View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Missouri: MOspace

CRYPTOGRAPHIC APPLICATIONS OF SPARSE POLYNOMIALS OVER FINITE RINGS

William D. Banks[1], Daniel Lieman[2], Igor E. Shparlinski[3] and Van Thuong To[4]

[1] Department of Mathematics, University of Missouri
Columbia, MO 65211, USA
bbanks@math.missouri.edu

[2] Department of Mathematics, University of Georgia
Athens, GA 30602, USA
dlieman@math.uga.edu

[3] Department of Computing, Macquarie University
Sydney, NSW 2109, Australia
igor@comp.mq.edu.au

[4] Department of Computing, Macquarie University
Sydney, NSW 2109, Australia
tto@comp.mq.edu.au

Abstract. This paper gives new examples that exploit the idea of using sparse polynomials with restricted coefficients over a finite ring for designing fast, reliable cryptosystems and identification schemes.

1 Overview

The idea of using polynomials with restricted coefficients in cryptography, though fairly new, has already found several cryptographic applications such as the NTRU cryptosystem [10], the ENROOT cryptosystem [6], the PASS identification scheme [9, 11], and the SPIFI identification scheme [2]; see also [8].

In contrast to the constructions of NTRU and PASS, which consider classes of low-degree polynomials with many "small" nonzero coefficients, ENROOT and SPIFI are based on the use of polynomials of high degree that are extremely sparse. Although these latter constructions were originally considered only over finite fields, in this paper we improve and extend the ideas of [2, 6] and show that both ENROOT and SPIFI can be generalized to the setting of an arbitrary finite ring. In this generality, the user can be assured of an extra degree of security by selecting rings in which the problem of solving polynomial equations is notoriously difficult, as in the case of the residue ring for an RSA-modulus $M = pl$, where p and l are two privately held primes. In this paper, we have also introduced several new security features for the ENROOT and SPIFI protocols. Quite recently several powerful attacks on the original versions of ENROOT and SPIFI and some of their modifications have been presented in [1]. In particular the present version is a result of our iterative, thanks to many fruitful discussions with the authors of [1]) attempts to make ENROOT and SPIFI resistant to attacks of the types described in [1]. Although we believe that there is no immediate "danger", it still seems that these attacks still present a serious threat to ENROOT and SPIFI. Nevertheless, these objects, sparse polynomials, look too nice and natural (easy to evaluate but hard to invert) not to try to use them for a public key cryptography. We hope that our paper may help to bring more attention to this area.

2 Notation and Definitions

Throughout this paper, $\log z$ denotes the binary logarithm of z.

Let $\mathcal{R}$ be an arbitrary finite ring, and let N denote a fixed multiple of the exponent $\exp(\mathcal{R}^\times)$ of the multiplicative group of units $\mathcal{R}^\times$; thus we have $a^{N+1} = a$ for all $a \in \mathcal{R}^\times \cup \{0\}$.

To illustrate our ideas below, we will sometimes consider two important special cases, which we refer to as the "$\mathbb{F}_q$-case" and the "$\mathbb{Z}_M$-case." In the $\mathbb{F}_q$-case, $\mathcal{R}$ is the finite field $\mathbb{F}_q$ with q elements, and we can take $N = \exp(\mathcal{R}^\times) = q - 1$. In the $\mathbb{Z}_m$-case, $\mathcal{R}$ is the ring $\mathbb{Z}/M\mathbb{Z}$ of residue classes with respect to an RSA modulus $M = pl$, where p and l are primes. In this case, we can either take $N = \varphi(M) = (p-1)(l-1)$, where φ is the Euler function, or $N = \lambda(M) = \mathrm{lcm}(p-1, l-1)$ (λ is the *Carmichael function*). We remark that in all of these cases, we have $\log|\mathcal{R}| \approx \log|\mathcal{R}^\times| \approx \log N$,

We also assume that any element of $\mathcal{R}$ can be encoded by using about $\log|\mathcal{R}|$ bits.

Let d be a fixed positive integer. Given a set $\mathcal{S} \subseteq \mathcal{R}$, we say that a polynomial $f(x_1, \ldots, x_d) \in \mathcal{R}[x_1, \ldots, x_d]$ is an *$\mathcal{S}$-polynomial* if every coefficient of f belongs to $\mathcal{S}$.

An expression of the form $ax_1^{e_1} \ldots x_d^{e_d}$ we call a *monomial* with the *coefficient* a and the *exponent* $(e_1, \ldots, e_d)$.

Finally, we say that a polynomial $f(x_1, \ldots, x_d) \in \mathcal{R}[x_1, \ldots, x_d]$ is *τ-sparse* if f has at most τ nonzero coefficients.

3 The SPIFI Identification Scheme

In this section, we describe a generalization of SPIFI (for *Secure Polynomial IdentiFIcation*; see [2]) for an arbitrary finite ring $\mathcal{R}$. For the sake of simplicity and practicality, we work only with polynomials of a single variable (that is, $d = 1$).

