A Public Key Encryption Scheme Based on the Polynomial Reconstruction Problem

Daniel Augot and Matthieu Finiasz

INRIA, Domaine de Voluceau
F-78153 Le Chesnay CEDEX

Abstract. The *Polynomial Reconstruction* problem (PR) has been introduced in 1999 as a new hard problem. Several cryptographic primitives established on this problem have been constructed, for instance Naor and Pinkas have proposed a protocol for oblivious polynomial evaluation. Then it has been studied from the point of view of robustness, and several important properties have been discovered and proved by Kiayias and Yung. Furthermore the same authors constructed a symmetric cipher based on the PR problem. In the present paper, we use the published security results and construct a new public key encryption scheme based on the hardness of the problem of Polynomial Reconstruction. The scheme presented is the first public key encryption scheme based on this Polynomial Reconstruction problem. We also present some attacks, discuss their performances and state the size of the parameters required to reach the desired security level. In conclusion, this leads to a cryptosystem where the cost of encryption and decryption per bit is low, and where the public key is kept relatively small.

1 Introduction

It is an important goal in cryptology to find new difficult problems to design cryptographic primitives, and it is a major area of research to establish these primitives and demonstrate their security through reductions to those new hard problems [3].

The problem of the decoding of Reed-Solomon codes is quite old and has received much interest from coding theorists since the introduction of these codes [12]. The goal of decoding is to retrieve a word of the Reed-Solomon code from a corrupted word, that is, a word containing a small number of errors. More recently, important progress has been done to extend the number of errors which can be corrected [5]. Thus the problem of decoding Reed-Solomon is easy when the number of errors is small.

On the other hand, attention has been recently given to the case when the number of errors is larger, and it turns out that the problem presents some interesting hardness properties [6, 7, 2]. This problem has an equivalent formulation under the name *Polynomial Reconstruction* (PR) [6, 7]. Consequently some effort has been done to construct cryptographic primitives for which the security is related to the hardness of Polynomial Reconstruction. For instance, in [10], the authors construct an oblivious polynomial evaluation scheme, and, in [6, 7], a semantically secure symmetric cipher is obtained.

In this paper we present a public key cipher related to the PR problem. In this scheme, Alice's public key is some kind of noise, which correspond to an instance of the Polynomial Interpolation Problem and which has been created by Alice from secret data. Then, when Bob wishes to send a message to Alice, he randomizes this noise, adds it to the message, and furthermore adds some small random noise generated by himself (in the same manner as in the McEliece public key cryptosystem [9]). Then Alice can use her secret data as a trapdoor to remove the noise added by Bob. Interestingly, our scheme relies on two new intractability assumption: while the hardness of PR is used to establish the security of the public key, the security of the encryption relies on the apparent difficulty of decoding a Reed-Solomon augmented by one word. From this point of view, it is radically different from the McEliece scheme.

Compared to the McEliece cryptosystem, where the public data is a whole generating matrix for a random-like error correcting code, the public key of our cryptosystem is a single word. This leads to a much shorter public key. The attacks we have investigated originate from coding theory, and the complexity of the best decoding algorithm [1] is discussed. This enables us to choose the parameters such that the work factor to recover the plaintext is above 2^{80}. For such a security, we get a public key of size ≈ 80000 bits, which can be reduced to 3072 bits, using particular instances of the Polynomial Reconstruction problem (subfield subcodes which shall be explained in Section 4.3).

The paper is organized as follows: in Section 2 we recall the definition and the properties of Reed-Solomon codes and we discuss some hardness properties of the decoding of Reed-Solomon codes, mainly following [6, 7]. Section 3 presents the encryption scheme, and Section 4 presents some attacks and their complexity, in order to determine practical security parameters. Finally, Section 4 presents the algorithmic performance of the system.

