A Public Key Cryptosystem Based on Sparse Polynomials

D. Grant, K. Krastev, D. Lieman, and I. Shparlinski

[1] Department of Mathematics, University of Colorado
Boulder, CO 80309, USA
grant@boulder.colorado.edu
[2] School of MPCE, Macquarie University
Sydney, NSW 2109, Australia
kate@mpce.mq.edu.au
[3] Department of Mathematics, University of Missouri
Columbia, MO 65211, USA
lieman@math.missouri.edu
[4] School of MPCE, Macquarie University
Sydney, NSW 2109, Australia
igor@mpce.mq.edu.au

Abstract. This paper introduces a new type of cryptosystem which is based on sparse polynomials over finite fields. We evaluate its theoretic characteristics and give some security analysis. Some preliminary timings are presented as well, which compare quite favourably with published optimized RSA timings. We believe that similar ideas can be used in some other settings as well.

1 Overview

In this paper we present a new idea for the construction of one-way functions.

The hard problem underlying our one-way functions can be stated as follows: *Given a system of sparse polynomial equations of high degree over certain large rings, it is hard to find a solution to this system.*

On the other hand, because the polynomials involved are sparse, their values at any point can be computed quite efficiently. We have conducted tests of our cryptosystem with parameter choices equivalent to four different levels of security, including the two most popular RSA levels of security and a 2^{80} level of security. Even with no serious attempt at optimization, our cryptosystem can encrypt and decrypt a message at speeds roughly equal to that of optimized RSA. In addition, key generation in our scheme is several orders of magnitude faster than in RSA.

We remark that several other cryptosystems based on polynomials have been developed, see [7, 9, 10] for example, but all of them exploit quite different ideas.

Througout the paper $\log x$ and $\ln x$ denote the binary logarithm and the natural logarithm of $x > 0$, respectively.

2 Construction of a Cryptosystem

Here we describe one of several possible variants of this cryptosystem, which we construct from polynomials over finite fields.

Following the established tradition, we call the communicating parties *Alice* and *Bob*.

The algorithm EnRoot (encryption with roots) can be described as follows:

Algorithm EnRoot

Step 1

Alice and *Bob* choose a large finite field $\mathbb{F}_q$, and positive integers k, s_i and t_i, $i = 1, \ldots, k$. This information is *public*.

Step 2

Alice puts $e_1 = 1$ and selects a random element $\vartheta \in \mathbb{F}_q$ and $k-1$ exponents $e_2, \ldots, e_k \in \mathbb{Z}/(q-1)$, which are all *secret*.

Step 3

Alice selects k random polynomials $h_i \in \mathbb{F}_q[X_1, \ldots, X_k]$ of degree at most $q-1$, containing at most $t_i - 1$ monomials, and makes the polynomials

$$f_i(X_1, \ldots, X_k) = h_i(X_1, \ldots, X_k) - h_i(a_1, \ldots, a_k), \qquad i = 1, \ldots, k,$$

public, where

$$a_i = \vartheta^{e_i}, \qquad i = 1, \ldots, k.$$

Step 4

To send a message $m \in \mathbb{F}_q$, *Bob* selects k random polynomials

$$g_i \in \mathbb{F}_q[X_1, \ldots, X_k], \qquad i = 1, \ldots, k,$$

of degree at most $q-1$, with each containing at most s_i monomials and having non-zero constant coefficients. *Bob* then computes the reduction Ψ of the polynomial $f_1 g_1 + \ldots + f_k g_k$ modulo the ideal generated by

$$X_1^q - X_1, \ldots, X_k^q - X_k,$$

and sends the polynomial $\Phi = m + \Psi$.

Step 5

To decrypt the message, *Alice* merely computes $\Phi(a_1, \ldots, a_k) = m$.

It is obvious that the computational cost of this algorithm is polynomial. More precisely, let us denote

$$M(r) = r \log r \log\log r.$$

It is known that the bit cost of multiplication and addition of r-bit integers as well as the bit cost of multiplication and addition over $\mathbb{F}_q$, where the prime power q is r-bits long, can be estimated by $O(M(r))$, see [1, 3, 12].

