Breaking Littlewood's cipher

Damien Stehlé

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Breaking Littlewood's cipher

Damien Stehlé

N° 4988

Novembre 2003

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4988--FR+ENG

INRIA
LORRAINE

Breaking Littlewood's cipher

Damien Stehlé

Thème 2 — Génie logiciel
et calcul symbolique
Projet Spaces

Rapport de recherche n° 4988 — Novembre 2003 — 17 pages

Abstract: In 1953, the celebrated mathematician John Edensor Littlewood proposed a stream cipher based on logarithm tables. Fifty years later, we propose the first analysis of his scheme. Littlewood suggests the idea of using real functions as tools to build cryptographic primitives. Even when considering modern security parameters, the original scheme can be broken by a simple attack based on differentiation. We generalise the scheme such that it resists this attack, but describe another attack which is derived from both polynomial approximation and Coppersmith's technique to find the small roots of modular multivariate polynomials. In contrast with these negative results we describe a candidate for a very efficient one-way function and present an open problem based on this work.

Key-words: Littlewood's cipher, real functions, lattices, Coppersmith's technique.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Cryptanalyse du chiffre de Littlewood

Résumé : En 1953, le célèbre mathématicien John Edensor Littlewood a proposé un schéma de chiffrement reposant sur les tables de logarithmes. 50 ans plus tard, nous décrivons la première cryptanalyse de ce protocole. Littlewood suggère l'idée d'utiliser les fonctions de la variable réelle comme outils de base pour construire des primitives cryptographiques. Même si l'on considère des paramètres de sécurité actualisés, le schéma original peut être cassé par une simple attaque utilisant des dérivées successives. Nous généralisons le protocole de telle sorte qu'il résiste à cette attaque simple, mais une autre attaque peut être réalisée, en utilisant d'une part des approximations polynomiales et d'autre part la méthode de Coppersmith pour trouver les petites racines de polynômes modulaires multivariés. En opposition à ces cryptanalyses, nous décrivons un nouveau candidat de fonction à sens uniqe très efficace et présentons un problème ouvert lié à ce travail.

Mots-clés : Schéma de chiffrement de Littlewood, fonctions de la variable réelle, réseaux Euclidiens, méthode de Coppersmith.

1 Introduction

Finding mathematical problems on which security primitives can be built is one of the major goals of cryptographic research. A huge number of fairly complicated structures have been studied in this light, among them finite fields; elliptic curves; lattices; linear codes; polynomial rings and braid groups. Surprisingly, the real numbers and the real functions like sin, log, exp, have found no such use. Demonstrating cryptographically interesting properties of these functions could be a major breakthrough since they are already widely used for other applications. Real functions are already implemented with very fast algorithms and are available in almost every calculator, some hardware devices and plenty of software packages. Hence, developing and distributing such a cryptographic primitive would not be a difficult task. The reader interested in the implementation of mathematical functions can find the state of the art in [18].

In 1953, the widely recognised number theoretist John Edensor Littlewood proposed a cipher based on the log function. Considering the clarity and the historical interest of his description, we give here his own words originally found in [16], chapter "Mathematics with minimum raw material":

> *The legend that every cipher is breakable is of course absurd, though still widespread among people who should know better. I give a sufficient example, without troubling about its precise degree of practicability. Suppose we have a 5-figure number N. Starting at a place N in a 7-figure log-table take a succession of pairs of digits $d_1 d'_1$, $d_2 d'_2$, ... from the last figures of the entries. Take the remainder of the 2-figure number $d_n d'_n$ after division by 26. This gives a 'shift' s_n, and the code is to shift the successive letters of the message by s_1, s_2, ... respectively.*
>
> *It is sufficiently obvious that a single message cannot be unscrambled, and this even if all were known except the key number N (indeed the triply random character of s_n is needlessly elaborate). If the same code is used for a number of messages it could be broken, but all we need do is to vary N. It can be made to be dependent on a date, given it in clear; the key might e.g. be that N is the first 5 figures of the 'tangent' of the date (read as degrees, minutes, seconds: 28^o 12' 52" for Dec. 28, 1952). This rule could be carried in the head, with nothing on paper to be stolen or betrayed. If any one thinks there is a possibility of the entire scheme being guessed he could modify 26 to 21 and use a date one week earlier than the one given in clear.*

Wilson describes very precisely in [22] how to mount a cryptanalysis of the described scheme. Indeed with these parameters it can be performed even by hand. Obviously, they are not up-to-date if one considers the actual power of computation, but the scheme can be easily generalised to larger parameters. To be precise, Littlewood defines a Pseudo-Random Number Generator (PRNG), and uses it as a one-time pad to obtain a stream cipher.

