Improved Secure Implementation of Code-Based Signature Schemes on Embedded Devices

Arnaud Dambra[1]*, Philippe Gaborit[2], Mylène Roussellet[3], Julien Schrek[2], and Nicolas Tafforeau[4],*

[1] UL Transaction Security, United Kingdom
`arnaud.dambra@ul.com`
[2] XLIM-CNRS, Université de Limoges, France
`{philippe.gaborit,julien.schrek}@xlim.fr`
[3] INSIDE Secure, Meyreuil, France
`mroussellet@insidefr.com`
[4] CLEARSY, Paris, France
`nicolas.tafforeau@clearsy.com`

Abstract. Amongst areas of cryptographic research, there has recently been a widening interest for code-based cryptosystems and their implementations. Besides the *a priori* resistance to quantum computer attacks, they represent a real alternative to the currently used cryptographic schemes. In this paper we consider the implementation of the Stern authentication scheme and one recent variation of this scheme by Aguilar *et al.*. These two schemes allow public authentication and public signature with public and private keys of only a few hundreds bits. The contributions of this paper are twofold: first, we describe how to implement a code-based signature in a constrained device through the Fiat-Shamir paradigm, in particular we show how to deal with long signatures. Second, we implement and explain new improvements for code-based zero-knowledge signature schemes. We describe implementations for these signature and authentication schemes, secured against side channel attacks, which drastically improve the previous implementation presented at Cardis 2008 by Cayrel *et al.*. We obtain a factor 3 reduction of speed and a factor of about 2 for the length of the signature. We also provide an extensive comparison with RSA signatures.

Keywords: Code-based Cryptography, Stern's Scheme, Signature, Authentication.

1 Introduction

Most of currently used public key schemes like RSA, ElGamal or ECDSA rely on the complex number theory based problems of integer factorization and the calculation of discrete logarithms [14,28]. Besides these well known problems, there exist a few other type of difficult problems on which it is possible to base cryptosystems, the most well known are: lattice problems, mutivariate problems or code-based problems. In this paper we are interested in code-based problems.

There are three main reasons to consider code-based cryptosystems. The first reason is that code-based cryptosystems are *a priori* resistant to quantum computer attack, which is not the case for classical number theory systems. The second reason is that code-based cryptoystems are usually faster than number theory based cryptosystems and have a real interest for low-cost cryptography. At last, the third reason is more semantic: it is not a good idea to put all its eggs in the same basket, especially for cryptography, since one is never sure of what the future may be and what new attack may appear. For instance very recently a new algorithm was found by Barbulescu *et al.* [3], which gives an heuristic algorithm with quasi-polynomial complexity for attacking the discrete logarithm problem in finite fields of small characteristic. Of course it does not correspond to the type of

* Part if this work was carried out when the author was doing his Master's thesis at Inside

parameters used in practice but still, it shows that one never knows what may happen in terms of new attacks for classical number theory based cryptography.

These three reasons make code-based cryptography a good alternative candidate for cryptosystems and makes very relevant to have real secure implementations of such schemes. Moreover in recent years there has been a burst of activity on this field of research and in particular on the complexity of attacking such cryptosystems which greatly improved the confidence in the security of such systems.

Code-based cryptography was initiated by McEliece in 1978 [23]. He defines an encryption algorithm based on the difficulty of decoding a linear code. Authentication schemes based on error correcting codes first appeared in the 1990's. The most well-known one was proposed by Stern at CRYPTO 1993 [29]. This zero-knowledge authentication scheme is a multi-round protocol in which each round is performed in three passes. Its security relies on the syndrome decoding problem which has been proven to be NP-complete. Studies on code-based cryptography mainly target theoretical aspects of the cryptosystems. The complexity associated with the difficulty of decoding a random code has been reduced in [4] and several improvements have also been introduced to make these schemes more practical by reducing the size of the keys [13] or the communication cost for authentication and signature schemes [9,24]. These improvements have allowed researchers to consider the use of code-based cryptosystems in embedded devices. A first smart-card implementation of the Stern scheme was realized in 2008 by Cayrel et al. on an 8051-architecture [8]. The authors presented a secure implementation using quasi-cyclic codes and developed an authentication in 6 seconds and estimated a signature could be computed in 24 seconds. In 2009 Eisenbarth et al. published the "MicroEliece", an implementation of McEliece scheme on an 8-bit AVR microprocessor and recently an implementation was presented in CHES 2013 on a new class of codes the QC-MDPC [17].

Our contribution. The contributions of the paper are twofold. First, whereas previous implementation of zero-knowledge code-based schemes were mainly focused authentication, we give a detailed implementation both of authentication and signature protocols on a smart card. The signature and authentication implementations aspects are different as signatures have to deal with larger data than for authentication in the Fiat-Shamir paradigm, which needs a really specific approach. We describe how to implement a code-based signature in a constrained device through the Fiat-Shamir paradigm and, in particular, we show how to deal with long signatures. Second, we implement and explain new improvements for code-based zero-knowledge signature schemes, some theoretical improvements coming from by Aguilar *et al.* [24] and some being new. We apply these improvements on the classical Stern scheme and on the recent scheme by Aguilar *et al.* (denoted AGS in the following).

Overall, our improvements allow signature lengths to be reduced by more than 30% for the Stern scheme, and by 15% for the AGS scheme compared to [24]. Based on the side-channel analysis realized by Cayrel *et al.* at CARDIS 2008 [8] and our new improvements, we obtain secure implementations for these schemes, which are three times faster than the previous ones. If we compare with protocols based on RSA cryptosystem, the results obtained show that for an 80-bit security the code-based signature is slightly faster than RSA without a coprocessor, and for a 110-bit security the signature with codes is more than 3 times faster.

