The Sidelnikov-Shestakov's Attack applied to the Chor-Rivest Cryptosystem

Sylvain Colin & Gaspard Férey

March 18, 2014

Contents

Abstract

In this article, we discuss about the Sidelnikov-Shestakov attack on cryptosystems based on Reed-Solomon codes. Then we describe how this algorithm can be used to improve the attack to the Chor-Rivest Cryptosystem proposed by Vaudenay [5].

1 Introduction

intro

Niederreiter and McEliece cryptosystem [3] was one of the first to use randomization in the encryption process. Indeed, in such a scheme, the encrypter uses a random error to prevent attacks on the codeword. This codeword is then decrypted using an error-correcting Reed-Solomon code. This algorithm has never gained much acceptance in the cryptographic community, but is a candidate for "post-quantum cryptography".

In 1992, Sidelnikov and Shestakov suggest an attack [4] on the public key of this cryptosystem using equivalences between private keys.

The Chor-Rivest cryptosystem [1] is a public key knapsack system which has been broken. However, it took longer to break than most, and is a very elegant use of finite fields.

The first efficient attack for the proposed parameters (i.e., $p \simeq 200$, $h \simeq 24$) has been obtained by Vaudenay in 2001, assuming h has a small factor.

1.1 Our Work

In this article, we first describe the Niederreiter and McEliece cryptosystem and present the original Sidelnikov and Shestakov's attack [4], which is also explained more clearly by Wieschebrink in Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes [6]. We try to make a mix between the simplicity of the arguments given by Wieshbrink and the generality of the article of Sidelnikov and Shestakov, who introduce a notion of infinity, not introduced by Wieshbrink, and which leads to get a better complexity.

On a second part, we focus on the Chor-Rivest cryptosystem and show some similarities between the "known g_{p^r} attack" introduced by Vaudenay and the Sidelnikov-Shestakov attack. This allows us to describe an algorithm for a "known g_{p^r} attack" which doesn't require r to be as big as in Vaudenay's article. In particular the term $p^{\sqrt{h}}$ in Vaudenay's complexity corresponding to an exhaustive research can be reduced to $p^{\log p}$ assuming we have $h \simeq \frac{p}{\log p}$ as suggested in Chor and Rivest's article and h has very small factors (around $\log p$). That last point should be verified since the scheme can only be used efficiently when h has a lot of divisors in order to be able to compute discrete logarithms for the public key.

:Prel

2 Preliminaries

2.1 A cryptosystem based on Reed-Solomon codes

2.1.1 Key generation

We study here the public-key cryptosystem introduced first by Niederreiter and McEliece [3] and based on generalized Reed-Solomon codes.

Let $\mathbb{F}_q$ be a finite field with $q = p^h$ elements and $\mathbb{F} = \mathbb{F}_q \cup \{\infty\}$, where ∞ has usual properties ($1/\infty = 0$, etc). We call G the following matrix:

$$G(\alpha_1, \ \dots \ , \alpha_n, z_1, \ \dots \ , z_n) := \begin{pmatrix} z_1\alpha_1^0 & z_2\alpha_2^0 & \cdots & z_n\alpha_n^0 \\ z_1\alpha_1^1 & z_2\alpha_2^1 & \cdots & z_n\alpha_n^1 \\ & & \ddots & \\ z_1\alpha_1^{k-1} & z_2\alpha_2^{k-1} & \cdots & z_n\alpha_n^{k-1} \end{pmatrix} \in \mathcal{M}_{\mathbb{F}}(k, n)$$

where $\alpha_i \in \mathbb{F}$ and $z_i \in \mathbb{F}_q \backslash \{0\}$ for all $i \in \{1, ..., n\}$. As in the article of Sidelnikov and Shestakov [4], if $\alpha_i = \infty$, the i-th column is defined by the vector $z_j(0, \ \dots \ , 0, 1)^T$ since the coefficient α_i^{k-1} dominates all the other coefficients α_i^j for $j < k - 1$. So, we concretely manipulate finite coefficients. The codes generated by such matrices are called Generalized Reed-Solomon (GRS) codes.

The cryptosystem is also based on a random nonsingular matrix H of size $k \times k$ with coefficients in $\mathbb{F}_q$. Finally, we call M the matrix of size $k \times n$ defined by $M = H \cdot G(\alpha_1, \ \dots \ , \alpha_n, z_1, \ \dots \ , z_n)$. It is clear that M has the following general form :

$$M := \begin{pmatrix} z_1 f_1(\alpha_1) & z_2 f_1(\alpha_2) & \cdots & z_n f_1(\alpha_n) \\ z_1 f_2(\alpha_1) & z_2 f_2(\alpha_2) & \cdots & z_n f_2(\alpha_n) \\ & & \ddots & \\ z_1 f_k(\alpha_1) & z_2 f_k(\alpha_2) & \cdots & z_n f_k(\alpha_n) \end{pmatrix}$$

where f_i is a polynomial of degree at most $k - 1$. We can notice that, if $\alpha_i = \infty$, $f_j(\alpha_i)$ is equal to the dominant coefficient of the polynomial f_j for all $1 \leq j \leq k$ because the last coefficient of the i-th column of G is equal to z_i and the others are equal to 0.

In the considered cryptosystem, the secret key consists of the set $\{\alpha_1, ..., \alpha_n\}$, the set $\{z_1, ..., z_n\}$ and the random matrix H.