Surprisingly, this scheme has attracted very little attention in the past fifty years, and the first known analysis miss this paper. We show how to break the PRNG by using a simple attack based on differentiation. Nevertheless, it can be easily modified to avoid this obvious attack, by introducing some randomness. This slightly modified PRNG seems to be more resistant, and indeed the only known attack is based on modern cryptanalytic techniques using the geometry of numbers, a field of mathematics created by Minkowski and which has seen significant algorithmic improvements in the last two decades. More precisely, we use the technique described by Coppersmith in [10], which helped, amongst others, to break certain very efficient variants of RSA (see [6] and [3] for the most powerful attacks).

The cryptanalyses described in this paper show that real functions are very poor objects from a cryptographic point of view. Nevertheless, the overall study naturally leads to the proposal of a candidate one-way function. Up to now, no subexponential algorithm to invert this function is known. Briefly, we consider a function $f : \mathbb{R} \to \mathbb{R}$, for example sin, which is public. The one-way function takes as input an n-bit integer x and outputs the n-bit integer $\lfloor 2^{2n} f(x/2^n) \rfloor \bmod 2^n$, that is to say the decimal bits $n+1$ to $2n$ of $f(x/2^n)$.

The remainder of the paper is organised as follows. In Section 2, we formalise the original Littlewood PRNG and give a simple attack that allows us to predict the sequence of numbers after having seen very few outputs. In Section 3, we generalise the scheme to obtain a stretching function quite similar to the one described in [7]. A heuristic attack based on polynomial approximation and lattice reduction is described in Section 4. Finally, in Section 5, we describe the candidate one-way function.

2 The original PRNG

Recall first which properties have to be satisfied by a PRNG for it to be said cryptographically secure. Intuitively, it has to be easily computable and unpredictable even if an adversary has already seen some of the output. Moreover, an adversary should not be able to distinguish the numbers he sees from random numbers. We give here the definition of a PRNG as presented in [11].

Definition 1 (PRNG). A deterministic polynomial-time algorithm G is called a pseudorandom generator if there exists an increasing function, $l : \mathbb{N} \to \mathbb{N}$, so that for any probabilistic polynomial-time algorithm D, for any positive polynomial p, and for all sufficiently large k's

$$|\Pr[D(G(U_k)) = 1] - \Pr[D(U_{l(k)}) = 1]| < \frac{1}{p(k)},$$

where U_n denotes the uniform distribution over $\{0,1\}^n$ and the probability is taken over U_k (resp., $U_{l(k)}$) as well as over the coin tosses of D.

We now define precisely the generator on which we are working.

Definition 2 (Littlewood's PRNG). Let n, n', m and t be parameters that can be adjusted to reach a certain security level.

1. Choose a random n-bit secret number N.

2. For i from 0 to t, output the decimal bits ranked $n'+1$ to $n'+m$ of $\log(N+i)$, that is to say: $\lfloor 2^{n'+m} \log(N+i) \rfloor \bmod 2^m$.

Intuitively, if we have a logarithm table with entries in rows, we return the block of bits between rows N and $N+t$ and between columns $n'+1$ and $n'+m$. The major weakness of this scheme is that the logarithms of two large consecutive numbers are very close and by using this property, given a small number of known outputs, the adversary can predict any subsequent outputs. Our attack is based on the following simple lemma:

Lemma 3. *Let Δ be the operator on sequences defined by $\Delta(u)_k = u_{k+1} - u_k$, and let $\Delta^{(i)}$ be this operator composed i times with itself. Let N be a n-bit integer and $a_k = \log(N+k)$, $k \geq 0$. Then for all $i \geq 1$ and $k \geq 0$,*

$$|\Delta^{(i)}(a_k)| \leq \frac{(i-1)!}{2^{i(n-1)}}.$$

Proof. We first extend Δ to the functions: $[\Delta(f)](x) = f(x+1) - f(x)$. Let $f(x) = \log(N+x)$, so $a_k = f(k)$. We apply the maximum-value theorem several times:

$$\begin{aligned}
|\Delta^{(i)}[f(k)]| &= |\Delta^{(i-1)}[f(k+1)] - \Delta^{(i-1)}[f(k)]| \\
&= |[\Delta^{(i-1)}(f)](k+1) - [\Delta^{(i-1)}(f)](k)| \\
&\leq \max(|[\Delta^{(i-1)}(f)]'(x)| \ / \ x \in [k, k+1]) \\
&\leq \max(|[\Delta^{(i-1)}(f')](x)| \ / \ x \in [k, k+1])
\end{aligned}$$

$$\begin{aligned}
&\leq \max(|[\Delta^{(i-2)}(f'')](x)| \ / \ x \in [k, k+2]) \\
&\vdots \\
&\leq \max(|f^{(i)}(x)| \ / \ x \in [k, k+i]) \\
&\leq \tfrac{(i-1)!}{N^i}.
\end{aligned}$$

