Parallel-CFS
Strengthening the CFS McEliece-Based Signature Scheme

Matthieu Finiasz

ENSTA

Abstract. This article presents a modification of the CFS code based signature scheme. By producing two (or more generally i) signatures in parallel, we show that it is possible to protect this scheme from "one out of many" decoding attacks. With this modification, and at the cost of slightly larger signatures, it is possible to use smaller parameters for the CFS signature, thus making this new Parallel-CFS construction more practical than standard CFS signatures.

Keywords: CFS digital signature scheme, code based cryptography, Syndrome Decoding problem.

Introduction

Published in 2001, the Courtois-Finiasz-Sendrier (CFS) code based signature scheme [4] was the first practical digital signature scheme with a security reduction to the Syndrome Decoding problem. Though practical, this scheme requires much more resources than traditional number theory signatures: both the public key size and the signature time are relatively large. However, it offers the shortest known digital signatures, with sizes down to 81 bits for the original parameters.

Some time after the publication of this construction, D. Bleichenbacher imagined a new attack based on the Generalized Birthday Algorithm [13] which made the original CFS parameters insecure. This unpublished attack is described in [7]. For a given security level, resisting Bleichenbacher's attack only requires a small parameter increase, but this small increase can have a heavy impact on the signature time or the public key size.

This article introduces the concept of parallel signature, that is, producing two CFS signatures (or more) in parallel for a same document. The aim of this construction is simple: make the application of Bleichenbacher's attack impossible, or at least, as expensive as standard decoding attacks. A similar idea was proposed by Naccache, Pointcheval and Stern in [10] under the name Twin-signature, but with a completely different purpose. Their construction allows to sign short messages without having to hash them, and thus, can be proven secure without the need for a random oracle. The main focus of the present article is practical security and we will thus use hash functions and consider them as random oracles.

A. Biryukov, G. Gong, and D.R. Stinson (Eds.): SAC 2010, LNCS 6544, pp. 161–172, 2011.

We start this article by recalling the original CFS signature construction and the existing attacks against it. In Section 2 we introduce Parallel-CFS and detail the gains it offers in terms of security. Finally, in Section 3, we explain how to select secure parameters for Parallel-CFS.

1 The Original CFS Signature Scheme

The hash-and-sign paradigm makes it possible to convert any public key encryption scheme into a digital signature scheme. Producing a signature consists in hashing the document to sign into a ciphertext and then decrypting this "random" ciphertext using a secret key. Verification of the signature simply consists in comparing the hash of the document with the encryption of the signature (as encryption only requires the knowledge of the public key associated to the signer's private key, anyone can verify the signature).

In order for this construction to apply (and be secure), it is necessary to have access to a public cryptographic hash function which outputs ciphertexts uniformly distributed among all the possible ciphertexts. In the case of code based cryptosystems like the McEliece [9] or Niederreiter [11] encryption schemes, designing such a hash function is however impossible: the set of possible ciphertexts is a subset of known size of binary vectors of a given length r, but no simple description of this subset exists. One can thus use a hash function with uniform output among the binary vectors of length r, but then, only a small part of the hashes can be decrypted. The CFS signature scheme proposes two workarounds for this problem.

1.1 The Niederreiter Encryption Scheme

The CFS signature is built upon the Niederreiter encryption scheme which is described by the three following algorithms.

Key Generation. Choose parameters m and t and let $n = 2^m$. Let $\Gamma(g, S)$ be a binary Goppa code defined by its polynomial g of degree t in $\mathbf{F}_{2^m}[x]$ and its support S, a permutation of the n elements of $\mathbf{F}_{2^m}$. This code can correct up to t errors. Let H be a systematic $mt \times n$ binary parity check matrix of $\Gamma(g, S)$. H is the public encryption key, g and S form the private decryption key.

Encryption. To encrypt a plaintext p, first convert it into an error pattern e_p of length n and Hamming weight at most t (using an invertible mapping φ_t). Compute $s = H \times e_p^T$, the syndrome associated to the error pattern e_p with respect to the public parity check matrix H. The syndrome s is the ciphertext.

Decryption. To decrypt the ciphertext s, one simply applies the decoding algorithm associated to g and S to the received syndrome s and obtains e_p. The plaintext p is recovered by applying φ_t^{-1}.

