Cryptanalysis of the Chor-Rivest Cryptosystem

Serge Vaudenay*

Ecole Normale Supérieure — CNRS
e-mail: Serge.Vaudenay@ens.fr

Abstract. Knapsack-based cryptosystems used to be popular in the beginning of public key cryptography before being all broken, all but the Chor-Rivest cryptosystem. In this paper, we show how to break this one with its suggested parameters: $\mathrm{GF}(p^{24})$ and $\mathrm{GF}(256^{25})$. We also give direction on possible extensions of our attack.

Recent interests about cryptosystems based on knapsacks or lattice reduction problems unearthed the problem of their security. So far, the Chor-Rivest was the only unbroken cryptosystem based on the subset sum problem [2, 3]. In this paper, we present a new attack on it which definitely breaks the system for all the proposed parameters in Chor-Rivest's final paper [3]. We also give directions to break the general problem, and related cryptosystems such as Lenstra's Powerline cryptosystem [8].

1 The Chor-Rivest Cryptosystem

We let $q = p^h$ be a power-prime (for a practical example, let $p = 197$ and $h = 24$). We consider the finite field $\mathrm{GF}(q)$ and we assume that its representation is public (*i.e.* there is a public h-degreed polynomial $P(x)$ irreducible on $\mathrm{GF}(p)$ and elements of $\mathrm{GF}(q)$ are polynomials modulo $P(x)$). We also consider a public numbering α of the subfield $\mathrm{GF}(p)$, *i.e.* $\{\alpha_0, \ldots, \alpha_{p-1}\} = \mathrm{GF}(p) \subseteq \mathrm{GF}(q)$.

Secret keys consist of

- an element $t \in \mathrm{GF}(q)$ with algebraic degree h
- a generator g of $\mathrm{GF}(q)^*$
- an integer $d \in \mathbf{Z}_{q-1}$
- a permutation π of $\{0, \ldots, p-1\}$

Public keys consist of all

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

for $i = 0, \ldots, p-1$. For this reason, the public parameters must be chosen such that the discrete logarithm is easy to calculate in $\mathrm{GF}(q)$. In the final paper, the authors suggested to use a relatively small prime power p and a

[*] Part of this work was done when the author was visiting AT&T Labs Research.

smooth power h, *i.e.* an integer with only small factors so that we can apply the Pohlig-Hellman algorithm [11].[1] Suggested parameters corresponds to the fields GF(197^{24}), GF(211^{24}), GF(243^{24}), and GF(256^{25}).

The Chor-Rivest cryptosystem works over a message space which consists of all p-bit strings with Hamming weight h. This means that the message to be encrypted must first be encoded as a bitstring $m = [m_0 \ldots m_{p-1}]$ such that $m_0 + \ldots + m_{p-1} = h$. The ciphertext space is $\mathbf{Z}_{q-1}$ and we have

$$E(m) = m_0 c_0 + \ldots + m_{p-1} c_{p-1} \bmod q - 1.$$

To decrypt the ciphertext $E(m)$, we compute

$$p(t) = g^{E(m)-hd}$$

as a polynomial in term of t over GF(p) with degree at most $h - 1$, which must be equal to

$$\prod_{m_i=1} (t + \alpha_{\pi(i)})$$

in GF(q). Thus, if we consider $\mu(x) + p(x)$ where $\mu(x)$ is the minimal polynomial of t, we must obtain the formal polynomial

$$\prod_{m_i=1} (x + \alpha_{\pi(i)})$$

whose factorization leads to m.

Although the public key generation relies on intricate finite fields computations, the decryption problem is based on the traditional subset sum problem (also more familiarly called ***knapsack problem***): given a set of pieces $c_0, \ldots, c_{p-1}$ and a target $E(m)$, find a subset of pieces so that its sum is $E(m)$. This problem is known to be hard, but the cryptosystem hides a trapdoor which enables the legitimate user to decrypt. This modifies the genericity of the problem and the security is thus open.

2 Previous Work

The Merkle-Hellman cryptosystem was the first subset-sum-based cryptosystem [10]. Although the underlying problem is NP-complete, it has surprisingly been broken by Shamir [12]. Later, many other variants have been shown insecure for any practical parameters by lattice reduction techniques (see [6] for instance). Actually, subset-sum problems can be characterized by the density parameter which is (with our notations) the ratio $d = p/\log_2 q$. When the density is far from 1 (which was the case of most of cryptosystems), the problem can efficiently

[1] This algorithm with Shanks' baby step giant step trick has a complexity of $O(h^3\sqrt{B}\log p)$ simple GF(p)-operations for computing one c_i where B is the largest prime factor of $p^h - 1$. (See Koblitz [7].) Since $p^r - 1$ is a factor of $p^h - 1$ when r is a factor of h, B is likely to be small when h only has small prime factors.

be solved by lattice reduction algorithms like the LLL algorithm [9]. The Chor-Rivest cryptosystem is an example of cryptosystem which achieves a density close to 1 (for $p = 197$ and $h = 24$, the density is 0.93). Its underlying problem has however the restriction that the subsets must have cardinality equal to h. Refinement of lattice reduction tools with this restriction have been studied by Schnorr and Hörner [13]. They showed that implementations of the Chor-Rivest cryptosystem with parameters $p = 151$ and $h = 16$ could be broken within a few days of computation on a single workstation (in 1995).