The result comes from the fact that $N \geq 2^{n-1}$. □

It follows from the previous result that the $(i+2)$-th output of the generator, that is to say $u_{i+1} = \lfloor 2^{n'+m} \log(N+i+1) \rfloor \mod 2^m$, can be estimated from the knowledge of the $i+1$ previous outputs by the formula:

$$u_{i+1} = (u_{i+1} - \Delta^{(i)}(u_0)) + \Delta^{(i)}(u_0).$$

The quantity $u_{i+1} - \Delta^{(i)}(u_0)$ depends only on u_0,...,u_i, and $\Delta^{(i)}(u_0)$ can be bounded by using the previous lemma. Thus, to recover at least the first bits of u_{i+1} from the previous outputs, it suffices that $|\Delta^{(i)}(u_0)| \leq 2^{n'-1}$, which is implied by the inequality $|2^{n'+m}\Delta^{(i)}(\log N)| \leq 2^{n'-1}$. It follows from Lemma 1 that the generator of Definition 2 is insecure as long as there exists an i such that:

$$i \geq \frac{m + 1 + \log((i-1)!)}{n-1}.$$

If m is significantly larger than n such an i may not exist but this attack works for any realistic parameters. For example, if $n = 100$, the attack can be mounted as long as $m \leq 10^{30}$ and if $n = 1000$, the bound on m becomes 10^{300}.

Remark: The same reasoning holds for a very large class of functions f instead of only log. It suffices that when N is large enough, there exist two constants $a > 0$ and b such that the successive derivatives of f satisfy:

$$f^{(i)}(x) = O(\frac{1}{N^{ai+b}}), \forall x \geq N, \forall i \geq i_0.$$

In particular, replacing log by a polynomial, a rational function or an α-th root does not save the scheme.

3 The modified scheme

Before presenting the modified PRNG, we introduce some notations and we recall some useful results about the computation of real functions.

3.1 Notations and definitions

In what follows, we denote by $[|a, b|]$ the set of the integers in the interval $[a, b]$ and by $[a, b]_n$ the set of real numbers between a and b that have at most n decimal bits in the standard binary representation, that is to say $\{\frac{\lceil 2^n a \rceil}{2^n}, \frac{\lceil 2^n a \rceil + 1}{2^n}, ..., \frac{\lfloor 2^n b \rfloor}{2^n}\}$. We also denote by $\text{bits}_{k_1..k_2}[x]$ the decimal bits from k_1 to k_2 of x, or more precisely the integer: $(\lfloor 2^{k_2} x \rfloor \bmod 2^{k_2 - k_1 + 1})$.

We also need a precise definition of what an easily computable real function is:

Definition 4 (Polynomial-time computable real function). Let $a, b \in \mathbb{R}$. A function $f : [a, b] \to \mathbb{R}$ is said polynomial-time computable (PTC) if there exist a deterministic algorithm A and a positive bivariate polynomial $P(x, y)$ such that given any $x \in [a, b]_n$, A computes the m most significant bits of $f(x)$ with at most $P(n, m)$ binary operations.

Of course, the elementary functions like ln; exp; sin; cos; tan; $\sin^{-1}$; $\cos^{-1}$; $\tan^{-1}$ are PTC on a given interval. Brent proved in [8] that computing cos, sin, arctan, ln and exp can be done in essentially the same time as a multiplication of two integers. The complexity of those functions is therefore almost quadratic, and even quasi-linear if one uses fast arithmetic [20].

Theorem 5 (Brent, 1976). *If π and $\ln 2$ are precomputed, the elementary function $f(x)$ can be evaluated to precision n in $(K + o(1))M(n)\log_2 n$ operations, where*

$$K = \begin{vmatrix} 13 & \text{if } f(x) = \ln(x) \text{ or } \exp(x) \\ 34 & \text{if } f(x) = \arctan(x) \text{ or } \sin(x) \text{ or } \cos(x) \end{vmatrix}$$

and $M(n)$ is the number of binary operations required to multiply two n-bit numbers.

3.2 A randomised Littlewood PRNG

The main problem of Littlewood's PRNG is that consecutive lines of the log-table are taken and hence the values of consecutive entries are too correlated when N grows to infinity. One way to solve this problem is to use random lines. Because it is more convenient, we do not use the domain $[|2^{n-1}, 2^n - 1|]$ anymore, but $[0, 1]_n$ instead. The object we introduce below is a PRNG very similar to the one described in [7]. Amazingly, the attacks on this new PRNG and on the one just mentioned are quite similar too.

Definition 6. (Hidden Real Number Problem HRNP) Let f be a PTC $[0,1] \to [0,1]$ function, e.g. sin. Let n, n', m and t be security parameters. Let $\alpha \in [|0, \frac{1}{2}|]_n$ be a secret number. The small real number problem consists of recovering α from the knowledge of:

$$[\ x_i, \mathrm{bits}_{(n'+1)..(n'+m)}[f(\alpha + x_i)]\ ],$$

for $0 \leq i \leq t$, and where the x_i's are chosen independently and uniformly in $[|0, \frac{1}{2}|]_n$.

