A Variant of a Public Key Cryptosystem Based on Goppa Codes

John P. Jordan

ABSTRACT

This paper suggests a way in which an interesting Public Key Cryptosystem based on Goppa Codes introduced by R. J. McEliece can be modified and used in a classical way to yield several advantages. The primary benefit is that the rate of the code (or data expansion reciprocal) can be increased by approximately 75%. Secondly (for public key and classical versions), by using a polynomial with no linear or repeated factors instead of an irreducible one to generate the Goppa Code, the code parameters remain the same and the decoding apparatus can be used to test candidate generators. Lastly, a wider range of transformations is made possible. No degradation in security is incurred.

John P. Jordan, 3L-315
Bell Laboratories
307 Middletown/Lincroft Road
Lincroft, New Jersey 07738

I. Introduction

A variant of an interesting Public Key Cryptosystem based on Goppa codes [1] is described. The advantage of this system is increased code rate and the disadvantage (if actually so) is that it is no longer a Public Key Cryptosystem even though the keys (kept secret now) can be distributed publicly using the original system. However, this should be no drawback and is more than compensated by the increased code rate during the data transmission sessions, making it a good alternative to the Data Encryption Standard. The security of the system depends on a number of factors but principally on the additional feature that the keys are kept hidden in the variant system. The Goppa polynomials can be chosen to be ones with no repeated factors and has no effect on parameter size for the binary case here in either the Public Key or variant system provided the polynomials also have no roots in the coefficient field. The Euclidean Algorithm in the decoding apparatus can then be used to test the polynomials and their derivatives for common factors. In addition, a wider range of transformations are made possible affording additional security if necessary.

II. The McEliece Cryptosystem

Following [1], the cryptosystem designer chooses an integer m and an irreducible polynomial of degree t over $GF(2^m)$. Corresponding to this polynomial is a Goppa code of length $n = 2^m$, dimension $k \geq n-tm$, capable of correcting all

patterns of t or fewer errors. A discussion of Goppa codes is given in [2, chapter 8] and a fast algorithm due to Patterson for decoding them is given in [2, problem 8.18]. (see [3] for decoding relative to Goppa polynomials with no repeated factors). After a (kXn) generator matrix G is obtained, it is pre-multiplied by a random dense (kXk) non-singular matrix S and post-multiplied by a random (nXn) permutation matrix P. The designer then makes public the matrix $G' = SGP$, which generates a linear code with the same rate and minimum distance as the code generated by G. To communicate with the designer, the data is broken up into k-bit blocks. If u is one such block, then $x = uG' + z$ is transmitted where z is a random vector of length n and weight t generated by the transmitting party. When x is received, the designer computes $x' = xP^{-1}$ where P^{-1} is the inverse of P. x' is a codeword in the previously chosen Goppa code. Patterson's algorithm is then used to get $u' = uS$ and finally $u = u'S^{-1}$.

III. Variant System

The above system requires no prior change of keys in a hidden way in order to communicate. However, the parameters of the system must be made large and the rate of the code is small. An example is $n = 2^{10}$ and $t = 50$, which gives a code rate approximately equal to 1/2. The variant system works as follows. A large public generator matrix G' as above is used by one party to distribute a polynomial with no linear

(optional) or repeated factors (as opposed to it being irreducible), a transformation S (not necessarily a matrix), and a permutation matrix P (can be a sequence of different numbers) to the other party in the communication. Preferably, the polynomial should have no zeros in the $GF(2^m)$ field used so as to make the dimension k and rate of the code as large as possible. The only requirement on the transformation S is that it be a one-one map of the set of all 2^k binary data vectors onto itself. For example, S could be obtained by considering the data vector as an element of $GF(2^k)$ and raising it to some power that is relatively prime to $2^k - 1$. The choice of a function S would be based on a tradeoff between ease of implementation, computation speed, data expansion, and security achieved. All these transformations (polynomial, S and P) are kept secret and therefore a higher rate code can be obtained for transmitting the data. The secrecy allows the reduction of t, the number of errors introduced into the data. If n is kept at the same large value then the code rate will be .9 for t = 10. If a smaller n is more desirable then this will produce a slight reduction in code rate. An example of the sizes that can be used are $n = 2^7$ and t = 3 to get k = 107 and a rate of .84 or $n = 2^8$ and t = 5 to get k = 216 and a rate of .84. The same hardware or software could be used for all parameter sizes. The decoding apparatus includes the Euclidean Algorithm, which can be used to determine if the Goppa polynomial has any repeated factors.

IV. Discussion of Security

After the transformations are distributed, a secret generator matrix G´ is formed as in the public system. The analysis will assume that the P matrix is not used and that S is a matrix. This will provide a conservative estimate of the security. The security of the secret system can be analyzed as follows. Because the elements of the system are kept secret, a plaintext - ciphertext attack is the only possible one. However, a large amount of plaintext and corresponding ciphertext must be available. Based on the attacks outlined below, this will be of the order k^6 and this has a low probability of being achieved. Assuming a sufficient amount of data has been obtained somehow, there are two approaches that the cryptanalyst might consider. The first is to use the data to obtain k^2 equations in the k^2 unknown matrix elements in the columns of G′ corresponding to the information bits of the codeword. This type of attack is suggested in [1] as the best attack to take for a public G´. Following [1], the probability of finding k^2 equations with no error in them is given by $(1 - t/n)^{k*k}$ and The work factor involved in solving k^2 equations is given by k^6. Therefore the expected work factor to get the k^2 matrix elements is given by $k^6/(1 - t/n)^{k*k}$. The $n = 2^7$ and $n = 2^8$ examples above give exceedingly high work factors. A better approach is to try to guess the Goppa polynomial and then decode to eliminate the errors, solve the equations for the k^2 unknown matrix

elements and then use other known plaintext and ciphertext to test the Goppa polynomial and matrix elements for correctness. The work factor associated with this approach would be bounded below by $(k^6)(2^{mt}/t)$ where the second factor in the product is the number of degree t irreducible (minor impact using this bound instead since t is small) polynomials over the Galois Field $GF(2^m)$. The $n = 2^7$ example gives a work factor of 10^{18} and the $n = 2^8$ example gives one of 10^{25}. Both these cases exceed the work factor of the Data Encryption Standard. More frequent changes of key can be done to enhance security.

References

[1] R. J. McEliece, "A Public Key System Based on Algebraic Coding Theory", JPL DSN Progress Report 42-44, Jan. and Feb. 1978, pp 114-116.

[2] R. J. McEliece, The Theory of Information and Coding, (Vol. 3 of The Encyclopedia of Mathematics and Its Applications) Reading, Mass., Addison-Wesley, 1977.

[3] Y. Sugiyama, m. Kasahara, S. Hirasawa, and T. Namekawa, "A Method for Solving Key Equation for Decoding Goppa Codes", Inform. Contr., Vol 27, pp. 87-99, Jan. 1975.