Cryptanalysis of Two Sparse Polynomial Based Public Key Cryptosystems

Feng Bao[1], Robert H. Deng[1], Willi Geiselmann[2], Claus Schnorr[3], Rainer Steinwandt[2], and Hongjun Wu[1]

[1] Kent Ridge Digital Labs
21 Heng Mui Keng Terrace, Singapore 119613
{baofeng,deng,hongjun}@krdl.org.sg
[2] Institut für Algorithmen und Kognitive Systeme
Arbeitsgruppen Computeralgebra & Systemsicherheit, Prof. Dr. Th. Beth,
Universität Karlsruhe, Am Fasanengarten 5, 76 131 Karlsruhe, Germany
{geiselma,steinwan}@ira.uka.de
[3] Department of Mathematics/Computer Science
Johann Wolfgang Goethe-University, Frankfurt am Main, Germany
schnorr@cs.uni-frankfurt.de

Abstract. The application of sparse polynomials in cryptography has been studied recently. A public key encryption scheme EnRoot [4] and an identification scheme SPIFI [1] based on sparse polynomials were proposed. In this paper, we show that both of them are insecure.
The designers of SPIFI proposed the modified SPIFI [2] after Schnorr pointed out some weakness in its initial version. Unfortunately, the modified SPIFI is still insecure. The same holds for the generalization of EnRoot proposed in [2].

1 Introduction

The commonly used public key cryptosystems today are based on either the factorisation problem or the discrete logarithm [3,7,8]. Some public key cryptosystems based on polynomials have been developed recently, such as the NTRU [6] and PASS [5]. EnRoot [4] and SPIFI [1] are two public key cryptosystems based on so-called sparse polynomials. The sparse polynomials used in EnRoot and SPIFI are of very high degree (as large as $2^{31} - 2$) while most of the terms are with zero coefficients.

EnRoot is a public key encryption scheme and is based on the difficulty of finding a solution to a given system of sparse polynomial equations over certain large rings. SPIFI is an identification scheme and is based on the difficulty of finding a sparse polynomial with specified values at some given points.

In this paper, we break EnRoot and SPIFI. Without dealing with the embedded hard problems, both cryptosytems could be broken easily: the plaintext of the original EnRoot could be recovered (without knowing the private key) faster than the decryption algorithm; the private key of the original SPIFI could

K. Kim (Ed.): PKC 2001, LNCS 1992, pp. 153–164, 2001.

be determined faster than the key generation process. We also break the modified SPIFI [2] in this paper. The modified SPIFI was proposed to eliminate the weakness of the original SPIFI pointed out by Schnorr.

This paper is organized as follows. Section 2 introduces the identification scheme SPIFI and the modified version. Section 3 gives the cryptanalysis of the original SPIFI and the modified version. Section 4 describes the public key encryption scheme EnRoot. The attack against EnRoot is given in Section 5. This attack also covers the generalized EnRoot system described in [2]. Section 6 concludes this paper.

2 Description of the Identification Scheme

The original SPIFI [1] (Secure Polynomial IdentiFIcation) and the modified version [2] are introduced in Subsections 2.1 and 2.2, respectively.

2.1 The Original SPIFI

We begin with the introduction of some terminology concerning polynomials.

Definitions. Given a finite field $\mathbb{F}_q$ and a set $S \subseteq \mathbb{F}_q$, we use the following terminology:

S-polynomial: a polynomial $G(X) \in \mathbb{F}_q[X]$ is an S-polynomial if every coefficient of G belongs to S.
Essentially S-polynomial: a polynomial $G(X) \in \mathbb{F}_q[X]$ is an essentially S-polynomial if $G(X) - G(0)$ is an S-polynomial.
τ-sparse: a polynomial $G(X) \in \mathbb{F}_q[X]$ is τ-sparse if it has at most τ non-zero coefficients.

The system parameters of SPIFI include

$\mathbb{F}_q$: a large finite field (where q is a large prime integer);
k: a small positive integer;
r, s, t: three small positive integers.