Definition 7. (Generalised Littlewood's PRNG) With the same notations, the generator is the following stretching function:

$$(x_0, ..., x_t) \in [|0, \frac{1}{2}|]_n^{t+1} \quad \to \quad < x_1, ..., x_t, b_1, ..., b_t >,$$

where $b_i = \mathrm{bits}_{(n'+1)..(n'+m)}[f(x_0 + x_i)]$, for $i = 1, \ldots, t$.

The security of the generalised PRNG is directly linked to the hardness of the HRNP. As usual, it is possible to re-key the generator from the previous output, by using the well-known technique described in [4]. Notice that if the x_i's are not chosen independently, especially if $x_{i+1} = x_i + 2^{-n}$, then the attack of the previous section still works.

There are two obvious attacks on this PRNG, which give two relations between the parameters in the case of a 2^s security level. The first one is based on the birthday paradox and gives $2m \geq s$. The second one consists of guessing the first n' decimal bits of $y = f(\alpha + x_i)$ and then computing $f^{-1}(y)$, when f^{-1} is defined and PTC. This attack has complexity $2^{n'}$ and thus gives the bound $n' \geq s$.

4 A polynomial approximation based attack

This attack breaks the scheme in Definition 7 but remains heuristic since it uses the lattice based technique of Coppersmith to find the small roots of multivariate modular polynomials. It consists of an algorithm that solves the HRNP efficiently. The first step is to approximate f by a polynomial with real coefficients on its domain of definition $[0,1]$. Then, we reduce the HRNP to the problem of finding the common integer roots of some multivariate polynomials over $\mathbb{Z}$. We finally use the multivariate Coppersmith technique to solve this last problem and recover the secret α. This attack is closely related to a new method [21] to solve the problem of finding the worst cases for correctly rounding the mathematical functions, also known as the table maker dilemma (TMD). The lattice part of the attack is very similar to the

technique used in [7].

In this section, we prove the following theorem:

Theorem 8. *Let $f : [0,1] \to [0,1]$ be any $\mathcal{C}^\infty$ PTC function such that, when d grows to infinity, $max_{x\in[0,1]}|f^{(d)}(x)| = O(d!2^d)$. Then there exists an algorithm working in time polynomial in n, n' and m, so that if t is larger than a bound growing at most polynomially with the parameters, then under a heuristic that will be explained below, it solves the HRNP.*

4.1 Approximating f by a polynomial

The approximation of continuous functions by polynomials is quite an old mathematical topic. A very good introduction to the implementation facet of this subject can be found in [18]. Here we consider approximations with respect to the L_∞ norm:

$$||f - P||_\infty = \sup(|f(x) - P(x)| \ / \ x \in [0,1]).$$

This is a well-known fact that theoretically there exists a polynomial that approaches any given continuous function as closely as desired:

Theorem 9 (Weierstrass,1885). *Let f be a continuous function. For any $\epsilon > 0$, there exists a polynomial P such that $||f - P||_\infty \leq \epsilon$.*

But this theorem does not help finding those polynomials and gives no indication of the convergence rate as a function of the degree of the polynomial. A very efficient method to compute a polynomial approximation of a $\mathcal{C}^\infty$ function is to use Chebyshev polynomials. The degree d Chebyshev approximate of f can be computed in polynomial time with numerical integration algorithms. More precisely, one can compute, in polynomial time in d and k, approximations of the coefficients of the degree d Chebyshev approximate of the function, with any number k of decimal bits. We will see in the next subsection that getting approximations of the coefficients suffices. Moreover, when d increases, the Chebishev approximates of f converge very quickly. For further details about Chebyshev approximation, we refer to [2], where one can find the following theorem:

Theorem 10 (Chebyshev approximation). *Let f be a $\mathcal{C}^\infty$ and $[0,1] \to [0,1]$ function, and P_d the degree d Chebyshev approximate of f. Then:*

$$||f - P_d||_\infty \leq \frac{||f^{(d+1)}||_\infty}{(d+1)!2^{2d+1}}.$$

In our case, we consider the smallest degree d such that $||f - P_d||_\infty \leq \frac{1}{2^{n'+m}}$. From what precedes, d is polynomially bounded in the variables n' and m, and we can compute approximations of the coefficients of P_d with a polynomially bounded number of decimal bits in time polynomial in all the parameters.

4.2 A problem over the integers

Suppose now that an adversary knows the quantities $[x_i, b_i]$ for i from 1 to t, with

$$b_i = \text{bits}_{(n'+1)..(n'+m)}[f(\alpha + x_i)] \ \in \ [|0, 2^m - 1|].$$

The next lemma shows how to transform each relation into a multivariate polynomial equation satisfied by integer unknowns:

Lemma 11. *Suppose that* $\text{bits}_{(n'+1)..(n'+m)}[f(x_0)] = b_i$ *with* $x_0 \in [|0,1|]_n$. *Let* P *be a degree* d *polynomial approximation of* f *on* $[0,1]$ *which satisfies* $||f - P||_\infty \leq \frac{1}{2^{n'+m}}$. *Let* $P'(x) = \lfloor 2^{n'+m} P \rceil(x)$. *This notation means that we round each coefficient of the polynomial* $2^{n'+m}P$ *to the nearest integer. Let* $Q(t) = 2^{dn} P'(\frac{t}{2^n})$. *Then* Q *is a degree* d *polynomial with integer coefficients, and:*

$$Q(2^n x_0) - 2^{dn} b_i = 2^{dn+m} y + z,$$

where y *and* z *are integer unknowns that satisfy* $0 \leq y < 2^{n'}$ *and* $|z| \leq (d+5)2^{dn-1}$.

