A SOFTWARE IMPLEMENTATION OF THE McELIECE PUBLIC-KEY CRYPTOSYSTEM

Bart Preneel[1,2], Antoon Bosselaers[1], René Govaerts[1] and Joos Vandewalle[1]

A software implementation of the McEliece public-key cryptosystem is presented together with some existing and new extensions. The important disadvantages of the scheme are the data expansion, the size of the keys and the fact that no digital signatures are possible. However, even a software implementation results in a reasonable speed of encryption and decryption. Moreover, the system can be used as a combined scheme that offers a both encryption and error correction at the cost of a decreased security level.

1 Introduction

In 1978, McEliece proposed a new public-key cryptosystem based on algebraic coding theory [13]. The system makes use of a linear error-correcting code for which a fast decoding algorithm exists, namely a Goppa code. The idea is to hide the structure of the code by means of a transformation of the generator matrix. The transformed generator matrix becomes the public key and the trapdoor information is the structure of the Goppa code together with the transformation parameters. The security is based on the fact that the decoding problem for general linear codes is NP-complete [4].

In a first section a mathematical description of McEliece's system will be given. Subsequently extensions and attacks will be summarized. Next some details are given on the implementation of key generation, the encryption and decryption. Finally we present our conclusions.

2 Description of McEliece's Public-Key Cryptosystem

For each irreducible polynomial $g(x)$ over $GF(2^m)$ of degree t, there exists a binary irreducible Goppa code of length $n = 2^m$ and dimension $k \geq n - mt$, capable of correcting any pattern of t or fewer errors[3]. As it is a linear code, it can be described by its $k \times n$ generator matrix G. With the aid of a regular $k \times k$ matrix S and an $n \times n$ permutation matrix P, a new generator matrix G' is constructed that hides the structure of G:

$$G' = S \cdot G \cdot P$$

[1]Katholieke Universiteit Leuven, Laboratorium ESAT, K. Mercierlaan 94, B–3001 Heverlee, Belgium.

[2]NFWO aspirant navorser, sponsored by the National Fund for Scientific Research (Belgium).

Appeared in *Proceedings of the 13th Symposium on Information Theory in the Benelux*, Werkgemeenschap voor Informatie- en Communicatietheorie, pp. 119–126, 1992.

The public key consists of G', and the matrices S and P together with $g(x)$ are the secret key. The new matrix G' is the generator matrix of another linear code, that is assumed to be difficult to decode if the trapdoor information is not known. The encryption operation consists of multiplication of the k-bit message vector by G' and the modulo 2 addition of an error vector e with Hamming weight t:

$$c = m \cdot G' \oplus e.$$

The first step of the decryption is the computation of $c \cdot P^{-1}$. Subsequently the decoding scheme makes it possible to recover $m \cdot S$ from

$$c \cdot P^{-1} = (m \cdot S \cdot G) \oplus (e \cdot P^{-1}).$$

The message m is finally constructed by a multiplication with S^{-1}.

The disadvantages of the scheme are the data expansion, the size of the keys, and the fact that no digital signatures are possible. On the other hand, the implementation of the encryption part is much simpler, and for a comparable security level, the speed of our general implementation is comparable to that of highly optimized code for the well known RSA cryptosystem [16]. Moreover, it is one of the few still unbroken public-key cryptosystems that is not based on any number-theoretic assumption.

3 Extensions

In this paper only public-key variations will be considered. A first extension originated by F. Jorissen[7]. The idea was to add only $t' < t$ errors, such that $t - t'$ additional errors can be corrected. This implies that the security level degrades: under worst case conditions no additional error occurs and the work factor of an attacker decreases significantly. For some applications however, it could be very attractive to have a combined system that automatically corrects some errors.

A second idea consists of improving the code rate by transferring some data through the pattern of the error bits [5, 14]. It is important to note that this has no effect on security if the data in the concealed channel is perfectly random, but otherwise an attacker could obtain an important advantage.

A third extension is the replacement of the requirement of irreducibility of $g(x)$ with a different condition: $g(x)$ must be the product of non-repeating factors of degree at least 2. It can be shown that in this case the error correcting capabilities are unchanged and the bound on the dimension remains valid. A first consequence is an increased key space. Secondly, the decoding algorithm has to be modified to take into account these changes. For the implications on key generation and decoding the reader is referred to section 5.

Appeared in *Proceedings of the 13th Symposium on Information Theory in the Benelux*, Werkgemeenschap voor Informatie- en Communicatietheorie, pp. 119–126, 1992.

4 Cryptanalysis

The known non-exhaustive attacks can be classified in three categories: a first type of attack tries to compute the key or an equivalent key. However, it is shown in [1] that the existence of an equivalent Goppa code is extremely unlikely. The conclusion in [6] is that the task of the researcher who wants to assess the security is as difficult as the task of an attacker who wants to break the scheme.