Key Generation. The key generation process consists of the following steps:

1. Randomly select k elements a_i $(i = 0, 1, \ldots, k-1)$, where each $a_i \in \mathbb{F}_q$.
2. Randomly select a t-sparse {0,1}-polynomial $\varphi(X) \in \mathbb{F}_q$ of degree at most $q-1$.
3. Compute $A = -\varphi(a_0)$, and let $f(X) = \varphi(X) + A$.
4. Compute $C_i = f(a_i)$ for $i = 1, 2, \ldots, k-1$.
5. The private key is the polynomial f, and the public key consists of the values $A, a_0, a_1, \ldots, a_{k-1}$ and $C_1, C_2, \ldots, C_{k-1}$.

Verification Protocol. It consists of the following steps:

1. The verifier selects at random an s-sparse essentially {0,1}-polynomial $h(X)$ from $\mathbb{F}_q[X]$ with $h(0) = B$ and sends $h(X)$ to the prover.

2. The prover randomly selects an r-sparse {0,1}-polynomial $g(X) \in \mathbb{F}_q[X]$ of degree at most $q-1$ with $g(0)=1$.
3. The prover computes $F(X)$ and D_i $(i=1,2,\ldots,k-1)$ as

$$F(X) \equiv f(X)g(X)h(X) \mod (X^q - X)$$

$$D_j = g(a_i) \qquad i = 1,\ldots,k-1$$

   Then $F(X)$ and D_i $(i=1,\ldots,k-1)$ are sent to the verifier.
4. The verifier computes

$$E_j = h(a_j), \qquad j = 1,2,\ldots,k-1$$

   and verifies that $F(X)$ is an rst-sparse {0,1,A,B,AB}-polynomial that satisfies $F(0)=AB$, and

$$F(a_j) = C_j D_j E_j, \qquad j = 0,1,\ldots,k-1$$

   where $D_0 = E_0 = 1, C_0 = 0$.

Eligibility of F

There is a negligible chance that the constructed polynomial $F(X)$ is not a $\{0,1,A,B,AB\}$-polynomial. However, if rst is substantially smaller than q, this chance is extremely small. The sparser the polynomials are, the smaller is the chance.

Recommended Parameters

It is claimed in [1] that parameters $q = 2^{31}-1$, $r=s=t=5$, and $k=3$ guarantee a security level of 2^{90} and a fast signature.

Hard Problem

It is claimed that the identification protocol is based on the following hard problem.

> *Given $2m$ arbitrary elements $\alpha_1,\ldots,\alpha_m,\gamma_1,\ldots,\gamma_m$ and a set $S \subseteq \mathbb{F}_q$ of small cardinality, it is unfeasible to find a τ-sparse S-polynomial $G(X) \in \mathbf{F}_q[X]$ of degree $deg(G) \leq q-1$ such that $G(\alpha_j)=\gamma_j$ for $j=1,\ldots,m$, provided that q is of "medium" size relative to the choice of $m \geq 1$ and $\tau \geq 3$.*

This hard problem ensures that it is hard to compute the private key $f(X)$ from the public key (the values of $f(X)$ at some given points).

2.2 The Modified SPIFI

We introduce only the modified SPIFI over the finite field $\mathbb{F}_q$. The general modified SPIFI can be found in [2]. The modified SPIFI differs from the original one only at the key generation process and the first step of the verification protocol.

Modified Key Generation. The key generation process consists of the following steps:

1. Randomly select k elements a_i $(i = 0, 1, \ldots, k-1)$, where each $a_i \in \mathbb{F}_q$.
2. Randomly select a $\frac{t}{2}$-sparse {0,1}-polynomial $f_1(X) \in \mathbb{F}_q[X]$ of degree at most $q-1$, $f_1(0) = 1$.
3. Randomly select a $\frac{t}{2}$-sparse {0,1}-polynomial $f_2(X) \in \mathbb{F}_q[X]$ of degree at most $q-1$, $f_2(0) = 0$, $f_2(a_0) \neq 0$ and $f_2(a_0) \neq -f_1(a_0)$.
4. Compute $A = -f_2(a_0)f_1(a_0)^{-1}$, and let $f(X) = Af_1(X) + f_2(X)$. Then $f(X)$ is a t-sparse $\{0, 1, A\}$-polynomial with $f(0) = A$ and $f(a_0) = 0$.
5. Compute $C_i = f(a_i)$ for $i = 1, 2, \ldots, k-1$.
6. The private key is the polynomial f, and the public key consists of the values $A, a_0, a_1, \ldots, a_{k-1}$ and $C_1, C_2, \ldots, C_{k-1}$.