3.1 A Hard Problem

The hard problem underlying our one-way functions can be stated as follows:

Given $2m$ arbitrary elements $\alpha_1, \dots \alpha_m, \beta_1, \dots, \beta_m \in \mathcal{R}$ and a set $\mathcal{S} \subseteq \mathcal{R}$ of small cardinality, it is not feasible to find a τ-sparse $\mathcal{S}$-polynomial $f(x) \in \mathcal{R}[x]$ of degree $\deg(f) \le N$ with $f(\alpha_j) = \beta_j$ for each $j = 1, \dots, m$, provided that N is of "medium" size relative to the choices of $m \ge 1$, the cardinality $|\mathcal{S}|$, and $\tau \ge 3$.

More precisely, we expect that if one fixes the number of points m, the cardinality $|\mathcal{S}|$, and the sparsity $\tau \ge 3$, then the problem requires exponential time as $N \to \infty$ (that is, exponential with respect to the bit length of N).

For example, let p be a prime, and consider the case where $\mathcal{R}$ is the finite field $\mathbb{F}_p$ with p elements. Let $a_{ij} \equiv \alpha_j^i \pmod p$ and $b_j \equiv \beta_j \pmod p$ be chosen so that $0 \le a_{ij}, b_j \le p-1$ for $i = 0, \dots, p-1$ and $j = 1, \dots, m$. Then in this simplified situation, the hard problem above is still equivalent to the problem of finding a feasible solution to the *integer programming problem*

$$\sum_{i=0}^{p-1} x_i \varepsilon_i a_{ij} + y_j p = b_j, \qquad j = 1, \dots, m, \qquad \sum_{i=0}^{p-1} \varepsilon_i \le \tau,$$

where $y_j \in \mathbb{Z}$, $x_i \in \mathcal{S}$, and $\varepsilon_i \in \{0, 1\}$ for all i and j.

3.2 Basic Idea

We fix the ring $\mathcal{R}$ and some integer parameters $k \ge 1$ and $r, s, t \ge 3$. This information is made *public*. The value of N may be kept *private*. Only *Alice* needs this value, so in this scenario the choice of the ring $\mathcal{R}$ (and the value of N) can be made by *Alice*.

In addition we require that $\mathcal{R}$ contains elements of multiplicative order in the interval $[0.5N^{1/4}, 2N^{1/4}]$. This certainly imposes some additional number theoretic requirements on N which in practice are easy to satisfy.

To create the signature *Alice* uses the following algorithm, which we still denote by SPIFI.

Initial Set-up

Step 1
Select at random k distinct elements $a_0, \dots a_{k-1} \in \mathcal{R}^\times$ where a_0 of multiplicative order in the interval $[0.5N^{1/4}, 2N^{1/4}]$.

Step 2
Select a random $\lceil t/2 \rceil$-sparse $\{0, 1\}$-polynomial $f_1(x) \in \mathcal{R}[x]$ with $\deg(f_1) \le N$ and $f_1(a_0) \in \mathcal{R}^\times$. Next, select a random $\lfloor t/2 \rfloor$-sparse $\{0, 1\}$-polynomial $f_2(x) \in \mathcal{R}[x]$ with $\deg(f_2) \le N$, $f_2(a_0) \ne 0$ and $f_2(a_0) \ne -f_1(a_0)$.

Step 3
Compute $A = -f_2(a_0) f_1(a_0)^{-1}$ and put $f(x) = Af_1(x) + f_2(x)$. Then f is a t-sparse $\{0, 1, A\}$-polynomial with $\deg(f) \le N$, and $f(a_0) = 0$. The polynomial f is the *private key*.

Step 4

Compute $C_j = f(a_j)$ for $j = 1, \ldots, k-1$.

Step 5

Publish the set of values $\{A, a_0, \ldots a_{k-1}, C_1, \ldots, C_{k-1}\}$ as the *public key*.

To verify *Alice*'s identity, *Alice* and *Bob* use the following procedure.

Verification Protocol

Step 1

Alice selects a random r-sparse $\{0,1\}$-polynomial $g(x) \in \mathcal{R}[x]$ with $\deg(g) \le N$ and $g(0) = 0$, computes

$$D_j = g(a_j), \qquad j = 1, \ldots, k-1,$$

and sends the sum $D = D_1 + \ldots + D_{k-1}$ to *Bob*.

Step 2

Bob selects a random s-sparse $\{0,1,B\}$-polynomial $h(x) \in \mathcal{R}[x]$ of degree $\deg(h) \le N$ and sends h to *Alice*. Here $B \ne 0, 1$ or A.

Step 3

Alice computes

$$F(x) \equiv f(x)g(x)h(x) \pmod{x^{N+1} - x}$$

and sends the polynomial F and $\{D_1, \ldots, D_{k-1}\}$ to *Bob*.

Step 4

Bob computes

$$E_j = h(a_j), \qquad j = 1, \ldots, k-1,$$

and verifies that

$$D_1 + \ldots + D_{k-1} = D$$

and that $F(x)$ is an rst-sparse $\{0,1,A,B,AB\}$-polynomial with $\deg(F) \le N$, $F(a_0) = 0$, and

$$F(a_j) = C_j D_j E_j, \qquad j = 1, \ldots, k-1.$$

Of course, there is a chance that the constructed polynomial $F(x)$ is not a $\{0,1,A,B,AB\}$-polynomial; however, if rst is substantially smaller than N, then this chance is negligible (and in this case, *Alice* and *Bob* can repeat the procedure).

