Reaction Attacks Against Several Public-Key Cryptosystems

Chris Hall[1], Ian Goldberg[2], and Bruce Schneier[1]

[1] Counterpane Systems
{hall,schneier}@counterpane.com
101 E. Minnehaha Pkwy
Minneapolis, MN 55419
(612) 832-1098

[2] U.C. at Berkeley
iang@cs.berkeley.edu
Soda Hall
Berkeley, CA 94720-1776

Abstract. We present attacks against the McEliece Public-Key Cryptosystem, the Atjai-Dwork Public-Key Cryptosystem, and variants of those systems. Most of these systems base their security on the apparent intractibility of one or more problems. The attacks we present do not violate the intractibility of the underlying problems, but instead obtain information about the private key or plaintext by watching the reaction of someone decrypting a given ciphertext with the private key. In the case of the McEliece system we must repeat the attack for each ciphertext we wish to decrypt, whereas for the Ajtai-Dwork system we are able to recover the private key.

Keywords: public-key cryptosystems, lattice-based cryptosystems, error-correcting codes, Ajtai-Dwork lattice cryptosystem, McEliece.

1 Introduction

In an attempt to design cryptosystems based upon (believed) intractible problems different from factoring and discrete logarithms, several public-key cryptosystems have been presented, including the two systems in [M78,AD97]. The first system is based upon the intractibility of decoding an arbitrary error-correcting code, and the second upon finding the shortest vector in a lattice. The authors of the former rely upon the fact that the respective problem is known to be **NP**-complete. However, it is not known whether the particular instances used in their cryptosystem are equally difficult to solve. The general shortest vector problem is known to be **NP**-hard, but the lattice-based systems depend on the apparent difficulty of an easier shortest vector lattice problem (the unique shortest vector lattice problem). We refer the reader to [AD97] for more details.

In this paper we present attacks against the McEliece Public-Key Cryptosystem (PKC), a McEliece variant [HR88], the Ajtai-Dwork PKC, and a modified version of the Ajtai-Dwork PKC that appears in [GGH97]. In these attacks an attacker presents the owner of the private key with a ciphertext that may contain one or more errors (that is, the ciphertext may decrypt to a plaintext which fails a simple signature or checksum verification). By watching the reaction of the owner in order to determine whether or not the ciphertext decrypted correctly, the attacker can usually determine information about the plaintext (in the McEliece system) or the private key (in the Ajtai-Dwork systems).

We feel that this is a legitimate class of attacks that one must be careful to guard against when implementing systems that use these ciphers. In the simplest case one may have a tamper-resistance module (such as a smartcard) which contains a copy of the secret key. By feeding erroneous ciphertexts to the card, one may be able to decrypt ciphertexts without the private key or even recover the private key. In other systems one may have to rely upon social engineering to finesse the attacks, but that's a small price if one can recover the private key. The basic principle behind the attack is that someone's reaction to a question often reveals information that they didn't intend to give.

The organization of our paper is as follows. In section 2 we describe the McEliece PKC, as well as a variant of it, and attacks against them. In section 3 we describe the original Ajtai-Dwork PKC as presented in [AD97], a modified version that appears in [GGH97], and attacks against both of these systems. Finally, in section 4 we discuss the properties of these public-key cryptosystems which allowed our attacks to work in order to try and give some design criterion for new public-key cryptosystems so that they will not be vulnerable to the same sort of attack.

2 McEliece

In [M78], McEliece outlined a public-key cryptosystem based upon error correcting codes. A user chooses a $n \times k$ generator matrix G (for a (n, k) error-correcting code which can correct up to t errors), a $k \times k$ non-singular matrix S, a $n \times n$ permutation matrix P, and publishes the matrix $G' = SGP$ as his public key. To encrypt a message a user chooses a random vector Z of weight t and sends:

$$C = MG' + Z.$$

To decrypt the message, the owner of the key computes:

$$CP^{-1} = MSG + ZP^{-1}$$

and corrects the t-bit error ZP^{-1} using the known error-correction algorithm. After that M can be recovered using S^{-1}.

This system, like other systems [HR88,J83,N86], depends on the fact that in the worst case, decoding an arbitrary error-correcting code is NP-complete [BMvT78]. It is hoped that G' represents one of these difficult cases.