Only the first step of the verification protocol is modified. It is modified as:

> The verifier selects at random an s-sparse $\{0, 1, B\}$-polynomial $h(X) \in \mathbb{F}_q[X]$ with $h(0) = B$ and sends $h(X)$ to the prover.

Recommended Parameters

It is claimed in [2] that the parameters $q = 2^{31} - 1$, $r = s = t = 5$, and $k = 3$ guarantee a security level of 2^{90}. These recommended parameters are the same as those in the original SPIFI.

3 Cryptanalysis of the Original SPIFI

We give two attacks to break the original SPIFI. One attack is to recover the private key $f(X)$ from $F(X)$ (part of the identification transcript), another one is to forge the identification transcript without knowing the secret key. They are illustrated in Subsection 3.1 and 3.2. We show that the modified SPIFI is still insecure in Subsection 3.3.

3.1 Recover the Private Key

In this subsection, we give an attack to recover the private key $f(X)$ from the identification transcript $F(X)$. Our attack involves only computations over $\mathbb{F}_q$. The direct computation over $\mathbb{F}_q[X]$ is too heavy for extremely large values of q and is thus not used in our attack.

The attack begins with recovering the polynomial $f(X)g(X)$ from $F(X)$. Let the polynomial $h'(X) = h(X) - B$, i.e., $h'(X)$ is a {0,1}-polynomial. Then the following relationship holds:

$$F(X) = f(X)g(X)h(X) = f(X)g(X)h'(X) + f(X)g(X) \cdot B.$$

Clearly, the product $f(X)g(X)h'(X)$ is a {0,1,A}-polynomial while $f(X)g(X) \cdot B$ is a {0,B,AB}-polynomial. It is thus straight forward to separate $Bf(X)g(X)$ from $F(X)$. The product $f(X)g(X)$ is obtained subsequently.

After knowing $f(X)g(X)$, we tend to recover $g(X)$. Denote the polynomial $f'(X) = f(X) - A$. Then

$$f(X)g(X) = f'(X)g(X) + Ag(X).$$

$f'(X)g(X)$ is a {0,1}-polynomial while $Ag(x)$ is a {0,A}-polynomial. The polynomial $g(X)$ could thus be obtained easily from $f(X)g(X)$.

We proceed to recover $f(X)$ from $f(X)g(X)$. Denote the set of all the non-zero-coefficient degrees of $f'(X)$ as $U = \{u_1, u_2, \ldots, u_t\}$. Denote the set of all the non-zero-coefficient degrees of $f(X)g(X)$ as $V = \{v_1, v_2, \ldots, v_{rt}\}$. Let two non-zero-coefficient degrees of $g(X)$ be w_1 and w_2. Then $X^{w_1}f(X)$ and $X^{w_2}f(X)$ are both in $f(X)g(X)$. Denote $U' = U + w_1$ as

$$U' = \{w_1 + u_1, w_1 + u_2, \ldots, w_1 + u_t\}$$

where $w_i + u_j$ is set as $(w_i + u_j - q + 1)$ if $w_i + u_j \geq q$. And let $U'' = U + w_2$. It is obvious that $U' \subset V$ and $U'' \subset V$, i.e., $U \subset (V - w_1)$ and $U \subset (V - w_2)$, where $v_i - w_j$ is set as $(v_i - w_j + q - 1)$ if $v_i - w_j \leq 0$. Due to the sparse nature of the problem, it is not difficult to show that $U = (V - w_1) \cap (V - w_2)$ holds with large probability. $f'(X)$ is thus determined and so is the private key $f(X) = f'(X) + A$.

The amount of computation needed in the attack to recover the private key is less than that required by the SPIFI key generation process.