The public key is given by the representation of the field $\mathbb{F}_q$ (ie. the polynomial used to define $\mathbb{F}_q$ over $\mathbb{F}_p$), the matrix M and the integer $t = \lfloor \frac{n-k}{2} \rfloor$.

2.1.2 Encryption and Decryption

The codewords are then the vectors $c = b \cdot M$ where $b \in \mathbb{F}_q^k$. So, the different codewords have necessarily the following form :

$$c = (z_i f_c(\alpha_i))_{1 \leq i \leq n}$$

where f_c is a polynomial whose degree is at most $k-1$. Once again, we can notice that, if $\alpha_i = \infty$, $f_c(\alpha_i)$ is equal to the dominant coefficient of the polynomial f_c.

Thus, given a message to send, which is actually a vector b of $\mathbb{F}_q^k$, one will have to transmit the vector $b \cdot M + e$ where e is a random vector of $\mathbb{F}_q^n$ with Hamming weight at most $t = \lfloor \frac{n-k}{2} \rfloor$. Since a GRS code can correct $\lfloor \frac{n-k}{2} \rfloor$ errors, this property remains true for M. Indeed, $b \cdot M$ remains a codeword of the GRS code, since it is the codeword obtained applying the GRS code to $b \cdot H$. So, in order to decrypt the message, we first compute $b' = b \cdot H$, finding the closest codeword from the

received message, using for example the Berlekamp-Welch algorithm. Then, we compute $b' \cdot H^{-1}$ to get the original message.

However, the original message can not be easily recovered when not knowing the GRS code used in the secret key.

2.2 Equivalence between Reed-Solomon codes

Sidelnikov and Shestakov show [4] that for all $a \in \mathbb{F}_q \backslash \{0\}$ and $b \in \mathbb{F}_q$, there exist $H_1, H_2, H_3 \in \mathcal{M}_{\mathbb{F}_q}(k,k)$ nonsingular, $(c_1, ..., c_n) \in \mathbb{F}_q \backslash \{0\}^n$ and $(d_1, ..., d_n) \in \mathbb{F}_q \backslash \{0\}^n$ such that:

$$H_1 \cdot G(a \cdot \alpha_1 + b,\ \dots\ , a \cdot \alpha_n + b, c_1 z_1,\ \dots\ , c_n z_n) = G(\alpha_1,\ \dots\ , \alpha_n, z_1,\ \dots\ , z_n) \quad (1) \quad \boxed{\texttt{eq1}}$$

$$H_2 \cdot G\left(\frac{1}{\alpha_1},\ \dots\ , \frac{1}{\alpha_n}, d_1 z_1,\ \dots\ , d_n z_n\right) = G(\alpha_1,\ \dots\ , \alpha_n, z_1,\ \dots\ , z_n) \quad (2) \quad \boxed{\texttt{eq2}}$$

$$H_3 \cdot G\left(\alpha_1,\ \dots\ , \alpha_n, a \cdot z_1,\ \dots\ , a \cdot z_n\right) = G(\alpha_1,\ \dots\ , \alpha_n, z_1,\ \dots\ , z_n) \quad (3) \quad \boxed{\texttt{eq3}}$$

This means that for any cryptosystem $M = H \cdot G(\alpha_1,\ \dots\ , \alpha_n, z_1,\ \dots\ , z_n)$, for any birationnal transformation

$$\phi : x \mapsto \frac{ax + b}{cx + d}$$

there exist a nonsingular matrix H_ϕ and a vector $(z'_1, ..., z'_n) \in \mathbb{F}_q \backslash \{0\}^n$ such that:

$$M = H_\phi \cdot G(\phi(\alpha_1),\ \dots\ , \phi(\alpha_n), z'_1,\ \dots\ , z'_n) \quad (4) \quad \boxed{\texttt{eq4}}$$

So, when M is given, it is impossible to compute the original matrices $G(\alpha_1,\ \dots\ , \alpha_n, z_1,\ \dots\ , z_n)$ and H since many pairs of such matrices lead to the same public matrix M. However, computing an equivalent pair is sufficient since it will allow to decrypt the messages as well as the original secret pair of matrices. It is the main idea of the attack.

ttack

3 The Sidelnikov-Shestakov Attack

3.1 A first important remark

By using the unique transformation ϕ that maps $(\alpha_1, \alpha_2, \alpha_3)$ to $(0, 1, \infty)$ in equation (4), we get that, for any cryptosystem $M = H \cdot G(\alpha_1, \ \dots\ , \alpha_n, z_1, \ \dots\ , z_n)$, M can be uniquely written

$$M = H' \cdot G(0, 1, \alpha'_3, \ \dots\ , \alpha'_k, \infty, \alpha'_{k+2}, \ \dots\ , \alpha'_n, z'_1, \ \dots\ , z'_n)$$

with H' nonsingular, $z'_i \neq 0$ and α_i distinct elements of $\mathbb{F}_q \backslash \{0, 1, \infty\}$. So, we can assume that $\alpha_1 = 0$, $\alpha_2 = 1$ and $\alpha_{k+1} = \infty$. Considering $a = z'^{-1}_{k+1}$ in equation (3), we can also assume that $z_{k+1} = 1$.