Since their introduction, public-key cryptosytems based on error-correcting codes have largely been of theoretical interest. They require enormous keys in order to achieve comparible security to currently implemented public-key cryptosystems based on factoring or the discrete logarithm problem. For example, the purported attack in [KT91] can break a cryptosystem based on a $(1024, 654)$ BCH code in 60 hours. Such a key would require a 654-kilobit generator matrix for part of the public key. This would seem to imply that secure keys would require one or more megabytes of storage for the generator matrix alone. Compare this to a 1024-bit RSA key which needs only a little more than 1Kb of storage and provides excellent security.

Large keys aside, error-correcting code cryptosystems have been touted as a potential saviour of public-key cryptography. These systems are based on the syndrome-decoding problem which was shown to be NP-complete in [BMvT78]. Other public-key cryptosystems, such as RSA, are based on problems such as factoring which are only thought to be difficult. It is possible that someone will discover an efficient factoring algorithm, whereas it would require proving that $P = NP$ to find an efficient general syndrome-decoding algorithm. Note, NP-completeness only implies that the worst case is hard to solve. It may be possible that the instances of problems produced by PKCs such as [M78] are much easier to solve.

Until recently, the two best attacks against McEliece's system appear in [AM87,KT91]. The attack in [AM87] relies on choosing k bits in an n-bit ciphertext that do not contain any errors. Given that the t incorrect bits are unknown, the probability of this event happening is low. However, once it occurs, one can solve for the message M using an algorithm outlined in the paper.

The attack outlined in [KT91] requires $O(n^3)$ operations for an arbitrary (n, k) code. It determines the error vector for the ciphertext C. The basic premise behind the attack is that a polynomial-time algorithm is introduced which will determine the error pattern for a received vector provided that the error pattern has weight at most $[(d-1)/2]$ (where d is the minimum distance of the code). The discovery of this algorithm does not contradict the NP-completness proof of [BMvT78] because it only corrects error patterns of weight at most $[(d-1)/2]$. If the received vector is further than $[(d-1)/2]$ from every codeword, then this algorithm cannot decode the vector to a codeword. Note, while the basic attack was outlined in [KT91], further information about the attack has failed to appear. It is possible that the authors were not able to make their attack scale as they had wished (when they present it, they had only run it against a toy problem).

An even more recent attack against the McEliece PKC was presented in [B97]. This attack relied about known linear relationships between two different ciphertexts (really their underlying plaintexts) in order to determine the error vectors used in encrypting the plaintexts. The author is quick to point out that the attack does not "break" the cryptosystem in the sense that it does not recover the private key, but it is still an interesting attack to note.

In the next subsections, we present a new attack which determines the error vector using $O(n)$ operations. It is a chosen ciphertext attack and does not require that the attacker see the resulting plaintext. The attacker only needs to be able to determine whether the chosen ciphertext had too many errors to be decoded correctly, or that it decoded to a plaintext other than the plaintext in question. The former can be distinguished in systems which send some sort of message indicating that a retransmission is necessary (such as network protocols). The latter can be distinguished in systems where a randomly chosen ciphertext has only a low probability of decrypting to a meaningful plaintext. In those cases we assume that some sort of retransmission request will also be sent.

2.1 Decoding Algorithms

Our attack rests largely upon one premise:

> *If a (n, k) error-correcting code is used which can correct t or fewer errors, then a decoder will not attempt to correct a vector which has $t+1$ or more errors.*

There are several decoders for Reed-Solomon and Goppa codes which meet this criterion, including those described in [B73,P75,SKHN76]. A common component of each of these algorithms is that they work to find an error-location polynomial $\sigma(z)$ of degree *at most* t. The decoders then determine the roots of $\sigma(z)$ in order to determine the location of the errors in the received vector (there is a one-to-one correspondance between the roots and the errors). Because $\deg(\sigma(z)) \leq t$, there are at most t roots and hence t errors that can be corrected. This means that these decoding algorithms are not be capabable of correcting $t + 1$ errors.

In principle these error-correcting codes are capable of correcting $t + 1$ errors *in some cases*. However, the conventional error-correction algorithms, based on the Berlekamp-Massey algorithm [B67,B68,M69,P75] and Euclid's Algorithm [SKHN74,SKHN76], are not capabable of correcting $t + 1$ errors, as they are of the above type. When a decoding algorithm is capable of decoding $t+1$ errors, we note that it is often possible to recover part of the permutation matrix P. This follows by observing which of the permuted error vectors of Hamming-Weight $t + 1$ are correctable and then determining the (possibly) unique permutation matrix P which would produce the permuted error vectors. One simply needs to observe the probability that bit i will be corrected and match it to a bit in the distribution on the unpermuted code.