2 Polynomial Reconstruction and Reed-Solomon decoding

2.1 Background on Reed-Solomon codes

Let F_q be the finite field with q elements, let $x_1, \ldots, x_n$ be n distincts elements of F_q, we denote by ev the following map

$$\mathrm{ev} : \begin{cases} F_q[X] \rightarrow F_q^n \\ p(X) \;\mapsto (p(x_1), \ldots, p(x_n)), \end{cases}$$

where $F_q[X]$ is the ring of univariate polynomials over F_q.

Definition 1 *The* Reed-Solomon *code of dimension k and length n over F_q, denoted RS_k is the following set of n-tuples (codewords)*

$$RS_k = \{ev(f); f \in F_q[X]; \deg f < k\}$$

where $F_q[X]$ is the set of univariate polynomials with coefficients in F_q. The set $(x_i)_{i\in\{1\ldots n\}}$ is called the support *of RS_k. We shall use the notation "*the $[n,k]_q$ Reed-Solomon code*". A* generating matrix *of RS_k is an $k \times n$ matrix whose rows generate RS_k as an F_q-vector space. Let $I \subseteq \{1\ldots n\}$, the* shortened *Reed-Solomon code at I is the Reed-Solomon code with support $(x_i)_{i\notin I}$.*

We shall need the following definition when discussing attacks on our scheme:

Definition 2 *The* dual $RS_k^\perp$ *of RS_k is the following set of words in F_q^n*

$$RS_k^\perp = \{x = (x_1,\ldots,x_n) \in F_q^n;\ \sum_{i=1\ldots n} x_i c_i = 0 \textit{ for all } c = (c_1,\ldots,c_n) \in RS_k\}$$

A parity check matrix H *of RS_k is a generating matrix of $RS_k^\perp$. The* syndrom *of a word x is Hx^t.*

The syndrom S of word x has the property to be zero when x belongs to the RS_k. Let $c \in F_q^n$ be given, *decoding* c is finding the *closest* element to c in RS_k. The distance in use is the *Hamming distance* d_H:

$$d_H(x,y) = \#\{i \in [1..n];\ x_i \neq y_i\},$$

and the weight of a word $c \in F_q^n$ is the number of non-zero coordinates of c. Formally the problem of decoding is:

Reed-Solomon decoding (RSD): *Given a $[n,k]$ Reed-Solomon code RS_k, w an integer and a word $y \in F_q^n$, find any codeword in RS_k at distance less than w of y.*

The minimum distance (the smallest weight of non zero codewords in RS_k) of the $[n,k]$ Reed-Solomon code is $(n-k+1)$. The classical goal of coding theory is to decode up to $w = (n-k+1)/2$ errors, in which case, the solution to RSD is guaranteed to be unique.

In terms of polynomials, the problem of decoding can be stated as follows, where the parameters t is $n-w$. The term *Polynomial Reconstruction* has been introduced in [10].

Polynomial Reconstruction (PR): *Given n, k, t and $(x_i, y_i)_{i=1\ldots n}$, output any polynomial $p(X)$ such that $\deg p(X) < k$ and $p(x_i) = y_i$ for at least t values of the index i.*

Note that the two formulation are strictly equivalent. Coding theorists use the notion of decoding, whereas other authors use the terms polynomial reconstruction. The problem is easy when $t \geq \frac{n+k}{2}$, where decoding can be done using the classical Berlekamp-Massey [8] algorithm or the Berlekamp-Welsh algorithm [13]. For this value it is indeed the classical problem of decoding Reed-Solomon up to their error capability. Note that this has been improved to $\sqrt{kn}$ by Guruswami and Sudan [5]. They introduced a polynomial time algorithm for listing all codewords at distance less than $n - \sqrt{kn}$ of the received word (*list decoding*). This implies that the PR problem is easy when $t \geq \sqrt{kn}$.

2.2 Cryptographic hardness of Polynomial Reconstruction

The PR problem has also been investigated for small values of t, that is to say, decoding when the error has a large weight. Naor and Pinkas made use of the PR problem to build a protocol for Oblivious Polynomial Evaluation [10]. Later, the security of PR has been fully investigated in [6, 7], and we informally summarize here their results.