First, we compute the echelon form of M:

$$E(M) = \begin{pmatrix} 1 & 0 & \cdots & 0 & b_{1,k+1} & \cdots & b_{1,n} \\ 0 & 1 & \cdots & 0 & b_{2,k+1} & \cdots & b_{2,n} \\ & & \ddots & & \vdots & & \vdots \\ 0 & \cdots & 0 & 1 & b_{k,k+1} & \cdots & b_{k,n} \end{pmatrix}$$

Since the echelon form can be computed only with left multiplication of the matrix M, the k lines of $E(M)$ are codewords. As a consequence, if we call f_{b_i} the polynomial associated to the i-th row, we have :

- $\forall\ 1 \leq i \leq k, f_{b_i}(\alpha_i) = 1$
- $\forall\ 1 \leq i \neq j \leq k, f_{b_i}(\alpha_j) = 0$
- $\forall\ 1 \leq i \leq k, \forall\ k+1 \leq j \leq n,\ f_{b_i}(\alpha_j) = b_{i,j}$

So, since all the α_i are different, the polynomial f_{b_i} has $k-1$ simple roots. Since its degree is at most $k-1$, we know all its roots. As a consequence, $b_{i,j} \neq 0$ for all $1 \leq i \leq k$ and $k+1 \leq j \leq n$. Moreover, we know the general form of the polynomial f_i :

$$f_{b_i}(X) = c_{b_i} \cdot \prod_{1 \leq j \leq k, i \neq j} (X - \alpha_j) \tag{5}$$ eq5

where $c_{b_i} \in \mathbb{F}_q \backslash \{0\}$.

Since $\alpha_{k+1} = \infty$, we have $z_{k+1} \cdot f_{b_i}(\alpha_{k+1}) = b_{i,k+1} = c_{b_i} \cdot z_{k+1} = c_{b_i}$. So, we know the coefficients c_{b_i} for all the rows of the matrix $E(M)$.

3.2 Computation of the α_i

Consider now the first and second rows. Using the fact that, for all $k+2 \leq i \leq n$, $b_{1,j} \neq 0$ and $b_{2,j} \neq 0$, and the form of the polynomials f_{b_i} given in equation (5), the following holds:

$$\frac{b_{1,j}}{b_{2,j}} = \frac{z_j \cdot f_{b_1}(\alpha_j)}{z_j \cdot f_{b_2}(\alpha_j)} = \frac{c_{b_1} \cdot (\alpha_j - \alpha_2)}{c_{b_2} \cdot (\alpha_j - \alpha_1)} = \frac{c_{b_1} \cdot (\alpha_j - 1)}{c_{b_2} \cdot \alpha_j} \tag{6}$$ eq6

So, we have that:

$$\forall k+2 \leq j \leq n, \alpha_j = \frac{b_{2,j} \cdot c_{b_1}}{b_{2,j} \cdot c_{b_1} - b_{1,j} \cdot c_{b_2}} \tag{7}$$ eq7

Consider now the same equation as in equation (6), replacing the second row by the i-th row where $3 \leq i \leq k$. We obtain:

$$\frac{b_{1,j}}{b_{i,j}} = \frac{z_j \cdot f_{b_1}(\alpha_j)}{z_j \cdot f_{b_i}(\alpha_j)} = \frac{c_{b_1} \cdot (\alpha_j - \alpha_i)}{c_{b_i} \cdot (\alpha_j - \alpha_1)} = \frac{c_{b_1} \cdot (\alpha_j - 1)}{c_{b_2} \cdot (\alpha_j - \alpha_i)} \tag{8}$$ eq8

Then, we get, considering for instance $j = k + 2$:

$$\forall 3 \leq i \leq k, \alpha_i = \alpha_{k+2} - \frac{b_{i,k+2}}{b_{1,k+2}} \cdot \frac{c_{b_1}}{c_{b_2}} \cdot (\alpha_{k+2} - 1) \tag{9}$$ eq9

So, we now found an equivalent set of α_i for the GRS code.

The next step is to find another set of α_i with no inifinite coefficient. The idea is to find an element α different from all the α_i we computed. We then apply the birationnal transformation

$$\phi : x \mapsto \frac{1}{x - \alpha}$$

Using equation (4), we get that there exist a nonsingular matrix H'' and a vector $(z''_1, ..., z''_n) \in \mathbb{F}_q \backslash \{0\}^n$ such that:

$$M = H'' \cdot G(\phi(\alpha_1), \ ... \ , \phi(\alpha_n), z''_1, \ ... \ , z''_n)$$

So, we have an equivalent set of α_i in which all the elements are finite.

3.3 Computation of the z_i

It remains to find the coefficients z''_i. To simplify the notations, we assume that the equivalent matrix we are looking for is exactly the matrix used to generate the code. So, $z''_i = z_i$. Once again, considering $a = z_{k+1}^{-1}$ in equation (3), we can also assume that $z_{k+1} = 1$.

First, note that equation (5) remains true but, since we applied a transformation on the set of α_i, the polynomials are not the same than in the previous part. In particular, we don't know the coefficients c_{b_i} anymore.

Let L_i be the polynomial defined by:

$$L_i(X) = \prod_{1 \leq j \leq k, i \neq j} (X - \alpha_j) = \frac{1}{c_{b_i}} \cdot f_{b_i}(X) \tag{10}$$ eq10

Since we know the values of the α_i, we can compute these polynomials.