3.2 Forge the Identification Transcript without the Private Key

The identification transcript of the SPIFI [1] could be forged without knowing the private key. Let Catherine be an attacker who does not know the prover's private key. She forges the identification transcript as follows:

1. Catherine receives an s-sparse essentially {0,1}-polynomial $h(X)$ $(h(0) = B)$ from the verifier.
2. Catherine gives an rt-sparse essentially {0,1,A}-polynomial $e(X) \in \mathbb{F}_q[X]$ such that $e(0) = A$ and $e(a_0) = 0$. (The detail will be discussed later in this subsection.)
3. Catherine computes $F(X) = e(X)h(X) \mod (X^q - X)$, sets $D_j = \frac{F(a_j)}{C_j E_j}$ $(E_j = h(a_j))$ for $j = 1, \ldots, k-1$, and sends $F(X)$ and $D_1, \ldots, D_{k-1}$ to the verifier.
4. The verifier checks that $F(X)$ is an rst-sparse $\{0, 1, A, B, AB\}$-polynomial with $F(0) = AB$, and $F(a_j) = C_j D_j E_j$ for $j = 1, \ldots, k-1$ and $F(a_0) = 0$.
5. Catherine successfully impersonates the prover.

In Step 2, Catherine needs to compute an rt-sparse essentially {0,1}-polynomial $e(X) \in \mathbb{F}_q[X]$ satisfying $e(0) = A$ and $e(a_0) = 0$. This problem is easy to solve as long as the value of q is not very large, such as the recommended $q = 2^{31} - 1$. Choose randomly an $(rt - 2)$-sparse essentially {0,1}-polynomial $e'(X)$. The required $e(X)$ is given as

$$e(X) = X(e'(X) + X^w) + A$$

where

$$w = \log_{a_0}\left(-\frac{A + a_0 e'(a_0)}{a_0}\right).$$

It is easy to compute the value of w over $\mathbb{F}_q$ for $q = 2^{31} - 1$.

If the value of q is too large, the attack in this subsection could not be applied. However, in that case, the SPIFI cryptosystem is based on the well-known discrete logarithm, instead of the hard problem related to the sparse polynomials. It is irrelevant for the attack how A and B appear in the polynomials. The attack applies to S-polynomials for any S.

3.3 Cryptanalysis of the Modified SPIFI

The SPIFI is modified to resist the attack in Subsection 3.1, i.e., to prevent the private key from being recovered from the identification transcript. However, our attack given in Subsection 3.2 could be applied (with only slight modification) to break the modified SPIFI, i.e., to forge the identification transcript without the private key (the details to forge the identification transcripts are ignored in this paper). Furthermore, we developed a new attack to recover the private key of the modified SPIFI. Details of this new attack are given in the following.

Recover the Private Key

The modified SPIFI still fails to hide the private key in a secure way. An extremely simple attack to recover the private key is given below.

Suppose now we obtained two identification transcripts. Denote these two polynomials in the transcripts as $F'(X)$ and $F''(X)$, where

$F'(X) \equiv f(X)g'(X)h'(X) \bmod (X^q - X)$ and

$F''(X) \equiv f(X)g''(X)h''(X) \bmod (X^q - X)$.

Let g', h', g'', h'' be randomly chosen sparse polynomials, $f(X)$ would be the only common terms of F' and F'' (with different coefficients). Thus $f(X)$ is recovered with negligible amount of computation. The probability of success of this attack would be very close to one due to the sparse nature of the involved polynomials.

Remarks.

1. To prevent the verifier from gaining the private key information by choosing certain special polynomials h, the SPIFI designers suggest to use two equivalent private keys alternatively [2]. This approach has little effect on our attack above. With at most three identification transcripts, we can recover one of the equivalent private keys and thus break the system. Generally, if r equivalent private keys are used, one of the keys could be determined with at most $r + 1$ identification transcripts.
2. The attack above can be applied directly to recover the private key in the original SPIFI. However, the attack in Subsection 3.1 requires only one identification transcript while the attack in this subsection requires two.