So far, the best known attack for secret key recovery is Brickell's attack which works within a complexity of $O(p^{2\sqrt{h}}h^2 \log p)$. It has been published in the final paper by Chor and Rivest [3]. This paper also includes several attempts of attacks when parts of the secret key is disclosed. In Sect. 5, we briefly review a few of them in order to show what all quantities in the secret key are for.

The Chor-Rivest cryptosystem has the unnatural property that the choice of the finite field $\mathrm{GF}(q)$ must be so that computing the discrete logarithm is easy. A variant has been proposed by Lenstra [8] which overcomes this problem. In this setting, any parameter can be chosen, but the encryption needs multiplications instead of additions. This variant has further been extended by Camion and Chabanne [1].

3 Symmetries in the Secret Key

In the Chor-Rivest cryptosystem setting, one has first to choose a random secret key, then to compute the corresponding public key. It relies on the difficulty of finding the secret key from the public key. It shall first be noticed that there are several *equivalent* secret keys, *i.e.* several keys which correspond to the same public key and thus which define the same encryption and decryption functions.

We first notice that if we replace t and g by their pth power (*i.e.* if we apply the Frobenius automorphism in $\mathrm{GF}(q)$), the public key is unchanged because

$$\log_{g^p}(t^p + \alpha_{\pi(i)}) = \frac{1}{p}\log_g((t + \alpha_{\pi(i)})^p) = \log_g(t + \alpha_{\pi(i)}).$$

Second, we can replace (t, α_π) by $(t + u, \alpha_\pi - u)$ for any $u \in \mathrm{GF}(p)$. Finally, we can replace (t, d, α_π) by $(ut, d - \log_g u, u.\alpha_\pi)$ for any $u \in \mathrm{GF}(p)$. Thus we have at least hp^2 equivalent secret keys. The Chor-Rivest problem consists of finding one of it.

Inspired by the symmetry use in the Coppersmith-Stern-Vaudenay attack against birational permutations [4], these properties may suggest that the polynomial $\prod_{i=0}^{h-1}\left(x - t^{p^i}\right)$ of whom all the equivalent t's are the roots plays a crucial role. This is actually the case as shown by the attacks in the following sections.

4 Relation to the Permuted Kernel Problem

Throughout this paper, we will use the following property of the Chor-Rivest cryptosystem.

Fact 1 *For any factor r of h, there exists a generator g_{p^r} of the multiplicative group of the subfield* GF(p^r) *of* GF(q) *and a polynomial Q with degree h/r and coefficients in* GF(p^r) *and such that $-t$ is a root and that for any i we have $Q(\alpha_{\pi(i)}) = {g_{p^r}}^{c_i}$.*

Proof. We let

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

where $g_{p^r} = \prod g^{p^{ri}}$ (g_{p^r} can be considered as the norm of g when considering the extension GF$(p^r) \subseteq$ GF(q)). We notice that we have $Q(x) \in$ GF(p^r) for any $x \in$ GF(p^r). Since $p^r > \frac{h}{r}$ we obtain that all coefficients are in GF(p^r). The property $Q(\alpha_{\pi(i)}) = {g_{p^r}}^{c_i}$ is straightforward. □

Since h/r is fairly small, it is unlikely that there exists some other (g_{p^r}, Q) solutions, and g_{p^r} is thus essentially unique. Throughout this paper we will use the notation

$$g_{q'} = g^{\frac{q-1}{q'-1}}.$$

If we consider the Vandermonde matrix

$$M = ({\alpha_i}^j)_{\substack{0 \leq i < p \\ 0 \leq j \leq h/r}}$$

and the vector $V = ({g_{p^r}}^{c_i})_{0 \leq i < p}$, we know there exists some vector X such that $M.X = V_{\pi^{-1}}$ where $V_{\pi^{-1}}$ is permuted from V through the permutation π^{-1}. By using the parity check matrix H of the code spanned by M (which is actually a Reed-Solomon code), this can be transformed into a permuted kernel problem $H.V_{\pi^{-1}} = 0$. It can be proved that all entries of H are actually in GF(p), thus this problem is in fact r simultaneous permuted kernel problems in GF(p). Actually, we can take $H = (A|I)$ where I is the identity matrix and A is the $(p - h/r - 1) \times (h/r + 1)$-matrix defined by

$$A_{i,j} = - \prod_{\substack{0 \leq k < h/r \\ k \neq j}} \frac{\alpha_{i+h/r} - \alpha_k}{\alpha_j - \alpha_k} \quad \begin{pmatrix} 1 \leq i < p - h/r \\ 0 \leq j \leq h/r \end{pmatrix}.$$

If we let V^i denotes the vector of the ith coordinates in vector V, we have

$$\forall i \quad H.V^i_{\pi^{-1}} = 0.$$

Unfortunately, there exists no known efficient algorithms for solving this problem. Since the matrix has a very special form, the author of the present paper believes it is still possible to attack the problem in this direction, which may improve the present attack.