We know that, for all $1 \leq i \leq k$ and $1 \leq j \leq n$,

$$b_{i,j} = z_j \cdot c_{b_i} \cdot L_i(\alpha_j) \tag{11}$$ eq11

Considering, for $1 \leq i \leq k$, $j = k + 1$ and $j = i$ in equation (11), we get that $b_{i,k+1} = z_{k+1} \cdot c_{b_i} \cdot L_i(\alpha_{k+1}) = c_{b_i} \cdot L_i(\alpha_{k+1})$ and $1 = b_{i,i} = z_i \cdot c_{b_i} \cdot L_i(\alpha_i)$. So, making the quotient of the two equalities, we obtain:

$$\forall 1 \leq i \leq k, z_i = \frac{L_i(\alpha_{k+1})}{b_{i,k+1} \cdot L_i(\alpha_i)} \tag{12}$$ eq12

Considering now, for $k + 2 \leq j \leq n$, $i = 1$ and $j = k + 1$ in equation (11), we get that $b_{1,j} = z_j \cdot c_{b_1} \cdot L_1(\alpha_j)$ and $b_{1,k+1} = z_{k+1} \cdot c_{b_1} \cdot L_1(\alpha_{k+1}) = c_{b_1} \cdot L_1(\alpha_{k+1})$. So, making the quotient of the two equalities, we obtain:

$$\forall k + 2 \leq j \leq n, z_j = \frac{b_{1,j}}{b_{1,k+1}} \cdot \frac{L_1(\alpha_{k+1})}{L_1(\alpha_j)} \tag{13}$$ eq12

So, we managed to compute the set $\{z_1, \ ... \ , z_n\}$ and we now know completely an equivalent GRS code G'. In order to compute the matrix H' corresponding to the equivalent code we found, we consider G_k and M_k the matrices formed by the k first columns of G' and M. We then have:

$$H' = M_k \cdot G_k^{-1}$$

So, we found a pair of matrices (G', H') equivalent to the original private key.

3.4 Some remarks about the attack

4 Application to the Chor-Rivest Cryptosystem

crypt

4.1 Notations

From now on, p is a power of a prime and $q := p^h$ with h integer. We call $\mathbb{F}_n := \mathrm{GF}(n)$ the finite field of order n. We often refer in this article to $\mathbb{F}_p$ and $\mathbb{F}_{p^r}$ (with r a divisor of h) both sub-fields of $\mathbb{F}_q$. The elements of $\mathbb{F}_p$ are $\{\alpha_0, ..., \alpha_{p-1}\}$.

When $n \in \mathbb{N}$, we define the weight of n in base p as follow

$$w_p : u = \sum_{i=0}^{d} u_i p^i \longmapsto \sum_{i=0}^{d} u_i \qquad \text{where } \ 0 \leq u_i \leq p-1$$

4.2 The Chor-Rivest Cryptosystem

The Chor-Rivest cryptosystem is a public key knapsack scheme using the

The description of the field $\mathbb{F}_q$ is public. The secret key consist of

- an element $t \in \mathbb{F}_q$ with algebraic degree h.
- a generator g of $\mathbb{F}_q^*$.
- an integer $0 \leq d < q$.
- a permutation π of $\{0, ..., p-1\}$.

Public keys consist of all

$$c_i := d + \log_g(t + \alpha_{\pi(i)}) \mod q - 1$$

To be encrypted, a message should first be encoded into a bitstring $m = [m_0 ... m_{p-1}]$ of length p and weight $h : \sum_i m_i = h$. The ciphertext is

$$E(M) := \sum_{i=0}^{p-1} m_i c_i \mod q - 1$$

To decipher this message, we compute

$$g^{E(M)-hd} = \prod_i \left(t + \alpha_{pi(i)}\right)^{m_i}$$

whose factorization leads to the message m.

To be able to compute efficiently the discrete logarithm, h should have a lot of small factors. Besides the density $d := p/\log_2 q$ which characterize subset-sum problems should stay close to 1, otherwise the problem could be attacked using lattice reduction algorithms.This allow us to suppose that

$$h = \Theta\left(\frac{p}{\log_2 p}\right)$$

4.2.1 Link with Reed-Solomon codes

Trying to attack this cryptosystem show some relations between this problem and the previous one studied in section 2. In particular we have the following theorem.

Theorem 1. *Let $2 \leq k \leq p-2$. Suppose there exists $(Q_i)_{0\leq i\leq k-1}$ k polynomials of $\mathbb{F}_p[X]$ linearly independent with degree smaller than $k-1$. Suppose the evaluations $m_{i,j} := Q_i(\alpha_{\pi(j)})$ is known for all i and j. Then the permutation π can be recovered in polynomial time using a Sidelnikov-Shestakov attack on the matrix $M = (m_{i,j})_{i,j} \in \mathcal{M}_{k,p}(\mathbb{F}_p)$.*

Proof. We suppose here that one of the Q_i has a degree exactly k. Then we write the square non singular matrix $H = (h_{i,j}) \in \mathcal{M}_k(\mathbb{F}_p)$ of the coefficients of the Q_i

$$Q_i(X) = \sum_{j=0}^{k-1} h_{i,j} X^j$$

If we still consider

$$G_k := \left(\alpha_{\pi(j)}^i\right)_{0\leq i<k, 0\leq j\leq p-1} \in \mathcal{M}_{k,p}(\mathbb{F}_p)$$

We have the equality

$$H \cdot G_k = M$$

with H non singular and since $k \leq p-2$, this is exactly the public key of a cryptosystem based on the Reed-Solomon codes described in Section 2.