From the point of view of establishing the hardness of PR, we will call parameters n, k, t *sound* if $\binom{n}{k}$ and $\binom{n}{t}$ are exponential in n, and if $t < \sqrt{kn}$. Such a requirement implies that exhaustive search on valid positions or on error positions is impossible, and that the Guruswami-Sudan algorithm does not apply. Naor and Pinkas have shown that, when the finite field is large ($\log q \geq 2n$), the proportion of instances in $S_{n,k,t}$ such that the number of solution polynomials is more than one is negligible (less than 2^{-n}). As a consequence, we may (and shall) assume that the number of solutions is one, and we will use the term *the* solution polynomial for the unique polynomial which is solution to the PR problem.

In order to study the hardness of the PR problem, the same authors identified a decisional problem: given a PR instance, determine whether the i-th point of the support belongs to the graph of the solution polynomial (that is, does we have $p(x_i) = y_i$, where $p(X)$ is the solution polynomial). They postulate the hardness of this problem under the name “Decisional PR-Assumption” (DPR).

Then the authors show the following properties:

- Hardness of partial information extraction: under the DPR, an adversary who wishes to compute their value of the solution on a new point has a negligible advantage over an adversary who does not see the instance.
- Under the DPR assumption, the family of PR instances is pseudorandom, for a poly-time observer.

These results indicates that the PR problem has good properties for cryptographic purposes. Note that these results hold for *sound* parameters. The authors succeed in building a symmetric cipher.

In [2], the following problem is proved to be NP-hard.
PolyAgree: *given* k, t, n *and a set of pairs* $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i, y_i \in F_q$, *does there exist a degree* k *polynomial* $p(X)$ *such that* $p(x_i) = y_i$ *for at least* t *different* i*'s?*

It is important to note that x_i's in the PolyAgree problem are not required to be different, and in fact the proof in [2] uses the fact that some of them may be equal. So the problem *PolyAgree* is not exactly the PR problem, but this result seems to indicate that PR is a hard problem.

3 The proposed system

The system is as follows.

Parameters q, n, k, W, w: q is the size of F_q, n is the length of the Reed-Solomon code, k its dimension, W is the weight of a large error ($W = n - t$ in terms of PR formulation), w is the weight of a small error. We ask for $n, k, n - W$ to be sound parameters for the PR problem.

Key generation Alice secretly generates a unitary polynomial $p(X)$ of degree equal to k, and an error E of weight W. She computes the codeword $c = \mathrm{ev}(p(X))$ of RS_k, and computes $c + E$. The public key is $c + E$, while c and E are kept secret.

Encryption Bob wishes to send a message m_0 of length k over the alphabet F_q. The message is first encoded into a codeword m in RS_{k-1}, by computing $m = \mathrm{ev}(m_0)$, where $m_0 = m_{0,0}, \ldots, m_{0,k-1}$ is seen as the polynomial $m_0(X) = m_{0,0} + m_{0,1}X + \ldots m_{0,k-1}X^{k-1}$.

Bob randomly generates $\alpha \in F_q$ and an error pattern e of weight w. Then he computes $y = m + \alpha \times (c + E) + e$, and transmits y to Alice.

Decryption Upon receipt of $y = m + \alpha \times (c + E) + e$, Alice considers only the positions i where $E_i = 0$. In terms of codes this means she considers the shortened code of length $N - W$, which is also a Reed-Solomon code of dimension k, that we shall denote by $\overline{RS}_k$. Let $\overline{m}$, $\overline{c}$, $\overline{e}$ correspond to the shortened m, c, e. Then Alice has to solve the equation

$$\overline{m} + \alpha \times \overline{c} + \overline{e} = \overline{y},$$

where the "big" error E has disappeared. Obviously $\overline{m} + \alpha \times \overline{c}$ belongs to $\overline{RS}_k$. Provided that w is less than the error correction capacity of $\overline{RS}_k$, Alice can correct $\overline{y}$ and find $\overline{m} + \alpha \times \overline{c}$. Now Alice uses Lagrange interpolation to computes the unique polynomial $q(X)$ of degree k such that $\mathrm{ev}(q(X)) = \overline{m} + \alpha \times \overline{c}$. Since m_0 is a polynomial of degree less than or equal to $k - 1$, and since $\overline{c} = \mathrm{ev}(p(X))$ and $p(X)$ has degree exactly k and is monic, α appears as the leading coefficient of $q(X)$. Knowing α and $p(X)$, Alice can compute $q(X) - \alpha p(X) = m_0(X)$.