4 Description of the Encryption Scheme EnRoot

The sparse polynomial public key encryption scheme proposed in [4] is called EnRoot. A generalized version of EnRoot is also considered in [2]. As the generalized system can be attacked in the same as the original one proposed in [4], in the sequel we restrict our attention to the original system.
The system parameters of EnRoot include

$\mathbb{F}_q$: a large finite field (where q is a large prime integer);
k: a small positive interger;
t: $t = (t_1, t_2, \ldots, t_k)$ where each t_i is a small positive integer;
s: $s = (s_1, s_2, \ldots, s_k)$ where each s_i is a small positive integer.

Key Generation. The key generation process consists of the following steps:

1. Randomly choose k integers $e_i \in \mathbb{Z}/(q-1)\mathbb{Z}$ $(i = 1, 2, \ldots, k)$.
2. Choose a random element $\vartheta \in \mathbb{F}_q$. Let $a_i = \vartheta^{e_i}$ for $i = 1, 2, \ldots, k$.
3. Select k random polynomials $h_i \in \mathbb{F}_q[X_1, X_2, \ldots, X_k]$ $(i = 1, 2, \ldots, k)$ of degree at most $q-1$. Each h_i contains at most $t_i - 1$ monomials.
4. For $i = 1, 2, \ldots, k$ compute

$$f_i(X_1, X_2, \ldots, X_k) := h_i(X_1, X_2, \ldots, X_k) - h_i(a_1, a_2, \ldots, a_k) \qquad (1)$$

5. The secret key is $a = (a_1, a_2, \ldots, a_k)$. The public key is $f = (f_1, f_2, \ldots, f_k)$.

Encryption. To encrypt a message $m \in \mathbb{F}_q$,

1. Select k random polynomials $g_i \in \mathbb{F}_q[X_1, X_2, \ldots, X_k]$ $(i = 1, 2, \ldots, k)$. Each g_i contains at most s_i monomials and has a non-zero constant coefficient.
2. Compute the reduction Ψ of the polynomial $f_1 g_1 + f_2 g_2 + \ldots + f_k g_k$ modulo the ideal generated by $(X_1^q - X_1, \ldots, X_k^q - X_k)$.
3. The message m is encrypted as the polynomial $\Phi = m + \Psi$.

Decryption. To decrypt the message, compute $m = \Phi(a_1, a_2, \ldots, a_k)$.

Recommended Parameters. It is claimed in [4] that the parameter values $q = 2^{31} - 1$, $k = 3$, $s = t = (4, 4, 4)$ guarantee a security level of 2^{70}.

Moreover, it is claimed in [4] that EnRoot is based on the following hard problem:

> *Given a system of sparse polynomial equations of high degree over certain large rings, it is hard to find a solution to this system.*

This means that the secret key $a = (a_1, a_2, \ldots, a_k)$ cannot be recovered from the public key $f = (f_1, f_2, \ldots, f_k)$, where a is a common root of those polynomials f_i $(i = 1, 2, \ldots, k)$. However, as is shown in the next section, EnRoot can be broken without knowing the private key.

5 Decryption without Secret Key

First we note, that for decrypting a ciphertext

$$c = m + \sum_{i=1}^{k} f_i \cdot g_i \mod (X_1^q - X_1, \ldots, X_k^q - X_k)$$

it is sufficient to reconstruct the constant coefficients $g_i(0)$ of Bob's polynomials g_i: The absolute coefficient of c is

$$c(0) = m + \sum_{i=0}^{k} f_i(0) \cdot g_i(0),$$

because of all non-constant monomials in the expression $\sum_{i=1}^{k} f_i \cdot g_i$ remaining non-constant after reduction modulo $(X_1^q - X_1, \ldots, X_k^q - X_k)$. Hence, knowing both Alice's public polynomials f_i and the constant coefficients $g_i(0)$ we can reveal the plaintext m via

$$m = c(0) - \sum_{i=0}^{k} f_i(0) \cdot g_i(0).$$

How can we find the constant coefficients $g_i(0)$? As there are $q^k - 1$ different non-constant terms (monic monomials) which are of degree $< q$ in each variable, with high probability the public polynomials f_i of Alice satisfy the condition

$$\mathrm{Terms}(f_i) \not\subseteq \bigcup_{j \neq i} \mathrm{Terms}(f_j) \quad (1 \leq i \leq k) \tag{2}$$