$$\begin{pmatrix} 1 & \cdots & 1 & \cdots & 1 \\ \alpha_1 & \cdots & \alpha_j & \cdots & \alpha_p \\ \vdots & & \vdots & & \vdots \\ \alpha_1^{k-1} & \cdots & \alpha_j^{k-1} & \cdots & \alpha_p^{k-1} \end{pmatrix} =: G_k$$

$$H := \begin{pmatrix} - & Q_1 & - \\ & \vdots & \\ - & Q_i & - \\ & \vdots & \\ - & Q_k & - \end{pmatrix} \quad \begin{pmatrix} Q_1(\alpha_1) & \cdots & Q_1(\alpha_p) \\ \vdots & \vdots & \vdots \\ Q_i(\alpha_1) & m_{i,j} & Q_i(\alpha_p) \\ \vdots & \vdots & \vdots \\ Q_k(\alpha_1) & \cdots & Q_k(\alpha_p) \end{pmatrix} =: M$$

□

So a possible way to attack the Chor-Rivest cryptosystem would be to find the evaluations of enough small degree polynomials in the $\alpha_{\pi(i)}$.

4.3 A First Attack using Reed-Solomon codes

When we attack this cryptosystem, we can consider a generator $g_0 = g^u$ with u unknown and $\gcd(u, q-1) = 1$ we then have

$$g_0^{c_i} = \left(g^d \left(t + \alpha_{\pi(i)}\right)\right)^u = \left(A + \alpha_{\pi(i)} \cdot B\right)^u$$

The unknown quantities, which can then be considered as an equivalent secret key, are

- $A \in \mathbb{F}_q$.
- $B \in \mathbb{F}_q$ such that $t = A \cdot B^{-1}$ has algebraic degree h.
- $0 < u < q-1$ prime with $q-1$.
- the permutation π of $\{0, ..., p-1\}$.

and public key consists in all the

$$d_i := \left(A + \alpha_{\pi(i)} \cdot B\right)^u \in \mathbb{F}_q$$

The ciphertext becomes

$$E'(M) := \prod_{i=0}^{p-1} d_i^{m_i} = g^{uE(M)} = B^{uh} \left(\prod_i \left(t + \alpha_{pi(i)}\right)^{c_i}\right)^u$$

Knowing u, B and h, it is easy to compute from $E'(M)$, the following quantity

$$\prod_i \left(t + \alpha_{pi(i)}\right)^{c_i}$$

which allow us to retrieve all the c_i.

We have for all j

$$g^{c_j} = g^d \cdot (t + \alpha_{\pi(j)}) = A + \alpha_{\pi(j)} \cdot B$$

where $\alpha_{\pi(j)} \in \mathbb{F}_p$ and A and B are elements of $\mathbb{F}_{p^h}$.

A naive attack would be then to try to guess at random the generator g and perform a polynomial "known g" attack. We will see that although finding the precise g is very unlikely, there is a family of generators that still allow to retrieve the permutation π and perform then a polynomial "known π" attack.

4.3.1 Small powers of g

As an attempt to guess g, we suppose that a random generator g_0 of $\mathbb{F}_q^*$ has been chosen. We know that $g_0 = g^u$ for a u integer such that $\gcd(u, p^h - 1) = 1$ and

$$g_0^{c_j} = (A + \alpha_{\pi(j)} \cdot B)^u$$

We chose now a base $(e_i)_{1 \leq i \leq h}$ of $\mathbb{F}_q$ as a $\mathbb{F}_p$-vector space. When written in this base, each coordinate of $g_0^{c_j}$ is a polynomial Q_i in the $\alpha_{\pi(j)}$.

$$g_0^{c_j} = \sum_{i=1}^{h} Q_i(\alpha_{\pi(j)}) e_i$$

where Q_i has its coefficients in $\mathbb{F}_p$. Q_i depends on A, B, u and obviously on i. However, Q_i does not depend on j.

Besides, we have

- $\deg Q_i \leq u$
- $\deg Q_i < p$ since $\alpha_{\pi(j)}^p = \alpha_{\pi(j)}$.

This means that we have access to the evaluations in the $\alpha_{\pi(j)}$ of h polynomials of degree smaller than u. According to Theorem 1, a sufficient condition for this attack to work is $u \leq h-1 \leq p-3$. The last inequality is always true for a big enough instance of the cryptosystem since h is supposed to be chosen close to $p/\log p$. However there are only $h-1$ different elements of $\mathbb{F}_{p^h}$ fulfilling the first inequality.

Considering all these acceptable generators only slightly improves the exhaustive research of g by a factor of h.

4.3.2 Wider set of acceptable generators

We can notice that if $u = u'p$,

$$g_0^{c_j} = (A + \alpha_{\pi(j)} \cdot B)^{u'p} = (A^p + \alpha_{\pi(j)} \cdot B^p)^{u'}$$

and the coordinates of this quantity are polynomials of degree u' in $\alpha_{\pi(j)}$. Actually, if u is written $u = \sum_{i=0}^{h-1} u_i p^i$ in base p, then we have

$$g_0^{c_j} = \prod_{i=0}^{h-1} (A^{p^i} + \alpha_{\pi(j)} \cdot B^{p^i})^{u_i}$$

whose coordinates are polynomials of degree $w_p(u) := \sum_{i=0}^{h-1} u_i$ in the $\alpha_{\pi(j)}$.