Comments The scheme has two parts for encryption: $m \mapsto m+\alpha\times(c+E)$ which is like a randomized one-time pad with $c+E$, and a McEliece part: $m+\alpha\times(c+E) \mapsto m+\alpha\times(c+E)+e$, where an error of small weight has been added. Also we must have $w \leq (n - W - k + 1)/2$ in order to have unique decoding for the shortened Reed-Solomon code.

4 Investigated attacks and security parameters

4.1 The relationship between m, α, e

In this Subsection, we show that the knowledge of any value among m, α, e implies the knowledge of the other two values. From a cryptographic point of view, any attack on one of these values enables to recover the other values.

Retrieval of m and e from α First suppose that the adversary knows α. He can compute $y-\alpha\times(c+E)=m+e$, then, since the Reed-Solomon code can decode up to w errors, he can decode, and find separately c and e.

Retrieval of m and α from e Second suppose that the small error e is known to the adversary. He computes $y-e=m+\alpha\times(c+E)$ and the syndrom of $y-e$: $S=H_{k-1}(y-e)^t$, where H_{k-1} is a parity check matrix for RS_{k-1}. Then $S=\alpha H_{k-1}(c+E)^t$ and α is recovered, since $c+E$ is public and its syndrom is known. Then $m=y-e-\alpha\times(c+E)$ is found.

Retrieval of α and e from m Finally we suppose that the adversary has a successful attack on the message m. Knowing y he computes $v:=y-m=\alpha\times(c+E)+e$. Doing componentwise division of v and $c+E$, the corresponding vector will have a large majority of α among its coefficients. Thus α is known and e also.

This interdependance of m,α,e implies that each of these value must be safe from the cryptographic point of view. For instance, to prevent exhaustive search on α, we must set $\alpha\in F_q$ where $q\geq 2^{80}$.

4.2 Attacks

There are two kinds of attacks on the system: structural attacks (universal attacks) which attempt to break the system by recovering the secret key from the public key, and decoding attacks (existential attacks) which will simply try to recover the plaintext from an encrypted message.

Structural attacks: recovering the secret key
The secret key of this system is the pair (c,E). Recovering it from the public key $c+E$ is a decoding problem. As all other decoding problems it can be treated in two ways: either the attacker tries to guess the positions where the error is located (i.e. indices i such that $E_i\neq 0$), shortens the code on these positions and tries to decode, or he tries to guess some positions where there are no errors and simply recovers the code word using a pseudo-inverse of the generator matrix of the code.

The first attack is called *Error Set Decoding* (ESD), it will take full advantage of the knowledge of the code structure to decode the largest possible number of errors remaining in the shortened code. The second attack is called *Information Set Decoding* (ISD). It will work as if the code was a random linear code and doesn't take advantage of it's structure.

Error Set Decoding for key recovery (ESD_W) is based on the following idea: with Guruswami-Sudan's algorithm one can decode up to $n-\sqrt{kn}$ errors. The weight W of E is chosen greater than this bound (otherwise decoding would be easy). However one can choose β positions in the code word and shorten the code on these β positions. In the new shortened code, Guruswami-Sudan's algorithm will then decode $W_\beta=n-\beta-\sqrt{k(n-\beta)}$ errors. The attacker will then be able to decode the error E if among the β chosen positions there are at least $W-W_\beta$

errors. This attack will be optimal when $\beta = W - W_\beta$ (that is $\beta = n - \frac{(n-W)^2}{k}$) and the attacker picks β positions which are all in the W error positions. The probability this will happen is $\binom{W}{\beta}/\binom{n}{\beta}$. Hence the security of the system against this kind of attack is $\binom{n}{\beta}/\binom{W}{\beta}$.