Remark: The classical description of the Niederreiter cryptosystem uses two scrambling matrices to "hide" the structure of the parity check matrix H. In practice, these matrices are not necessary (see [2] for more details): the $n \times n$ permutation matrix simply corresponds to the order of the elements of the support S, and the $mt \times mt$ invertible matrix is not necessary when H is given in systematic form.

1.2 The CFS Signature Scheme

The CFS signature scheme uses the hash-and-sign paradigm. The hash function used should have an output size of $r = mt$ bits. The document hash is then viewed as a ciphertext which has to be decrypted. However, only ciphertexts corresponding to syndromes of error patterns of weight t or less can be decrypted as the decoding algorithm only corrects up to t errors. This means than only $\binom{2^m}{t} \simeq \frac{2^{mt}}{t!}$ out of the 2^{mt} possible syndromes can be decoded. Thus, only one document out of $t!$ can be signed, which makes its direct application impractical. There are two methods to get round this problem, however, both require to perform $t!$ decryption attempts in average. For this reason, only very small values of t are acceptable.

Appending a Counter to the Message. The first method which was proposed consists in appending a counter to the document to sign. We denote by D the document to sign and h the hash function. Instead of computing $s = h(D)$, one computes a sequence of hashes $s_i = h(D \,||\, i)$ where $||$ represents the concatenation. Starting with s_0 and incrementing i each time, the signer tries to decrypt the syndrome s_i, until a decodable syndrome is found. Let i_0 be the first index for which s_{i_0} can be decrypted. The signature is then $(p_{i_0} \,||\, i_0)$ where p_{i_0} is the plaintext corresponding to s_{i_0}, that is $H \times \varphi_t(p_{i_0})^T = s_{i_0}$. In order to verify this signature, one has to verify the equality $H \times \varphi_t(p_{i_0})^T \overset{?}{=} h(D \,||\, i_0)$.

Note that by bounding the space in which the counters are selected, L. Dallot was able to formally prove the security of this construction in the random oracle model [5].

Performing Complete Decoding. The idea of this second method is to be able to decode/decrypt any syndrome given by the hash function. For this, one needs to modify the decoding algorithm in order to be able to decode more than t errors. A first step towards complete decoding could be the use of Bernstein's list decoding algorithm [1], however, for code rates close to 1 (which is the case here for very small values of t), this algorithm becomes less efficient than simple exhaustive search. The idea is thus to use exhaustive search for the additional errors we want to decode. Let δ be the smallest integer such that $\binom{2^m}{t+\delta} > 2^{mt}$, then, any syndrome can (with a good probability) be decoded into an error pattern of weight $t + \delta$ or less.

With this method, the signer will go through all error patterns $\varphi_\delta(i)$ of weight δ and try to decode the resulting syndrome $s_i = h(D) + H \times \varphi_\delta(i)^T$. Once a decodable syndrome is found for a given index i_0, as in the previous method, we have found a p'_{i_0} such that $H \times \varphi_t(p'_{i_0})^T = s_{i_0} = h(D) + H \times \varphi_\delta(i_0)^T$. The signature is then $p_{i_0} = \varphi^{-1}_{t+\delta}(\varphi_t(p'_{i_0}) + \varphi_\delta(i_0))$, that is, the message corresponding to the error pattern of weight $t + \delta$.

In order to verify the signature, one simply verifies the equality:

$$H \times \varphi_{t+\delta}(p_{i_0})^T \overset{?}{=} h(D).$$

1.3 Existing Attacks - One Out of Many Syndrome Decoding

Two kind of attacks exist against a signature scheme: key recovery attacks, which aim at recovering the private key from the public key, and signature forgeries, which try to produce a valid signature for a message without the knowledge of the private key. Key recovery attacks against McEliece type cryptosystems are traditionally less efficient than decoding attacks, however, recent developments in Goppa code distinguishing [6] could bring new attacks into consideration. For the moment no such attack has been found and we therefore focus on decoding attacks which, in the context of signatures, are signature forgeries.