5 Partial Key Disclosure Attacks

In this section we show that we can mount an attack when any part of the secret key is disclosed. Several such attacks have already been published in [3]. Some have been improved below and will be used in the following.

Known t Attack. If we guess that $\pi(0) = i$ and $\pi(1) = j$ (because of the symmetry in the secret key, we know that an arbitrary choice of (i, j) will work), we can compute $\log(t + \alpha_i)$ and $\log(t + \alpha_j)$ then solve the equations

$$c_0 = d + \frac{\log(t + \alpha_i)}{\log g}$$
$$c_1 = d + \frac{\log(t + \alpha_j)}{\log g}$$

with unknowns d and $\log g$.[2]

Known g Attack. If we guess that $\pi(0) = i$ and $\pi(1) = j$ (because of the symmetry in the secret key, we know that an arbitrary choice of (i, j) will work), we can compute

$$g^{c_0 - c_1} = \frac{t + \alpha_i}{t + \alpha_j}$$

then solve t.[3]

Known π Attack. We find a linear combination with the form

$$\sum_{i=1}^{p-1} x_i(c_i - c_0) = 0$$

with relatively small integral coefficients x_i's. This can be performed through the LLL algorithm [9]. We can expect that $|x_i| \leq B$ with $B \approx p^{\frac{h}{p-1}}$. Exponentiating this we get some equation

$$\prod_{i \in I} (t + \alpha_{\pi(i)})^{x_i} = \prod_{j \in J} (t + \alpha_{\pi(j)})^{-x_j}$$

with non-negative small powers, which is a polynomial equation with low degree which can be solved efficiently.[4]

Brickell's attack with nothing known consists of finding a similar equation but with a limited number ℓ of $\alpha_{\pi(i)}$ and then exhaustively finding for those $\pi(i)$'s. There is a tradeoff on ℓ: the LLL algorithm may product x_i's smaller than $B = p^{\frac{h}{\ell}}$, the root finding algorithm requires $O(B^2 h \log p)$ GF(p)-operations and the exhaustive search requires $O(p^\ell)$ trials. (For more details and better analysis, see [3].)

Known g_{p^r} and π Attack. Since we will use this attack several times in the following, we put it here. We can interpolate the $Q(x)$ polynomial of Fact 1 with $h/r + 1$ pairs $(\alpha_{\pi(i)}, {g_{p^r}}^{c_i})$. We thus obtain a h/r-degree polynomial whose roots are conjugates of $-t$. We can thus solve it in order to get t and perform a known t attack.

[2] Another attack attributed to Goldreich was published in [3].

[3] Another attack was published in Huber [5].

[4] This attack was attributed to Odlyzko and published in [3].

6 Known g_{p^r} Attack

Here we assume we know the g_{p^r} value corresponding to a subfield GF(p^r) (see Fact 1).

Let $i_0, \dots, i_{h/r}$ be $h/r+1$ pairwise distinct indices from 0 to $p-1$. Because of Fact 1 we can interpolate $Q(x)$ on all $\alpha_{\pi(i_j)}$'s, which leads to the relation

$$g_{p^r}{}^{c_i} = \sum_{j=0}^{h/r} g_{p^r}{}^{c_{i_j}} \prod_{\substack{0\leq k\leq h/r\\ k\neq j}} \frac{\alpha_{\pi(i)} - \alpha_{\pi(i_k)}}{\alpha_{\pi(i_j)} - \alpha_{\pi(i_k)}} \tag{2}$$

for $i = 0, \dots, p-1$. Actually, we can even write this as

$$g_{p^r}{}^{c_i} - g_{p^r}{}^{c_{i_0}} = \sum_{j=1}^{h/r} \left(g_{p^r}{}^{c_{i_j}} - g_{p^r}{}^{c_{i_0}}\right) \prod_{\substack{0\leq k\leq h/r\\ k\neq j}} \frac{\alpha_{\pi(i)} - \alpha_{\pi(i_k)}}{\alpha_{\pi(i_j)} - \alpha_{\pi(i_k)}}. \tag{3}$$

Because of the symmetry of π in the secret key, we can arbitrarily choose $\pi(i_1)$ and $\pi(i_2)$ (see Sect. 3).