A second type of attack aims at recovering directly the message m. The main idea is to select and solve k of n equations obtained from c and G'. This attack was already mentioned in [13], but has been significantly improved in by Lee and Brickell [9] and by van Tilburg [18]. A first element is that the agreement between the resulting message and the original message is systematically checked. The second improvement is that j errors ($j \geq 1$) in the k equations are allowed. The attack is then optimized with respect to j.

A recent attack by Korzhik and Turkin [8] is based on an iterative optimization algorithm, and the claimed number of operations to correct an error pattern with weight at most t is claimed to be $20 \cdot n^3$. This would mean a major breakthrough, but the validity of the work is in question and remains to be verified. Note that this attack does not contradict the fact that the decoding problem for a general linear code is NP-complete, as only error patterns with Hamming weight $\leq t$ can be corrected.

The conclusion is that for $m = 10$ a maximal work factor of $2^{71.1}$ is obtained if $t = 39$. This implies that the resulting code has a length $n = 1024$, and a dimension $k \geq 634$. The information rate equals 0.619. The concealed channel could contain 235 bits.

5 Implementation

In this section, the key generation, encryption and decryption will be discussed in more detail. For execution time and memory requirements, data will be given for the case $m = 10$ and $t = 39$. Execution times were measured on a 16 MHz IBM PS/2 Model 80 containing a 80386 processor running under DOS. No use was made of 32-bit 80386 instructions.

5.1 Key Generation

This is certainly the most complicated part of the algorithm. It is certainly less critical than the encryption and decryption, but the nature of the operations (e.g. inversion of a $k \times k$ matrix) requires careful coding if a reasonable performance is expected. The key generation involves following steps:

Step 1 Select a primitive polynomial of degree m and prepare log and antilog tables for $GF(2^m)$.

Appeared in *Proceedings of the 13th Symposium on Information Theory in the Benelux*, Werkgemeenschap voor Informatie- en Communicatietheorie, pp. 119–126, 1992.

Step 2 Select a generator polynomial of degree t, with no linear or repeating factors. The first condition is easily checked, while the second is fulfilled if

$$\gcd(g(x), g'(x)) = 1.$$

Note that the probability that a random polynomial is irreducible, as was required in the original scheme, equals approximately $1/t$, while a constant fraction of about $1/e$ of all polynomials satisfies the relaxed conditions. A proof of this assertion is given in [19]. As a consequence, the time for finding a $g(x)$ is reduced.

Step 3 Compute the parity matrix H over $GF(2^m)$ and H_{bin}, the parity matrix over $GF(2)$. The generator matrix G_{bin} can now easily be computed.

Step 4 Select a random $n \times n$ permutation matrix P and construct the invertible scramble matrix S as follows: $S = S_1 \cdot S_2$, where S_1 is a lower triangular matrix over $GF(2)$ with random entries and S_2 is an upper triangular matrix over $GF(2)$ with random entries and with diagonal elements equal to 1. The inverse S^{-1} is easily computed as $S_2^{-1} \cdot S_1^{-1}$.

Step 5 The public key G' is computed as $S \cdot G \cdot P$, and the secret key consists of S, P an $g(x)$.

This key generation takes about 5 minutes, and program plus data require together about 560 K, which is quite close to the 640 K limit of the DOS operating system. The size of the public key G' is 79.3 K, while the secret key consists of S (49.1 K), P (1.3 K) and $g(x)$. The size of the secret key could be reduced significantly if S is generated from a small seed. In [18] it is shown that a (different) decomposition of G' can be made public without decreasing the security level. This results in a public key only slightly larger than $k(n-k)$ in stead of kn. For the usual parameters this would be about 30.2 K. The price paid for this is in both cases that the scramble matrix must be computed when it is needed.

5.2 Encryption

The encryption operation is very simple: it consists of a vector-matrix multiplication followed by an addition of t random errors. The number of operations is kn, resulting in a number of operations per bit of k with respect to the code bits and n with respect to the information bits. The implementation in assembly language achieves a speed of respectively 6 Kbit/sec and 3.7 Kbit/sec. With special hardware, this encryption operation would certainly allow for speeds in the order of several Mbit/sec.

Appeared in *Proceedings of the 13th Symposium on Information Theory in the Benelux*, Werkgemeenschap voor Informatie- en Communicatietheorie, pp. 119–126, 1992.

5.3 Decryption

The most time consuming and complex step in the decryption is the computation of the error locator polynomial.

Step 1 Apply the permutation P^{-1} to c.

Step 2 Compute the error locator polynomial corresponding to $c \cdot P^{-1}$. The algorithm of Patterson [15] is modified to take into account the fact that $g(x)$ is not necessary irreducible. In Step 2.2 an additional gcd has to be computed.