When forging a CFS signature, one has to find a valid signature/document pair, and therefore has to solve an instance of the Syndrome Decoding problem: find a low weight error pattern corresponding to a given syndrome. However, as opposed to the standard Syndrome Decoding problem, one only has to solve one instance out of many.

Problem 1 (One out of Many Syndrome Decoding (OMSD)). Given parameters n, r, t, and N, a binary $r \times n$ matrix H and a set of N binary vectors S_i of r bits, find a binary error pattern e of Hamming weight t or less such that:

$$\exists i \in [1, N] \ s.t. \ H \times e^T = S_i.$$

In order to solve this problem, one can use the same methods as for the standard single Syndrome Decoding (SD) problem and see how they can be improved when several instances (for the same matrix) are available. As described in [7], when designing a code based cryptosystem the two main attacks to consider are Information Set Decoding (ISD) and the Generalized Birthday Algorithm (GBA) [13].

Information Set Decoding. ISD algorithms are particularly efficient for solving instances of the SD problem which have a single (or a few) solutions. The application of ISD to instances of OMSD has been studied by Johansson and Jönsson in [8]. They modify the Canteaut-Chabaud algorithm [3] so as to take into account the number of target syndromes and study the efficiency of their new algorithm against various code based cryptosystems including the CFS signature scheme. However, the authors do not give any asymptotic analysis of the gain when targeting N syndromes instead of 1. Applying their technique to the "idealized" algorithm described in [7] cannot gain more than an order $\sqrt{N}$, which is exactly what the Generalized Birthday Algorithm described in the following section offers. Both algorithms should thus have a similar asymptotic behavior, but the complexity of the next algorithm is well known and easier to study. The reader should thus keep in mind that the attack complexities given in Section 3.2 might be slightly improved using an idealized version of Johansson and Jönsson's algorithm, but the gain cannot be more than a small constant factor.

In the rest of this article, we will denote by ISD the Information Set Decoding attack targeting a single syndrome. This attack is an exact replacement for the secret decoding algorithm, so any signature the legitimate signer can produce can be forged using this algorithm. Its complexity which is close to $2^{\frac{mt}{2}}$ will thus serve as a reference security level.

Generalized Birthday Algorithm. As opposed to ISD, the GBA algorithm can only work when there are many solutions. When attacking a single instance of SD with the CFS parameters, there is less than a solution in average so GBA cannot be applied directly. However, when attacking OMSD, if enough instances are considered simultaneously, the number of solutions can become very large. This idea came from Daniel Bleichenbacher who proposed to build a list of target syndromes and use this list as one of the starting lists of GBA. Compared to ISD, this allows to significantly decrease the cost of a signature forgery. The complexity of this attack against CFS with a counter is given by the formulas:

$$C_{\text{GBA}} = L\log(L), \text{ with } L = \min\left(\frac{2^r}{\binom{n}{t-\lfloor t/3\rfloor}}, \sqrt{\frac{2^r}{\binom{n}{\lfloor t/3\rfloor}}}\right).$$

The size L of the largest list used in the algorithm is roughly $2^{\frac{mt}{3}}$ (which is what one would expect using GBA with 4 lists). In practice, it is a little larger than this because $\binom{n}{t} < 2^{mt}$ in the counter version of CFS, so lists formed by XORs of columns of H are too small and this has to be compensated by using a larger list of target syndromes.

In the case of CFS using complete decoding, the target is a word of weight $t+\delta$ and, as $\binom{n}{t+\delta} > 2^{mt}$, lists formed of XORs of columns of H are large enough and the size L of the largest list is very close to $2^{\frac{mt}{3}}$.

2 Parallel-CFS: Cost and Gain

The possibilities offered by OMSD against the original CFS signature are not devastating: asymptotically, the security can be decreased from $2^{\frac{mt}{2}}$ to $2^{\frac{mt}{3}}$. However, this is enough to require a slight increase in the parameters m or t, which translates into a significant increase in public key size (which is exponential in m) or signature cost (which is exponential in t). For this reason, keeping the parameters as small as possible is critical. The aim of Parallel-CFS is to make the application of OMSD improvements impossible.