A straightforward algorithm for finding π consists of exhaustively look for the values of $\pi(i_j)$ for $j = 0, 3, \dots, h/r$ until Equation (2) gives a consistent permutation π. It is illustrated on Fig. 1. The complexity of this method if roughly $O(hp^{h/r-1})$ computations in GF(p).

```
**Input** GF($q$) descriptors, $\alpha$ numbering, $c_0, \dots, c_{p-1}$, $r|h$, $g_{p^r}$
**Output** a secret key whose corresponding public key is $c_0, \dots, c_{p-1}$
  1. choose pairwise different $i_0, \dots, i_{h/r}$ in $\{0, \dots, p-1\}$
  2. choose different $\pi(i_1)$ and $\pi(i_2)$ arbitrarily in $\{0, \dots, p-1\}$
  3. for all the possible values of $\pi(i_0), \pi(i_2), \dots, \pi(i_{h/r})$ (i.e. all values such that
     $\pi(i_0), \dots, \pi(i_{h/r})$ are pairwise different and in the set $\{0, \dots, p-1\}$), we set
     $S = \{\pi(i_0), \dots, \pi(i_{h/r})\}$ and do the following
     (a) for all $j$ which is not in $S$, compute the right-hand term of Equation (2)
         with $\alpha_j$ instead of $\alpha_{\pi(i)}$. If it is equal to $g_{p^r}{}^{c_i}$ such that $\pi(i)$ has not been
         defined, set $\pi(i) = j$, otherwise continue loop in step 3.
     (b) perform a known $g_{p^r}$ and $\pi$ attack.
```

Fig. 1. An $O(p^{\frac{h}{r}-1})$ Known g_{p^r} Attack.

When r is large enough, there is a much better algorithm. Actually, if $h/r \leq r$ (*i.e.* $r \geq \sqrt{h}$), the coefficients in Equation (2) are the only GF(p) coefficients which write $g_{p^r}{}^{c_i} - g_{p^r}{}^{c_{i_0}}$ in the basis $g_{p^r}{}^{c_{i_0}} - g_{p^r}{}^{c_{i_0}}, \dots, g_{p^r}{}^{c_{i_{h/r}}} - g_{p^r}{}^{c_{i_0}}$. Let a^i_j be the coefficient of $g_{p^r}{}^{c_{i_j}} - g_{p^r}{}^{c_{i_0}}$ for $g_{p^r}{}^{c_i} - g_{p^r}{}^{c_{i_0}}$. We have

$$\frac{a^i_2}{a^i_1} = u\frac{\alpha_{\pi(i)} - \alpha_{\pi(i_1)}}{\alpha_{\pi(i)} - \alpha_{\pi(i_2)}} \tag{4}$$

where u is an element of GF(p) which does not depend on i. Hence, if we randomly choose i_j for $j = 0, \ldots, h/r$, we can write all ${g_{p^r}}^{c_i} - {g_{p^r}}^{c_{i_0}}$'s in the basis $({g_{p^r}}^{c_{i_0}} - {g_{p^r}}^{c_{i_0}}, \ldots, {g_{p^r}}^{c_{i_{h/r}}} - {g_{p^r}}^{c_{i_0}})$. Now if we guess the GF(p)-value of u, we obtain $\pi(i)$ from the above equation. This is a polynomial algorithm in p, h, r for getting π.

```
Input  GF($q$) descriptors, $\alpha$ numbering, $c_0, \ldots, c_{p-1}$, $r|h$, $g_{p^r}$ s.t. $r \geq \sqrt{h}$
Output  a secret key whose corresponding public key is $c_0, \ldots, c_{p-1}$
    1. choose pairwise different $i_0, \ldots, i_{h/r}$ in $\{0, \ldots, p-1\}$ and precompute the basis
       transformation matrix for the basis $({g_{p^r}}^{c_{i_0}} - {g_{p^r}}^{c_{i_0}}, \ldots, {g_{p^r}}^{c_{i_{h/r}}} - {g_{p^r}}^{c_{i_0}})$
    2. choose different $\pi(i_1)$ and $\pi(i_2)$ arbitrarily in $\{0, \ldots, p-1\}$
    3. for all possible $u$ in GF($p$), do the following
       (a) for all $i$, write ${g_{p^r}}^{c_i} - {g_{p^r}}^{c_{i_0}}$ in the basis and get $a_0^i$ and $a_1^i$. From Equation
           (4) get $\pi(i)$. If it is not consistent with other $\pi(i')$, continue loop in step
           3.
       (b) perform a known $g_{p^r}$ and $\pi$ attack.
```

Fig. 2. A Polynomial Known g_{p^r} Attack for $r \geq \sqrt{h}$.

In the rest of the paper, we show how to find g_{p^r} with a choice of r so that these known g_{p^r} attacks can be applied.

7 Test for g_{p^r}

Equation (3) means that all ${g_{p^r}}^{c_i}$'s actually stand on the same h/r-dimensional affine subspace of GF(p^r) over GF(p). Thus, if we assume that $h/r + 1 \leq r$ (*i.e.* $r \geq \sqrt{h + \frac{1}{4}} + \frac{1}{2}$), this leads to a simple test for g_{p^r}.