Note, at least one system explicitly states that vectors with more than t errors should be ignored. For example, see [HW92].

2.2 Removing an Error Vector

Suppose that we have a ciphertext C which we wish to decrypt to its corresponding plaintext M. To find the message, we present an algorithm which allows the

attacker to determine the error vector Z used to encrypt M. We do this by making small one-bit modifications to C until we obtain a ciphertext C' which either will not decode at all (it contains $t+1$ or more errors) or decrypts to a different plaintext (contains t or fewer errors, but decodes to the wrong codeword).

Once we obtain such a ciphertext we can flip each bit once in n different trials, and see whether or not the resulting ciphertext decrypts correctly. If it does, then we know that we have found an erroneous bit and we keep track of it. Otherwise we know that the bit is correct. Once all erroneous bits have been found they can be flipped to produce $C - Z$.

We now give the algorithms used in full:

Algorithm A: *Determining C' (a modified ciphertext with $t+1$ errors)*

1. Let $i = 1$.
2. Flip bits 1 through i of C to form C'.
3. Have C' decrypted and determine whether or not it decrypted to M. Based on a comment made in the introduction, this should be possible (we watch the reaction of the key owner).
4. If C' did not decrypt correctly, halt the procedure and continue onto the next algorithm. Otherwise, increment i and goto step 2.

This algorithm will halt within $2t+1$ steps for in the worst case we will "correct" the t incorrect bits and introduce $t+1$ additional errors (we never correct an error that we introduce).

Once this algorithm halts we know that we have a vector C' with exactly $t+1$ errors. If it had fewer, then the decoder would have succeeded in removing the errors and decrypting to M. If it had more ($t+j$ errors for $j > 1$), then at some previous trial we must have had a vector C' with $t+1$ errors (because we know that C has at most t errors and we made one bit modifications to obtain C'). However, that vector should not have decrypted to M because that would violate the known properties of the decoder. Therefore we would have stopped the procedure sooner. So we must have that C' is a message with exactly $t+1$ errors. Note, $t < n/2$ for these error correction codes.

We then continue with the next algorithm:

Algorithm B: *(determining which bits of C' are errors)*

1. Let $i = 1$.
2. Flip bit i of C' to form C''.
3. Have C'' decrypted and determine whether or not it decrypted to M.
4. If C'' decrypted correctly, then bit i is in error. Record i, add 1 to i, and goto step 2 if $i \leq n$. Otherwise halt.

In the worst case we only need to examine the first $n-1$ bits of C'. Hence we will require at most $n-1$ queries.

Once this algorithm halts, we can flip all those bits of C' recorded in step 4 of the above algorithm. This will result in a ciphertext without any errors.

It works because any C'' which decrypts correctly must have at most t errors. However, it has Hamming Distance 1 from C' which we know has $t+1$ errors, so C'' must have exactly t errors. This means that the bit we flipped was an incorrect bit. If C'' does not decrypt correctly, then it must have $t+2$ bit errors and hence we introduced another error when we flipped bit i.

The first algorithm will halt within $2t+1$ steps and the second will require at most $n-1$ steps. So the total attack takes at most $n+2t$ queries and comparable time.

2.3 Attacking McEliece

Once we have executed the algorithms described in the previous section, we will obtain a ciphertext $C - Z$ which is error-free and we can then solve for the message M using the method outlined in [AM87]. Let C_c denote the corrected ciphertext vector.

Briefly, the algorithm selects k bits from the vector C_c to form C'_c and the corresponding k columns from the generator matrix G. The resulting matrix G' will be a $k \times k$ matrix and we know that

$$C'_c = MG'$$

so we can solve for M by inverting G'

$$M = C'_c G'^{-1}.$$

The attack as it is presented in [AM87] requires that the k selected bits be error-free in order to recover M. Consequently their attack requires $C(n,k)/C(n-t,k)$ times as much work since that is how many G' they must try on average. Others have improved on this attack, including [LB88,vT90]. However, the attacks are still expensive.

2.4 Attacking Hwang-Rao

The algorithms described in section 2.1 can also be used to attack the system described in [HR88]. The system is very similar to the McEliece cryptosystem in that it adds a random error vector E to a codeword prior to transmission. Again, it is assumed that decoding the resulting codeword is as hard as decoding an aribtrary error-correcting code.

Once we have managed to remove the error-vector from the ciphertext, we can apply an attack suggested by the Hwang and Rao in their paper. There are a couple of different versions of their system described in their paper and corresponding attacks are also described. We refer the reader to [HR88] for details.