Put

$$T = \sum_{i=1}^{k} t_i, \qquad S = \sum_{i=1}^{k} s_i, \qquad R = \sum_{i=1}^{k} t_i s_i.$$

Theorem. *Let a prime power q be r-bits long. The Algorithm* ENROOT *has the following characteristics:*

- *the complexity of generating the public key, that is, the set of polynomials $f_1, \ldots, f_k$, is $O((k+r)TM(r))$ bit operations plus the cost of generating $O(krT)$ random bits;*
- *the size of the public key is $O(krT)$ bits;*
- *the complexity of encryption, that is, generating the polynomial Φ, is $O(kRM(r))$ bit operations plus the cost of generating $O(krS)$ random bits;*
- *the size of the encrypted message is $O(krR)$ bits;*
- *the complexity of decryption, that is, finding the plain text message $m \in \mathbb{F}_q$, is $O((k+r)RM(r))$ bit operations.*

Proof. First of all, we remark that the value of any monomial $X_1^{n_1} \ldots X_k^{n_k}$, with exponents $0 \le n_1, \ldots, n_k \le q-1$, can be computed at $(a_1, \ldots, a_k)$ with $O((k+r)M(r))$ bit operations, by using repeated squaring. Indeed, first of all one may compute

$$E \equiv \sum_{i=1}^{k} e_k n_k \pmod{q-1}, \quad 1 \le E \le q-1,$$

with $kM(r)$ bit operations. After this the computation of

$$a_1^{n_1} \ldots a_k^{n_k} = \vartheta^E$$

can be done with $O(rM(r))$ bit operations.

To generate ϑ and the exponents $e_2, \ldots, e_k$ we need to generate $O(rk)$ random bits.

To generate the coefficients of the polynomials $f_1, \ldots, f_k$, we need to generate $T-k$ random elements of $\mathbb{F}_q$. This requires $O(rT)$ random bits. We also need to generate $T-k$ random k-tuples $(n_1, \ldots, n_k)$, with $0 \le n_1, \ldots, n_k \le q-1$, giving the exponents of the $T-k$ non-constant monomials involved in these polynomials. This requires $O(krT)$ random bits and, as it follows from the above remark, $O((k+r)TM(r))$ bit operations.

Similar analysis applies to the cost of generating $g_1, \ldots, g_k$.

Next, the cost of computing the sum of the products $f_i g_i$, $i = 1, \ldots, k$ is $O(kRM(r))$, which consists of the cost of computing $O(R)$ products over $\mathbb{F}_q$ and the cost of computing kR sums of $O(r)$ bit integers (to compute th the exponents for each monomial in the product). The cost of reduction modulo the ideal generated by $X_1^q - X_1, \ldots, X_k^q - X_k$ involves only $O(kR)$ subtractions of $O(r)$ bit integers.

Noting that Φ contains at most $O(R)$ monomials and that each of them can be computed at $(a_1, \ldots, a_k)$ with $O((k+r)M(r))$ bit operations, we obtain the desired result. □

We remark that the implied constants in these estimates can be easily evaluated.

3 Security Considerations

One possible attack on this cryptosystem is to try to find a solution to the system of equations

$$f_i(x_1, \ldots, x_k) = 0, \qquad i = 1, \ldots, k. \tag{1}$$

All known algorithms to solve systems of polynomial equations of total degree n require (regardless of sparsity) time polynomial in n, see [6, 12], but the degree of the polynomials in (1) is very large in our settings, namely it can be of order q. Thus this attack is totally infeasible, taking into account that n is exponentially large in our setting.

Another possible attack is to guess a solution. However, one expects that a system of k sparse polynomial equations in k variables of high degree over $\mathbb{F}_q$ has few zeroes over $\mathbb{F}_q$. Thus the probability that such a random guess gives a solution is, apparently, very small. The best known estimate on this problem when $k = 1$ is given in [2, 5] and it confirms that sparse polynomials over $\mathbb{F}_q$ have very few zeros in $\mathbb{F}_q$. Thus this brute force attack should take about $0.5q^k$ trials "on average".

Of course, it is very tempting to select $k = 1$. Unfortunately it seems that in this case there are more intelligent attacks, one of which is based upon considering the difference set of the powers of monomials of the polynomial Φ.

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 $n_1 = m_1 = 0$, then $\Phi(X)$ contains st monomials $C_{ij} X^{r_{ij}}$, where

$$r_{ij} \equiv n_i + m_j \pmod{q-1}, \qquad i = 1, \ldots, t;\ j = 1, \ldots, s.$$

In particular, for any pair $j_1, j_2 = 1, \ldots, s$, we have

$$r_{ij_1} - r_{ij_2} \equiv m_{j_1} - m_{j_2} \pmod{q-1},$$

for any $i = 1, \ldots, t$.