3.3 Efficiency

The sparsity of the polynomials involved guarantees computational efficiency for this scheme. Using (naive) repeated squaring, one can compute the power a^e for any $a \in \mathcal{R}$ and $0 \le e \le N$ in about $2 \log N$ arithmetic operations in $\mathcal{R}$ in the worst case, or about $1.5 \log N$ arithmetic operations "on average"; see Section 1.3 of [3], Section 4.3 of [4], or Section 2.1 of [5]. Consequently, any

τ-sparse polynomial $f(x) \in \mathcal{R}[x]$ of degree at most N can be evaluated at any point in about $O(\tau \log N)$ arithmetic operations in $\mathcal{R}$.

We recall that any element of $\mathcal{R}$ can be encoded by using about $\log |\mathcal{R}|$ bits.

Finally, we remark that if $0 \in \mathcal{S} \subseteq \mathcal{R}$, then any τ-sparse $\mathcal{S}$-polynomial $f(x) \in \mathcal{R}[x]$ of degree at most N can be encoded with about $\tau \log(N|\mathcal{S}| - N)$ bits. To do this, we have to identify at most τ positions at which f has a nonzero coefficient. The encoding of each position requires about $\log N$ bits, and for each such position, about $\log(|\mathcal{S}| - 1)$ bits are then required to determine the corresponding element of $\mathcal{S}$.

For example, the signature must encode rst positions of the polynomial F (corresponding to its nonzero coefficients), which takes about $rst \log N$ bits. Each position requires two additional bits to distinguish between the possible nonzero coefficients 1, A, B and AB. The encoding of $D_1, \ldots, D_{k-1}$ and their sum D requires about $k \log |\mathcal{R}|$ bits. Hence the total signature size is about $rst \log(4N) + k \log |\mathcal{R}|$ bits.

Putting everything together, after simple calculations we derive that (using the naive repeated squaring exponentiation)

- the *initial set-up* takes $O(kt \log N)$ arithmetic operations in $\mathcal{R}$;
- the *private key size* is about $t \log(2N)$ bits;
- the *public key size* is about $k(\log |\mathcal{R}| + \log |\mathcal{R}^{\times}|)$ bits;
- *signature generation*, that is, computation of the polynomial F, elements D_j, $j = 1, \ldots, k-1$, and their sum D, takes $O(rst)$ arithmetic operations with integer numbers in the range $[0, 2N]$ and $O((k-1)r \log N)$ arithmetic operations in $\mathcal{R}$;
- the *signature size* is about $rst \log(4N) + k \log |\mathcal{R}|$ bits;
- *signature verification*, that is, computation of $D_1 + \ldots + D_{k-1}$, $F(a_j)$ and the products $C_j D_j E_j$, $j = 1, \ldots, k-1$, takes about $O(krst \log N)$ arithmetic operations in $\mathcal{R}$.

We remark that the practical and asymptotic performance of the SPIFI scheme can be improved if one uses more sophisticated algorithms to evaluate powers and sparse polynomials; see [3–5, 13, 15]. In particular, one can use precomputation of certain powers of the a_j, $j = 1, \ldots, k-1$, and several other clever tricks which we do not consider in this paper.

3.4 Possible Attacks

It is clear that recovering or faking the private key (that is, finding a t-sparse $\{0, 1, A\}$-polynomial polynomial $\widetilde{f}(x) \in \mathcal{R}[x]$ with $\widetilde{f}(a_0) = 0$ and $\widetilde{f}(a_j) = C_j$ for $j = 1, \ldots, k-1$) or faking the signature (that is, finding a rst-sparse $\{0, 1, A, B, AB\}$-polynomial $\widetilde{F}(x) \in \mathcal{R}[x]$ with $\widetilde{F}(a_0) = 0$ and $\widetilde{F}(a_j) = C_j D_j E_j$ for $j = 1, \ldots, k-1$) are versions of the hard problem mentioned in Section 3.1 (with slightly different parameters).

We also remark that that without the reduction

$$f(x)g(x)h(x) \pmod{x^{N+1} - x},$$

one of the one possible attacks might be via polynomial factorization. In a practical implementation of this scheme, one should make sure that both f and g have terms of degree greater than $N/2$ so there are some reductions. Even without the reduction modulo $x^{N+1} - x$, the factorization attack does not seem to be feasible because of the large degrees of the polynomials involved; all known factorization algorithms (as well as their important components such as irreducibility testing and the greatest common divisor algorithms) do not take advantage of sparsity or any special structure of the coefficients; see [4, 14]. Moreover, the first factor that any of these algorithms will find would be the trivial one, that is, $(x - a_0)$. But the quotient $F(x)/(x - a_0)$ is most likely neither sparse nor an $\mathcal{S}$-polynomial for any small set $\mathcal{S}$. Finally, we remark that if one works in the setting of a ring $\mathcal{R}$ that is *not* a field (such as the $\mathbb{Z}_M$-case), then the problem of factorization becomes much more complicated, so this type of attack is even less likely to succeed.

It is possible that by using some "clever" choice of polynomials h, after several rounds of identification, *Bob* might be able to gain some information about f. But the polynomials g are specifically designed to prevent him from doing this. In Section 3.5 below, we present another idea which should render this attack completely infeasible, at least in the $\mathbb{F}_q$-case.