2.1 Description of Parallel-CFS

The idea in Parallel-CFS is very simple: instead of producing one hash (using a function h) from a document D and signing it, one can produce two hashes (using two functions h_1 and h_2) and sign both $h_1(D)$ and $h_2(D)$ in parallel. This way, if one wants to use OMSD, he will have to produce two forgeries, one for h_1 and one for h_2, but these forgeries also have to be for the same document D. More generally, one can also use i different hash functions and produce i signatures in parallel. We will denote the construction using i hash functions as order i Parallel-CFS.

Using CFS with a Counter is Impossible. The first thing to note when using Parallel-CFS is that the version of CFS using a counter is not suitable. If we simply compute two CFS signatures for D using both h_1 and h_2, we will obtain two signatures $(p_{i_1} \,||\, i_1)$ and $(p_{i_2} \,||\, i_2)$. The problem here is that these two signatures are not linked through the hash function: they are signatures for $h_1(D \,||\, i_1)$ and $h_2(D \,||\, i_2)$, and using enough different counter values one simply has to solve two independent instances of OMSD. In order to link the signatures together, they have to be signatures for the exact same message: a single counter has to be used for both signatures. One has to find an index i_0 such that both $h_1(D \,||\, i_0)$ and $h_2(D \,||\, i_0)$ can be decoded. The problem now is that instead of having $t!$ attempts to perform, one needs $t! \times t!$ decodings in average! Of course this is not acceptable, and we therefore propose to use the other version of CFS.

Using CFS with Complete Decoding. When using this second version of CFS, we no longer have any counter problems. Parallel-CFS can thus be described by the following algorithms.

Key Generation. This step is similar to the Niederreiter cryptosystem. Choose parameters m, t and let $n = 2^m$. Select δ such that $\binom{n}{t+\delta} > 2^{mt}$. Choose a Goppa polynomial g of degree t in $\mathbf{F}_{2^m}[x]$ and a support S (a permutation of the n elements of $\mathbf{F}_{2^m}$). Let H be a $mt \times n$ systematic parity check matrix of the Goppa code $\Gamma(g, S)$. H is the public verification key, g and S form the private signature key. The parameters m, t, and δ as well as two cryptographic hash function h_1 and h_2 are also made public.

Signature. When signing a document D, compute $h_1(D)$ and $h_2(D)$. Then, as in the original CFS, using the Goppa code decoding algorithm, compute two error patterns e_1 and e_2 of weight at most $t + \delta$ such that $H \times e_1^T = h_1(D)$ and $H \times e_2^T = h_2(D)$. The signature is $\left(\varphi_{t+\delta}^{-1}(e_1) \,||\, \varphi_{t+\delta}^{-1}(e_2)\right)$.

Verification. When verifying a signature $(p_1 \,||\, p_2)$ for a document D. One verifies if $H \times \varphi_{t+\delta}(p_1)^T \stackrel{?}{=} h_1(D)$ and $H \times \varphi_{t+\delta}(p_2)^T \stackrel{?}{=} h_2(D)$. If both equalities are verified, the signature is valid.

2.2 Cost of Parallel-CFS

The computational cost of Parallel-CFS is easy to evaluate: instead of producing one CFS signature, the signer has to produce two (or more generally i) signatures.

Compared to the original CFS, the signature time is doubled (or multiplied by i) and the signature size is also doubled (or multiplied by i), the key sizes remain the same, and the verification time is doubled (or multiplied by i).

Signature Failure Probability. One of the main issues with the complete decoding version of CFS (as opposed to the version using a counter) is that some documents might not be signable: some syndromes might not correspond to error patterns of weight $t+\delta$. Complete decoding is only possible if $t+\delta$ is equal (or larger) to the *covering radius* of the Goppa code we use. Unfortunately, computing the covering radius of a Goppa code is difficult. When selecting δ we thus consider that the words in the balls or radius $t+\delta$ centered on the codewords are randomly distributed in the space. This way, we can compute an estimate of the probability $\mathrm{P_{noCFS}}$ of not being able to perform one CFS signature, and the probability $\mathrm{P_{fail}}$ of not being able to sign with order i Parallel-CFS.

$$\mathrm{P_{noCFS}} \simeq \left(1 - \frac{1}{2^{mt}}\right)^{\binom{n}{t+\delta}}$$

$$\mathrm{P_{fail}} = 1 - \left(1 - \mathrm{P_{noCFS}}\right)^{i}.$$

Even if this probability is negligible for the proposed parameters (see Section 3.2, Table 1), the protocol has to be ready to handle this kind of problem, typically by modifying (or expanding) the document to sign.