(here $\mathrm{Terms}(f_j)$ denotes the set of terms occuring in f_j with non-zero coefficient). In other words, for each f_i we can find a term

$$X^{\mu_i} := \prod_{j=1}^{k} {X_j}^{\mu_{ij}} \in \mathrm{Terms}(f_i)$$

such that X^{μ_i} does not occur in any polynomial f_j with $j \neq i$. Now let $i \in \{1, \ldots, k\}$ be arbitrary but fixed, and denote by a_{μ_i} the (non-zero) coefficient of X^{μ_i} in f_i. Then the coefficient of the term X^{μ_i} in the ciphertext $c = \sum_{i=1}^{k} f_i \cdot g_i$ is with high probability equal to $g_i(0) \cdot a_{\mu_i}$: If this coefficient were different from a_{μ_i} we had

$$X^{\alpha} \cdot X^{\beta} \equiv X^{\mu_i} \mod (X_1^q - X_1, \ldots, X_k^q - X_k)$$

for some $(X^{\alpha}, X^{\beta}) \in \mathrm{Terms}(f_j) \times \mathrm{Terms}(g_j)$ with $j \in \{1, \ldots, k\}$. However, the non-constant terms of the (sparse) polynomials f_j, g_j were chosen at random, and as there are $q^k - 1$ possible non-zero exponent vectors the chance of this to happen is negligible. So in practice we can reveal $g_i(0)$ by simply reading off the coefficient of X^{μ_i} in the ciphertext, followed by dividing this coefficient by a_{μ_i}.

To illustrate this astonishingly simple attack let us consider an example with the parameters $q = 2^{31} - 1$ (a prime number), $k = 3$, $s_1 = t_1 = \ldots = s_3 = t_3 = 4$ considered in [4]:

Example 1. By means of the `randpoly` function of the computer algebra system *Maple V Realease 4* we derive the following public polynomials for Alice—the secret common zero is

$$(x_1, x_2, x_3) := (723264497, 1295378210, 230009212) :$$

$$\begin{aligned}
f_1 &= 349502340 \cdot X_1^{809446137} \cdot X_2^{956143141} \cdot X_3^{1600225079} + \\
&\quad 313871617 \cdot X_1^{779070143} \cdot X_2^{727160601} \cdot X_3^{1344053701} + \\
&\quad 1715097824 \cdot X_1^{1172854581} \cdot X_2^{559420076} \cdot X_3^{439722691} + \\
&\quad 116222600 \\
f_2 &= 200180663 \cdot X_1^{1823184387} \cdot X_2^{1504204554} \cdot X_3^{472267093} + \\
&\quad 1678471703 \cdot X_1^{759656320} \cdot X_2^{273015567} \cdot X_3^{1022563056} + \\
&\quad 10188499 \cdot X_1^{92552998} \cdot X_2^{1050663882} \cdot X_3^{371683973} + \\
&\quad 1218489385 \\
f_3 &= 942412531 \cdot X_1^{1346986246} \cdot X_2^{1330841188} \cdot X_3^{1657353576} + \\
&\quad 539695881 \cdot X_1^{332026853} \cdot X_2^{273278370} \cdot X_3^{1260893325} + \\
&\quad 1786359577 \cdot X_1^{900024634} \cdot X_2^{592601620} \cdot X_3^{182312333} + \\
&\quad 931260911
\end{aligned}$$

For our attack we use the following terms (any other combination is possible as well):

$$\begin{aligned}
X^{\mu_1} &:= X_1^{809446137} \cdot X_2^{956143141} \cdot X_3^{1600225079} \\
X^{\mu_2} &:= X_1^{1823184387} \cdot X_2^{1504204554} \cdot X_3^{472267093} \\
X^{\mu_3} &:= X_1^{1346986246} \cdot X_2^{1330841188} \cdot X_3^{1657353576}
\end{aligned}$$

