Short McEliece-based Digital Signatures

Nicolas Courtois, Matthieu Finiasz, Nicolas Sendrier
Projet Codes, INRIA Rocquencourt
BP105, 78153 Le Chesnay - Cedex France
`courtois@minrank.org, {Matthieu.Finiasz, Nicolas.Sendrier}@inria.fr`

Abstract — **The McEliece public key encryption scheme [1] was invented in 1978. Though very secure (all known attacks are still exponential) it has not been very successful as far as applications are concerned. We show here how this scheme could be used for a new application: short digital signatures.**

I. Introduction

For a long time, it has been a general belief that signature was impossible using code-based cryptosystems. Especially because signature requires to decode a random word (obtained through hashing), which in most cases (in fact all the interesting cases) is impossible. The main idea of our scheme is that instead of being able to decode any random word it is enough to decode a small part of them.

II. A Brand New Signature Scheme

In place of the McEliece cryptosystem we will use Niederreiter's variant [2]. It is completely equivalent on a security point of view, but it manipulates syndromes instead of code words. Hence a typical signature scheme would be the following:

- hash the document to sign into a syndrome,
- decode it using the trapdoor (the structure of the underlying Goppa code),
- the clear-text obtained can be used as the signature,
- to verify the signature: rehash the document and encode the clear-text with the public parity check matrix,
- signature is valid if both syndromes are the same.

The only problem lies at the second step: it is necessary to decode a random syndrome. With the original McEliece parameters this has one chance in 2^{216} of being possible. Therefore we need to modify the first two steps of the process. We first concatenate a counter to the document, and hash both of them at the same time. Then we try to decode this random syndrome. If it is not decodable we just increment our counter, re-hash and try again, with a new random syndrome, and so on until we obtain a valid syndrome. We will then need to append the counter value to the signature so it can be verified.

III. The Right Parameters

The parameters McEliece used for the Goppa code are not suitable for our scheme: they would require far too many decoding attempts. We need to find parameters for which the average number of attempts is small and which, at the same time, have a good security (see [3] for the best attacks).

The average number of tries needed to find a decodable syndrome is approximately $t!$ (where t is the number of errors corrected by the underlying binary Goppa code). More details can be found in [4]. If we don't want a signature time of more than an hour, we will have to take a t not greater than 10.

If we want our scheme to be secure we will then need a large n (the code length) of at least 2^{16} for $t = 9$ or 2^{15} for $t = 10$. In both cases an attacker will need at least 2^{80} binary operations to forge a signature which is enough to be considered secure.

IV. Signature Length

In the Niederreiter scheme a clear-text (and therefore one of our signatures) is a word of length n and weight t. With $t = 9$ and $n = 2^{16}$ about 125 bits are required to represent such a word. To that we need to add an average of 19 bits for the counter. Signatures of this length have a very fast verification (about 40 CPU operations). We have the following trade-offs between signature length and verification time:

- 132 bits signatures $\simeq 1\mu$s verification
- 119 bits signatures $\simeq$ 1ms verification
- 81 bits signatures $\simeq$ 30s verification

Each of them has the same security level.

The only way to shorten the signature is to transmit less information. The signature being a word of low weight, it simply contains the positions of the non-zero bits. We omit r of these positions to shorten the signature. The verification will then require the decoding of an error of weight r using brute force. The first trade-off corresponds to $r = 1$, the second to $r = 2$. In the third trade-off we use an additional trick: we partition the support of the code into blocks of 16 bits and only give the positions of the non-zero blocks. We also omit 3 positions so that the verifier simply has to perform a decoding, but this time on a shortened code. The cost of the Gaussian elimination required is less than the cost of the brute force decoding for 3 positions, so partitioning the support won't increase the verification time significantly. However, this reduces the signature length of 24 bits.

V. Conclusion

We demonstrated how to achieve digital signature with the McEliece public key cryptosystem. In the same time we obtain a secure signature scheme with either very short signatures or a very fast verification. An analysis of security and asymptotic behavior is given in [4]. It shows that this system is and should remain secure, as long as there is no major breakthrough in general purpose decoding algorithms.

References

[1] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Prog. Rep.,* Jet Prop. Lab., California Inst. Technol., Pasadena, CA, pages 114–116, January 1978.

[2] H. Niederreiter. Knapsack-type crytosystems and algebraic coding theory. *Prob. Contr. Inform. Theory*, 15(2):157–166, 1986.

[3] A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code... *IEEE Transactions on Information Theory*, 44(1):367–378, January 1998.

[4] N. Courtois, M. Finiasz, and N. Sendrier. How to achieve a McEliece-based digital signature scheme. In *ASIACRYPT 2001*, Springer-Verlag, 2001.