2.3 Gain of Parallel-CFS

The only gain compared to the original CFS is on the security side. Security against ISD attacks is the same as before and is easy to evaluate: you need to forge two CFS signatures, so the cost is twice the cost of attacking CFS. Security against GBA and attacks like that of Bleichenbacher is a little more complicated to evaluate. There are two strategies to exploit OMSD in Parallel-CFS: either consider it as one big signature scheme, or try to chain two attacks on the first and second signatures (or, in general, chain i attacks).

Parallel-CFS as One Big Signature. This first strategy considers Parallel-CFS as a single OMSD problem and tries to solve a problem of the form:

$$\left(\begin{array}{c|c} H & 0 \\ \hline 0 & H \end{array}\right) \times \left(\begin{array}{c|c} e_1 & e_2 \end{array}\right)^T = \left(\begin{array}{c} h_1(D) \\ \hline h_2(D) \end{array}\right)$$

In this case, Bleichenbacher's attack applies directly, but on a problem where all parameters are doubled. The size L of the lists used in the attack is roughly $2^{\frac{2mt}{3}}$ which, as intended, is larger than the complexity of $2^{\frac{mt}{2}}$ offered by ISD attacks. This strategy is thus less efficient than applying 2 ISD attacks independently.

Chaining Attacks against Parallel-CFS. This second strategy considers the two signatures sequentially. Bleichenbacher's attack was originally described to produce a single solution, however, GBA can also be used to produce several solutions quite efficiently. The idea is thus to use this attack to produce a large number of valid signatures for the first hash function h_1, and use all these solutions as possible targets of the second hash function. It is then possible to benefit from OMSD also for the second signature. However, the attack against the first signature will cost more than for a single solution.

In order to simplify computations, we will make the following assumptions (similar to those used in [7]):

- it is possible to XOR non-integer numbers of vectors, thus XORs of $\frac{t+\delta}{3}$ columns of H are always possible,
- the number $\binom{n}{t+\delta}$ of syndromes we can build is larger than 2^{mt}, but is not much larger. We consider that we can build 3 lists L_0, L_1, and L_2, respectively of XORs of w_0, w_1, and w_2 columns of H, with $w_0 + w_1 + w_2 = t + \delta$ such that $|L_0| \times |L_1| \times |L_2| = 2^{mt}$. In practice these lists can even be a little larger than 2^{mt}, but we ignore this small factor as the improvement it can yield is negligible.

With these assumptions, the cost of Bleichenbacher's attack against a single CFS signature is exactly $L \log L$ with $L = 2^{\frac{mt}{3}}$.

Now, suppose one has forged 2^c signatures for 2^c different hashes $h_1(D_i)$. Then there are 2^c target hashes $h_2(D_i)$ for the second function. One then builds three lists L_0, L_1, and L_2 of XORs of columns of H such that $|L_0| = |L_1| = 2^{\frac{mt+c}{4}}$ and $|L_2| = 2^{\frac{mt-c}{2}}$. In accord with our assumption, this choice satisfies $|L_0| \times |L_1| \times |L_2| = 2^{mt}$ and is thus possible. Then, if we merge lists L_0 and L_1 together and list L_2 with our 2^c hashes, by zeroing c bits in the first step of a GBA it is possible to find a valid signature with a cost of $2^{\frac{mt-c}{2}}$.

In order to determine the optimal value of c, we need to know the cost of the forgery of 2^c signatures for the first hash function. Thanks to our assumptions this is quite easy: we can find one solution with GBA and a complexity of $2^{\frac{mt}{3}}$ by using $2^{\frac{mt}{3}}$ target syndromes $h_1(D_i)$. If we want more solutions we have to take more syndromes: the number giving the smallest complexity is $2^{\frac{mt+2c}{3}}$ syndromes. Then one can choose $|L_0| = |L_1| = 2^{\frac{mt+c/2}{3}}$ and $|L_2| = 2^{\frac{mt-c}{3}}$ and, by zeroing $\frac{mt-c}{3}$ bits in the first merge, obtain exactly 2^c solutions in average. The complexity of this step is $2^{\frac{mt+2c}{3}}$.