We also highlight the fact that our implementation provides an efficient and practical code-based authentication scheme on a real industrial product.

Roadmap. This paper is organized as follows. Section 2 recalls some authentication protocols based on error correcting codes and the constraints faced by cryptographic implementations in embedded devices. Section 2.3 presents the efficient implementations we obtained for authentication and signature schemes with theoretical improvements and practical results. We extend the analysis to secure implementations in section 4. We conclude this work in section 5.

2 Background

2.1 Theoretical background on code-based cryptography

In number theory cryptosystems security is based on factorisation or discrete logarithm problems. In coding theory there exists other difficult problems such as the Syndrome Decoding.

Definition 1 (Syndrome Decoding Problem). *Given a matrix H of size $k \times n$ over $\mathbb{F}_2$, a syndrome s in $\mathbb{F}_2^k$ and a positive integer w.*
Is it possible to find a word x in $\mathbb{F}_2^n$ of weight w such as $Hx^t = s^t$?

Security of code-based signature and authentication with zero-knowledge. The security of the syndrome decoding problem is now very well known as showed by a list of papers recently published in the best cryptographic conferences [6,4]. There are different types of protocols. Code-based encryption schemes are usually based on the Mc-Eliece settings, which consists in hiding a decoding structure. The use of compact matrices (such as cyclic matrices) makes these schemes more vulnerable to structural attacks and in practice some parameters have been attacked in such schemes (even if the schemes in themselves still hold) [11,5]. In authentication schemes the situation is different. There is no hidden structure only a random double-circulant matrix and a general instance of the problem. The difficulty of the problem relies solely on this general instance, as it is the case for protocols based on the discrete logarithm. To our knowledge, there are no known attacks on this type of random double-circulant code. One simple generic attack, the DOOM attack [28], exists but it only decreases the security by a small factor (proportional to the square root of the code length). The situation is comparable to lattice-based cryptography for which no attack has been found using the compact structures of NTRU or ring-LWE.

Previous Work. We recall here some identification protocols based on the syndrome decoding problem. In 1993 Stern presented [29] the first code-based zero-knowledge protocol. It consists of a three-pass interaction between a prover (denoted P) and a verifier (denoted V). P generates 3 commitments from secret elements and sends them to V. From a random number provided by V, and referred to as the challenge, he reveals some values (answers) allowing V to check 2 of the commitments. The probability of an adversary pretending to be the prover is 2/3. To reach an appropriate confidence level the protocol is repeated several times. Generally we consider that the probability of cheating must be lower than 2^{-16} which means this protocol must be repeated at least 28 times. Stern also presented another protocol in 5 passes allowing to reduce the probability of cheating to 1/2. In this case only 16 rounds are necessary.

In 1996 Veron [30] proposed another identification scheme with a probability of cheating close to 1/2. In this scheme the communication cost was reduced but the key length was increased.

The main drawback of these schemes is that they rely on matrix storage which represents more around $120,000$ bits. The main improvement to reduce the matrix size was proposed by Gaborit and Girault in [13]. They introduced the use of double circulant matrices which reduces memory requirements to a few hundred bits. A circulant matrix has the property of being fully generated from its first line by using rotations. A double circulant matrix is the concatenation of the identity matrix and a circulant matrix. Therefore only k bits are needed to generate a $k \times 2k$ matrix.

More recently, two new protocols were proposed by Cayrel *et al.* [9] and Aguilar *et al.* [24] (we will denote these schemes CVE and AGS). They use the notion of quasi-cyclic codes to obtain compact keys of a few hundred bits.

Table 1 summarizes the parameters of these schemes.

	Stern 3	Stern 5	Veron	CVE	AGS
Rounds	28	16	28	16	18
Matrix size (bits)	122,500	122,500	122,500	32,768	350
Public key (bits)	350	2,450	700	512	700
Private key (bits)	700	4,900	1,050	1,024	700
Communication cost (bits)	41,066	62,272	35,486	31,888	20,080

Table 1. Comparison of several zero-knowledge schemes for a 2^{-16} cheating probability

In this paper we consider the implementation of the AGS protocol, which is the most efficient and most recent.

Fiat-Shamir paradigm. An identification scheme can be transformed into a signature scheme through the Fiat-Shamir paradigm [12]. The main difference between authentication and signature is the number of characters involved. An authentication protocol consists of an interaction between a prover and a verifier while a signature only needs one signer. The idea of Fiat and Shamir is to generate the challenge with a random oracle. The general principle is the following. The signer first generates all the commitments at once. By applying an oracle to these elements, he obtains challenges that are used to compute the answers to send to the verifier. In practice we use a hash function to simulate the random oracle and the challenges are deduced from the hash value of the message and the commitments.

The Fiat-Shamir paradigm has been proved secure for the three-pass protocols in [27] and more recently for five-pass protocols in [2].

2.2 Implementation Constraints in Embedded Devices

Implementations in embedded devices such as smart cards face two main issues: efficiency and tamper resistance. The last decade has seen significant improvements in smart card technology (increased memory size, clock frequency, technology shrink ...). However developing cryptographic libraries for these devices remains a technical challenge. It requires constant compromises between memory size (volatile and non-volatile) and execution time: the best algorithm in a minimum of time using a minimum of memory.

The cryptographic functions must also be protected against side-channel attacks. Electronic devices are composed of thousands of logical gates that switch differently depending on the executed operations and the data manipulated. Power consumption or electromagnetic radiation generated by the gates switches, can provide information on the executed instruction and the manipulated data. The goal of side-channel attacks is to analyse this leakage in order to recover secret information.