Proof. By the definition of P', for all x in $[0,1]$, $|P'(x) - 2^{n'+m}P(x)| \leq \frac{d+1}{2}$. Thus, by the triangle inequality, there exists an integer $0 \leq y < 2^{n'}$ such that $|P'(x_0) - 2^m y - b_i| \leq \frac{d+1}{2} + 2$. Consequently, $|Q(2^n x_0) - 2^{dn+m} y - 2^{dn} b_i| \leq (d+5)2^{dn-1}$. □

We apply Lemma 11 to the polynomial P_d found during the polynomial approximation phase of the algorithm. Let Q be the degree d polynomial over $\mathbb{Z}$ obtained by applying the previous lemma to P_d. For each couple $[x_i, b_i]$ we have an equation:

$$Q(2^n(\alpha + x_i)) - 2^{dn} b_i - 2^{dn+m} y_i + z_i = 0 \quad (E_i),$$

where α, y_i and z_i are the unknowns with $0 \leq y_i < 2^{n'}$ and $|z_i| \leq (d+5)2^{dn-1}$. We transform the equations (E_i) over $\mathbb{Z}$ into modular equations:

$$Q_i(a) + z_i = 0 \bmod A \quad (E_i'),$$

where $A = 2^{dn+m}$, $a = 2^n \alpha \in [|0, 2^{n-1}|]$, $Q_i(a) = Q(a + 2^n x_i) - 2^{dn} b_i$ and $|z_i| \leq (d+5)2^{dn-1}$. The problem is now to find a, given that $Q_i(a)$ is small modulo A,

with known polynomials Q_i. This is a very typical application of lattice reduction algorithms.

Remark: As claimed in the previous subsection, only a polynomial number of bits of the coefficients of P_d have to be computed to finally get the right polynomial Q. Because of the definition of P' in the previous lemma, it suffices that $n' + m + 1$ decimal bits of each coefficient of P_d have been calculated.

4.3 Some lattice background

We recall here some basic facts about lattices. A comprehensive introduction to the computational aspect of that subject can be found in [17].

A *lattice* is a discrete subgroup of $\mathbb{R}^n$, or, equivalently, the set of all integral linear combinations of $l \leq n$ linearly independent vectors $\mathbf{b}_i \in \mathbb{R}^n$:

$$L = \{\sum_{i=1}^{l} x_i \mathbf{b}_i \ / \ x_i \in \mathbb{Z}\}.$$

The *determinant*, also called *volume*, of the lattice L is $\det(L) = \prod_{i=1}^{l} ||\mathbf{b}_i^*||$, where $||.||$ is the Euclidean norm and $[\mathbf{b}_1^*, ..., \mathbf{b}_l^*]$ is the Gram-Schmidt orthogonalisation of $[\mathbf{b}_1, ..., \mathbf{b}_l]$. The determinant of a lattice is particularly easy to compute when $l = n$ and the matrix expression of the $\mathbf{b}_i$'s is triangular: in that case the determinant is the absolute value of the product of the diagonal elements. A given lattice L can be spanned by several different *bases* $[\mathbf{b}_1, ..., \mathbf{b}_l]$, but generally, only those consisting of short and rather orthogonal vectors are of practical interest. For $i \leq l$, the i-th *minimum* of L, denoted by $\lambda_i(L)$, is defined as the smallest r such that there are at least i linearly independent lattice vectors in the hypersphere of radius r centred in 0. A very good basis would ideally reach the first minima, but as soon as $l \geq 5$, there are lattices such that no basis satisfies this property. Nevertheless the well-known LLL algorithm [15] computes a basis for which the vectors "almost reach" the minima. This means that the length of the i-th shortest vector returned by LLL is below $c\lambda_i(L)$, for some constant c (when the dimension is fixed). More precisely, we have the following theorem:

Theorem 12. *For any constants l and n there exists $c_1 > 1$ such that given a basis $[\mathbf{b}_1, ..., \mathbf{b}_l]$ of a lattice $L \subset \mathbb{Z}^n$ as input, the LLL algorithm outputs in time polynomial in the bit length of the input, a basis $[\mathbf{v}_1, ..., \mathbf{v}_l]$ satisfying:*

$$||\mathbf{v}_i|| \leq c_1 \lambda_i(L), \quad \forall i \in [|1, l|].$$

Moreover, since the Minkowski theorem guarantees that $\Pi_{i=1}^{l}\lambda_i(L) \leq c_2 det(L)$ *for a certain constant* c_2*, there exists a constant* c_3 *such that the returned basis satisfies:*

$$||\mathbf{v}_i|| \leq c_3 det(L)^{\frac{1}{l-i+1}}, \quad \forall i \in [|1,l|].$$

In 1996, Coppersmith (see [9] or [13] for the easiest descriptions) found a novel consequence of this theorem: heuristically, the small roots of a multivariate polynomial modulo some integer A can be computed in time polynomial in $\log A$. Since its publication, this very general method proved useful in many areas, among them integer factorisation [10], RSA cryptanalysis [6] and noisy CRT list decoding [5].