3 Ajtai-Dwork

In [AD97] the authors present a PKC based on the assumption that the Unique Shortest Vector Problem[1] (SVP) is computationally infeasible in the worst-case

[1] Unique up to a polynomial factor.

complexity. Briefly, a public key in the system consists of $m + n$ n-dimensional vectors $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ and $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_m)$. How these vectors are chosen is beyond the scope of this paper, but details can be found in [AD97] and [GGH97].

Encryption is simple in this system, but only one bit can be encrypted at a time. However, there are multiple encryptions of both '0' and '1', and under the asumption that SVP is infeasible, it is impossible to distinguish between two ciphertexts without knowing u. To encrypt a '0' a user chooses $b_1, \ldots, b_m \in \{0, 1\}$ and computes

$$x = (\sum_{i=1}^{m} (b_i \cdot v_i)) \bmod P(\mathbf{w})$$

where $P(\mathbf{w})$ is the parallelepiped formed by the w_i. To encrypt a '1' a user chooses x uniformly in the parallelepiped $P(\mathbf{w})$.

To decrypt a message, the user computes $\tau = <x, u>$. If τ is within $1/n$ of an integer, then the ciphertext is decrypted as '0'. Otherwise the ciphertext is decrypted as '1'. It is interesting to note that an encrypted '1' will decrypt to a '0' with probability roughly $2/n$. However, such a point is beyond the scope of this paper (although it is addressed by the authors in [GGH97] and is the motivation for their modification).

We have not included our chosen-ciphertext attack against Ajtai-Dwork which allows us to recover the private key. It is a slightly more complicated version of the attack described in the next section and we leave it as an exercise to the reader to extend the attack.

3.1 Attacking the GGH Variant

In [GGH97] the authors present a modification of the Ajtai-Dwork PKC. In the original Ajtai-Dwork system as presented in [AD97], a '0' will always decrypt to a '0'. However, there is a $2/n$ probability that a '1' will decrypt to a '0' (where n is the dimension of the associated lattice). The enhanced system eliminates this problem by first specifying a vector v_{i_1} such that $<u, v_{i_1}>$ is an odd integer.

To encrypt '0', one still uniformly selects $b_1, \ldots, b_m \in \{0, 1\}$ and reduces the vector $\sum_{i=1}^{m} b_i \cdot v_i$ modulo the parallelepiped $P(\mathbf{w})$. The resulting vector x is the ciphertext. To encrypt '1', one uniformly selects $b_1, \ldots, b_m \in \{0, 1\}$ and reduces the vector $\frac{1}{2} v_{i_1} + \sum_{i=1}^{m} b_i \cdot v_i$ modulo the parallelepiped $P(\mathbf{w})$. Again, the resulting vecotr x is the ciphertext.

To decrypt, the user computes $\tau = <u, x>$ and decrypts the ciphertext as '0' if τ is within $1/4$ of some integer and decrypts the ciphertext as '1' otherwise. Our attack makes use of an oracle $O(v)$ which computes $< u, v >$ and returns the resulting plaintext. Note, such an oracle exists since we could send a legitimate message to a user which has a MAC computed as if the ciphertext in question decrypted to a '0'. If the ciphertext decrypts correctly, then we know that $O(v) =$'0'. Otherwise we know that $O(v) =$'1'.

Given that we can send a user a vector v such that $v_i = x$ for some non-zero x and $v_j = 0$ for $i \neq j$, we can focus on one component of u at a time. Hence we assume a simplified oracle $O_i(x)$ which computes $u_i \cdot x$ and returns '0'

(resp. '1') if the result is (resp. is not) within 1/4 of an integer. Assuming that $|u_i| = d_0.d_1 \ldots d_r$ we can determine $|u_i|$ using the following algorithm:

Algorithm D: *(determining $|u_i|$)*

1. Let $j = 0$.
2. If $O_i(2^{j+1}) =$'0', then $d_j d_{j+1} \in \{00, 11\}$. Otherwise $d_j d_{j+1} \in \{01, 10\}$.
3. Let $j = j + 1$. If $j \leq r$ goto step 2.
4. Pick $d_0 = 0$ and choose the remaining d_j appropriately. [2]

This algorithm will determine $|u_i|$ within 2^{-r} of the correct value. That's because $1.0 = 0.\overline{1}$ in binary and our test (in step 2) may choose the alternate expansion for the remaining digits of $|u_i|$.