Side-channel attacks were introduced by Kocher in 1996 [19] and completed in [20]. The Simple Power Analysis (SPA) consists in finding information on secret elements by observing a single power consumption trace. Analysis of this trace can provides information on the structure of an algorithm and the of implementation used. The Differential Power Analysis (DPA) requires several power consumption traces obtained from different messages. It consists in validating an hypothesis on some key bits using statistical tests applied to the traces. Among the possible statistical tests we can mention the correlation coefficient [7] or the mutual information [15]. An extension of DPA called High-Order Differential Power Analysis [25] combines many instants of the power traces (instead of one).

The most common countermeasure to protect cryptographic algorithms against statistical attacks is the blinding method [1,26]. A random value, called the mask, is applied to the internal data to mask the operations. In this case the attacker cannot validate a key bit hypothesis as another unknown value has been added to the manipulated data.

2.3 Background on protocols: the Stern protocol and its Aguilar-Gaborit-Schrek variation

The Stern protocol, which is fully described in [8], is a 3-pass zero-knowledge protocol with a cheating probability of 2/3. As it is well known, we focus on the description of the AGS variation in this section. The details of the Stern scheme can be found in the appendix.

In 2011 Aguilar *et al.* [24] published a new identification scheme based on Veron's protocol (which is already a variation of the Stern authentication scheme). This protocol uses the fact that, because of the cyclicity, it is possible to decrease the probability of cheating from 2/3 to 1/2 asymptotically - the authors propose practical parameters for which the cheating probability is 0.53. Moreover the authors present a convincing sketch of proof of soundness. The possibility of reducing the cheating probability for one round is very important as it is directly related to overall probability of cheating. For the Stern derived signature, it is necessary to consider 140 rounds to reach a 2^{80} level of security when only 88 are needed for AGS because of the reduced cheating probability.

The protocol consists in a five-pass scheme with a probability of cheating close to 1/2 and uses the cyclicity properties of a double-circulant code.

Let $G = (I_k|A)$ be a double circulant matrix of size $k \times n$ where I_k refers to the identity matrix of size k, A is a circulant $k \times k$ matrix defined by a $k-$bit vector $a = (a_0 \dots a_{k-1})$ and n is equal to $2k$. Let e be a random element in $\mathbb{F}_2^n$ of weight w and m a random element of $\mathbb{F}_2^k$. We obtain:

private key (e, m)
public key (G, x, w) with $x = mG + e$

Let define ρ_s a rotation of s bits of an element of $\mathbb{F}_2^k$. Thanks to the double circulant property of the code, the rotation ρ_s applied on $x = (x_l, x_r)$ can be translated on vectors m and $e = (e_l, e_r)$:

$$x = mG + e \Leftrightarrow (\rho_s(x_l), \rho_s(x_r)) = \rho_s(m)G + (\rho_s(e_l), \rho_s(e_r)) = m_sG + e_s$$

Figure 1 describes one round of the AGS protocol where h denotes a hash function. This protocol needs to be repeated 18 times to obtain a probability of cheating lower than 2^{-16}.

1. (First commitment) P randomly chooses $y \in \mathbb{F}_2^k$ and a permutation σ of $\{1, 2, \ldots, n\}$. P computes and sends to V the commitments c_1 and c_2:

   $$c1 = h(\sigma) \qquad c2 = h(\sigma(yG))$$

2. (First part of the challenge) V sends a value s between 0 and $k-1$ to P (it represents the number of shifted positions).

3. (Second commitment) P builds $e_s = \rho_s(e)$ and sends the last part of the commitment:

   $$c3 = h(\sigma(yG \oplus e_s))$$

4. (Challenge) V sends $b \in \{0, 1\}$ to P.

5. (Answer) Two possibilities:
   - b=0: P reveals $(y \oplus m_s)$ et σ
   - b=1: P reveals $\sigma(yG)$ et $\sigma(e_s)$

6. (Verification) Two possibilities:
   - b=0: V verifies validity of $c1$ and $c3$
   - b=1: V verifies validity of $c2$, $c3$ and that the weight of $\sigma(e_s)$ is ω

7. Repeat N times step 1 to 6 to reach a good security level.

Fig. 1. Identification scheme from Aguilar, Gaborit and Schrek

The Fiat-Shamir paradigm allows a transformation of this protocol into a signature scheme. In order to reach a security of 2^{80}, the signature must perform 88 rounds.

Another constraint regarding implementation in embedded devices is related to the size of the elements sent. The following formula provides the number of bits involved in the communication for the authentication and the signature depending on the size k of elements.

$$Comm_{AGS}(k) = N \times (3 \cdot l_{hash} + (k + l_\sigma + 2 \cdot k + 2 \cdot k)/2) \tag{1}$$

where N refers to the number of rounds performed, l_{hash} the length of the hash value and l_σ the length of parameters needed for the pseudo random permutation σ.

Similarly the communication cost for the Stern protocol is given by the following formula:

$$Comm_{Stern}(k) = N \times (3 \cdot l_{hash} + (4 \cdot n + 2 \cdot l_\sigma)/3) \tag{2}$$

3 Efficient Implementation for Embedded Devices

3.1 Improvements for reducing communication costs

The main drawback of the Fiat-Shamir paradigm is the signature length. There are different ways to decrease the communication cost without altering the global security of the scheme. In our case, we present three improvements.