Information Set Decoding for key recovery (ISD_W) consists (in its first form) in finding k positions with no errors. As the code dimension is k this is enough to recover the complete codeword and deduce the error from it. The attack would be the following: choose k random positions, recover the complete associated code word, check that the corresponding error is of the right weight, if not, try again with another k positions. This attack would have a complexity of $\binom{n}{k}/\binom{n-W}{k}$.

However in most cases it can be improved: instead of trying to guess k positions with no errors the attacker can look for k positions containing only a small given number p of errors and then try to guess which are these p positions. This can be done very efficiently using the Canteaut-Chabaud algorithm [1]. Depending on the parameters of the code the complexity of the attack can decrease a lot when p is well chosen. Its optimal value will depend widely on the parameters (n, k, W) of the code and on the size of the field in which we work. We call this attack CC_W for "Canteaut-Chabaud".

Decoding attacks: decoding in a Reed-Solomon after adding a word
The problem of decoding in a Reed-Solomon code after adding a random word $c + E$ to it seems to be hard when the alphabet F_q is large. The only way to take advantage of the underlying Reed-Solomon structure would be to test for all the possible values of α if there is one for which we can decode. However this will only work if the code is defined on a small alphabet, that is, if the finite field in which α is taken is small enough. If this can not be done there is no known technique to decode using the structure of the code. Hence the attacker will have to consider the code as a random linear code, and the only possible attacks will be generic decoding attacks. As for the key recovery we will then distinguish an ESD attack and an ISD attack.

Error Set Decoding for message recovery (ESD_w) is not the best attack for message recovery. In the previous case ESD_W for key recovery worked well because the shortened code still had a good error correction capacity. This time we consider the code as a random code which has no error correction capacity. The attack then becomes very basic: it consists in finding all the w error positions to decode a message. Therefore this attack is necessarily more expensive than the ISD which only needs to find k positions containing no errors.

Information Set Decoding for message recovery (ISD_w) will work exactly as for the key recovery, except this time the dimension of the code is one more and the number of errors to be corrected is much smaller: $w = n - W - \sqrt{(n-W)k}$. We then obtain the following security diagrams. We also have a refined attack using the Canteaut-Chabaud algorithm (CC_w).