This means that all u such that $w_p(u) < h$ allow to retrieve the permutation and break the cryptosystem. The number of such u is

$$\left(\binom{h+1}{h-1}\right) = \binom{2h}{h-1} = \Theta\left(4^h\sqrt{h}\right)$$

This is a drastic improvement in the exhaustive research of g. However this remains quite small compared to the number $\phi(p^h - 1)$ of different generators in $\mathbb{F}_{p^h}$ which is comparable to p^h.

5 Vaudenay attack

c:Vau

The previous attack requires to find a generator of $\mathbb{F}_q$ among the elements that can be written g^u with $w_p(u) < h$. Since there are very few of such elements compared to the $\phi(q-1)$ different generators of $\mathbb{F}_q$, Vaudenay suggests [5] to consider a generator g_{p_r} of the subfield $\mathbb{F}_{p^r} \subseteq \mathbb{F}_{p^h}$. He introduces the following theorem

Theorem 2. *For any factor r of h, there exists a generator g_{p^r} of the multiplicative group of the subfield $\mathbb{F}_{p^r}$ of $\mathbb{F}_q$ and a polynomial Q with degree h/r whose coefficients are in $\mathbb{F}_{p^r}$ and such that $-t$ is a root and for all j , we have $Q(\alpha_{\pi(j)}) = g_{p^r}^{c_j}$.*

Proof. The expression of such a polynomial is given by

$$Q(X) = g_{p^r}^d \prod_{i=0}^{h/r-1} \left(X + t^{p^{ri}}\right)$$

where $g_{p^r} = g^{\frac{q-1}{p^r-1}}$. □

If we chose a base $(e_i)_{1\leq i\leq r}$ of $\mathbb{F}_{p^r}$ (seen as a $\mathbb{F}_p$-vector space), we can write the coefficients of $g_{p^r}^{c_j}$ in this base as polynomials Q_i in $\alpha_{\pi(j)}$. We get

$$g_{p^r}^{c_j} = \sum_{i=1}^{r} Q_i(\alpha_{\pi(j)}) e_i$$

with $\deg Q_i \leq h/r$. We get the evaluation of r polynomials of degree smaller than h/r.

This means that instead of searching a generator among the approximately p^h generators of $\mathbb{F}_q$, we could search only within $\mathbb{F}_{p^r}$ with r as small as possible. We notice that to be able to apply Theorem thm:link 1, we must have $h/r < r$. This yields the following theorem.

Theorem 3. *When $r > \sqrt{h}$, there exists a polynomial "known g_{p^r}" attack on the Chor-Rivest cryptosystem.*

This theorem is basically the main result from Vaudenay's article [5]. It states that to attack the Chor-Rivest cryptosystem, one can only search for the generator g_{p^r} among $\phi(p^r-1)$ (which is about p^r) possible choices instead of the exhaustive research for g (around p^h choices).

However the "known g_{p^r} attack" suggested in Vaudenay's article can be improved in two ways.

- First it is only polynomial when $r > \sqrt{h}$. We will see that using a Reed-Solomon attack, we can still retrieve π in polynomial time provided we manage to get enough linearly independent polynomials in the $\alpha_{\pi(j)}$ which is possible even for small values of r using $g_{p^r}^u$ with u small enough.
- Besides, the Reed-Solomon attack doesn't require the knowledge of all the c_j. Only $O(h)$ (or a little more when considering the attack for r small) of them. So actually only a small proportion of them is enough. This makes this attack strong.

5.1 Generating more rows...

We now describe a "known g_{p^r}" attack which allow the divisor r of h to be far smaller. We still consider the expression of elements of $\mathbb{F}_{p^r}$ in a given base $(e_i)_{1\leq i\leq r}$.

When we find g_{p^r} such that $g_{p^r}^{c_j} = Q(\alpha_{\pi(j)})$ with $\deg Q \leq h/r$, we only have r polynomials corresponding to the r different coordinates of $g_{p^r}^{c_j}$ in a certain base of $\mathbb{F}_{p^r}$. Being able to generate more polynomials would allow to chose a lower r and improve drastically the attack.

We could consider now the coordinates in the equation $Q^u(\alpha_{\pi(j)}) = g_{p^r}^{uc_j}$ for $u \in [1, p^r - 1]$. This yields again r polynomials of degree smaller than uh/r. Actually, since $P \mapsto P^p$ let the degree invariant we know that the degree of Q^u is actually at most $w_p(u)h/r$.

To generate more polynomials, we could then decide to consider all the equations $Q^u(\alpha_{\pi(j)}) = g_{p^r}^{uc_j}$ for all $u \in U_w^{(r)} := \{u \in [1, p^r - 1] | w_p(u) \leq w\}$ This yields $r \cdot |U_w^{(r)}|$ polynomials of degree at most wh/r.

Besides, assuming $w \gg r$, the number of such polynomials can be up to

$$|U_w^{(r)}| = \left(\binom{r+1}{w}\right) = \binom{r+w+1}{w} = \binom{r+w+1}{r+1} = \Theta\left(\frac{w^{r+1}}{(r+1)!}\right).$$

This means that we can choose r far smaller than $\sqrt{h}$. Indeed, even if we consider polynomials of higher degree (wh/r), the number of such polynomials we can get is far greater than r and we can hope that if we choose w high enough, we can get enough different polynomials to use a Sidelnikov-Shestakov attack.