Next, we encrypt the plaintext message $1234567890 \in \mathbb{F}_{2^{31}-1}$ with the polynomials g_i below (these were also created by means of *Maple V*'s `randpoly` function):

$$\begin{aligned}
g_1 &= 757504042 \cdot X_1^{1902769822} \cdot X_2^{1021006850} \cdot X_3^{1121824348} + \\
&\quad 1024142914 \cdot X_1^{522845576} \cdot X_2^{176006881} \cdot X_3^{1459236022} + \\
&\quad 1452872129 \cdot X_1^{227645716} \cdot X_2^{24530405} \cdot X_3^{1104197961} + \\
&\quad 1655562558 \\
g_2 &= 199017912 \cdot X_1^{2009873524} \cdot X_2^{590749267} \cdot X_3^{1358354210} + \\
&\quad 394475909 \cdot X_1^{1337441105} \cdot X_2^{915805516} \cdot X_3^{971137190} + \\
&\quad 497731252 \cdot X_1^{533944316} \cdot X_2^{641808520} \cdot X_3^{85031460} + \\
&\quad 1158943955 \\
g_3 &= 1552167047 \cdot X_1^{765919171} \cdot X_2^{2067090688} \cdot X_3^{1623699208} + \\
&\quad 140676907 \cdot X_1^{1867678520} \cdot X_2^{717664997} \cdot X_3^{703320394} + \\
&\quad 478601450 \cdot X_1^{99708880} \cdot X_2^{707867631} \cdot X_3^{2047224198} + \\
&\quad 1010407045
\end{aligned}$$

The constant coefficient of the resulting ciphertext c is $c(0) = 1881347037$. Further on, c contains—among others—the monomials $1290226445 \cdot X^{\mu_1}$, $463293963 \cdot X^{\mu_2}$, and $1216948431 \cdot X^{\mu_3}$. Using these monomials we can reveal the constant coefficients of the secret polynomials g_i as described above:

$$\begin{aligned} g_1(0) &= 1290226445 \cdot 349502340^{-1} = 1655562558 \\ g_2(0) &= 463293963 \cdot 200180663^{-1} = 1158943955 \\ g_3(0) &= 1216948431 \cdot 942412531^{-1} = 1010407045 \end{aligned}$$

Consequently, the plaintext m computes to

$$m = c(0) - f_1(0) \cdot g_1(0) - f_2(0) \cdot g_2(0) - f_3(0) \cdot g_3(0) = 1234567890.$$

5.1 Modifying EnRoot

Grant et al. also mention the following modification of their public key cryptosystem: instead of choosing the polynomials h_i at random in the key generation phase, each h_i is chosen as an $\mathbb{F}_q$-linear combination

$$h_i = \sum_{j=1}^{z} a_{i\nu_j} X^{\nu_j}$$

of a fixed set of (w. l. o. g. non-constant) terms $X^{\nu_1}, \ldots, X^{\nu_z}$. The other parts of the cryptosystem, in particular the encryption and decryption phase, remain unaltered.

In this setting the above attack does not immediately apply, because we cannot assume condition (2) to be fulfilled. However, the next section shows that this modified scheme is insecure, too.

5.2 Decryption without Secret Key in the Modified System

W. l. o. g. we may assume that the public polynomials $f_1, \ldots, f_k$ are linearly independent over $\mathbb{F}_q$.—In the contrary, it is sufficient to apply the attack described subsequently to a maximal linearly independent subset of $\{f_1, \ldots, f_k\}$, because such a subset is in particular a generating set of the ideal $(f_1, \ldots, f_k) \trianglelefteq \mathbb{F}_q[X]$.