It's important to understand that this attack determines the projection of the hyperplanes onto all of the coordinate axis. For any particular axis it sets all other coordinates to zero and performs a binary search on the boundary midway between the hyperplane through the origin and the next hyperplane along the axis. However, there's nothing that says the attacker must use the same notion of coordinates axis as everyone else. In fact, she could rotate the axis and perform a binary search on the modified axis (that's in fact what we do below in order to determine the sign of the u_i). Therefore one cannot counter this attack simply by refusing to decrypt all ciphertext such that $u_i = 0$ for all but one i.

Once we determine $|u_i|$ for all i it remains to determine the respective signs. Since the private key u is equivalent to the private key $-u$ we assume without loss of generality that $u_1 \geq 0$ (more properly that the first non-zero $u_i > 0$, but we can permute the vectors and bases appropriately so that $u_1 > 0$). Given $|u_j|$ for some $j > i$, we can compute $u_i + u_j$ for both values of the sign for u_j. The result will be different in each case and let k be the first bit that both sums differ in. Then by computing

$$O(0, \ldots, 2^{k-1} \cdot u_i, \ldots, 2^{k-1} \cdot u_j, \ldots)$$

we can determine the sign of u_j. That is, we use the oracle to determine which sum is correct when considering bit k and hence whether $u_j > 0$ or $u_j < 0$.

Using this the oracle $O(v)$ (which was used to construct $O_i(x)$) we are able to compute u. Notice that each invocation of the oracle returns one bit of information. Correspondingly we use each invocation of the oracle to determine one bit of the private key. Hence our attack determines the private key u using the information theoretic minimum number of oracle invocations. Therefore we know that this is the best possible attack against u using an oracle.

We should point out that this is a *non-adaptive* attack. Unfortunately, one must modify the attack to use one adaptation in order to achieve the information theoretically minimal attack. This is due to the fact that one must see

[2] If $|u_i| = 1$ for some i, then $u_j = 0$ for all $j \neq i$. This situation occurs with negligible probability, but it is also easy to detect, so we assume that $|u_i| < 1$ for all i and hence $d_0 = 0$.

the $|u_i|$ before one can determine ciphertexts which will give their signs. From a reaction attack standpoint, this is a minute difference since we gauge the reaction to each ciphertext separately. However, it makes a difference from a chosen-ciphertext standpoint because non-adaptive attacks are considered more powerful than adaptive attacks.

4 Future Systems

The success of our attacks rely upon a common weakness in the PKCs we examined: given a ciphertext C, an attacker can produce a second ciphertext C' which has non-neglible probabilities of decoding to the same plaintext and to a different plaintext. For the McEliece system of [M78], whether or not C' decodes correctly gives information. For the Ajtai-Dwork and Goldreich-Goldwasser-Halevi systems of [AD97,GGH97], whether or not C' decodes correctly gives information about the private key.

Each of these systems could be considered a closest-point cryptosystem. That is, the ability to decode a ciphertext depends upon the ability to determine the closest "point" to the ciphertext in some linear space. For error-correcting code systems, this equates to the ability to determine the closest codeword in the linear space of codewords. For lattice-based systems, this equates to finding the closest point in a lattice. In all of these systems one could consider the class of ciphertexts corresponding to a particular plaintext to be a sphere surrounding a point in the respective space (where the boundaries of the sphere are determined by an appropriate distance metric).

By examining ciphertexts which are close to each other in the space (but possibly in different classes) we can determine the boundaries of this sphere and hence the center of the sphere. For some of the systems this will only give us the closest point in the respective space. In those systems, an attacker can already produce points in the space at will using the public key. So determining the closest point doesn't give any more information about the private key that the public key does. For other systems, the security of the system relies upon the inability to determine points in the respective space. Hence the ability to determine a closest point leads to the ability to determine the private key.

We feel that any other public-key cryptosystem with these properties will be vulnerable to the same sorts of attacks we present here.

5 Conclusion

We feel that the existance of these attacks effectively limits these ciphers to theoretical considerations only. That is, any implementation of the ciphers will be subject to the attacks we present and hence not safe. We should point out that we have *not* broken these cryptosystems in the sense that we are able to mathematically derive the private key. Instead our attacks rely upon the fact that someone within the system knows the private key (and hence the answer to the intractible problem). Therefore we would encourage further research into

the properties of these cryptosystems in order to determine if a modified cryptosystem can be designed with the same theoretical basis and that is also safe against our attacks.

References

[AM87] C. M. Adams, H. Meijer, "Security-Related Comments Regarding McEliece's Public-Key Cryptosystem," *Advances in Cryptology—Proceedings of CRYPTO '87*, Springer-Verlag, 1988, pp.224–230.