The first idea we present is related to the seed generation and is well known. The second idea on decreasing the number of hashes was suggested in [24]. The third method is new and allows a reduction of communication cost of 10% to 15% depending on the scheme. We present these methods in the context of the AGS protocol since it is the most recent variation of the Stern protocol, but all these methods can be used and adapted for the Stern protocol and all its variations. Overall, depending on the scheme, they can decrease the communication cost by more than 40%.

Efficient Seed Generation Method. The random permutation σ and the random vector y are generally generated from random seeds. The prover and the verifier can agree on a common algorithm to generate σ and y from these seeds. Usually only the seed is sent instead of the whole σ or the whole y. In our case only the permutation is returned during the answer step. For a security of 2^{80}, only 80 bits for σ seed are sent instead of few hundred bits depending of the permutation generation chosen.

Equation (1) remains the same but l_σ becomes 80 bits for a security of 2^{80}.

$$Comm_{AGS}(k) = N \times (3 \cdot l_{hash} + (k + 80 + 2 \cdot k + 2 \cdot k)/2) \tag{3}$$

Minimizing the number of hashes to send. In [24] the authors propose a way to decrease the number of hashes to send based on the fact the verifier can recover two commitments over three per round. The prover generates the commitments of all the N rounds and then, computes and sends a general hash of all these values to the verifier. The verifier sends the challenges of the N rounds from which the prover constructs the answer. This answer allows two of the commitments to be recovered, the last one being sent by the prover with the answer. The verifier is then able to compute the general hash provided by the prover at the first stage and validate the authentication.

1. (First commitments) P builds $c_{1,1} \ldots c_{1,N}$ and $c_{2,1} \ldots c_{2,N}$, sends to V:
   $$C_{1,2} = h(c_{1,1}, c_{2,1}, \ldots, c_{1,N}, c_{2,N})$$
2. (First part of the challenges) V sends N values $0 \leq s_i \leq k-1$ to P with i from 1 to N
3. (Second commitments) P computes $c_{3,1} \ldots c_{3,N}$ and sends the last part of the commitment:
   $$C = h(C_{1,2}, c_{3,1}, \ldots, c_{3,N})$$
4. (Challenges) V sends a binary word $(b_1 \ldots b_N)$ to P.
5. (Answers) P reveals $((\epsilon_1, c_{2-b_1}) \ldots (\epsilon_N, c_{2-b_N}))$
6. (Verification) V needs to verify C

Fig. 2. Minimizing the number of commitments sent

With this method, one hash value is sent rather than three for each round. In case of 160-bit hashes, 320 bits are saved per round which represents around $5,760$ bits for the 18 rounds of the AGS protocol.

Figure 2 illustrates this method, where ϵ_i represents the answer associated to challenge b_i for i from 1 to N.

With this second method equation (1) becomes:

$$Comm_{AGS}(k) = N \times (l_{hash} + (k + l_\sigma + 2 \cdot k + 2 \cdot k)/2) + l_{hash} \tag{4}$$

Small weight word compression. During the protocol the prover may have to send in the answer a transformation $\sigma(e_s)$ of its secret e. The vector $\sigma(e_s)$ is n-bit long and has a fixed weight of w bits. Authors suggested in [24] to reduce the number of n bits sent by taking advantage of the low weight of the vector. For this purpose they propose to use classical algorithms such as the one employed in Niederreiter encryption scheme. However this type of algorithms is very time consuming. In a zero-knowledge authentication scheme it has to be done for each round so cannot be used in practice.

Our idea is to use the Huffman compression algorithm which is possible in the AGS scheme as the number of sent bits has not to be fixed. In practice, we fix a small bit-length d and encode all the d-bit possible words by predetermined symbol sequences depending on the probability of these d-bit words. A detailed example of how to determine the encoded sequences is given in appendix B.

We obtain an average compression slightly higher than 50% meaning that $\sigma(e_s)$ is defined on k bits instead of n.

With this last method to reduce communication size, equation (1) becomes:

$$Comm_{AGS}(k) = N \times (l_{hash} + (k + l_\sigma + 2 \cdot k + k)/2) + l_{hash} \tag{5}$$

Improvement results on Stern scheme. The methods presented above can also be applied to the Stern protocol to reduce the communication cost. Equation 2 becomes:

$$Comm_{Stern}(k) = N \times (l_{hash} + (3 \cdot n + 2 \cdot l_\sigma + k)/3) \tag{6}$$

3.2 Efficiency Analysis

Consider a security level of 2^{80}. The corresponding parameters for AGS scheme are $k = 349$, $w = 70$ and the hash function returns a 160-bit long value. We consider the permutation σ is generated using the method presented by Luby and Rackoff in [21,22]. This method involves two pseudo random functions f and g defined in $\mathbb{F}_2^m$ (where $m = \lceil \log_2(2k) \rceil /2$) in a Feistel scheme. The pseudo-random permutation is then defined by these two functions f and g, representing $2 \times m \times 2^m = 320$ bits. The authentication scheme needs 18 rounds and the signature 88 rounds.

From equations (1) to (6) we can create Table 2. The various improvements represent a gain of around 40% on the signature size. In particular the use of Huffman compression we propose reduces the signature length from 94,000 bits to 79,000 bits.

3.3 Implementation Considerations

Authentication Implementation. The second improvement method presented in section 3.1 is to send only one hash value for all commitments of all rounds. The implementation of this method requires either a large amount of memory to save as many elements as possible, or a re-computation of several elements, which would take a longer time. To reduce the memory cost and remain efficient we propose to generate two or three hash values instead of only one for all rounds.