One might also consider lattice attacks. In particular, one can try to select a rt-sparse $\{0, 1, A\}$-polynomial $e(X) \in \mathcal{R}[x]$ with $e(a_0) = 0$, compute

$$D_j = e(a_j)C_j^{-1}, \qquad j = 1, \ldots, k - 1,$$

and then send these values together with

$$F(x) \equiv e(x)h(x) \pmod{x^{N+1} - x}$$

to the verifier. In principal this attack could succeed but finding such a polynomial e is kind of knapsack problem and since the dimension of the corresponding lattice would be equal to the (very large) degree N of the polynomials involved, any such attack seems completely infeasible at this time. With current technology, one can reduce lattices of degrees only in the hundreds, while in a practical implementation of this scheme our lattices will have dimension N of much large order of magnitude. Another attempt to construct such a polynomial e could be via solving the discrete logarithm in $\mathcal{R}$ to base a_0, see [1]. However because a_0 is selected to be of small order this attack is very unlikely to succeed either.

There is also another way to avoid the above attack via constructing a rt-sparse $\{0, 1, A\}$-polynomial $e(X) \in \mathcal{R}[x]$ with $e(a_0) = 0$. This way does not require any restrictions of the order of a_0 and thus can be used for arbitrary N. Namely we request that for each of the polynomials $f(x)$, $g(x)$ and $h(x)$ the sum of the degrees of the monomials is divisible by N. In this case the same condition also holds for $F(x)$. Indeed, if an s-sparse polynomial and a t-sparse polynomial have monomials of degrees $n_1, \ldots, n_s$ and $m_1, \ldots, m_t$, respectively, with

$$\sum_{i=1}^{s} n_i \equiv 0 \pmod{N} \qquad \text{and} \qquad \sum_{j=1}^{t} m_j \equiv 0 \pmod{N}$$

then their product has st monomials $n_i + m_j$, $i = 1, \ldots, s$, $j = 1, \ldots, t$ (unless a collision occurs which is very unlikely). Then

$$\begin{aligned}\sum_{i=1}^{s}\sum_{j=1}^{t}(n_i + m_j) &= \sum_{i=1}^{s}\left(tn_i + \sum_{j=1}^{t} m_j\right)\\ &= t\sum_{i=1}^{s} n_i + s\sum_{j=1}^{t} m_j \equiv 0 \pmod{N}\end{aligned}$$

as claimed. Therefore the aforementioned discrete logarithm attack from [1] is very unlikely (with probability about $1/N$) to produce a polynomial $e(x)$ which besides the aforementioned conditions also has the sum of the degrees of the monomials which is divisible by N.

We remark that, although using composite moduli may add some additional security features, the security of the SPIFI scheme is not compromised even if the factorization of M is known. In fact, we believe that even in the case where the modulus is a (sufficiently large) prime (that is, in the $\mathbb{F}_q$-case), the scheme is still very secure.

It has turned out that *Alice* must make some commitment about the values of $D_1, \ldots, D_{k-1}$ before she receives the polynomial h from *Bob*, otherwise there is a very simple attack on this scheme. On the other hand, sending the whole set to *Bob* before he selects his polynomial h may open some ways of attacking for “cheating” verifier. Sending the sum $D = D_1 + \ldots + D_{k-1}$ is just one of many possible ways for *Alice* undertake some commitments about the values of $D_1, \ldots, D_{k-1}$ (just reducing the probability of the aforementioned “on-line” attack to $1/N$). Probably a more practical way would be just sending about a half of the bits of $D_1, \ldots, D_{k-1}$ at Step 1 (instead of computing and sending D) and then sending the rest of the bits at Step 3 (just reducing the probability of the aforementioned “on-line” attack to about $N^{-(k-1)/2}$).

Moreover, the SPIFI scheme is easily modified so that the value $N = \varphi(M)$ or $\lambda(M)$ (see Section 2) remains secret. Indeed, *Alice* can choose g in Step 1 of the verification protocol so that the reduction modulo $x^{N+1} - x$ that occurs in Step 3 produces a polynomial whose degree is not “too close” to N. In fact, “on average” it should be about $N(1 - 1/2rst)$ for the SPIFI scheme and about $N(1 - 1/2R)$ for the ENROOT scheme (see Section 4 below) since the corresponding polynomials are rst-sparse and R-sparse, respectively. Thus, in the case that $N = \varphi(M)$, the degree of F gives a worse approximation to N than the value of M itself, at least if $M = pl$ is a product of two primes of the same order.

3.5 Modification of the Basic Scheme

In this subsection, we consider only the case of a finite field $\mathcal{R} = \mathbb{F}_q$. It is very likely, however, that these ideas can be generalized to the setting of an arbitrary ring.

In order to prevent *Bob* from gaining any useful information about f by selecting certain special polynomials h, *Alice* can initially construct *two* polynomials f_1 and f_2, either one of which can serve as her private key. That is, for some $A, C_1, \ldots, C_{k-1} \in \mathbb{F}_q$ with $A \neq 0, 1$ and distinct $a_0, \ldots, a_{k-1} \in \mathbb{F}_q^\times$, *Alice* can find two t-sparse $\{0, 1, A\}$-polynomials $f_1(x), f_2(x) \in \mathbb{F}_q[x]$ of degree at most N that satisfy

$$f_1(a_j) = f_2(a_j) = C_j, \qquad j = 0, \ldots, k-1, \tag{1}$$

for some $C_0, C_1, \ldots, C_{k-1} \in \mathbb{F}_q$.