Unfortunately, these polynomials are strongly linearly dependent. For example, the coordinates of $g_{p^r}^{pc_j}$ are linearly dependent on the coordinates of $g_{p^r}^{c_j}$. Indeed, when decomposed in a normal base of $\mathbb{F}_{p^r}$, these two sets of vectors of coordinates only differs by a rotation.

6 Simulations and expected complexity

Simul

In order to demonstrate the efficiency of the "known g_p" attack algorithm presented in Section 5 with small r, we run simulations on the number of linearly independent lines that can be obtained from the coordinates of $g_{p^r}^{uc_j}$.

6.1 First examples

For example, see Example A.1 in appendix, we try $h = 24$ and $r = 3 < \sqrt{h}$. As a result, we manage to obtain $11 \times 3 = 33$ lines of coordinates linearly independent (the simulation is the verification of this independence). This would allow the attack to retrieve the permutation π using the attack on the cryptosystem based on Reed-Solomon codes.

This proves that it is possible to duplicate the number of lines at the expense of the degree of the polynomials considered. Here, the polynomials considered are of degree 32 instead of 8 but we manage to generate 33 linearly independent lines instead of only 3. Therefore the condition $r \geq \sqrt{h}$ is not absolutely necessary and we can hope to get a (far) smaller lower bound on r.

In the Example A.1 (see appendix), we chose $h = 27$, $p = 53$ and we manage to get 46 linearly independent lines using $r = 3$ only. An exhaustive research for g_{p^3} would require circa 149.000 verifications whereas using $r = 9 > \sqrt{h}$, we would need at least $3.3 \cdot 10^{15}$ steps, which is out of most computer range.

6.2 Number of linearly independent polynomials

On several simulations for different values for p and h, see Results B in appendix, we choose, each time, the smallest value for r that allow an attack on the system. We present in the appendix the maximum weight w chosen for the exponents u_i of g_{p^r} and the number of linearly independent rows these $g_{p^r}^{u_i}$ allow to generate (should be $w \cdot \frac{h}{r} + 1$).

Let r be a divisor of h, we call u_w the number of linearly independent rows of coordinates we can get using the coordinates of $g_{p^r}^{uc_j}$ for all u such that $w_p(u) \leq w$.

We have $u_0 = 1$ obviously because using $g_{p^r}^0$, we only get one row filled with 1.

We also have $u_1 = r + 1$ because we can only get r more lines using g_{p^r} corresponding to the r different coordinates of $g_{p^r}^{c_j}$ in $\mathbb{F}_{p^r}$. The coordinates given by the generator $g_{p^r}^{p^i}$ are the same as those given by g_{p^r} in a normal base. This mean that whatever the base chosen, they are at least linearly dependent from each other and so using g_{p^r} doesn't allow to generate more useful polynomials.

6.3 Postulate

We notice experimentally the following behavior

Postulate 4. *We have for $r > 2$*

$$u_w = \min\left(\binom{w+r}{r}, w\frac{h}{r} + 1, p\right)$$

This behavior has been observed for the following range of parameters

r	w	h/r
2	1-17	1-2
3	1-17	1-17, 30
4	1-17	1-17, 30
5	1-17	1-17, 30

For $r = 2$ and $h/r \geq 3$, we notice a different behavior...

6.4 Optimum choice for r

We suppose that we have found r allowing to attack a Chor-Rivest cryptosystem. Being able to perform a Sidelnikov-Sjestakov attack imposes that there exists w such that

$$\begin{aligned} wh/r + 1 &\leq p - 2 \\ wh/r + 1 &\leq u_w \end{aligned}$$

We choose the biggest w verifying the first condition to have the weakest condition on r : $w = \lfloor r\frac{p-3}{h} \rfloor$. Besides we know that h is suppose to be chosen around $\frac{p}{\log p}$. This means that we have $w = \Theta(r \log p) \geq Cr \log p$.

We suppose that we remain in the domain $w \leq h/r$ and that the postulate in the Section 6 is true. The second condition is then easily equivalent to

$$\upsilon(w) = \binom{w+r}{r} \geq w\frac{h}{r} + 1$$

and since

$$\binom{w+r}{r} \geq \frac{w^r}{r!} \quad \text{and} \quad p \geq w\frac{h}{r} + 1$$

a sufficient condition would be

$$\begin{aligned} & \frac{w^r}{r!} && \geq && p \\ \Leftarrow \quad & Cr^r(\log p)^r && \geq && p \cdot r! \\ \Leftarrow \quad & (e \log p)^r && \geq && \frac{p}{C\sqrt{2\pi r}} \geq C'p \\ \Leftarrow \quad & r && \geq && \frac{\log p + C''}{\log\log p + 1} \sim \frac{\log p}{\log\log p} \end{aligned}$$

This mean that the exhaustive research can be done in time $O\left(p^{\frac{\log p}{\log\log p}}\right)$, far faster than $O\left(p^{\sqrt{\frac{p}{\log p}}}\right)$.

The condition $rw \leq h$ yields an upper bound on the r.

$$C'r^2 \leq \frac{p}{\log^2 p}$$

which is asymptotically always verified given the upper bound on r found.

7 Conclusions

We can notice that this attack on Chor-Rivest cryptosystem is only effective when h possesses a small factor.

References

[1] B. Chor and R. L. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Trans. Inform. Theory*, 34(5):901–909, 1988.

[2] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Probl. Control and Inform. Theory*, 15:19–34, 1986.