Fact 2 *If there exists a factor r of h such that $r \geq \sqrt{h + \frac{1}{4}} + \frac{1}{2}$ if we let g_{p^r} denotes $g^{1+p^r+\ldots+p^{h-r}}$, then all ${g_{p^r}}^{c_i}$'s stands on the same h/r-dimensional affine space when considering* GF(p^r) *as an r-dimensional* GF(p)*-affine space.*

The existence of such an r can be seen as a bad requirement for this attack, but since the parameters of the Chor-Rivest cryptosystem must make the discrete logarithm easy, we already know that h has many factors, so this hypothesis is likely to be satisfied in practical examples. Actually, h with no such factors are prime and square-prime numbers. The real issue is that r shall not be too large.

Thus there is an algorithm which can check if a candidate for g_{p^r} is good: the algorithm simply check that all ${g_{p^r}}^{c_i}$'s are affine-dependent. The algorithm has an average complexity of $O(h^3/r)$ operations in GF(p). Since there are $\varphi(p^r - 1)/r$ candidates, we can exhaustively search for g_{p^r} within a complexity of $O(h^3 p^r / r^2)$. Since r has to be within the order of $\sqrt{h}$, this attack is better than Brickell's attack provided that such an r exists. The algorithm is depicted on Fig. 3.

```
Input GF($q$) descriptors, $\alpha$ numbering, $c_0, \ldots, c_{p-1}$, $r|h$ s.t. $r \geq \sqrt{h + \frac{1}{4}} + \frac{1}{2}$
Output possible values for $g_{p^r}$
  1. choose pairwise different $i_0, \ldots, i_{h/r}$ in $\{0, \ldots, p-1\}$
  2. for any generator $g_{p^r}$ of GF($p^r$), do the following
     (a) get the equation of the affine space spanned by $({g_{p^r}}^{c_{i_0}}, \ldots, {g_{p^r}}^{c_{i_{h/r}}})$
     (b) for all other $i$, check that ${g_{p^r}}^{c_i}$ in the space. If not, continue loop in step 2.
     (c) perform the known $g_{p^r}$ attack of Fig. 2.
```

Fig. 3. An $O(p^r)$ Attack for $r \geq \sqrt{h + \frac{1}{4}} + \frac{1}{2}$.

With the parameter $h = 24$, we can take $r = 6$. We have about 2^{41} candidates for g_{p^r} so we can find it within 2^{52} elementary operations, which is feasible with modern computers.

Here we also believe we can still adapt this attack for smaller r values. The next section however gives an alternate shortcut to this issue.

8 On the Use of all the c_i's

In his paper [8], Lenstra suspected that disclosing all the c_i's in the public key was a weakness. Actually, this property enables to drastically improve the previous algorithm by using all the factors of h.

We have the following fact.

Fact 3 *Let $Q(x)$ be a polynomial over* GF(p^r) *with degree d and let e be an integer such that $1 \leq e < \frac{p-1}{d}$. We have*

$$\sum_{a \in \mathrm{GF}(p)} Q(a)^e = 0.$$

This comes from the fact that $Q(x)^e$ has a degree less than $p - 1$ and that $\sum a^i = 0$ for any $i < p - 1$. This proves the following fact.

Fact 4 *For any $1 \leq e < (p-1)r/h$ we have*

$$\sum_{i=0}^{p-1} {g_{p^r}}^{ec_i} = 0.$$

This provides a much simpler procedure to select all g_{p^r} candidates. Its main advantage is that it works in any subfield. For instance, we can consider $r = 1$

and find the only g_p such that for all $1 \leq e < (p-1)r$ we have $\sum g_{p^r}{}^{ec_i} = 0$. The average complexity of checking one candidate is $O(p)$ GF(p)-computations: it is unlikely that a wrong candidate will not be thrown by the $e = 1$ test. Hence, we can recover g_p within $O(p^2)$ simple computations.

Unfortunately, the g_{p^r} cannot be used efficiently when r is too small. We can still use g_{p^r} in smaller subfields to compute it in large ones. Our goal is to compute g_{p^r} with r large enough. Let us consider the problem of computing g_{p^r} when $r_1, \ldots, r_k$ are factors of r with the knowledge of $g_{p^{r_i}}$. Since we have $g_{p^{r_i}} = g_{p^r}^{1+p^{r_i}+\ldots+p^{r-r_i}}$, we obtain that

$$\log g_{p^r} = \frac{\log g_{p^{r_i}}}{1 + p^{r_i} + \ldots + p^{r-r_i}} \pmod{p^{r_i} - 1}. \tag{5}$$