Our idea is to split the N rounds into 2 groups of $N/2$ rounds or more generally u groups of N/u rounds and to compute one hash for all the commitments of a group of rounds. This means that u hash values will be sent to the verifier instead of only one. We define r as the number of rounds per group ($N = r \times u$) and obtain the algorithm presented on Figure 3.

A. For j from 1 to u

1. (First commitments) P builds $c_{1,1} \dots c_{1,r}$ and $c_{2,1} ... c_{2,r}$, sends to V:
$$C^j_{1,2} = h(c_{1,1}, c_{2,1}, \dots, c_{1,r}, c_{2,r})$$
2. (First part of the challenges) V sends r values $0 \leq s_i \leq k-1$ to P with i from 1 to r
3. (Second commitments) P computes $c_{3,1} \dots c_{3,r}$ and sends the last part of the commitment:
$$C^j = h(C^j_{1,2}, c_{3,1}, \dots, c_{3,r})$$
4. (Challenges) V sends a binary word $(b_1 \dots b_r)$ to P.
5. (Answers) P reveals $((\epsilon_1, c_{2-b_1}) \dots (\epsilon_r, c_{2-b_r}))$

B. (Verification) V needs to verify $C^1 \dots C^u$

Fig. 3. Implementation of AGS authentication

Although it slightly increases the communication cost of some hash values, this technique reduces the memory cost of our implementation. Several intermediate values such as yG or $\sigma(yG)$ are used several times in a round, and their computation is time consuming. Keeping them in memory reduces the execution time which is a very important aspect for a real product.

From Figure 3 we determine the time needed to compute the authentication process on the prover side, as a function of k. We denote respectively by T_{hash}, T_P, T_σ, T_{ρ_s} and T_{Huff}, the execution time of the routines for computing a hash value, the vector-matrix product yG, the pseudo-random function $\sigma(x)$, the rotation of s positions and the Huffman compression.

$$T_{auth}(k) = u \cdot (r \cdot (T_{c1}(l_\sigma) + T_{c2}(k) + T_{c3}(k) + (T_{\rho_s}(k) + T_{Huff}(2k))/2 + T_{hash}(3 * l_{hash})))$$

where

$$T_{c1} = T_{hash}(l_\sigma) \qquad T_{c2} = T_P(k) + T_\sigma(2k) + T_{hash}(2k)$$
$$T_{c3} = T_{\rho_s}(2k) + T_\sigma(2k) + T_{hash}(2k)$$

Signature Implementation. To implement the signature in embedded devices we need to adapt the second method of section 3.1. The 79,000 bits of the signature cannot be stored in such devices. Our idea is to send the signature on the fly which means that once the challenges are determined the signer sends the answers round by round.

Figure 4 details the different steps to compute a signature using AGS scheme. We consider three stages: the computation of all commitments, the generation of the challenges and the answer construction. In the signature the parameter r (number of rounds per group) depends on the security level, more precisely it is related to the number of challenges s_j which can be determined from the hash of the message and commitments.

The values s_j are defined on $log_2(k)$ bits. To respect the general security level of 2^{80} the generation of r values s_j must satisfy:

$$r \times log_2(k) > 80$$

In our implementation $k = 349$ therefore r must be greater than 9.

Hash initialization $H1_0 = h(m)$ and $H2_0 = hashinit()$

Commitments For i from 1 to $u = N/r$

1. (First commitment) Build $c_{1,1} \dots c_{1,r}$ and $c_{2,1} \dots c_{2,r}$ and compute:

$$H1_i = h(H1_{i-1}, (c_{1,1}, c_{2,1}, \dots, c_{1,r}, c_{2,r}))$$

2. (First part of the challenge) Determine r values $0 \leq s_j \leq k-1$ from $H1_i$ for j from 1 to r

3. (Second commitment) Build $c_{3,1} \dots c_{3,r}$ and compute:

$$H2_i = h(H2_{i-1}, (c_{3,1}, \dots, c_{3,r}))$$

Challenges Compute and send $H = h(H1_u, H2_u)$
Determine t bits b_i for i from 1 to t

Answers For i from 1 to t
if $b_i = 0$ reveal $(y_i \oplus m_{s_i}, \sigma_i, c_{2,i})$
if $b_i = 1$ reveal $(\sigma_i(y_iG), \sigma_i(e_{s_i}), c_{1,i})$

Fig. 4. Implementation of AGS signature

In the implementation described on Figure 4 we optimized the memory management while remaining efficient for the generation of all commitments. We determine the time needed to compute the signature process as a function of k. In this formula we consider that we keep in memory the data $\sigma(yG)$ between the computation of $c2$ and $c3$.

$$\begin{aligned}T_{sign}(k) = {} & u \cdot (r \cdot (T_{c1}(l_\sigma) + T_{c2}(k) + T_{hash}(2 * l_{hash} + (T_{c3}(k) + T_{hash}(l_{hash})) \\ & + T_{hash}(2l_{hash}) \\ & + r.u\,(T_{\rho_s}(k) + T_{c2}(k) + T_P(k) + 2T_\sigma(2k) + T_{\rho_s}(2k) + T_{Huff}(2k) + T_{c1}(l_\sigma))/2\end{aligned}$$

3.4 Practical results

We implement the AGS protocol in authentication and signature modes with the improvements described above. The device used was a AT90SC chip embedding the 8-bit AVR core running at 30MHz.

This chip contains a hardware random generator and a hardware AES used to generate the random values y and σ. The hash function is made from a Davies-Meyer based construction using AES. The pseudo-random permutation was implemented using the method presented by Luby and Rackoff in [21,22]. A function ϕ made of four Feistel rounds takes the index of a bit in the vector to permute and computes the new position in the result vector.