To do this, *Alice* selects a certain parameter n and considers certain random $\{\pm 1, \pm A\}$-polynomials $\psi(x)$ of degree $4n$, looking for roots in $\mathbb{F}_q^\times$. For values of n of reasonable size this can be done quite efficiently, at least in probabilistic polynomial time; see [4, 14].

It follows from Theorem 3 of [12] that for sufficiently large q, the probability that a random monic polynomial of degree $4n$ over $\mathbb{F}_q$ will have $k+1$ distinct roots in $\mathbb{F}_q$ is given by

$$P_{k+1}(4n, q) = \sum_{m=k+1}^{\infty} \binom{q}{m} q^{-m} \sum_{l=0}^{4n-m} (-1)^l \binom{q-m}{l} q^{-l}.$$

In particular,

$$\lim_{n\to\infty} \lim_{q\to\infty} P_{k+1}(4n, q) = \frac{1}{(k+1)\,!\, e^{k+1}}.$$

Letting A vary randomly over $\mathbb{F}_q/\{0, 1\}$, *Alice* considers $\{\pm 1, \pm A\}$-polynomials $\psi(x) \in \mathbb{F}_q[x]$ of degree $4n$ which have n coefficients of each type 1, -1, A or $-A$. Since

$$(q-2)\frac{(4n)\,!}{(n\,!)^4 (k+1)\,!\, e^{k+1}}$$

is large, after $O((k+1)!e^{k+1})$ trials *Alice* will find with high probability such a polynomial with $k+1$ distinct roots $a_0, \ldots, a_k \in \mathbb{F}_q$. By reordering if necessary, we can assume that $a_0, \ldots, a_{k-1}$ are distinct elements in the multiplicative group $\mathbb{F}_q^\times$. *Alice* now writes $\psi(x) = \varphi_1(x) - \varphi_2(x)$ where φ_1, φ_2 are $2n$-sparse $\{0, 1, A\}$-polynomials of degree at most $4n+1$, and each φ_i has n coefficients of each type 1 or A. Moreover, $\varphi_1 \neq \varphi_2$, but clearly

$$\varphi_1(a_j) = \varphi_2(a_j), \qquad j = 0, \ldots, k-1.$$

Now *Alice* selects a random $(\lceil t/2 \rceil - n)$-sparse $\{0, 1\}$-polynomial $\psi_1(x) \in \mathbb{F}_q[x]$ with $\deg(\psi_1) \le q-1$ and $\psi_1(a_0) \neq 0$. *Alice* then selects a random $(\lfloor t/2 \rfloor - n)$-sparse $\{0, 1\}$-polynomial $\psi_2(x) \in \mathbb{F}_q[x]$ with $\deg(\psi_2) \le q-1$. Assuming that ψ_1 and ψ_2 have been selected so that the non-constant monomials that occur in them have degree greater than $4n+1$, *Alice* can now define

$$f_i(x) = A\psi_1(x) + \psi_2(x) + \varphi_i(x), \qquad i = 1, 2.$$

Then f_1 and f_2 are t-sparse $\{0, 1, A\}$-polynomials in the correct form for the SPIFI protocol, and they satisfy (1) some $C_0, C_1, \dots, C_{k-1} \in \mathbb{F}_q$ We remark that in this case the value $C_0 = f_1(a_0) = f_2(a_0)$ must be published as well (although the scheme can easily be modified in such a way that as before $f_1(a_0) = f_2(a_0) = 0$ thus this value need not be sent).

Now *Alice* can alternate between f_1 and f_2 in a random order to confound *Bob*'s attempts to gain useful information about the private key.

It is easy to see that instead of the sum $A\psi_1(x) + \psi_2(x) + \varphi_i(x)$ for $i = 1, 2$, one can also consider more complicated expressions involving $\{0, 1\}$-polynomials. For example, one can put

$$f_i(x) = A\psi_1(x) + \psi_2(x) + \varphi_i(x)\varphi(x), \qquad i = 1, 2,$$

for appropriately chosen $\{0, 1\}$-polynomials $\psi_1(x), \psi_2(x)$ and $\varphi(x)$.

3.6 Remarks

It is natural to try to construct and utilize more than two t-sparse $\{0, 1, A\}$-polynomials that take the same values at k distinct points. However our approach of Section 3.5 does not seem to extend to this case.

Although we have not done so here, it can be interesting to extend our construction to multivariate polynomials.

4 The ENROOT Cryptosystem

In this section, we describe a generalization of ENROOT (for *ENcryption with ROOTs*; see [6]) for an arbitrary finite ring $\mathcal{R}$. We will now consider polynomials in $\mathcal{R}[x_1, \dots, x_d]$, where $d \geq 2$ is fixed. Accordingly, we will often employ vector notation, writing $f(\boldsymbol{x})$ for $f(x_1, \dots, x_d)$, $\mathcal{R}[\boldsymbol{x}]$ for $\mathcal{R}[x_1, \dots, x_d]$, etc.