The knowledge of all $g_{p^{r_i}}$'s thus gives the knowledge of $\log g_{p^r}$ modulo

$$\ell = \operatorname{lcm}\{p^{r_1} - 1, p^{r_2} - 1, \ldots, p^{r_k} - 1\}.$$

Thus we need only $(p^r - 1)/\ell$ trials to recover g_{p^r}. The algorithm is illustrated on Fig. 4. It is easy to see that each loop controlled in step 2 requires on average $O(pr^2)$ operations in GF(p).

```
Input  GF($q$) descriptors, $\alpha$ numbering, $c_0, \ldots, c_{p-1}$, $r_i | r | h$ and $g_{p^{r_i}}$, $i = 1, \ldots, k$
Output  set of possible $g_{p^r}$ values
    1. solve the Equation System (5) for $i = 1, \ldots, k$ and obtain that $g_{p^r} = \beta.\gamma^x$ for
       unknown $x$
    2. for $x = 0, \ldots, (p^r - 1)/\mathrm{lcm}\{p^{r_i} - 1; i = 1, \ldots, k\} - 1$ do the following
        (a) compute $\sum \beta^{ec_i}\gamma^{ec_i x}$ for $e = 1, \ldots (p-1)r/h - 1$ and if one sum is non-zero
            continue loop on step 2.
        (b) output $g_{p^r} = \beta.\gamma^x$
```

Fig. 4. Getting g_{p^r} from the $g_{p^{r_i}}$.

Thus we can define an algorithm for dedicated h's by a graph.

Definition 5. *Let G be a rooted labeled direct acyclic graph in which the root is labeled by a finite field* GF(p^r) *and such that whenever there is a $u \dashrightarrow v$ edge in G then the label $L(u)$ of u is a subfield of the label $L(v)$ of v and an extension of* GF(p). *We call G a "p-factoring DAG for* GF(p^r)*".*

To G and an integer p we associate the quantity

$$C(G) = \sum_v \frac{\#L(v) - 1}{\operatorname{lcm}\{\#L(w) - 1; v \leftarrow w\}}.$$

(By convention, lcm of an empty set is 1.) We can define an algorithm for computing g_{p^r} with complexity $O(pr^2 C(G))$. Thus, we can break the Chor-Rivest cryptosystem with parameter h which is neither prime nor a square prime within a complexity essentially

$$O\left(\min_{\substack{r|h\\ r\geq\sqrt{h}}}\ \min_{\substack{G \text{ is a } p-\text{factoring}\\ \text{DAG for } \mathrm{GF}(p^r)}} pr^2C(G)\right).$$

The corresponding algorithm is illustrated on Fig. 5.

```
Input  $\mathrm{GF}(p^h)$ descriptors, $\alpha$ numbering, $c_0, \ldots, c_{p-1}$,
Output  a possible secret key
  1. for the smallest factor $r$ of $h$ such that $r \geq \sqrt{h+\frac{1}{4}}+\frac{1}{2}$, find the $p$-factoring
     DAG with minimal $C(G)$
  2. for any $u$ in $G$ such that for all $u \leftarrow u_i$, $u_i$ has been visited, visit $u$ by doing
     the following
     (a) perform the algorithm of Fig. 4 with $\mathrm{GF}(p^r) = L(u)$ and $\mathrm{GF}(p^{r_i}) = L(u_i)$
         and obtain $g_{p^r}$
  3. perform the known $g_{p^r}$ attack of Fig. 2
```

Fig. 5. An Efficient Attack Dedicated for h.

Example 6 ($h = 25$). We can solve the $h = 25$ case with a trivial G p-factoring DAG for $\mathrm{GF}(p^5)$ which consists of two vertices labeled with $\mathrm{GF}(p)$ and $\mathrm{GF}(p^5)$. From g_{p^5} we can then apply the algorithm of Fig. 2. We have

$$C(G) = \frac{p^5-1}{p-1} + p - 1 \approx p^4$$

so the corresponding complexity is $O(p^5)$.

Example 7 ($h = 24$). Here is another dedicated attack for $h = 24$. We can choose $r = 6$ for which we have $h/r + 1 \leq r$. Recovering g_{p^6} requires firstly, $O(p)$ trials to get g_p, secondly, $O(p)$ trials to get g_{p^2} with g_p, thirdly, $O(p^2)$ trials to get g_{p^3} with g_p, finally, $O(p^2)$ trials to get g_{p^6} with g_{p^2} and g_{p^3}. The maximum number of trials is thus $O(p^2)$. Hence the complexity is $O(p^3)$ multiplications in $\mathrm{GF}(p^6)$. Actually, this attack corresponds to the p-factoring DAG for $\mathrm{GF}(p^6)$ depicted on Fig. 6. For this DAG we have

$$C(G) = \frac{p^6-1}{\mathrm{lcm}(p^2-1, p^3-1)} + \frac{p^3-1}{p-1} + \frac{p^2-1}{p-1} + p - 1$$

thus $C(G) = 78014$ for $p = 197$. We thus need about 2^{29} operations in $GF(197)$ to break the Chor-Rivest cryptosystem in $\mathrm{GF}(197^{24})$.

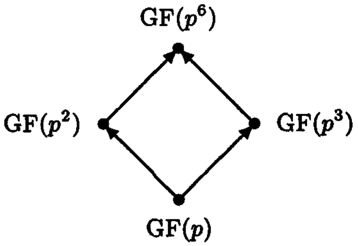

Fig. 6. A Factoring DAG for GF(p^6).