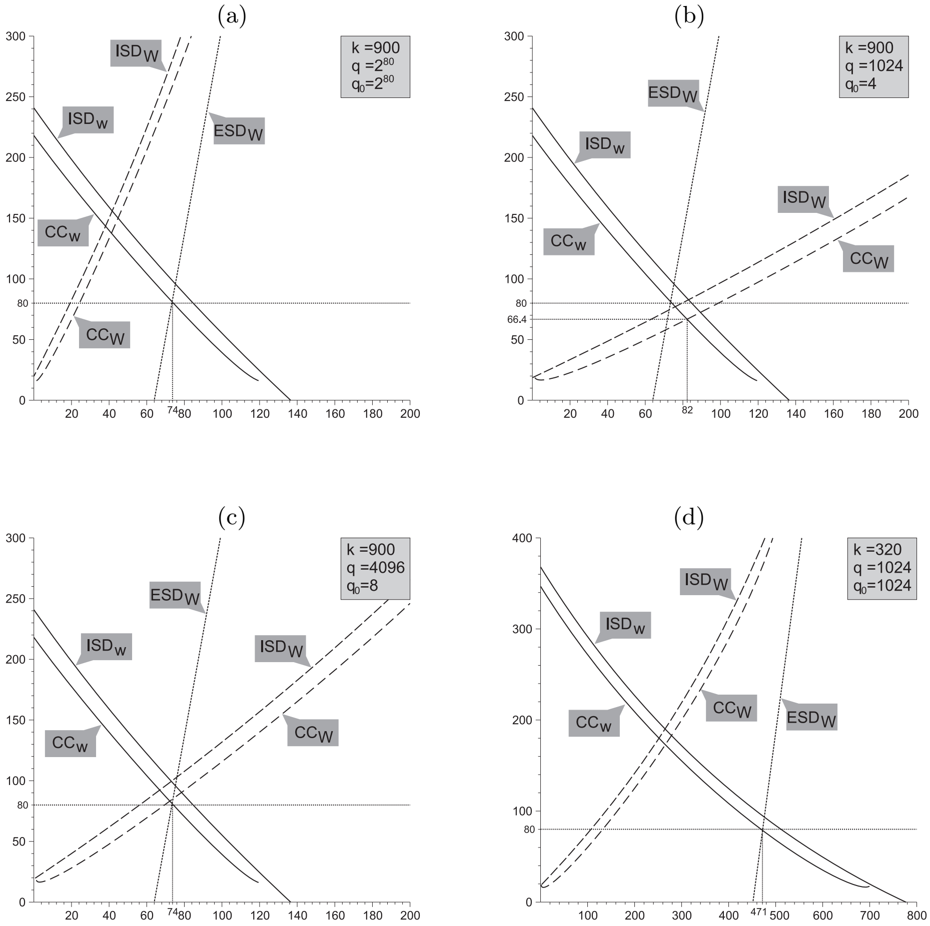

Fig. 1. Assumed $\log_2$(Security) for different parameters of the system as a function of W

4.3 Security parameters

In the following discussion, we assumed that there are no other attacks than those presented above.

Aiming at a security level of 2^{80}, it is not possible to devise our system using a very short length n. Thus we must settle the length to be 1024. In that case, Figure 1(a) shows the work factor as a function of W for the various attacks: ESD_W, ISD_W and CC_W for key recovery ; and ISD_w and CC_w for message recovery. In particular it can be seen that $W = 74$ is required to reach the 2^{80} barrier.

4.4 Shortening the public key

The above security parameters ($n = 1024$, $k = 900$, $W = 74$, $w = 25$, $q = 2^{80}$), lead to a public key of size $n \times \log q = 81920$ bits. We note that this is shorter than the public key in McEliece cryptosystems, or the HFE cryptosystem [11]. However, we can make the public key smaller, using the notion of *subfield subcode.*

Definition 3 *Let a Reed-Solomon code C be defined over F_q where $q = q_0^m$. The F_{q_0}-subfield subcode of C set is the set of codewords of C whose coordinates belongs to the subfield F_{q_0}.*

If the code C has length n and dimension k, then the F_{q_0}-subfield subcode has dimension between k and $n - (n - k)m$. Conversely, given a generating matrix G of a Reed-Solomon code of length n and dimension k defined over F_{q_0}, it can be used as a generating matrix for the Reed-Solomon code of the same length and dimension, defined over some extension F_q of F_{q_0}.

Considering the Reed-Solomon code defined over $F_{2^{80}}$, we shorten the public key by considering the Reed-Solomon code of length 1024 defined over the field $F_{2^{10}}$. Note that there is no decrease of dimension since all elements of the support are in $F_{2^{10}}$. In that case the public key is $c + E$ where c is a codeword of the $[n, k]_{2^{10}}$ Reed-Solomon code, E is an error of weight W with coordinates in $F_{2^{10}}$. Encryption is still done over the extension field $F_{2^{80}}$. This leads to a public key of size $10 \times 1024 = 10240$ bits.

We can further reduce the size of the public key, by considering subfield subcodes of the $[n, k]_{2^{10}}$ Reed-Solomon code, for instance the $[n, k']_{2^2}$ subfield subcode. The field $F_{2^{10}}$ is an extension of degree 5 of F_{2^2} and this code has dimension k' of order $1024 - 5 \times 124 = 404$. The public key would then be defined over F_{2^2} and have length 2048. But this is not secure enough because of the huge dimension decrease, and falls under the decoding attack, see Figure 1(b).