4.1 Another Hard Problem

Our one-way functions are based on the following hard problem:

> *Given the τ-sparse polynomials $f_1(\boldsymbol{x}), \dots, f_d(\boldsymbol{x}) \subset \mathcal{R}[\boldsymbol{x}]$ of degree at most N, it is not feasible to find an element $\boldsymbol{a} = (a_1, \dots, a_d) \in \mathcal{R}^d$ with $f_j(\boldsymbol{a}) = 0$ for $j = 1, \dots, d$, provided that N is sufficiently large relative to the choices of $d \geq 2$ and $\tau \geq 3$.*

Again, we expect that if one fixes the number $d \geq 2$ and the sparsity $\tau \geq 3$, then the problem requires exponential time as $N \to \infty$ (see Section 4.4 below).

4.2 Basic Idea

We fix the ring $\mathcal{R}$ and the integers $d > \ell \geq 3$, $s_j, t_j \geq 3$, $j = 1, \ldots, d$ such that $t_1 = \ldots = t_\ell$. This information is made *public*. The value of N may be kept *private*. In fact, only *Bob* needs this value so in this scenario the choice of the ring $\mathcal{R}$ (and thus the value of N) is made by *Bob*.

The algorithm ENROOT can be described as follows.

ENROOT Algorithm

Step 1
Alice selects d random elements $a_1, \ldots, a_d \in \mathcal{R}^\times$ which form her *private key*.

Step 2
Alice selects d random polynomials $h_j(\boldsymbol{x}) \in \mathcal{R}[\boldsymbol{x}]$, of degree at most $|\mathcal{R}|$, containing at most $t_j - 1$ monomials, $j = 1, \ldots, d$, such that the first ℓ polynomials $h_1(\boldsymbol{x}), \ldots, h_\ell(\boldsymbol{x})$ have the same set $\mathcal{E}$ of exponents of their monomials.

Step 3
Alice publishes the polynomials $f_j(\boldsymbol{x}) = h_j(\boldsymbol{x}) - h_j(\boldsymbol{a})$ for $j = 1, \ldots, d$ as her *public key*, where $\boldsymbol{a}$ is the vector $(a_1, \ldots, a_d) \in (\mathcal{R}^\times)^d$.

Step 4
To send a message $m \in \mathcal{R}$, *Bob* selects d random polynomials $g_j(\boldsymbol{x}) \in \mathcal{R}[\boldsymbol{x}]$ of degree at most N, containing at most $s_j - 1$ monomials such that one monomial has an exponent from the set $\mathcal{E}$ and having nonzero constant coefficients. *Bob* then computes the reduction $F(\boldsymbol{x})$ of the polynomial

$$m + f_1(\boldsymbol{x})g_1(\boldsymbol{x}) + \ldots + f_d(\boldsymbol{x})g_d(\boldsymbol{x})$$

modulo the ideal in $\mathcal{R}[\boldsymbol{x}]$ generated by $\{x_1^{N+1} - x_1, \ldots, x_d^{N+1} - x_d\}$, and he sends $F(\boldsymbol{x})$ to *Alice*.

Step 5
To decrypt the message, *Alice* simply computes $F(\boldsymbol{a}) = m$.

4.3 Efficiency

The sparsity of the polynomials involved again provides computational efficiency for this scheme. Using repeated squaring, one can compute the monomial $a_1^{e_1} \ldots a_d^{e_d}$ for any $(a_1, \ldots, a_d) \in \mathcal{R}^d$ and $0 \leq e_j \leq |\mathcal{R}|$, $j = 1, \ldots, d$, in about $O(d \log(2|\mathcal{R}|))$ arithmetic operations in $\mathcal{R}$. Consequently, any τ-sparse polynomial $f(\boldsymbol{x}) \in \mathcal{R}[\boldsymbol{x}]$ of degree at most $|\mathcal{R}|$ can be evaluated at any point in $\mathcal{R}^d$ in about $O(\tau d \log(2|\mathcal{R}|))$ arithmetic operations in $\mathcal{R}$.

We remark that any τ-sparse polynomial $f(\boldsymbol{x}) \in \mathcal{R}[\boldsymbol{x}]$ of degree at most $|\mathcal{R}|$ can be encoded with about $\tau((d+1)\log|\mathcal{R}|)$ bits. To do this, we have to identify at most τ monomials for which f has a nonzero coefficient. The encoding of each monomial $x_1^{e_1} \ldots x_d^{e_d}$ requires about $d \log |\mathcal{R}|$ bits, and for each such monomial about $\log |\mathcal{R}|$ bits are then required to encode the coefficient.

Let us set

$$T = \sum_{j=1}^{d} t_j, \qquad S = \sum_{j=1}^{d} s_j, \qquad R = \sum_{j=1}^{d} t_j s_j, \qquad Q = (d+1)\log|\mathcal{R}|.$$

Then after simple calculations we derive that (using the naive repeated squaring exponentiation)

- ∘ *generation of the public key*: to produce the vector $\boldsymbol{a}$ requires $O(d \log|\mathcal{R}|)$ random bits; to construct the polynomials $h_j(\boldsymbol{x})$ requires the generation of another $(T-d)Q$ random bits; the computation of the $h_j(\boldsymbol{a})$, $j = 1, \ldots, d$, takes $O(Td\log(2|\mathcal{R}|))$ arithmetic operations in $\mathcal{R}$;
- ∘ the *private key size* is about $d\log|\mathcal{R}| + (T-d)Q$ bits;
- ∘ the *public key size* is about TQ bits;
- ∘ *cost of encryption*: to construct the polynomials $g_j(\boldsymbol{x})$ requires the generation of about $d\log|\mathcal{R}| + (S-d)Q$ random bits; the computation of the polynomial $F(\boldsymbol{x})$ requires about R arithmetic operations in $\mathcal{R}$ plus Rd additions in $\mathbb{Z}/N\mathbb{Z}$;
- ∘ the *size of the encrypted message* is about RQ bits;
- ∘ the *cost of decryption*: the evaluation of $F(\boldsymbol{a}) = m$ takes about $O(Rd\log(2N))$ arithmetic operations in $\mathcal{R}$.