9 Generalization

In this section we generalize our attack in order to cover the GF(256^{25}) case *i.e.* when p is a power-prime: there is no reason why to restrict our attacks to finite fields which are extensions of GF(p) since we have many other subfields. For this we need to adapt the algorithm of Fig. 5 with generalized factoring DAGs, *i.e.* when the labels are not extensions of GF(p). We first state generalized version of Fact 1.

Fact 8 *Let* GF(q') *be a subfield of* GF(q) *i.e.* $q = q'^s$. *We let*

$$Q(x) = N(g^d(x+t)) \bmod (x^p - x)$$

where $N(y) = y^{\frac{q-1}{q'-1}}$. $Q(x)$ *is a polynomial such that* $Q(\alpha_{\pi(i)}) = N(g)^{c_i}$. *In addition, if we have* $\gcd(s,h) < p_0$ *where* $p_0 = q^{\frac{1}{\operatorname{lcm}(s,h)}}$ *then the degree of* $Q(x)$ *is* $\gcd(s,h)\frac{p-1}{p_0-1}$.

Proof. $Q(\alpha_{\pi(i)}) = N(g)^{c_i}$ is obvious since $\alpha_{\pi(i)}$ is a root of $x^p - x$. The useful part of this fact is the distance between the degree of $Q(x)$ and p.

We have

$$Q(x) \equiv N(g).N(x+t) \equiv N(g) \prod_{i=0}^{s-1} \left(x^{q'^i} + t^{q'^i}\right) \quad (\text{mod } (x^p - x)).$$

We notice that

$$x^i \bmod (x^p - x) = x^{(i-1) \bmod (p-1) + 1}$$

thus if we let

$$d = \sum_{i=0}^{s-1} \left(\left(q'^i - 1\right) \bmod (p-1) + 1\right)$$

the degree of $Q(x)$ is d provided that $d < p$. Let $p_0 = q^{\frac{1}{\operatorname{lcm}(s,h)}}$ and $p = p_0^g$. We have

$$d = \frac{s}{g} \sum_{i=0}^{g-1} \left((p_0^i - 1) \bmod (p_0^g - 1) + 1\right) = \frac{s}{g} \sum_{i=0}^{g-1} p_0^i = \frac{s}{g} \frac{p-1}{p_0 - 1}.$$

We further notice that $\frac{s}{g} = \gcd(s,h)$ and that $d < p$. □

As a consequence we obtain a generalized form of Fact 4.

Fact 9 *Let* $q = p^h = q'^s$ *and* $p_0 = q^{\frac{1}{\mathrm{lcm}(s,h)}}$ *be such that* $\gcd(s,h) < p_0 - 1$*. We have*

$$\sum_{i=0}^{p-1} g_{q'}{}^{ec_i} = 0$$

for any $1 \le e < \frac{p_0 - 1}{\gcd(s,h)}$.

We can thus generalize the attack of Fig. 5 whenever each $\mathrm{GF}(q^{1/s})$ label fulfill the assumption $\gcd(s,h) < p_0 - 1$ where $p_0 = q^{\frac{1}{\mathrm{lcm}(s,h)}}$.

Example 10 *($q = 256^{25}$).* The GF(16) field does not fulfill the assumption. However, the GF(256), $\mathrm{GF}(16^5)$ and $\mathrm{GF}(256^5)$ fields do. We can thus start the attack with the field GF(256) and then obtain g_{16} from g_{16^2} as illustrated by the (generalized) factoring DAG of $\mathrm{GF}(256^5)$ illustrated on Fig. 7. We have

$$C(G) = \frac{256^5 - 1}{\mathrm{lcm}(255, 16^5 - 1)} + \frac{16^5 - 1}{15} + \frac{15}{255} + 255 = 131841 + \frac{1}{17}$$

thus we need about 2^{29} GF(16)-operations to break the Chor-Rivest cryptosystem in $\mathrm{GF}(256^{25})$.

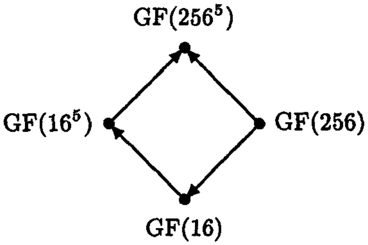

Fig. 7. A Generalized Factoring DAG for $\mathrm{GF}(256^5)$.

We believe there is no need for formalizing further generalizations in the Chor-Rivest cryptosystem context. We believe that the more we have some subfield choices of $\mathrm{GF}(q)$, the lower is the complexity of the best attack.

10 Conclusion

We have described general attack when the parameter h has a small factor r greater than $\sqrt{h+\frac{1}{4}}+\frac{1}{2}$ which has a complexity $O(h^3p^r/r^2)$. We also have solved one of Lenstra's conjectures arguing that keeping all the c_i coefficients in the public key is a weakness by exhibiting a shortcut algorithm in the previous attack.

The attack has been successfully implemented on an old laptop with the suggested parameters $\mathrm{GF}(p^{24})$ by using hand-made (inefficient) arithmetic libraries. Recovering the secret key from the public key takes about 15 minutes. But computing the public key from the secret key takes much longer...