As Bob chooses the polynomials g_i at random, analogously as in Section 5 we see that the coefficient c_{ν_j} of X^{ν_j} $(1 \leq j \leq z)$ in the ciphertext c equals $\sum_{i=1}^{k} a_{i\nu_j} \cdot g_i(0)$ with high probability. In other words, we obtain the following system of linear equations for the constant coefficients $g_i(0)$:

$$\begin{pmatrix} a_{1\nu_1} & \ldots & a_{k\nu_1} \\ \vdots & & \vdots \\ a_{1\nu_z} & \ldots & a_{k\nu_z} \end{pmatrix} \cdot \begin{pmatrix} g_1(0) \\ \vdots \\ g_k(0) \end{pmatrix} = \begin{pmatrix} c_{\nu_1} \\ \vdots \\ c_{\nu_z} \end{pmatrix} \tag{3}$$

¿From $f_1, \ldots, f_k$ being linearly independent we can conclude that also $h_1, \ldots, h_k$ are linearly independent (cf. the defining equations (1)). Therefore the coefficient

matrix on the left-hand side of Equation (3) is of rank k, and applying Gauß' algorithm to this equation yields a unique solution for $g_1(0), \ldots, g_k(0)$. Finally, as in our attack on the original EnRoot cryptosystem the plaintext computes to

$$m = c(0) - \sum_{i=0}^{k} f_i(0) \cdot g_i(0).$$

Remarks.

1. One of the referees brought up the idea that the system might be resistent to our attack, if Bob chooses the random polynomials so that some of the resulting monomials of the ciphertext have an identical exponent vector, say Bob chooses one term X^β of his polynomials in such a way that $X^{\mu_i} = X^\alpha \cdot X^\beta$, where X^{μ_i} and X^α are terms of Alice's public key.
   In this case obviously we cannot reconstruct the plaintext with the above mentioned procedure. But it is easy to detect the number of such "collisions" of monomials just by counting the number of monomials in the ciphertext. Thus in this case one of the randomly chosen terms has to be of the form $X^\beta := X^{\mu_i - \alpha} \bmod (X_1^q - X_1, \ldots, X_k^q - X_k)$ for some terms X^{μ_i}, X^α of the public key. Such a term X^β can easily be disclosed, as (with high probability) it is a term of the ciphertext, too. With the knowledge of the coefficient of X^β the above attack can be adapted appropriately and works again.
2. The secret k polynomials used in the encryption process could also be recovered easily. The attack is similar to that in Section 3.1. The details of this attack are ignored here.

6 Conclusions

In this paper we showed that the sparse polynomial based SPIFI (together with the modified SPIFI) and EnRoot (as well as its generalization) are insecure. Whether secure sparse polynomial based public key cryptosystems exist or not is still an open problem.

References

1. W. Banks, D. Lieman and I. Shparlinski, "An Identification Scheme Based on Sparse Polynomials", in Proceedings of PKC'2000, LNCS 1751, Springer-Verlag, pp. 68–74, 2000.
2. W. Banks, D. Lieman and I. Shparlinski, "Cryptographic Applications of Sparse Polynomials over Finite Rings", to appear in ICISC'2000.
3. T. ElGamal. "A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms". *IEEE Transactions on Information Theory*, **31** (1985), 469–472.
4. D. Grant, K. Krastev, D. Lieman and I. Shparlinski, "A Public Key Cryptosystem Based on Sparse Polynomials", in Proceedings of an International Conference on Coding Theory, Cryptography and Related Areas, LNCS, Springer-Verlag, pp. 114–121, 2000.

5. J. Hoffstein, D. Lieman and J.H. Silverman, "Polynomial Rings and Efficient Public Key Authentication", Proceedings of the International Workshop on Cryptographic Techniques and E-Commerce, pp. 7–19, M. Blum and C. H. Lee, eds., July 5–8, 1999, Hong Kong. At the time of writing also available at the URL `http://www.ntru.com/technology/tech.technical.htm`.
6. J. Hoffstein, J. Pipher and J.H. Silverman, "NTRU: A Ring Based Public Key System", Proceedings of ANTS III, Porland (1998), Springer-Verlag.
7. V. Miller, "Uses of Elliptic Curves in Cryptography", in *Advances in Cryptology–Crypto'85*, LNCS 218, Springer-Verlag, pp. 417–426, 1986.
8. R. L. Rivest, A. Shamir, and L. Adleman, "A method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Commun. ACM*, vol. 21, no. 2, pp. 158–164, Feb. 1978.