The optimal choice of c is when both steps of the forgery have the same cost, that is: $\frac{mt-c}{2} = \frac{mt+2c}{3}$ and $c = \frac{1}{7}mt$. Chaining two OMSD attacks to forge a Parallel-CFS signature thus costs $2L \log L$ with $L = 2^{\frac{3}{7}mt}$.

Security of Order *i* Parallel-CFS. Using two CFS signatures in parallel increases the cost of GBA-based attacks from $2^{\frac{1}{3}mt}$ to $2^{\frac{3}{7}mt}$. If one uses i signatures in parallel an attacker has to chain i GBA algorithms and the security of the construction should be even closer to $2^{\frac{1}{2}mt}$. In order to evaluate this more precisely we need to determine the exact cost of a signature forgery starting from 2^{c_j} syndromes and producing $2^{c_{j+1}}$ signatures.

Similarly to the second step of the attack against order 2 Parallel-CFS, one chooses starting lists such that $|L_0| = |L_1| = 2^{\frac{mt+c_j}{4}}$ and $|L_2| = 2^{\frac{mt-c_j}{2}}$, but instead of zeroing c_j bits during the first merge operations, one can only zero $c_j - c_{j+1}$ bits. The complexity of the attack is thus $2^{\frac{mt-(c_j-2c_{j+1})}{2}}$.

When forging an order i Parallel-CFS signature one has to select indices $(c_1, ..., c_i)$ such that each step has the same complexity 2^K, that is:

- for the first signature forgeries $K = \frac{mt+2c_1}{3}$,
- for each of the following steps $\forall j \in [1, i-1],\ K = \frac{mt-(c_j-2c_{j+1})}{2}$,
- at the end one wants one valid signature, so $c_i = 0$.

Simple computations lead to the solution $c_j - 2c_{j+1} = \frac{mt}{2^{i+1}-1}$ and the global complexity of an attack chaining i OMSD attacks is of order $iL\log L$ with $L = 2^{\frac{2^i-1}{2^{i+1}-1}mt}$.

As one can see, order i Parallel-CFS cannot achieve an asymptotic security of $2^{\frac{mt}{2}}$, but comes very close to it. In practice, order 2 or 3 Parallel-CFS will be the best choice for most parameters. The security gain using 4 parallel signatures can never compensate the cost in signature time and size of a fourth signature.

3 Parameter Selection

3.1 Signature Size

It is always possible to reduce the size of a signature at the cost of an increase in the verification time. As signature verification is exceptionally fast for CFS (one only has to XOR $t + \delta$ columns of H) it is not necessarily a problem to increase its cost. The signature shortening techniques presented in [4] can still be applied for Parallel-CFS. Compared to the original proposition, the replacement of the counter by δ additional positions does not change much and the two following reduction methods can still be applied:

- omit the w first positions of the $t + \delta$ positions of a signature: the verifier has to exhaustively search for $w - 1$ of these positions,
- split the length n in $\frac{n}{s}$ intervals of size s (typically s will be close to m) and only include the interval containing the error in the signature (not the exact position): the verifier has to perform a Gaussian elimination on H (the cost of which is not negligible if less than 3 positions were omitted).

If one wants very fast signature verification ($t + \delta$ column operations on H), omitting a single column is the right choice and the size of the signature of order i Parallel-CFS is $i \times \log_2 \binom{n}{t+\delta-1}$. For the original $t = 9$ and $m = 16$ parameters this is $139 \times i$ bits. If one aims for short signatures with a longer verification ($3\binom{n}{w}/\binom{t+\delta}{2}$ column operations on H), omitting 3 columns and using intervals of size m, the size of the signature of order i Parallel-CFS is $i \times \log_2 \binom{n/m}{t+\delta-3}$. For the original parameters this is $81 \times i$ bits.