As a consequence we need to find a code with higher dimension to prevent this attack. We change the big field F_q to be $F_{2^{84}}$, consider the Reed-Solomon of length 1024 and dimension 900 over the field $F_{2^{12}}$. Since $F_{2^{12}}$ admits F_{2^3} as a subfield, we publish the key as belonging to F_{2^3}. We get a public key of size $3 \times 1024 = 3072$, and a dimension k' for the subfield subfield larger than or equal to $1024 - 4 \times 124 = 528$, which is now secure under the decoding attack (Figure 1(c)). Such a size of key is twice the one of a RSA key for the same level of security (1536 bits for a security of 2^{80} as estimated in [4]).

Security Considerations: using such a small field, we loose the fact that a PR instance has a unique solution with overwhelming probability. Furthermore the adversary is faced with a very particular case of the PR problem, for which the solution polynomial which is sought for is such that it evaluates over the medium field into values in the small field. Nevertheless, to our knowledge, no better attack than the decoding attack seem to apply.

4.5 Complexity of encryption/decryption

One main drawback of our scheme is that the block length is very large: approximately 84000 bits, which may be convenient for some applications, but too large for others, eg. transmitting the short keys of a symmetric cipher. There are two steps for encryption: encoding the message into a Reed-Solomon codeword, and adding noise to the codeword. Encoding is done in the following way: the message $m_0(X)$ is evaluated at each point of the support of the Reed-Solomon code. This is done using n Horner evaluations of a polynomial of degree k. Each evaluation has a complexity k, thus a total cost of nk operations in $F_{2^{84}}$. Adding the noise consists in doing the multiplication $\alpha \times (c + E)$, which costs n multiplications. The cost of adding the small noise e is negligible. Counting $\log^2 q$ bit operations for multiplication if F_q, the total cost is of order $O(nk \log^2 q)$ for encryption, that is, $O(n \log q)$ bit operations per bit of the plaintext.

Decryption is more expensive, since it consists in decoding of a Reed-Solomon code, which is done at cost $O((n - W)^2)$ (See [8] for instance) arithmetic operations in F_q, that is $O((n - W)^2 \log^2 q)$ bit operations. Then the polynomial must be found by interpolation, at cost $O((n-W)^2 \log^2 q)$ operations. Thus the total cost is $O(n - W)^2 \log^2 q)$ operations and the cost per bit of plaintext is $O(\frac{(n-W)^2}{k} \log q)$.

If one has in mind to reach the fastest decryption time, it can further be accelerated by letting W be large, such that the cost $O((n - W)^2)$ of decoding is relatively small. For instance we could use the parameters $W = 470$, $k = 320$ (see Figure 1(d)).

4.6 Asymptotic behavior

We have pointed out some parameters giving our cryptosystem a security of 2^{80} CPU operations. For this security level we have seen that it is easy to adjust these parameters either to have a shorter public key, or a faster encryption/decryption time. However for our system to be of real interest it is necessary to think about its future. For instance, a security of 2^{80} won't be considered sufficient in a few years. Therefore we need to see how the parameters for this system scale.

We will only consider the case where $q = q_0 = n$ and the transmission rate is $k/n = R$. We then express all the parameters of the system as functions of n and R. For the sake of simplicity we only consider the basic forms of ISD since asymptotically the difference between CC and ISD isn't really significant. Moreover, as $q = q_0$ and W is always greater than w (as long as W is greater than the Sudan bound $n - \sqrt{kn}$), the ISD for key recovery will always be more expensive than the ISD for message recovery (ISD_w). Hence, we need to find the intersection point between the two curves ESD_W and ISD_w. It will give us the optimal value for W from a security point of view.

