CFS Software Implementation

Gregory Landais Nicolas Sendrier

INRIA Paris-Rocquencourt, Project-Team SECRET

December 12, 2012

Motivations

CFS has serious benefits :

- Cheapest verification of all known secure digital signature schemes ($\approx$ 30 XORs)
- Scales as technology progresses
- Few secure digital signature schemes exist
- Quantum Computer resilient

Common beliefs

- Public key is very very big. $\rightarrow$ That's true but some use cases can deal with it.
- Long signature time $\rightarrow$ This talk.

Purpose of this talk

1. Demonstrate CFS is practical
2. Study its algorithmic difficulties
3. Illustrate with a classical implementation

CFS

First code-based signature scheme. Relies on :

- hardness of the syndrome decoding problem
- the undistinguishability of a binary Goppa code

CFS instance

A CFS instance is defined by a binary Goppa code Γ

- of length $n \leq 2^m$
- of support $L = (\alpha_0, \ldots, \alpha_{n-1})$, an ordered sequence of distincts elements of $\mathbb{F}_{2^m}$
- of polynomial generator g of degree t
- with an algebraic t-error correcting procedure
- of dimension $k \leq n - m \times t$
- of parity check matrix $H \in \{0, 1\}^{n \times (n-k)}$

Parameters : m, t ($, \lambda$ for Parallel-CFS)

Public key : H

Secret key : L, g

Definition

Key generation

Pick a random Goppa code.

Signing

Hash the message to a syndrome, decode it and use the error vector as the signature.

Verifying

Multiply the error vector by the parity check matrix and check whether it matches the hash of the message.

Scalability/Security

2001 Publication by N. Courtois, M. Finiasz, N. Sendrier.

Signature cost	$t!m^2t^2$
Signature length	mt
Verification cost	mt^2
Public-key size	$tm2^m$
Security	$2^{tm/2}$

Scalability/Security

2003 Unpublished attack from D. Bleichenbacher.

Signature cost	$t!m^2t^2$
Signature length	mt
Verification cost	mt^2
Public-key size	$tm2^m$
Security	$2^{tm/3}$

Scalability/Security

2010 Parallel-CFS countermeasure by M. Finiasz.

Signature cost	$\lambda t! m^2 t^2$
Signature length	$\lambda m t$
Verification cost	$\lambda m t^2$
Public-key size	$t m 2^m$
Security	$2^{tm \frac{2^\lambda - 1}{2^{\lambda+1} - 1}}$

Key security issue

2011 Distinguisher for high rate Goppa codes by Faugère, Gauthier, Otmani, Perret & Tillich

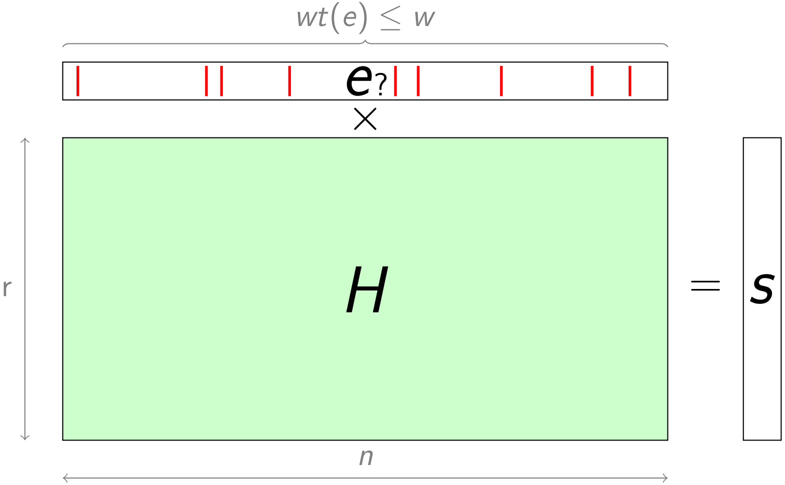
- Invalidates the security reduction.
- Do not lead to an attack.

Message security

Computational Syndrome Decoding Problem

Given $H \in \{0,1\}^{r \times n}$, $s \in \{0,1\}^r$ and $w \in \mathbb{N}$, find $e \in \{0,1\}^n$ such as $hamming\_weight(e) \leq w$ and $H \times e^T = s$.