The implementation has been executed for parameters $k = 349$, $w = 70$, $N = 18$ rounds for authentication and $N = 88$ rounds for the signature. Execution time and memory cost for non-secure implementations are presented on Table 3.

Table 3 highlights the interest of splitting the protocol into groups of rounds instead of performing all rounds in one step. The memory used is divided by more than 2 while the execution time remains equivalent. From now on we consider the authentication in 3 groups of 6 rounds.

From these results we validate the formulas defined in section 3.3 and can now estimate the execution time and memory cost for larger parameters. We provide results for $k = 419$ (correspond to a security of 2^{100}) and $k = 479$ (correspond to a security of 2^{110}) on Table 4.

4 Side-Channel Protected Implementation

4.1 Secure Implementation

As presented in section 2.2, implementations in embedded devices require to be resistant against side-channel attacks. In [8] the authors present a secure implementation of the Stern authentication scheme. The protections they use can be applied to the AGS protocol as the same operations are involved.

The straightforward method to protect linear functions such as the matrix-vector product consists in masking the sensitive values involved in the computation. The pseudo-random function is defined by two non-linear functions f and g that we implement with look-up tables. In this case two other tables f^* and g^* are computed by applying random masks on f and g.

The formulas defined in section 3.3 to compute execution time ($T_{auth}(k)$, $T_{sign}(k)$) remain the same. However, the execution time of the internal functions T_P, T_σ and T_{ρ_s} increases. The blinding method requires computing operations on the masked data and the masks separately. The execution time of the hash function, based on a secure AES, remains the same.

4.2 Practical Results

We implement the AGS protocol in a secure way for authentication and signature. The parameters used remain $k = 349$, $w = 70$, $N = 18$ rounds for authentication and $N = 88$ rounds for the signature.

From these results and the formulas defined in section 3.3 we can estimate the execution time and memory cost for larger parameters for a secure implementation. We provide results for $k = 419$ (correspond to a security of 2^{100}) and $k = 479$ (correspond to a security of 2^{110}) on Table 6.

The improvements presented in the previous sections can be applied to any variation of the Stern scheme. From the results obtained for a secure implementation of the AGS protocol we estimate the execution time and memory cost needed for a secure implementation of the Stern algorithm. These results are given on Table 7 for $k = 349$, $N = 28$ rounds for authentication and $N = 140$ rounds for the signature.

We obtain an authentication in 630ms at 30MHz for $k = 349$. The last implementation of the Stern protocol provided an authentication in 5,911ms at 8MHz for $k = 256$. The authentication execution time increases linearly with the size of parameter k. With $k = 256$ our Stern authentication would be executed in 460ms at 30MHz. At the same frequency our implementation appears to be more than 3 times faster compared to the original one.

4.3 Comparison with RSA

In this section we compare our implementation with a software implementation of RSA. The goal is to determine the advantage of using error correcting codes on low resource devices.

We use the results of Gura *et al.* [16] for RSA estimations. The authors present an hybrid multiplication they tested on two components. We use the results obtained on the processor ATmega128 which is an 8-bit AVR micro-processor as in our implementation.

The authors present results for an optimized implementation of RSA using a fast squaring operation. In secure implementations we avoid using a fast squaring operation in order to counter SPA attacks. Generally secure RSA are implemented using the atomicity principle [10]. Each operation is computed with a multiplication and tests on the secret exponent are performed in constant time.

Based on the results obtained by Gura *et al.* we estimate the execution time of a secure RSA-CRT. We consider that all operations are performed with the multiplication (atomicity principle). To be resistant to side-channel attacks the message and the modulus are randomized with a 64-bit random value, and the exponent is randomized with a 32-bit random value. Finally the public exponentiation is computed at the end to check the result and protect the algorithm against fault attacks.

For a n-bit modulus, if we denote the execution time needed for a multiplication T_{mult}, for a squaring operation T_{square}, for the CRT recombination T_{CRT} and for the public exponentiation $T_{pub-exp}$, the execution time of a RSA-CRT is obtained thanks to the following formula:

$$T_{RSA-CRT}(n) = 2 \cdot ((512 + 32) \cdot T_{mult}(n/2) + (256 + 16)T_{square}(n/2)) + T_{CRT}(n) + T_{pub-exp}(n)$$

From [16] we have: $T_{mult}(512) = 65,000$ clock cycles. For a secure implementation we have $T_{square}(x) = T_{mult}(x)$ and for a non-secure implementation we have $T_{square}(x) = 0.78 * T_{mult}(x)$. We also consider the time needed for CRT recombination as negligible compared to exponentiation ($T_{CRT} \approx 0$).

Based on these elements we obtain estimations given in Table 8 for various secure RSA-CRT. To make the comparison easier with our implementations we provide results for a chip running at 30MHz.

4.4 Synthesis

Results presented in this paper are summarised in two graphs. Figure 5 represents the evolution of the execution time for non-secure implementations of AGS and RSA signatures. Figure 6 shows the evolution of the execution time for secure implementations of these two schemes.

These figures show that the execution time of signatures based on AGS protocol increases linearly whereas it increases quadratically with the size of parameters for RSA computations.

For a security of 2^{80} the time needed to compute a signature with a RSA-CRT is equivalent to the time needed to compute a signature based on AGS protocol. However when increasing the security level, the ASG signature becomes more interesting. For example a RSA-CRT 2048 bits is more than 4 times slower than an AGS signature with $k = 479$.