Therefore, finding the repeated elements in the difference set

$$\Delta = \{r_{i_1 j_1} - r_{i_2 j_2} \pmod{q-1} \ :\ i_1, i_2 = 1, \ldots, t;\ j_1, j_2 = 1, \ldots, s\},$$

which is considered as a subset of the residue ring $\mathbb{Z}/(q-1)\mathbb{Z}$, may reveal some information about the polynomial g.

In addition, if $k = 1$, one may also compute the greatest common divisor of $f(X)$ with $X^q - X$. This yields a product of the linear factors of f. If f has few roots, it may be easy to find a root of this new polynomial, which will have much smaller degree than f. Although it is not clear how to do this in time that would be polynomial in the sparsity t (rather than in the degree of f, which is of order q) and $\log q$, potentially this may be a threat.

On the other hand, even for $k = 2$, these attacks seem to fail. Indeed, the first attack may help to get some information about the total set of monomials in all the polynomials $g_1, \ldots, g_k$, but does not provide any information about the individual polynomials because it is not clear which monomial comes from which product $f_i g_i$, $i = 1, \ldots, k$. In order to try all possible partitions into k groups of $s_i t_i$ monomials, $i = 1, \ldots, k$, one should examine

$$N = \frac{R!}{(s_1 t_1)! \ldots (s_k t_k)!} \tag{2}$$

combinations. In particular, in the most interesting case when all s_i are of approximately the same size and so are t_i, that is, if $s_i \sim s$, $t_i \sim t$, $i = 1, \ldots, k$, then

$$\log N \sim R \log k.$$

Thus the number N of combinations to consider grows exponentially with respect to all parameters, provided that $k \geq 2$.

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

Moreover, it may be that if the polynomials $h_1, \ldots, h_k$ contain the same monomials (or monomials which differ by the same degrees), then the cryptosystem is more secure, and it may also help to reduce the computational cost of the encryption and decryption.

We have also constructed several lattice attacks to recover the private key, but these attacks are based on lattices of dimension equal to the cardinality of the base field. They are thus completely impractical provided the size of the base field is large, as in the sample parameters below.

4 Parameter Choices and Runtimes

We have tested EnRoot with four parameter choices which provide different levels of security. In all our experiments we use $k = 3$ and work over the prime field $\mathbb{F}_q$ with $q = 2^{31} - 1$. Thus for these values of parameters the brute force attack of searching for a common root of the polynomials f_i, $i = 1, \ldots, k$, takes about 2^{92} trials.

Our implementation uses the NTL library [13] quite substantially. Replacing some of the general purpose programs of this library by some more specialized and better tuned to our applications programs should provide an essential speeding up of the process.

We tested the following combination parameters $\mathbf{s}_i = (s_{i1}, s_{i2}, s_{i3})$ and $\mathbf{t}_i = (t_{i1}, t_{i2}, t_{i3})$

$$\begin{array}{llll} \mathbf{s}_1 = (4,4,4), & \mathbf{s}_2 = (4,4,4), & \mathbf{s}_3 = (4,4,5), & \mathbf{s}_4 = (4,5,5); \\ \mathbf{t}_1 = (3,3,4), & \mathbf{t}_2 = (4,4,4), & \mathbf{t}_3 = (4,4,4), & \mathbf{t}_4 = (4,4,4). \end{array}$$

From (2) we estimate the corresponding security levels with respect to our best idea of attack

$$N_1^{ER} = 2^{57}, \qquad N_2^{ER} = 2^{70}, \qquad N_3^{ER} = 2^{76}, \qquad N_4^{ER} = 2^{82}.$$

With these parameter choices, the total time required to execute a complete cycle of loading the cryptosystem; choosing a private key; constructing a public key; encrypting a message and decrypting that message is given below

$$T_1^{ER} = 0.009 \text{ sec}, \quad T_2^{ER} = 0.010 \text{ sec}, \quad T_3^{ER} = 0.011 \text{ sec}, \quad T_4^{ER} = 0.013 \text{ sec}.$$

These times are on a 600 MHz DEC AlphaStation.

The results are compared with corresponding results for RSA (the RSA time is scaled from the runtimes announced in [8] on a 255 MHz DEC AlphaStation - these times may be compared directly to those in the previous paragraph). Note that the RSA times include only encryption and decryption, and do not include substantial key generation times (as much as 1 second!).