In the $\mathbb{F}_q$-case, the above scheme can be accelerated if *Alice* sets $e_1 = 1$, selects a random element $a \in \mathcal{R}^\times$ and $d-1$ random exponents $e_2, \ldots, e_d \in \mathbb{Z}/(q-1)\mathbb{Z}$, and defines $\boldsymbol{a}$ as $(a^{e_1}, \ldots, a^{e_d}) \in (\mathcal{R}^\times)^d$.

Again we mention that the performance of the ENROOT algorithm can be improved if one uses more sophisticated algorithms to evaluate powers and sparse polynomials; see [3–5, 13, 15].

Another possible way to improve performance is to use at Step 4 only $k < d$ randomly selected polynomials from the set $\{f_1, \ldots, f_d\}$. For the same level of security, there will be a trade-off between the complexity of Step 2 (hence the size of the private key) and the complexity of Step 4.

4.4 Security Considerations

The obvious way to attack the ENROOT cryptosystem is to try to find a simultaneous solution to the system of polynomial equations

$$f_j(\boldsymbol{x}) = 0, \qquad j = 1, \ldots, d, \tag{2}$$

which amounts to solving the hard problem in Section 4.1. All known algorithms to solve systems of polynomial equations of total degree n require (regardless of sparsity) an amount of time that is polynomial in n (that is, exponential time with respect to the bit length of n); see [7, 14]. Since the degrees of the polynomials in (2) will be very large in practical implementations (about the size of N), this attack is totally infeasible.

Another possibility is to simply "guess" a solution. One expects that a system of τ-sparse polynomial equations in d variables of high degree over $\mathcal{R}$ will have very few zeros if $d \geq 2$, even though the number of zeros of a polynomial over an arbitrary ring is not necessarily bounded by the degree. Working heuristically, if we view the vector of polynomials $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_d(\boldsymbol{x}))$ as defining a "random" map $\boldsymbol{f} : \mathcal{R}^d \to \mathcal{R}^d$, then the expected number of roots common to all of the polynomials $f_j(\boldsymbol{x})$ (that is, the cardinality of the kernel of $\boldsymbol{f}$) is given by

$$\frac{1 - |\mathcal{R}|^{d(1-|\mathcal{R}|^d)}}{1 - (1 - |\mathcal{R}|^{-d})^{|\mathcal{R}|^d}} \approx \frac{1}{1 - e^{-1}} \approx 1.5819,$$

hence this brute force attack will take roughly $0.245|\mathcal{R}|^d$ trials "on average" to succeed. For arbitrary rings, we expect the choice $d \geq 2$ will provide the 2^{90} level of security against this attack if N is at least 50 bits long.

Although it is tempting to choose $d = \ell = 1$, in this case there are more sophisticated attacks that provide some information about *Alice*'s private key. One of these is based upon consideration of the difference set of the powers of monomials occurring in the polynomial $F(x)$. Indeed, if

$$f(x) = \sum_{i=1}^{t} A_i x^{n_i} \qquad \text{and} \qquad g(x) = \sum_{j=1}^{s} B_i x^{m_j}$$

are the polynomials selected by *Alice* and *Bob*, respectively, with $n_1 = m_1 = 0$ (and such that the sets $\{n_1, \ldots, n_t\}$ and $\{m_1, \ldots, m_s\}$ have a reasonably large intersection) then $F(x)$ contains at most $\tau \leq st$ monomials $C_\nu x^{r_\nu}$, where

$$r_\nu \equiv n_i + m_j \pmod{N}, \qquad \nu = 1, \ldots, \tau,$$

for some $i = 1, \ldots, t$ and $j = 1, \ldots, s$.

Consequently, finding the repeated elements in the difference set

$$\Delta = \{r_\nu - r_\eta \pmod{N} \,|\, \nu, \eta = 1, \ldots, \tau\},$$

may reveal some information about the polynomial $f(x)$.

In addition, if $d = 1$, one can also compute the greatest common divisor of $f(x)$ with $x^{N+1} - x$. Since this polynomial will have lower degree than f in general, it would be easier to find a root from a theoretical standpoint. Although it is not clear how to do this in an amount of time that is polynomial in the sparsity rather than in the degree of $f(x)$, it remains a potential threat.

On the other hand, these attacks on ENROOT fail when $d > \ell \geq 2$. Indeed, the first attack may help to gain some information about the total set of monomials in all of the polynomials $f_1, \ldots, f_d$, but it does not provide any information about the individual polynomials since it is impossible to determine which monomial comes from which product $f_j g_j$, $j = 1, \ldots, d$. In order to try all possible partitions into d groups of $s_j t_j$ monomials, $j = 1, \ldots, d$, needs to examine

$$\mathcal{P} = \frac{R\,!}{(s_1 t_1)\,! \ldots (s_d t_d)\,!} \tag{3}$$

total combinations. In particular, the most interesting case is when $s_1, \ldots, s_d$ are of approximately the same size as well as $t_1, \ldots, t_d$, that is, when $s_j \sim s$ and $t_j \sim t$ for $j = 1, \ldots, d$. In this case

$$\log \mathcal{P} \sim st(d \log d),$$

hence the number $\mathcal{P}$ of combinations to consider grows exponentially with respect to all parameters, provided that $d > \ell \geq 2$.