Let L be a subset of $GF(2^m)$ with the property that no element of $GF(2^m)$ is a root of $g(x)$ and let $M = \{\gamma \in L \mid e_\gamma = 1\}$. The syndrome $S(x)$ is then defined as

$$S(x) = \sum_{\gamma \in L} \frac{c_\gamma}{x - \gamma} \bmod g(x).$$

The error locator polynomial $\sigma(x)$ is defined as

$$\sigma(x) = \sum_{\gamma \in M} (x - \gamma).$$

Decoding of the binary Goppa code means solving the **key equation**:

$$S(x) \cdot \sigma(x) \equiv \sigma'(x) \bmod g(x)$$

where $\sigma'(x)$ denotes the formal derivative of $\sigma(x)$. The solution of this key equation requires following steps:

Step 2.1 Split $\sigma(x)$ in an even and an odd part:

$$\sigma(x) = \alpha^2(x) + x \cdot \beta^2(x).$$

The key equation then becomes

$$S(x)(\alpha^2(x) + x \cdot \beta^2(x)) \equiv \beta^2(x) \bmod g(x).$$

As $\deg \sigma(x) \leq t$, we can conclude that $\deg \alpha(x) \leq \lfloor t/2 \rfloor$ and $\deg \beta(x) \leq \lfloor (t-1)/2 \rfloor$.

Step 2.2 Compute $g_1(x) = \gcd(S(x), g(x))$. Note that if $g(x)$ is irreducible, $g_1(x)$ is constant. The factor $g_1(x)$ can then be removed from $S(x)$ and $g(x)$:

$$\begin{aligned} g(x) &= g_1(x) \cdot g_2(x) \\ S(x) &= g_1(x) \cdot h(x). \end{aligned}$$

Appeared in *Proceedings of the 13th Symposium on Information Theory in the Benelux*, Werkgemeenschap voor Informatie- en Communicatietheorie, pp. 119–126, 1992.

Step 2.3 As $h(x)$ is relatively prime to $g(x)$, its inverse modulo $g(x)$ can be computed. The resulting equation is:

$$\beta^2(x)(x \cdot g_1(x) + h^{-1}(x)) = g_1(x)\left(\alpha^2(x) + q^*(x) \cdot g_2(x)\right)$$

Step 2.4 With following definitions:

$$\begin{aligned} \beta^2(x) &= g_1^2(x) \cdot \beta_1^2(x) \\ d_1^2(x) &\equiv (x \cdot g_1(x) + h^{-1}(x)) \bmod g(x) \\ d_2^2(x) &\equiv g_1(x) \bmod g(x) \\ d^2(x) &= d_1^2(x) \cdot d_2^2(x) \end{aligned}$$

the equation simplifies to:

$$\beta^2(x) \cdot d^2(x) \equiv \alpha^2(x) \bmod g_2(x) \qquad (\dagger)$$

To compute $d_1(x)$ and $d_2(x)$, a square root has to be extracted modulo $g(x)$. Two techniques are known to us to solve the equation $a^2(x) \equiv b(x) \bmod g(x)$.

- Extracting a square root is easy if $b(x)$ has only even powers of x. It can be shown [6] that every $b(x)$ can be written in this form by addition of
$$\left(g'^{-1}(x) \cdot b'(x) \bmod g(x)\right) \cdot g(x).$$
- Squaring is a linear operation and thus extracting the square root can be done by multiplying with a precomputed matrix.

To optimize the speed of the implementation, the second alternative was chosen.

Step 2.5 It is clear that a solution of

$$\beta_1(x) \cdot d(x) \equiv \alpha(x) \bmod g_2(x)$$

will result in a solution of (†) and thus of the key equation. The solution will be unique if

$$\deg \alpha(x) \leq \deg g_2(x) - \deg \beta_1(x) - 1.$$

In case $g_2(x)$ is a power of x, Berlekamp's algorithm [2] can be used to compute $\alpha(x)$ and $\beta_1(x)$. Patterson [15] has extended this algorithm for an arbitrary $g_2(x)$, but note that his paper contains an error, as was discovered independently in [6]. An easier way to solve this equation is to apply

Appeared in *Proceedings of the 13th Symposium on Information Theory in the Benelux*, Werkgemeenschap voor Informatie- en Communicatietheorie, pp. 119–126, 1992.

Euclid's algorithm, till following conditions are satisfied:

$$\begin{aligned} \deg \alpha(x) &\leq \lfloor \frac{t}{2} \rfloor \\ \deg \beta_1(x) &\leq \lfloor \frac{\deg g_2(x) - \deg g_1(x) - 1}{2} \rfloor. \end{aligned}$$

The error locator polynomial is than given by

$$\sigma(x) = \alpha^2(x) + x \cdot g_1^2(x)\beta_1^2(x).$$

Step 3 The roots of $\sigma(x)$ indicate the error positions. For the usual parameters the most efficient way of determining the roots is simply trying all elements of the finite field, with removing the corresponding linear factor when a root is found.