We also generalized our attack in order to break the $\mathrm{GF}(256^{25})$ proposal. In Appendix, we even suggest an improvement of the presented attacks when h does not have a small factor r greater than $\sqrt{h+\frac{1}{4}}+\frac{1}{2}$.

In order to repair the Chor-Rivest cryptosystem, we believe that

- we must choose a finite field $\mathrm{GF}(p^h)$ where p and h are both prime;
- we must not put all the c_is in the public key.

It is then not clear how to choose the parameters in order to make the discrete logarithm problem easy, and to achieve a good knapsack density in order to thwart the Schnorr-Hörner attack.

One solution is to use Lenstra's Powerline cryptosystem, or even its recent generalization: the Fractional Powerline System (see Camion-Chabanne [1]). We however have to fulfill the two requirements above. The security in this setting is still open, but we suspect that the simultaneous permuted kernel characterization of the underlying problem may lead to a more general attack on this cryptosystem with any parameters. We highly encourage further work in this direction.

Acknowledgment

The author thanks Andrew Odlyzko for many helpful discussions and AT&T for inviting to perform those researches.

References

1. P. Camion, H. Chabanne. On the Powerline system. In *Advances in Cryptology, ICICS'97*, Beijing, China, Lectures Notes in Computer Science 1334, pp. 381–385, Springer-Verlag, 1997.
2. B. Chor, R.L. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. In *Advances in Cryptology CRYPTO'84*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science , pp. 54–65, Springer-Verlag, 1985.

3. B. Chor, R.L. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, vol. IT-34, pp. 901–909, 1988.
4. D. Coppersmith, J. Stern, S. Vaudenay. The security of the birational permutation signature schemes. *Journal of Cryptology*, vol. 10, pp. 207–221, 1997.
5. K. Huber. Specialised attack on Chor-Rivest public key cryptosystem. *Electronics Letters*, vol. 27, no. 23, pp. 2130, 1991.
6. A. Joux, J. Stern. Lattice Reduction: a Toolbox for the Cryptanalyst. To appear in *Journal of Cryptology*.
7. N. Koblitz. *A Course in Number Theory and Cryptography*, 2nd Edition, Graduate Texts in Mathematics 114, Springer-Verlag, 1994.
8. H.W. Lenstra, Jr. On the Chor-Rivest Knapsack Cryptosystem. *Journal of Cryptology*, vol. 3, pp. 149–155, 1991.
9. A.K. Lenstra, H.W. Lenstra Jr., L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, vol. 261, pp. 515–534, 1982.
10. R.C. Merkle, M. Hellman. Hiding information and signatures in trap-door knapsacks. *IEEE Transactions on Information Theory*, vol. IT-24, pp. 525–530, 1978.
11. S. Pohlig, M. Hellman. An improved algorithm for computing logarithms over GF(q) and its cryptographic significance. *IEEE Transactions on Information Theory*, vol. IT-24, pp. 106–110, 1978.
12. A. Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, Chicago, Illinois, U.S.A., pp. 145-152, IEEE, 1982.
13. C.P. Schnorr, H.H. Hörner. Attacking the Chor-Rivest Cryptosystem by improved lattice reduction. In *Advances in Cryptology EUROCRYPT'95*, Saint-Malo, France, Lectures Notes in Computer Science 921, pp. 1–12, Springer-Verlag, 1995.

A Extension of Algorithm of Fig. 2

Equation (4) is a simple way to solve the problem when $r \geq \sqrt{h}$. We still believe we can adapt the above attack for any value of r by more tricky algebraic computations.

Actually, let us consider a value r such that $\frac{h}{r} \geq r$ and $\ell = \frac{h}{r} - r$. Let e_i denotes ${g_{p^r}}^{c_{i_j}} - {g_{p^r}}^{c_{i_0}}$ for $i = 1, \ldots, h/r$. There may exist some $\sum_j u_{k,j} e_j = 0$ equations, namely ℓ of it. Hence if we write ${g_{p^r}}^{c_i} - {g_{p^r}}^{c_{i_0}} = \sum_j a^i_j e_j$, there may exist some x^i_k coefficients such that

$$a^i_j - \sum_k x^i_k u_{k,j} = \prod_{\substack{0 \leq k \leq h/r \\ k \neq j}} \frac{\alpha_{\pi(i)} - \alpha_{\pi(i_k)}}{\alpha_{\pi(i_j)} - \alpha_{\pi(i_k)}}$$

for $j = 1, \ldots, h/r$. When considering a set of n values of i, we have nh/r algebraic equations with $n(\ell + 1) - 1 + h/r$ unknowns x^i_k, $\alpha_{\pi(i_j)}$, $\alpha_{\pi(i)}$. Thus if $r > 1$ we can take n large enough as long as $p(r-1) + 1 \geq h/r$. We thus believe further algebraic tricks may leads to the solution for any $r > 1$ as long as $p + 1 \geq h/2$.