5 Conclusion

We have provided and implemented an efficient secure signature scheme based on error correcting codes on an embedded device. For a security level of 2^{80} our signature implementation is processed in 3.6 seconds which is more than 3 times faster than the previous

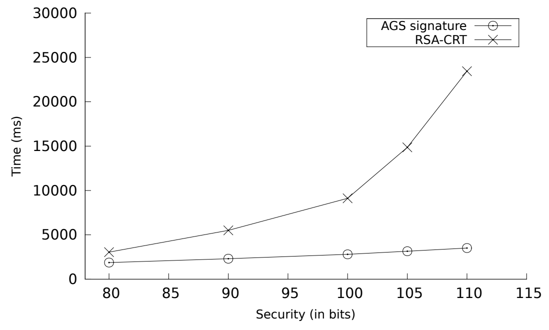

Fig. 5. Execution time for different security levels for non-secure implementations

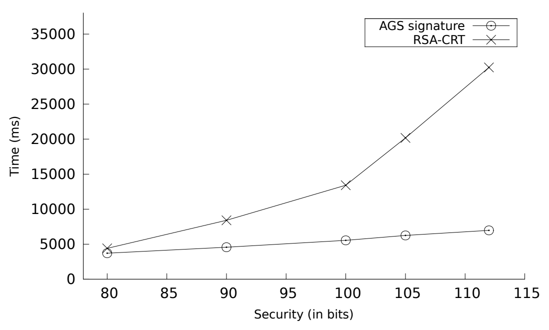

Fig. 6. Execution time for different security levels for secure implementations

one. We also obtain a secure authentication implementation in 410ms with communication of $16,000$ bits. Our results show that code-based schemes are a good alternative to RSA when no coprocessor is needed. This type of implementation is particularly suited for low-cost devices. The system we implemented highlights that error correcting codes cryptography can be used for industrial products.

References

1. M.-L. Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.
2. M. El Yousfi Alaoui, Ö. Dagdelen, P. Véron, D. Galindo, and P.-L. Cayrel. Extended Security Arguments for Signature Schemes. In A. Mitrokotsa and S. Vaudenay, editors, *AFRICACRYPT*, volume 7374 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2012.
3. Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In *IACR eprint*, 2013/400.
4. A. Becker, A. Joux, A. May, and A. Meurer. Decoding Random Binary Linear Codes in 2 n/20: How $1 + 1 = 0$ Improves Information Set Decoding. In *EUROCRYPT*, pages 520–536, 2012.
5. T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani. Reducing Key Length of the McEliece Cryptosystem. In *AFRICACRYPT*, pages 77–97, 2009.
6. D. J. Bernstein, T. Lange, and C. Peters. Smaller Decoding Exponents: Ball-Collision Decoding. In *CRYPTO*, pages 743–760, 2011.
7. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In Joye and Quisquater [18], pages 16–29.
8. P.-L. Cayrel, P. Gaborit, and E. Prouff. Secure Implementation of the Stern Authentication and Signature Schemes for Low-Resource Devices. In G. Grimaud and F.-X. Standaert, editors, *Smart Card Research and Advanced Applications - CARDIS 2008*, volume 5189 of *Lecture Notes in Computer Science*, pages 191–205. Springer, 2008.
9. P.-L. Cayrel, P. Véron, and S. M. El Yousfi Alaoui. A Zero-Knowledge Identification Scheme Based on the q-ary Syndrome Decoding Problem. In A. Biryukov, G. Gong, and D. Stinson, editors, *Selected Area in Cryptography - SAC 2010*, volume 6544 of *Lecture Notes in Computer Science*, pages 171–186. Springer, 2011.
10. B. Chevallier-Mames, M. Ciet, and M. Joye. Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity. *IEEE Transactions on Computers*, 53(6):760–768, 2004.
11. J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic Cryptanalysis of McEliece Variants with Compact Keys. In *EUROCRYPT*, pages 279–298, 2010.
12. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In A. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1987.
13. P. Gaborit and M. Girault. Lightweight Code-based Identification and Signature. In *IEEE Transactions on Information Theory - ISIT*, 2007.
14. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Computers*, 31(4):469–472, 1985.

15. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.
16. N. Gura, A. Patel, A. Wander, H. Eberle, and S. Chang Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In Joye and Quisquater [18], pages 119–132.
17. Stefan Heyse and Tim Güneysu. Code-based cryptography on reconfigurable hardware: tweaking Niederreiter encryption for performance. *J. Cryptographic Engineering*, 3(1):29–43, 2013.
18. Marc Joye and Jean-Jacques Quisquater, editors. *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*. Springer, 2004.
19. P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
20. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
21. M. Luby and C. Rackoff. Pseudo-random Permutation Generators and Cryptographic Composition. In *STOC*, pages 356–363, 1986.
22. M. Luby and C. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM J. Comput.*, 17(2):373–386, 1988.
23. R. J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. 1978.
24. C. Aguilar Melchor, P. Gaborit, and J. Schrek. A new zero-knowledge code based identification scheme with reduced communication. *CoRR*, abs/1111.1644, 2011.
25. T. S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.
26. T. S. Messerges. Securing the AES Finalists Against Power Analysis Attacks. In Bruce Schneier, editor, *FSE*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2001.
27. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *J. Cryptology*, 13(3):361–396, 2000.
28. R. L. Rivest, A Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM 21*, pages 120–126, 1978.
29. J. Stern. A New Identification Scheme Based on Syndrome Decoding. In D. Stinson, editor, *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 13–21. Springer, 1994.
30. P. Véron. Improved Identification Schemes Based on Error-correcting Codes. *Appl. Algebra Eng. Commun. Comput.*, 8(1):57–69, 1996.