CSD



NP-hard problem; conjectured difficult on average for suitable parameters.

Signing with CFS

function $\textsc{sign}(M)$ ▷ input: message M
  $S \leftarrow \mathrm{syndromes}(M)$ ▷ S is a family of syndromes (typically obtained by hashing)
  for all $s \in S$ **do**
    $e \leftarrow \mathrm{algebraic\_decoder}(s)$
    if $e \neq$ **fail then**
      return e, s
    end if
  end for
end function

Probability of success of the decoding $\approx \frac{1}{t!}$

Generating the family of syndromes

1. Counter appending : append a counter to the message before hashing it to a syndrome.
   - Hashing performed on the target architecture
   - Variable signature size
   - No Parallel-CFS counter measure

   BAD IDEA

2. Complete decoding : hash the message to a unique syndrome and try to guess δ elements of the corresponding error vector.
   - Adds a recoverable signature failure probability

   BETTER IDEA

Complete decoding

```
function SIGN(M)                                ▷ input: message M
    s_0 ← hash(M)
    for all e' of weight δ do
        s ← s_0 + syndrome(e')
        e ← algebraic_decoder(s)
        if e ≠ fail then
            return e, e', s
        end if
    end for
end function
```

Let's open the black box

function $\textsc{sign}(M)$ ▷ input: message M

  $s_0 \leftarrow \mathrm{hash}(M)$

  for all e' of weight δ **do**

    $s \leftarrow s_0 + \mathrm{syndrome}(e')$

    $\sigma(z) \leftarrow \mathrm{solve\_key\_eq}(s)$

    if $\sigma(z)$ splits in $\mathbb{F}_{2^m}[z]$ **then**

      return $\mathrm{roots}(\sigma(z)), e', s$

    end if

  end for

end function

Decoding methods

Several decoding methods exist.
We considered two of them :

- Berlekamp-Massey
- Patterson

Let's count

		critical				non critical	
(m,t)	type	(1)	(2)	(3)	(1)+(2)+(3)	(4)	(5)
(18,9)	BM	58	180	840	1078	2184	3079.1
(18,9)	Pat.	38	329	840	1207	1482	3079.1
(20,8)	BM	52	144	747	943	1950	3024.6
(20,8)	Pat.	34	258	747	1039	1326	3024.6

(1) syndrome adjustment
(2) key equation solving
(3) split checking
(4) initial syndrome
(5) root finding

Table: Number of field operations (excluding additions) per decoding

Finite field arithmetic

Store logarithm and the exponentiation of each element in base α, a primitive element of $\mathbb{F}_{2^m}$.
Space used :

$\mathbb{F}_{2^{20}}$ $2^{20} \times 2 \times 4\mathrm{B} = 8192\mathrm{KB}$

$\mathbb{F}_{2^{10}}$ $2^{10} \times 2 \times 2\mathrm{B} = 4\mathrm{KB}$

Cache size of Intel XEON W3550 :

L1 $4 \times 32\mathrm{KB}$

L2 $4 \times 256\mathrm{KB}$

L3 $8192\mathrm{KB}$

Timings of my implementation

	(m, t, δ, λ)			
	(18,9,2,3)	(18,9,2,4)	(20,8,2,3)	(20,8,1,5)
decoding	1 117 008	1 489 344	121 262	360 216
BM	14.70 s	19.61 s	1.32 s	3.75 s
Pat	15.26 s	20.34 s	1.55 s	4.26 s
security bits	83.4	87.0	82.5	87.3

Table: Average number of algebraic decoding and running time per signature

75% of the CPU time for the field multiplication

Conclusion

- Signing with codes and 80 bits of security in less than 1 second is possible.
- Berlekamp-Massey is better for CFS
- Most optimisation efforts should focus on the finite field arithmetic

Further works

- Make the code public
- Benchmark it (eBACS)
- Bit-slice it (joint work with Peter Schwabe)
- FPGA it (joint work with Jean-Luc Beuchat)

Further works

- Make the code public
- Benchmark it (eBACS)
- Bit-slice it (joint work with Peter Schwabe)
- FPGA it (joint work with Jean-Luc Beuchat)

Thank you