3.2 Parameter Examples

Table 1 presents a few sets of parameters that can be used for Parallel-CFS. These parameters are not all intended to be optimal choices but simply serve as examples to illustrate the flexibility of this new construction. If one aims for a security of 2^{80}, four sets of parameters (m, t, δ) stand out:

- Parameters $(20, 8, 2)$: if one wants fast signature, signature time will be 10 times smaller than for other parameters with a same security level. In order to have a sufficient security level 3 signatures have to be produced in parallel leading to a signature size of 294 bits, which is acceptable. The downside of this set of parameters is the public key size: 20 Mbytes is a lot.

Table 1. Some parameter examples for Parallel-CFS. Using $i = 1$ corresponds to the orginal CFS signature scheme and can serve for comparison

parameters				ISD	orig. CFS	security against	sign. failure	public	sign.	sign.
m	t	δ	i[a]	security[b]	security[c]	(chained) GBA	probability	key size	cost[d]	size[e]
20	8	2	1	$2^{81.0}$	$2^{66.4}$	$2^{59.1}$	~ 0	20.0 MB	$2^{15.3}$	98
–	–	–	2	–	–	$2^{75.7}$	~ 0	–	$2^{16.3}$	196
–	–	–	3	–	–	$2^{82.5}$	~ 0	–	$2^{16.9}$	294
16	9	2	1	$2^{76.5}$	$2^{63.3}$	$2^{53.6}$	2^{-155}	1.1 MB	$2^{18.5}$	81
–	–	–	2	–	–	$2^{68.7}$	2^{-154}	–	$2^{19.5}$	162
–	–	–	3	–	–	$2^{74.9}$	2^{-153}	–	$2^{20.0}$	243
18	9	2	1	$2^{84.5}$	$2^{69.5}$	$2^{59.8}$	2^{-1700}	5.0 MB	$2^{18.5}$	96
–	–	–	2	–	–	$2^{76.5}$	2^{-1700}	–	$2^{19.5}$	192
–	–	–	3	–	–	$2^{83.4}$	2^{-1700}	–	$2^{20.0}$	288
19	9	2	1	$2^{88.5}$	$2^{72.5}$	$2^{62.8}$	~ 0	10.7 MB	$2^{18.5}$	103
–	–	–	2	–	–	$2^{80.5}$	~ 0	–	$2^{19.5}$	206
–	–	–	3	–	–	$2^{87.7}$	~ 0	–	$2^{20.0}$	309
15	10	3	1	$2^{76.2}$	$2^{63.1}$	$2^{55.6}$	~ 0	0.6 MB	$2^{21.8}$	90
–	–	–	2	–	–	$2^{71.3}$	~ 0	–	$2^{22.8}$	180
–	–	–	3	–	–	$2^{77.7}$	~ 0	–	$2^{23.4}$	270
16	10	2	1	$2^{86.2}$	$2^{66.2}$	$2^{59.1}$	2^{-13}	1.2 MB	$2^{21.8}$	90
–	–	–	2	–	–	$2^{75.7}$	2^{-12}	–	$2^{22.8}$	180
–	–	–	3	–	–	$2^{82.5}$	$2^{-11.3}$	–	$2^{23.4}$	270
17	10	2	1	$2^{90.7}$	$2^{69.3}$	$2^{62.5}$	2^{-52}	2.7 MB	$2^{21.8}$	98
–	–	–	2	–	–	$2^{80.0}$	2^{-51}	–	$2^{22.8}$	196
–	–	–	3	–	–	$2^{87.2}$	2^{-50}	–	$2^{23.4}$	294

[a] Number of parallel signatures to perform.

[b] Cost of an ISD attack on the associated SD instance. This is an upper bound on the security we can obtain from parallel signatures.

[c] Security against GBA of the original CFS signature (using a counter with no parallel signatures) with the same parameters.

[d] Average number of decoding attempts required to sign a document.

[e] Size in bits of a signature with all size reductions applied (omitting 3 columns and using intervals).

– Parameters $(18, 9, 2)$: these are what we would call parameters for "everyday's use." A key of 5 Mbytes is not negligible, but can be accepted for certain applications and signature time is still reasonable. Order 3 Parallel-CFS still has to be used, leading to signatures of 288 bits.
– Parameters $(19, 9, 2)$: these parameters offer short 206 bits signatures as 2 parallel signatures are enough for 80 bits of security. While the signature time is still reasonable, the key size can already be problematic.
– Parameters $(16, 10, 2)/(17, 10, 2)$: choosing $t = 10$ allows for much smaller keys but significantly increases the cost of the signature. These parameters are similar to the two previous sets, but with a smaller key and larger signature time.

