfect sense in cryptography, where keys are chosen at random, and define a probability distribution on problem instances. Knowing that a problem is easy or hard in the average case, tells you if your randomly chosen key can or cannot be cracked. Notice how both cryptanalysis problems defined a certain (natural) distribution on lattices, and we proved that the relevant lattice

problem could be solved with high probability when the lattice is chosen according to that distribution. Average case complexity is relevant not only in cryptanalysis, but also in most practical setting, when the input distribution is known. For this reason, we will be primarily interested in the average case complexity of lattice problems.