[AD97] M. Ajtai, C. Dwork, "A Public-Key Cryptosystem with Worst-Case/ Average-Case Equivalence," In *29th ACM Symposium on Theory of Computing*, 1997, pp. 284–293.

[B67] E.R. Berlekamp, "Nonbinary BCH Decoding," paper presented at the 1967 International Symposium on Information Theory, San Remo, Italy.

[B68] E.R. Berlekamp, *Algebraic Coding Theory*, New York: mcGraw-Hill, 1968.

[B73] E.R. Berlekamp, "Goppa Codes," *IEEE Transactions on Information Theory*, Vol. IT-19, No. 5, pp. 590–592, September 1973.

[BMvT78] E.R. Berlekamp, R.J. McEliece, H. van Tilborg, "On the Inherent Intractability of Certain Coding Problems," *IEEE Transactions on Information Theory*, Vol. 24, 1978, pp. 384–386.

[B97] T. Berson, "Failure of the McEliece Public-Key Cryptosystem Under Message-Resend and Related-Message Attack," *Advances in Cryptology—CRYPTO '97 Proceedings*, Springer-Verlag, 1997, pp. 213–220.

[GGH97] O. Goldreich, S. Goldwasser, S. Halevi, "Eliminating Errors in the Ajtai-Dwork Cryptosystem," *Advances in Cryptology—CRYPTO '97 Proceedings*, Springer-Verlag, 1997, pp. 105–111.

[HR88] T. Hwang, T.R.N. Rao, "Secret Error-Correcting Codes (SECC)," *Advances in Cryptology—CRYPTO '88 Proceedings*, Springer-Verlag, 1990, pp. 540–563.

[HW92] L. Harn, D.C. Wang, "Cryptanalysis and Modification of Digital Signature Scheme Based on Error-Correcting Codes," *Electronics Letters*, v. 28, n. 2, 10 Jan 1992, p. 157–159.

[J83] J.P. Jordan, "A Variant of a Public-Key Cryptosystem Based on Goppa Codes," *Sigact News*, 1983, pp. 61–66.

[KT91] V.I. Korzhik, A.I. Turkin, "Cryptanalysis of McEliece's Public Key Cryptosystem," *Advances in Cryptology—EUROCRYPT '91 Proceedings*, Springer–Verlag, 1991, pp. 68–70.

[LB88] P.J. Lee, E.F. Brickell, "An Observation on the Security of McEliece's Public-Key Cryptosystem," *Advances in Cryptology—EUROCRYPT '88 Proceedings*, Springer–Verlag, 1988, pp. 275–280.

[LDW94] Y.X. Li, R.H. Deng, X.M. Wang, "On the Equivalence of McEliece's and Niederreiter's Public-Key Cryptosystems," *IEEE Transactions on Information Theory*, Vol. 40, 1994, pp. 271–273.

[M69] J.L. Massey, "Shift Register Synthesis and BCH Decoding," *IEEE Transactions on Information Theory*, Vol. IT-15, No. 1, pp. 122–127, Jan. 1969.

[M78] R.J. McEliece, "A Public-Key Cryptosystem Based on Algebraic Coding Theory," *Deep Space Network Progress Report 42-44*, Jet Propulsion Laboratory, California Institute of Technology, 1978, pp. 104–113.

[N86] H. Niederreiter, "Knapsack-Type Cryptosystems and Algebraic Coding Theory," *Problems of Control and Information Theory*, v. 15, n. 2, 1986, pp. 159–166.

[P75] N.J. Patterson, "The Algebraic Decoding of Goppa Codes," *IEEE Transactions on Information Theory*, Vol. IT-21, No. 2, pp. 203–207, March 1975.

[SKHN74] Y. Sugiyama, M. Kasahara, S. Hirasawa, T. Namekawa, "A method for solving the key equation for decoding Goppa codes," presented at the IEEE Int. Symp. Information Theory, Notre Dame, Ind., Oct. 27–31, 1974.

[SKHN76] Y. Sagiyama, M. Kasahara, S. Hirasawa, T. Namekawa, "An Erasures-and-Errors Decoding Algorithm for Goppa Codes," *IEEE Transactions on Information Theory*, pp. 238–241, March 1976.

[vT90] J. van Tilburg, "On the McEliece Cryptosystem," *Advances in Cryptology—CRYPTO '88 Proceedings*, Springer-Verlag, 1990, pp. 119–131.