3.3 Hiding the Structure in Parallel-CFS?

A natural idea to improve Parallel-CFS would be to combine the two independent signature problems into one large signature: this way, an attacker would have to solve an instance of OMSD with doubled parameters. Unfortunately, the structure in the combined matrix is too strong and cannot be efficiently hidden (while preserving the legitimate signature algorithm).

The best that can be done in terms of structure hiding is to select a $2n \times 2n$ permutation matrix P, a $2mt \times 2mt$ invertible matrix S, and a random $mt \times n$ binary matrix Y and build the matrix:

$$H' = S \times \left(\begin{array}{c|c} H_1 & 0 \\ \hline Y & H_2 \end{array} \right) \times P.$$

Where H_1 and H_2 are two distinct Goppa code parity check matrices. Signing in that case requires to sign the first hash $h_1(D)$, and then sign the second hash XORed to the product of the first signature by Y.

As we said before, this is unfortunately not sufficient to hide the structure in matrix H'. Raphael Overbeck already studied this structure under the name "permuted reducible codes" and showed in [12] that separating H_1 and H_2 was an easy problem. We leave it as an exercise to the reader to find out how this can be done.

4 Conclusion

With Parallel-CFS we propose to add some flexibility in the parameter choices of the CFS signature scheme. In order to resist OMSD attacks like that of Bleichenbacher, it is possible to increase the number of signatures to produce instead of increasing the signature parameters. In addition to the two previous tradeoffs where security could be gained at the cost of a larger public key or of a much longer signature time, we now have a third tradeoff where increasing security mostly increases the signature size. Thanks to this, parameter sets which had become insecure can be used again, making Parallel-CFS much more practical than standard CFS.

References

1. Bernstein, D.J.: List decoding for binary goppa codes. Preprint (2008), `http://cr.yp.to/codes/goppalist-20081107.pdf`
2. Biswas, B., Sendrier, N.: McEliece cryptosystem implementation: Theory and practice. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 47–62. Springer, Heidelberg (2008)
3. Canteaut, A., Chabaud, F.: A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. IEEE Transactions on Information Theory 44(1), 367–378 (1998)
4. Courtois, N., Finiasz, M., Sendrier, N.: How to achieve a mcEliece-based digital signature scheme. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 157–174. Springer, Heidelberg (2001)
5. Dallot, L.: Towards a concrete security proof of courtois, finiasz and sendrier signature scheme. In: Lucks, S., Sadeghi, A.-R., Wolf, C. (eds.) WEWoRC 2007. LNCS, vol. 4945, pp. 65–77. Springer, Heidelberg (2008)
6. Faugère, J.-C., Otmani, A., Perret, L., Tillich, J.-P.: Algebraic cryptanalysis of mcEliece variants with compact keys. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 279–298. Springer, Heidelberg (2010)
7. Finiasz, M., Sendrier, N.: Security bounds for the design of code-based cryptosystems. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 88–105. Springer, Heidelberg (2009)
8. Johansson, T., Jönsson, F.: On the complexity of some cryptographic problems based on the general decoding problem. IEEE Transactions on Information Theory 48(10), 2669–2678 (2002)
9. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. DSN Progress Report, Jet Prop. Lab., California Inst. Technol., Pasadena, CA, pp. 114–116 (January 1978)
10. Naccache, D., Pointcheval, D., Stern, J.: Twin signatures: an alternative to the hash-and-sign paradigm. In: ACM Conference on Computer and Communications Security, ACMCCS 2001, pp. 20–27. ACM, New York (2001)
11. Niederreiter, H.: Knapsack-type crytosystems and algebraic coding theory. Prob. Contr. Inform. Theory 15(2), 157–166 (1986)
12. Overbeck, R.: Recognizing the structure of permuted reducible codes. In: Augot, D., Sendrier, N. (eds.) International Workshop on Coding and Cryptography, WCC 2007, pp. 269–276 (2007)
13. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–304. Springer, Heidelberg (2002)