[3] V. M. Sidelnikov and S. O. Shestakov. On insecurity of cryptosystems based on generalized reed-solomon codes. *Discrete Math. Appl.*, 2(4):439–444, 1992.

[4] S. Vaudenay. Cryptanalysis of the chor–rivest cryptosystem. *Journal of Cryptology*, 14:87–100, 2001.

[5] C. Wieschebrink. Cryptanalysis of the niederreiter public key scheme based on grs subcodes. Federal Office for Information Security (BSI), Godesberger Allee 185-189, 53175 Bonn, Germany.

A Simulation examples

A.1 Example 1

Ex:1

We choose the following parameters

- $p = 197$, $h = 24$, $r = 3 < \sqrt{h}$.
- We define $\mathbb{F}_q$ as the quotient of $\mathbb{F}_p[X]$ by the polynomial

$$\begin{aligned} X^{24} &+ 192X^{23} + 152X^{22} + 25X^{21} + 75X^{20} + 67X^{19} + 92X^{18} + 23X^{17} + 45X^{16} + 97X^{15} \\ &+ 2X^{14} + 21X^{13} + 106X^{12} + 130X^{11} + 128X^{10} + 136X^9 + 195X^8 + 95X^7 + 155X^6 \\ &+ 34X^5 + 51X^4 + 180X^3 + 97X^2 + 23X + 87 \end{aligned}$$

- We choose $g := X + 2$ the private multiplicative generator.
- We compute

$$\begin{aligned} g_{p^r} &= g^{\frac{p^h-1}{p^r-1}} \\ &= 153X^{23} + 168X^{22} + 167X^{21} + 45X^{20} + 128X^{19} + 68X^{18} + 103X^{17} + 11X^{16} \\ &\quad +139X^{15} + 190X^{14} + 75X^{13} + 73X^{12} + 190X^{11} + 64X^{10} + 173X^9 + 34X^8 \\ &\quad +88X^7 + 30X^6 + 139X^5 + 146X^4 + 111X^3 + 80X^2 + 136X + 48 \end{aligned}$$

- We choose different values for d, t and π, the results remain the same.
- We choose the base $\left(g_{p^r}, g_{p^r}^{p}, g_{p^r}^{p^2}\right)$ for the $\mathbb{F}_p$-vector space $\mathbb{F}_{p^r}$ (this is however of no influence on the results).
- We choose $(u_i)_{1\leq i\leq 11} = (1, 2, p+1, 3, 2p+1, p+2, 4, p+3, 3p+1, 2p+2, 2p^2+p+1)$
  We have $w_p(u_i) \leq 4$ so the coordinates of $g_{p^r}^{u_i c_j}$ are polynomials of degree smaller than $4h/r + 1 = 33$ in the $\alpha_{\pi(j)}$.

A.2 Example 2

Ex:2

p	53
h	27
r	$3 < \sqrt{h}$
$\mathbb{F}_{p^q}$	$\mathbb{F}_p[X]/(P)$ with $$\begin{aligned} P := X^{27} &+ X^{26} + X^{25} + 6X^{24} + 15X^{23} + 13X^{22} \\ &+ 40X^{21} + 34X^{20} + 10X^{19} + 3X^{17} + 34X^{16} \\ &+ 40X^{15} + 49X^{14} + 42X^{13} + 20X^{12} + 6X^{11} \\ &+ X^{10} + 48X^9 + 35X^8 + 41X^7 + 27X^6 \\ &+ 12X^5 + 34X^4 + 38X^3 + 47X^2 + 19X + 1 \end{aligned}$$
g	$X + 2$
g_{p^r}	$$\begin{aligned} 2X^{26} &+ 20X^{25} + 38X^{24} + 32X^{23} + 15X^{22} + 28^{21} \\ &+ 39X^{20} + 26X^{19} + 39X^{18} + 36X^{17} + 21X^{16} \\ &+ 40X^{15} + 38X^{14} + 40X^{13} + 51X^{12} + 32X^{11} \\ &+ 50X^{10} + 51X^9 + 2X^8 + 48X^7 + 17X^6 \\ &+ 24X^5 + 31X^4 + 42X^3 + 6X^2 + 46X + 16 \end{aligned}$$
$(u_i)_{1 \le i \le 16}$	$\{0, 1, 2, p{+}1, 3, p{+}2, 2p{+}1, 4, p{+}3, 3p{+}1, 2p{+}2, 2p^2{+}p{+}1, 5, 4p{+}1, 3p{+}2, p^2{+}p{+}3\}$
t, d, π	Several values chosen at random
Rank	46

m:res

B Simulation results

p	h	r	$w_p(u_i)$	Number of linearly independent lines
197	24	2	w	$\frac{w(w+3)}{2} < 12w + 1$ Attack impossible.
197	24	3	4	33
197	24	4	2	13
197	24	$r \geq 6$	1	$1 + h/r$
193	36	2	w	$\frac{w(w+3)}{2} < 18w + 1$ Attack impossible.
193	36	3	6	73
193	36	4	3	28
193	36	6	2	13
193	36	$r \geq 9$	1	$1 + h/r$
251	60	2	w	$\frac{w(w+3)}{2} < 30w + 1$ Attack impossible.
251	60	3	8	161
251	60	4	4	61
251	60	5	3	37
251	60	6	2	21
251	60	$r \geq 10$	1	$1 + h/r$