4.4 Recovering a by Coppersmith's technique

The last step to recover the secret α, or equivalently a, is the use of Coppersmith's technique on the equations (E_i') we found by polynomial approximation: we reduce the problem of finding a to a short lattice vectors problem, which we solve with the LLL algorithm. More precisely, we construct a lattice for which short linearly independent lattice vectors give multivariate polynomial equations over the integers satisfied by variables linked to a. Under the heuristic that these polynomial equations are algebraically independent, computing their resultant towards all the unknowns except one leads to a univariate polynomial over $\mathbb{Z}$. The roots of this polynomial are computed, and they directly give a.

Suppose that we have t equations $(E_1'), ..., (E_t')$. Let $q_i(x, y_1, ..., y_t) = Q_i(x) + y_i$. We know that modulo A,

$$q_1(a, z_1, ..., z_t) = Q_1(a) + z_1 = 0 \quad , ..., \quad q_t(a, z_1, ..., z_t) = Q_t(a) + z_t = 0.$$

We consider the following family of polynomials:

$$\{A^2 x^i \ / \ 0 \leq i \leq 2d\} \cup \{Ax^i q_j \ / \ 0 \leq i \leq d, 1 \leq j \leq t\} \cup \{q_j q_k \ / \ 1 \leq j, k \leq t\}.$$

The $(t+1)$-uple $(a, z_1, ..., z_t)$ is a root of each of them modulo A^2, whence of each integral linear combination of them. The task of the lattice reduction step is to build $t+1$ such linear integral combinations which have small values for small values of the variables $x, y_1, ..., y_t$.

We now suppose that a and the z_i's are bounded: $|a| \leq X$ and $|z_i| \leq Y$ for all $i \in [|1,t|]$. We build the lattice L naturally associated with the polynomials modulo A^2. Its columns correspond to the terms and its rows to the polynomials we selected. The coefficient of the matrix for a term $T = x^i y_1^{i_1} ... y_t^{i_t}$ and a given polynomial Q is

the coefficient of Q for T, multiplied by the weight of T, *ie* $X^i Y^{\sum_{j=1}^{t} i_t}$. For example, if $d = 2$ and $t = 2$ we get a lattice of the form[1]:

	1	x	x^2	x^3	x^4	y_1	y_2	xy_1	xy_2	x^2y_1	x^2y_2	${y_1}^2$	${y_2}^2$	y_1y_2
(A^2)	A^2													
(A^2X)		A^2X												
(A^2X^2)			A^2X^2											
(A^2X^3)				A^2X^3										
(A^2X^4)					A^2X^4									
(AE'_1)	$-$	$-$	$-$			AY								
(AE'_2)	$-$	$-$	$-$				AY							
(AXE'_1)		$-$	$-$	$-$				AXY						
(AXE'_2)		$-$	$-$	$-$					AXY					
$(AX^2E'_1)$			$-$	$-$	$-$					AX^2Y				
$(AX^2E'_2)$			$-$	$-$	$-$						AX^2Y			
$({E'_1}^2)$	$-$	$-$	$-$	$-$	$-$	$-$		$-$		$-$		Y^2		
$({E'_2}^2)$	$-$	$-$	$-$	$-$	$-$		$-$		$-$		$-$		Y^2	
$(E'_1E'_2)$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$			Y^2

The dimension of the lattice is the number of considered polynomials, *i.e.* $\frac{(t+2)(2d+t+1)}{2}$, and its determinant is easy to compute since we have a triangular matrix:

$$X^{\frac{d}{2}(dt+4d+t+2)} \cdot Y^{t(d+t+2)} \cdot A^{2+t+4d+td}.$$

Because there are $t+1$ variables $(x, y_1, ..., y_t)$, in order to eliminate enough of them by the resultant computation, we need $t+1$ vectors of length less than A^2. By Theorem 12, we can compute such short linearly independent lattice vectors as long as:

$$X^{\frac{d}{2}(dt+4d+t+2)} \cdot Y^{t(d+t+2)} \cdot A^{2+t+4d+td} \leq A^{2td+t^2+t+4d+2} \cdot c_3^{-\frac{1}{4}(t+2)(2d+t+1)(2td+t^2+t+4d+2)},$$

which gives when t grows to infinity:

$$X^{\frac{d(d+1)}{2t}+o(1)} Y^{1+o(1)} \leq A.$$

In the attack, we can make t grow polynomially in n, n' and m, and choose $X = 2^n$. Therefore we obtain $t+1$ polynomials $R_1, ..., R_{t+1}$ with $(t+1)$ variables $x, y_1, ..., y_t$. $(a, z_1, ..., z_t)$ is a root for each of them modulo A^2. Since $|a|, |z_1|, ..., |z_t|$ are small, $(a, z_1, ..., z_t)$ is a root for each of them over the integers. We now compute the resultant $R(x) = Res_{y_1,...,y_t}(R_1, ...R_{t+1})$, which can be done in polynomial time since this is essentially a determinant computation. The polynomial $R(x)$ is then factored over $\mathbb{Z}$ (see for example [12]) and as long as $R(x)$ is not zero, a is recovered since it is a root of $R(x)$ over $\mathbb{Z}$.

[1] entries denoted by the symbol $-$ are those that can be non-zero

Consequently, the statement claimed in Theorem 8 holds under the following heuristic:

Coppersmith's heuristic: The $t+1$ linearly independent short vectors obtained by reducing the initial basis of the lattice L have a non-zero resultant towards the variables y_i.

Similar assumptions have been made very often in cryptography (see [6] and [9] among others) and seem reasonable in practice, although are known to fail sometimes [3].

4.5 A faster and improved attack

The lattice step of the attack can be improved by using the fact that in general, the short linearly independent lattice vectors have similar lengths. More precisely, for a "random"[2] d-dimensional lattice L, the d first minima have all lengths around $\det(L)^{1/d}$, to within some multiplicative constant. This explains why lattices and lattice reduction algorithms behave in some cases far better in practice than in theory. In particular, Theorem 12 gives a very bad bound for "random" lattices. A "probabilistic" version of that result would be:

Given a random basis $[\mathbf{b}_1, ..., \mathbf{b}_l]$ *of a lattice* $L \subset \mathbb{Z}$*, the LLL algorithm outputs with high probability (towards the randomness of the lattice) in time polynomial in the bit length of the input, a basis* $[\mathbf{v}_1, ..., \mathbf{v}_l]$ *satisfying, for some constant* c*:*

$$||\mathbf{v}_i|| \leq c \cdot det(L)^{\frac{1}{l}}, \quad \forall i \in [|1, l|].$$

With this result, considering the polynomials q_i's themselves modulo A suffices (instead of considering products of them modulo A^2). For example, with $d = 3$ and $t = 4$, one gets a lattice of the form:

[2] "Random" is in brackets since it is somehow hard to define a natural randomness on lattices (see [1] for more details), and it is not clear how "random" our lattices are.

$$\begin{array}{c|cccccccc} & 1 & x & x^2 & x^3 & y_1 & y_2 & y_3 & y_4 \\ (A) & A & & & & & & & \\ (AX) & & AX & & & & & & \\ (AX^2) & & & AX^2 & & & & & \\ (AX^3) & & & & AX^3 & & & & \\ (E'_1) & - & - & - & - & Y & & & \\ (E'_2) & - & - & - & - & & Y & & \\ (E'_3) & - & - & - & - & & & Y & \\ (E'_4) & - & - & - & - & & & & Y \end{array}$$

From the definition of the q_i's, we have that the entries of the column x^d are identical for the rows (E'_i). Consequently it is interesting to consider the sublattice spanned by the lines (AX^i) for $0 \leq i \leq d-1$ and $(E'_j) - (E'_1)$ for $2 \leq j \leq t$. This provides a lattice of dimension $d+t-1$ and determinant:

$$X^{\frac{d(d-1)}{2}} Y^{t-1} A^d.$$

By taking $t+1$ small vectors of this lattice and computing the integer roots of the resultant of the corresponding polynomials towards the variables y_i's, we expect finding a as long as:

$$X^{\frac{d(d-1)}{2t}} Y^{1+o(1)} \leq A,$$

when t is large enough.

5 Conclusion and open question

The pseudo-random number generator proposed by Littlewood and its generalisation were shown to be insecure. A simple differentiation attack breaks the original scheme and an algorithm based on polynomial approximation and Coppersmith's technique recovers the secret in the case of the generalised PRNG. Thus real functions seem inadequate for building cryptographic primitives. However, the general idea of using real functions in cryptography cannot entirely be disregarded. The following problem, known as the Table Maker Dilemma (TMD), is a well-known question in the field of computational arithmetic:

TMD: Let n, n', m be some parameters. Given a $\mathcal{C}^\infty$ PTC function $f : [0,1] \to [0,1]$, find all the solutions $x \in [|0,1|]_n$ to the equation:

$$bits_{(n'+1)..(n'+m)}[f(x)] = 0.$$

This question has been studied extensively because it is the key to correctly rounding the elementary functions (see [18], [23], [14] among others). Despite some recent progress based on lattices [21], this problem has no satisfying answer. When $m \approx n$, a simple probabilistic argument suggests that there is only one solution, but the best known algorithm runs in time $2^{0.6n+\epsilon}$. This motivates the following question:

Open question: Let n be a parameter. Let $f : [0,1] \to [0,1]$ be a $\mathcal{C}^\infty$ PTC function. We define the $[|0, 2^n|] \to [|0, 2^n|]$ function:

$$\Phi(x) = bits_{(n+1)..(2n)}[f(\frac{x}{2^n})].$$

Can we consider Φ as a one-way function?