Step 4 Compute m by multiplying $m \cdot S$ on the right with S^{-1}.

Most routines were coded in assembly language. Different parts were optimized such that the execution time is evenly distributed over the different steps. The resulting decryption speed is 1.7 Kbit/sec, and about 1 Kbit/sec with respect to the information bits.

6 Conclusion

A software implementation of McEliece public-key cryptosystem was presented. The key space was extended by allowing not only irreducible polynomials, but also polynomials that have no linear or repeating factors. For the parameters $m = 10$ and $t = 39$, an encryption speed of 6 Kbit/sec and a decryption speed of 1.7 Kbit/sec were obtained on a 16 MHz IBM PS/2 Model 80. We believe this can be speeded up with a factor of at least 2 by fixing all parameters (the current program is very flexible), by coding all routines in assembly language and by using the powerful 32-bit instructions of the 80386.

Acknowledgement

We would like to thank Luc Provoost, Luc Vanderghote, Frank Windmolders and Patrick Wuytens for their important contributions to this work.

References

[1] C.M. ADAMS & H. MEIJER, "Security-related comments regarding McEliece's public-key cryptosystem", *IEEE Trans. Info. Theory*, **35** (1989) 454–455.

[2] E. BERLEKAMP, *"Algebraic coding theory"*, McGraw-Hill, New York, 1968.

Appeared in *Proceedings of the 13th Symposium on Information Theory in the Benelux*, Werkgemeenschap voor Informatie- en Communicatietheorie, pp. 119–126, 1992.

[3] E. BERLEKAMP, "Goppa codes", *IEEE Trans. Info. Theory*, **19** (1973) 590–592.

[4] E. BERLEKAMP, R.J. McELIECE & H.C.A. van TILBORG, "On the inherent intractability of certain coding problems", *IEEE Trans. Info. Theory*, **24** (1978) 384–386.

[5] D.W. DAVIES & W.L. PRICE, *"Security for computer networks"*, John Wiley & Sons, 1984.

[6] P.J.M. HIN, *Channel-error-correcting privacy cryptosystems (in Dutch)*, Thesis, Delft University of Technology, 1986.

[7] F. JORISSEN, "A security evaluation of the public-key cipher system proposed by McEliece, used as a combined scheme", *ESAT report K.U.Leuven*, 1986.

[8] V.I. KORZHIK & A.I. TURKIN, "Cryptanalysis of McEliece's public-key cryptosystem", *Adv. in Cryptology, Proc. Eurocrypt'91, LNCS 547*, Springer Verlag (1991) 68–70.

[9] P.J. LEE & E.F. BRICKELL, "An observation on the security of McEliece's public cryptosystem", *Adv. in Cryptology, Proc. Eurocrypt '88*, Springer-Verlag (1988) 153–157.

[10] R. LIDL & H. NIEDERREITER, *"Finite fields"*, Addison Wesley, 1983.

[11] L. PROVOOST & L. VANDERGHOTE, *Study and implementation of McEliece's public-key cryptosystem (in Dutch)*, Thesis, ESAT K.U.Leuven, 1989.

[12] R.J. McELIECE, *"The theory of information and coding"*, (Vol. 3 of the *Encyclopedia of Mathematics and its Applications*), Addison-Wesley, Reading, MA, 1977.

[13] R.J. McELIECE, "A public-key cryptosystem based on algebraic coding theory", *DSN Progress Report 42–44*, Jet Propulsion Laboratory, Pasadena, CA, (1978), 114–116.

[14] C.S. PARK, "Improving code rate of McEliece's public-key cryptosystem", *Electronic Letters*, **25** (1989) 1466–1467.

[15] N.J. PATTERSON, "The algebraic decoding of Goppa codes", *IEEE Trans. Info. Theory*, **21** (1975) 203–207.

[16] R.L. RIVEST, A. SHAMIR & L. ADLEMAN, "A method for obtaining digital signatures and public-key cryptosystems", *Comm. ACM*, **21** (1978) 120–126.

[17] F.J. MacWILLIAMS & N.J.A. SLOANE, *"The theory of error-correcting codes"*, North Holland, 1978.

[18] J. van TILBURG, "On the McEliece cryptosystem", *Adv. in Cryptology, Proc. Crypto'88, LNCS 403*, Springer Verlag (1988) 119–131.

[19] F. WINDMOLDERS & P. WUYTENS, *Study and implementation of cryptosystems based on algebraic coding theory (in Dutch)*, Thesis, ESAT K.U.Leuven, 1990.

Appeared in *Proceedings of the 13th Symposium on Information Theory in the Benelux*, Werkgemeenschap voor Informatie- en Communicatietheorie, pp. 119–126, 1992.