It appears, in first order approximation, that, at the optimal point, all the variables of the system (W, w and β) can be put in the form $C \times n$ where C is a constant depending only on the chosen R. The diagram in Figure 2(a) shows the optimal value for W/n as a function of R. However, the most interesting part about this approximation is the security: at the optimal point, once all the parameters are put under the form $C \times n$, the security of the system takes the form B^n. The value of B is given in Figure 2(b). It is important to note that

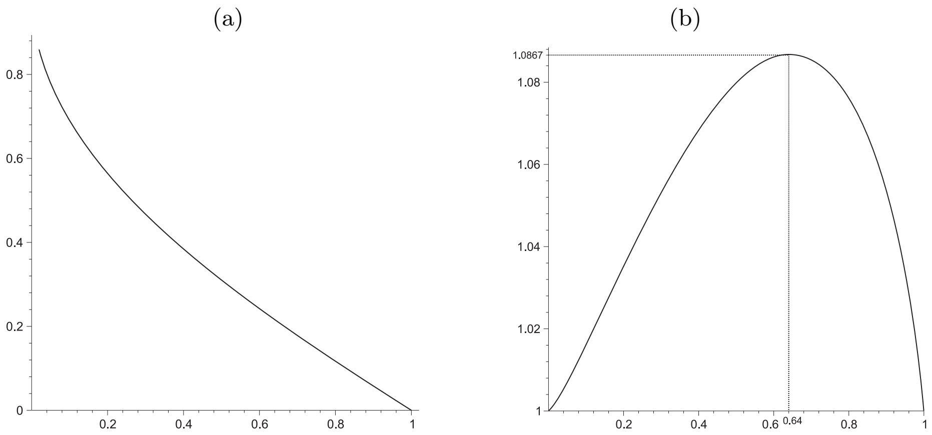

Fig. 2. optimal asymptotic values for W/n (on the left) and the security of the system (on the right) as functions of R

this approximation is only a first order approximation. However it turns out to be quite good as the security levels it gives are really close to those obtained through exact calculations, for given parameters.

This means that, whatever the value of R, it is always possible to get an exponential security, as long as W is well chosen. The best choice from a security point of view will be $R \simeq 0.64$ which can lead to a security as high as 2^{122} for a length $n = 1024$. Moreover, if one accepts loosing a little security, it is possible to either increase the transmission rate to minimize the amount of data transfered during an encrypted communication, or decrease the transmission rate to reduce the amount of calculation and speed up the encryption/decryption processes.

5 Conclusion

We have proposed a public key encryption scheme based on a new intractable problem, the *Polynomial Reconstruction* problem, which has already been identified as a hard problem by Kiayias and Yung. The cryptosystem is fast for encryption and decryption and presents a variant with a small public key (3072 bits). The public and secret keys are very easy to generate, even in large quantities. Furthermore when the parameters are carefully selected, the security of the system seems truly exponential in terms of the parameters. Thus the system should remain secure under every known attack.

References

1. A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: application to primitive narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, January 1998.

2. O. Goldreich, R. Rubinfeld, and M. Sudan. Learning polynomials with queries: the highly noisy case. *SIAM J. of Discrete Math*, 13(4):535–570, 2000.
3. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.
4. L. Granboulan. Short signature in the random oracle model. In Y. Zheng, editor, *Advances in Cryptology, Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 364–378, 2002.
5. V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and Algebraic-Geometric codes. *IEEE Transactions on Information Theory*, 45:1757–1767, 1999.
6. A. Kiayias and M. Yung. Cryptographic hardness based on the decoding of Reed-Solomon codes with applications. Technical report, Electronic Colloquium on Computational Complexity, http://eccc.uni-trier.de/eccc/, 2002.
7. A. Kiayias and M. Yung. Cryptographic hardness based on the decoding of Reed-Solomon codes with applications. In Springer-Verlag, editor, *ICALP 2002*, number 2380 in LNCS, pages 232–243, 2002.
8. F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes.* North-Holland, 1977.
9. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. Technical report, Jet Prop. Lab., 1978.
10. M. Naor and B. Pinkas. Oblivious transfer and polynomnial evaluation. In ACM, editor, *STOC 99*, pages 245–254, 1999.
11. J. Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of symmetric algorithms. In *Advances in Cryptology, Eurocrypt'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48, 1996.
12. I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *J. SIAM*, 8:300–304, 1960.
13. L. R. Welch and E. R. Berlekamp. Error correction for algebraic block codes. US Patent 4 633 470, 1986.