The second attack fails as well since the notion of greatest common divisor for multivariate polynomials is not defined, and taking resolvents to reduce to one variable is too costly.

However the case $d = 2$ is still not secure. Indeed, in this case we have either $d = \ell = 2$ or $\ell = 1$. In either case there are very simple linear algebra attacks which do not apply when $d > \ell \geq 2$ which we believe to be completely secure against the aforementioned attacks. There are some other alternative ways to guarantee that there are sufficiently many common elements in the sets of exponents of monomials of $f_1, \ldots, f_d$. In particular, the first few monomials of each polynomial $h_1, \ldots, h_d$ can be selected the same exponents.

Finally, we remark that the ENROOT cryptosystem is probably secure against lattice attacks for the same reason that the SPIFI scheme is secure (see Section 3.4): most lattice attacks would necessarily be based on lattices of dimension equal to the cardinality of $|\mathcal{R}|$, and in practical implementations this number would be very large.

4.5 Remarks

The ENROOT cryptosystem is well-suited to private key sharing among multiple parties. For parameter choices and approximate runtimes in the $\mathbb{F}_q$-case, we refer the reader to [6]. The main inherent weakness of this cryptosystem is its high message expansion cost. It is likely that working over certain noncommutative rings or rings that are not principal ideal domains may improve the overall security, so that smaller rings could be employed. Working over these rings, it would be interesting to have a more thorough security analysis than we have attempted here.

Acknowledgments. The authors are grateful to Claus Schnorr for pointing out an error in the initial version of the SPIFI scheme. The authors also thank the authors of [1] for very interesting discussions of other attacks on SPIFI and ENROOT and generally the perspectives of sparse polynomial based cryptography.

Thanks also go to Kwangjo Kim and Arjen Lenstra for their assistance in contacting with the authors of [1].

This work was done during a series of visits by W. B. and D. L. to Macquarie University, whose hospitality and support are gratefully acknowledged.

Work also supported in part, W. B. by NSF grant DMS-0070628, for D. L. by NSF grant DMS-9700542, and a Big 12 Faculty Fellowship from the University of Missouri and for I. S. by ARC grant A69700294.

References

1. F. Bao, R. H. Deng, W. Geiselmann, C. Schnorr, R. Steinwandt and H. Wu, 'Crytoanalysis of two sparse polynomial based cryptosystems', *Proc. Int. Conf. on Public Key Cryptography, PKC'2001, Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, 2001, to appear.
2. W. Banks, D. Lieman and I. E. Shparlinski, 'An identification scheme based on sparse polynomials', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **1751**, 68–74.
3. H. Cohen *A course in computational algebraic number theory*, Springer-Verlag, Berlin, 1997.
4. J. von zur Gathen and J. Gerhard, *Modern computer algebra*, Cambridge Univ. Press, Cambridge, 1999.
5. D. M. Gordon, 'A survey of fast exponentiation methods', *J. Algorithms*, **27** (1998), 129–146.
6. D. Grant, K. Krastev, D. Lieman and I. E. Shparlinski, 'A public key cryptosystem based on sparse polynomials', *Proc. International Conference on Coding Theory, Cryptography and Related Areas, Guanajuato, 1998*, Springer-Verlag, Berlin, 2000, 114–121.
7. M.-D. A. Huang and Y.-C. Wong, 'Solving systems of polynomial congruences modulo a large prime', *Proc. 37 IEEE Symp. on Found. of Comp. Sci.*, 1996, 115–124.
8. J. Hoffstein, B. S. Kaliski, D. Lieman, M. J. B. Robshaw and Y. L. Yin, 'A new identification scheme based on polynomial evaluation', *US Patent*, No. **No. 6076163**, 2000.
9. J. Hoffstein, D. Lieman and J. H. Silverman, 'Polynomial Rings and Efficient Public Key Authentication', *Proc. the International Workshop on Cryptographic Techniques and E-Commerce*, City University of Hong Kong Press, to appear.
10. J. Hoffstein, J. Pipher and J. H. Silverman, 'NTRU: A ring based public key cryptosystem', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **1433** (1998), 267–288.
11. J. Hoffstein and J. H. Silverman, 'Polynomial rings and efficient public key authentication II', *Proc. the International Workshop on Cryptography and Computational Number Theory, Singapore, 1999*, Birkhäuser, to appear.
12. A. Knopfmacher and J. Knopfmacher, 'Counting polynomials with a given number of zeros in a finite field', *Linear and Multilinear Algebra*, **26** (1990), 287–292.
13. N. Pippenger, 'On the evaluation of powers and monomials', *SIAM J. Comp.*, **9** (1980), 230–250.
14. I. E. Shparlinski, *Finite fields: Theory and computation*, Kluwer Acad. Publ., Dordrecht, 1999.
15. A. C.-C. Yao, 'On the evaluation of powers', *SIAM J. Comp.*, **5** (1976), 100–103.