A The Stern authentication scheme

This scheme was developed in 1993 (see [29]), it provides a zero-knowledge authentication scheme, not based on number theory problems. Let h be a hash function. Given a public random matrix H of size $(n-k)\times n$ over $\mathbb{F}_2$. Each user receives a secret key s of n bits and of weight ω. A user's public identifier is obtained from: $i = Hs^t$. It is calculated once in the lifetime of H. It can thus be used by several future identifications. Let us suppose that L wants to prove to V that he is indeed the person corresponding to the public identifier i_L. L has his own private key s_L such that $i_L = Hs_L^t$.

Our two protagonists follow the following protocol :

It is proven in [29] that this scheme is a zero-knowledge Fiat-Shamir like scheme with a probability of cheating in 2/3 (rather than in 1/2 for Fiat-Shamir).

B Details on the secret compression

We fix $d = 4$. The element has a low weight meaning that the most likely word of 4 bits is $'0000'$. Then this word is encoded by the unique symbol $'0'$. For the other 4-bit words we must to determine their apparition probability.

1. [Commitment Step] L randomly chooses $y \in \mathbb{F}^n$ and a permutation σ of $\{1, 2, \ldots, n\}$. Then L sends to V the commitments c_1, c_2 and c_3 such that :

$$c_1 = h(\sigma|Hy^t);\ c_2 = h(\sigma(y));\ c_3 = h(\sigma(y \oplus s))$$

   where $h(a|b)$ denotes the hash of the concatenation of the sequences a and b.
2. [Challenge Step] V sends $b \in \{0, 1, 2\}$ to L.
3. [Answer Step] Three possibilities :
   - if $b = 0$: L reveals y and σ.
   - if $b = 1$: L reveals $(y \oplus s)$ and σ.
   - if $b = 2$: L reveals $\sigma(y)$ and $\sigma(s)$.
4. [Verification Step] Three possibilities :
   - if $b = 0$: V verifies that c_1, c_2 have been honestly calculated.
   - if $b = 1$: V verifies that c_1, c_3 have been honestly calculated.
   - if $b = 2$: V verifies that c_2, c_3 have been honestly calculated, and that the weight of $\sigma(s)$ is ω.
5. Iterate the steps 1,2,3,4 until the expected security level is reached.

Fig. 7. Stern's protocol

We denote by $t = \frac{w}{n}$ the probability to have a bit 1 in a element of n bits of a weight w. The probability that j bits are equal to 1 in a d-bit word, and $d - j$ bits equal to 0, is $t^j \cdot (1 - j)^{d-j}$.

To illustrate this length reduction improvement, let's consider the case of the 16 sequences of 4 bits, with $n = 700$, $w = 70$ and $d = 4$. We obtain the following encoding:

sequence	encoding	sequence	encoding	sequence	encoding	sequence	encoding
0000	0	0001	1110	0110	1111011	1101	111111101
1000	100	1100	1111000	0101	111110	1011	111111110
0100	101	1010	1111001	0011	1111110	0111	1111111110
0010	110	1001	1111010	1110	111111100	1111	1111111111

In this case, the 4-bit words are encoded with sequences of size 1.9702 bits on average. A 700-bit word of weight 70 can then be reduced into a word of 344.785 bits, meaning an average compression of more than 50%. It is worth to notice that other algorithms may also be used, and give the same type of compression around 50% for our case.

	Basic scheme	Replace σ by its seed	Decrease hash number	Compression of $\sigma(e)$
Authentication - Stern	45,472	40,992	32,192	28,935
Signature - Stern	227,360	204,960	160,320	144,033
Authentication - AGS	27,025	25,065	19,465	16,324
Signature - AGS	133,100	122,540	94,540	79,184

Table 2. Communication cost given in bits

	k = 349	
Implementations	time (ms)	RAM (byte)
Authentication / 3 × 6 rounds	213	2,400
Authentication / 2 × 9 rounds	214	3,200
Authentication / 1 × 18 rounds	212	5,600
Signature / 8 × 11 rounds	1,877	3,600

Table 3. Authentication and Signature results for a non-secure implementation

	k = 419		k = 479	
Implementations	time (ms)	RAM (byte)	time (ms)	RAM (byte)
Authentication / 3 × 6 rounds	252	2,700	287	3,000
Signature / 10-11 × 11 rounds	2,801	4,300	3,512	4,700

Table 4. Authentication and Signature estimations for a non-secure implementation

	k = 349	
Implementations	time (ms)	RAM (byte)
Authentication / 3 × 6 rounds	415	2,400
Signature / 8 × 11 rounds	3,684	3,600

Table 5. Authentication and Signature results for a secure implementation

	k = 419		k = 479	
Implementations	time (ms)	RAM (byte)	time (ms)	RAM (byte)
Authentication / 3 × 6 rounds	457	2,700	521	3,000
Signature / 10-11 × 11 rounds	5,553	4,300	6,972	4,700

Table 6. Authentication and Signature estimations for a secure implementation

	k = 349	
Implementations	time (ms)	RAM (byte)
Authentication / 4 × 7 rounds	630	2,660
Signature / 14 × 10 rounds	4,700	3,600

Table 7. Authentication and Signature results for a secure implementation of Stern

	RSA-CRT 1024 bits	RSA-CRT 1536 bits	RSA-CRT 2048 bits
non-secure RSA (time in ms)	3,050	9,110	23,440
Secure RSA (time in ms)	4,500	13,420	30,240

Table 8. RSA-CRT estimations for non-secure and secure versions