To estimate the level of security of RSA we use the formula

$$T = \exp\left(1.639 \ln^{1/3} M \ln^{2/3} \ln M\right)$$

from [4] for the expected complexity of factoring of an integer M by the number field sieve.

For the key lengths (in bits)

$$K_1^{RSA} = 512, \qquad K_2^{RSA} = 768, \qquad K_3^{RSA} = 1024$$

and the security levels

$$N_1^{RSA} = 2^{55} \qquad N_2^{RSA} = 2^{65} \qquad N_3^{RSA} = 2^{74}.$$

we have corresponding times

$$T_1^{RSA} = 0.004 \text{ sec}, \qquad T_2^{RSA} = 0.011 \text{ sec}, \qquad T_3^{RSA} = 0.019 \text{ sec}.$$

Moreover, the key generation time for ENROOT is several orders of magnitude faster than for RSA. Please note that the highest security level tested for RSA is the same (roughly) as our medium security level, and that the ENROOT times do include key generation!

5 Concluding Remarks

Clearly, our cryptosystem is naturally suited to private key sharing among multiple parties.

The initial set and decryption can probably be accelerated in ENROOT if one uses more sophisticate algorithms to evaluate sparse polynomials, see [11, 14].

We remark that this entire cryptosystem is based on a special case of the following problem: *Let $\mathcal{R}$ be a commutative ring with identity. Given a set of elements $f_1, \ldots f_k$ in an $\mathcal{R}$-algebra S, find an $\mathcal{R}$-algebra homomorphism $\varphi : S \to \mathcal{R}$ such that $\varphi(f_i) = 0$ for all $i = 1, \ldots, k$.*

Even more generally, the problem could be stated as: *Given a morphism of schemes $f : X \to Y$, find a section $s : Y \to X$ for f.*

One inherent weakness of our cryptosystem is its high message expansion cost. Perhaps working with noncommutative rings or rings which are not principal ideal domains will allow the possibility of more secure or more efficient implementations of the above algorithm.

Acknowledgments

The authors would like to thank Michael Larsen for a number of fruitful discussions, Michael Johnson for support and Richard Miller for help with computation.

A part of this work was done during visits by I. S. to the University of Missouri and by D. L. to Macquarie University and the University of Colorado, whose hospitality and support are gratefully acknowledged.

Work supported in part, for D. L. by the National Science Foundation and a Big 12 Faculty Fellowship from the University of Missouri and for I. S. by the Australian Research Council.

References

1. A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The design and analysis of computer algorithms*, Addison-Wesley, Reading, MA, 1975.
2. R. Canetti, J. Friedlander, S. Konyagin, M. Larsen, D. Lieman and I. E. Shparlinski, 'On the statistical properties of the Diffie–Hellman distribution', *Israel J. Math*, to appear.
3. D. G. Cantor and E. Kaltofen, 'On fast multiplication of polynomials over arbitrary algebras', *Acta Inform.*, **28** (1991), 693–701.
4. D. Coppersmith, 'Modifications to the number field sieve', *J. Cryptology*, **6** (1993), 169–180.
5. J. Friedlander, M. Larsen, D. Lieman and I. E. Shparlinski, 'On correlation of binary M-sequences', *Designs, Codes and Cryptography*, to appear.
6. 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.
7. N. Koblitz, *Algebraic aspects of cryptography*, Springer-Verlag, Berlin, 1998.

8. NTRU Cryptosystems, Inc., 'The NTRU public key cryptosystem: Operating characteristics and comparison with RSA, ElGamal, and ECC cryptosystems', `http://www.ntru.com/tutorials/operatingchar.htm`, 1998.
9. J. Patarin, Asymmetric cryptography with a hidden monomial, *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **1109** (1996), 45–60
10. J. Patarin, L. Goubin and N. Courtois, 'Improved algorithm for isomorphism of polynomials', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **1403** (1998), 184–200.
11. N. Pippenger, 'On the evaluation of powers and monomials', *SIAM J. Comp.*, **9** (1980), 230–250.
12. I. E. Shparlinski, *Finite fields: Theory and computation*, Kluwer Acad. Publ., Dordrecht, 1999.
13. V. Shoup, 'NTL: A library for doing number theory (version 3.1b)', `http://www.cs.wisc.edu/~shoup/ntl/`, 1998.
14. A. C.-C. Yao, 'On the evaluation of powers', *SIAM J. Comp.*, **5** (1976), 100–103.