The arguments above suggest that although Φ is easy to compute, it is probably hard to invert. The main advantage of Φ is that it is very easy to implement given any device that can compute the mathematical functions to a high enough precision. If inverting Φ is asymptotically exponential, then this one-way function is quite competitive towards the other number-theoretic one-way functions, like factorisation and discrete logarithms on finite fields or elliptic curves. From Theorem 1, we know that computing Φ is almost as efficient as computing a product of two integers. Moreover, it would be as hard to invert as the discrete logarithm on elliptic curves is supposed to be. From a practical point of view, if the $2^{0.6n+\epsilon}$ algorithm reveals the best possible, for a security level of 2^{80}, $n = 135$ seems to suffice.

Acknowledgements

The present work has been done at the University of Stanford during a research exchange. The author thanks Dan Boneh for having invited him, for presenting him Littlewood's cipher and for helpful discussions and comments.

References

1. Ajtai, M. A conjectured 0-1 law about the polynomial time computable properties of random lattices, I. Electronic Colloquium on Computational Complexity, Report No. 61. 2002.
2. Atkinson, K.E. An introduction to Numerical Analysis. 2nd Edition. John Wiley, New York. 1989.
3. Blömer, J., and May, A. Low Secret Exponent RSA Revisited. Proceedings of CALC'01, volume 2146 of Lecture Notes in Computer Science, pages 4-19. Springer-Verlag. 2001.

4. Blum, M., and Micali, S. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. SIAM J. Computing, 13(4):850-864. 1984.
5. Boneh, D. Finding smooth integers in short intervals using CRT decoding. Journal of Computer and System Sciences, 64:768-784. 2002.
6. Boneh, D., and Durfee, G. Cryptanalysis of RSA with private key less than $n^{0.292}$. Proceedings of Eurocrypt'99, volume 1592 of Lecture Notes in Computer Science, pages 1-11. Springer-Verlag. 1999.
7. Boneh, D., and Halevi, S., and Howgrave-Graham, N. The Modular Inversion Hidden Number Problem. Proceedings of Asiacrypt'01, volume 2248 of Lecture Notes in Computer Science, pages 36-51. Springer-Verlag. 2001.
8. Brent, R. Fast multiple precision zero-finding methods and the complexity of elementary function evaluation. Journal of the ACM, 23:242-251. 1976.
9. Coppersmith, D. Finding small solutions to small degree polynomials. Proceedings of CALC'01, volume 2146 of Lecture Notes in Computer Science, pages 20-31. Springer-Verlag. 2001.
10. Coppersmith, D. Finding a small root of a bivariate integer equation; factoring with high bits known. Proceedings of Eurocrypt'96, volume 1070 of Lecture Notes in Computer Science, pages 326-337. Springer-Verlag. 1996.
11. Goldreich, O. Modern Cryptography, Probabilistic Proofs and Pseudorandomness. Algorithms and Combinatorics 17. Springer-Verlag. 1999.
12. Van Hoeij, M. Factoring Polynomials and the Knapsack Problem. Journal of Number Theory, 95:167-189. 2002
13. Howgrave-Graham, N. Computational mathematics inspired by RSA. PhD thesis of the University of Bath. 1998.
14. Lefèvre, V. Moyens Arithmétiques pour un calcul fiable. Thèse de Doctorat de l'École Normale Supérieure de Lyon. 2000.
15. Lenstra, A.K., and Lenstra, H.W., and Lovász, L. Factoring polynomials with rational coefficients. Mathematische Annalen, 261:515-534. 1982.
16. Littlewood, J.E. A Mathematician's Miscellany. Methuen and Co. Ltd. London. 1953.
17. Lovász, L. An algorithmic theory of numbers, graphs and convexity. SIAM lecture series, 50. 1986.
18. Muller, J.M. Elementary Functions, algorithms and implementation. Birkhauser, 1997.
19. Schneier, B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wi,ley and Sons. 1995.
20. Schönhage, A. Schnelle Multiplikation grosser Zahlen. J. of Computing, 7:281-292. 1971.
21. Stehlé, D., and Lefèvre, V., and Zimmermann, P. Worst Cases and Lattice Reduction. Proceedings of ARITH'16. IEEE Computer Society. 2003.
22. Wilson, D. Littlewood's cipher. Cryptologia 3(2):120-121 and 3(3):172-176. 1979.
23. Ziv, A. Fast evaluation of elementary mathematical functions with correctly rounded last bit. ACM Transactions on Mathematical Software, 17(3):410-423. 1991.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399