Short McEliece-based Digital Signatures

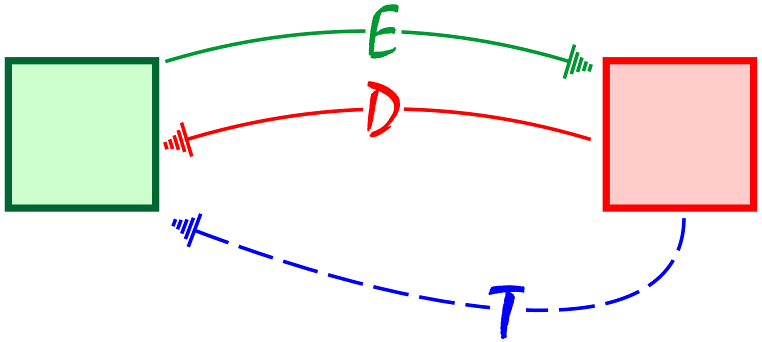
Nicolas Courtois Matthieu Finiasz Nicolas Sendrier

ISIT 2002 – Lausane

INRIA

Public-key cryptography

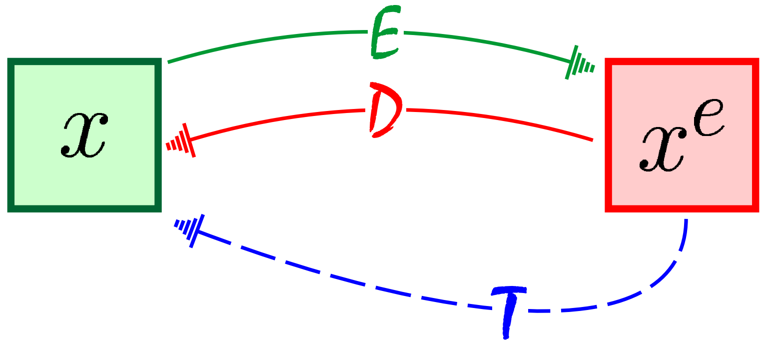
Some definitions



$\mathcal{E}$ = computationnaly easy one-way function
$\mathcal{D}$ = computationnaly difficult inversion problem
$\mathcal{T}$ = trap: secret inversion algorithm

For example with RSA we have:

$\mathcal{E}$ = modular exponentiation: $x \mapsto x^e$
$\mathcal{D}$ = e^{th}-root extraction problem
$\mathcal{T}$ = $x \mapsto x^d$

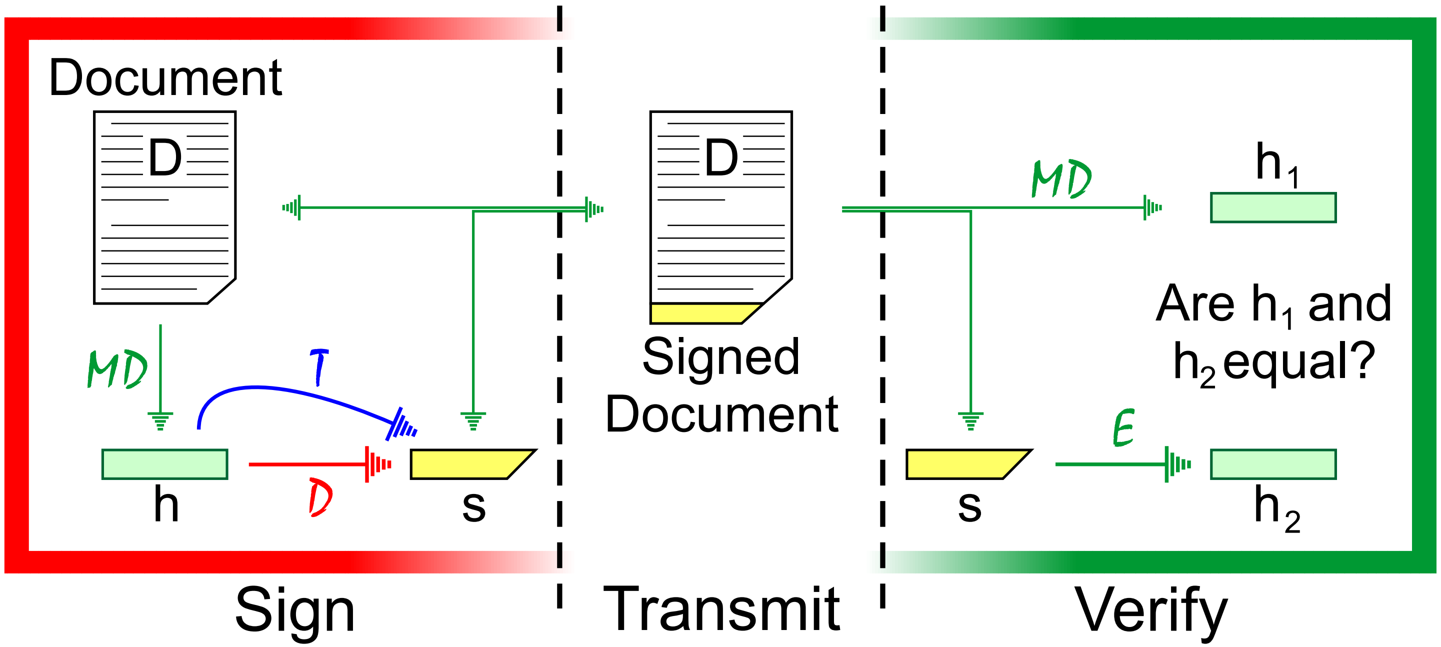


From one-way functions to signature

When $\mathcal{A}$ signs a document D he computes $s(D,\mathcal{A})$ with the following properties:

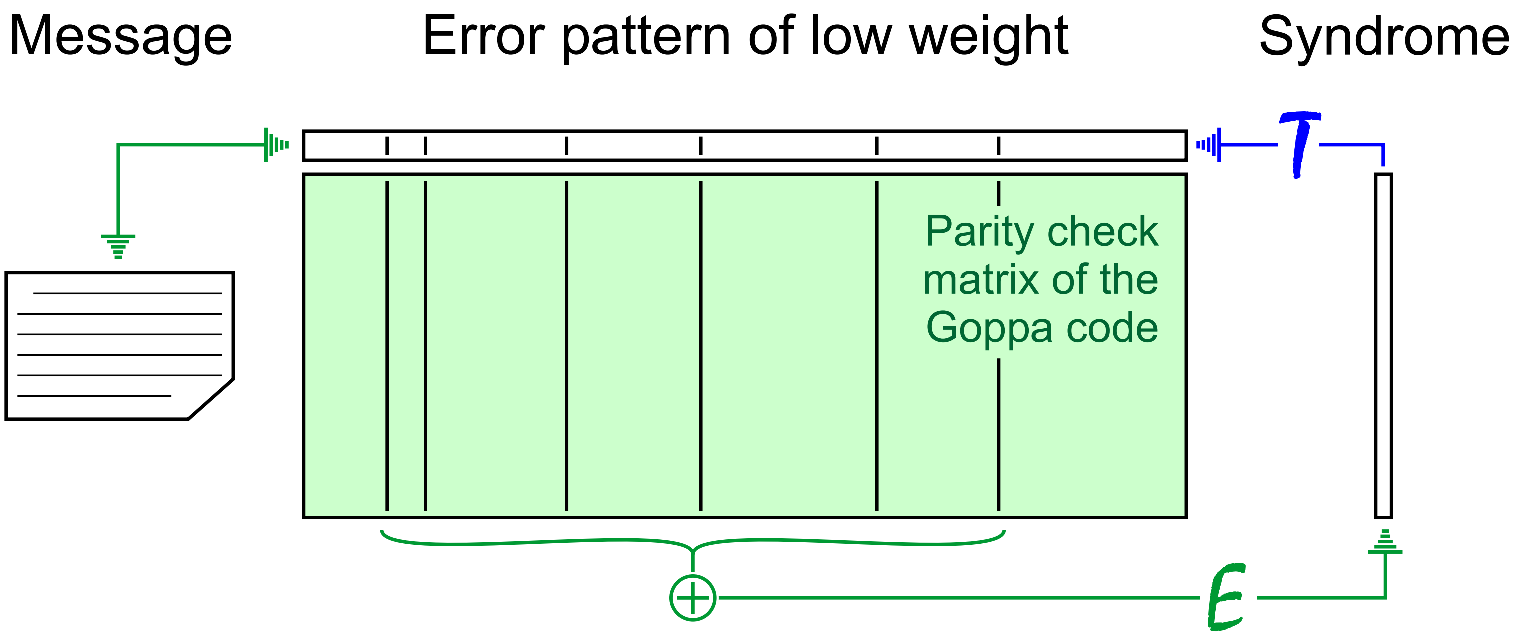
- for a given D, only $\mathcal{A}$ can compute $s(D,\mathcal{A})$
- for a given σ it is impossible to find D such that $s(D,\mathcal{A}) = \sigma$

It is possible to achieve this with a one-way hash function *MD* and a one-way function *E* with a trap *T*.



Using error-correcting codes: the McEliece cryptosystem

$\Rightarrow$ We can use Niederreiter's variant of the McEliece cryptosystem



To sign we will:

- ◇ hash the document (using whatever message digest $\mathcal{MD}$) into a syndrome
- ◇ decode this syndrome into an error pattern using the trap $\mathcal{T}$
- ◇ use the equivalent message as signature

To verify the signature we simply compare the results of $\mathcal{E}$ and $\mathcal{MD}$

Inversion problems

In this scheme we need to apply $\mathcal{T}$ to the result of $\mathcal{MD}$

$\Rightarrow$ we need to decode a “random” syndrom

$\Rightarrow$ the trap $\mathcal{T}$ can decode syndromes corresponding to error patterns of Hamming weight $\leq t$

$\Rightarrow$ we can only decode a small ratio of syndromes

With the original McEliece parameters: $t = 50$, $m = 10$, $n = 1024$.

- there are 2^{500} different syndromes (of length 500)
- there are $\binom{1024}{50} \simeq 2^{284}$ error patterns of weight less than t

$\Rightarrow$ This makes a ratio of 1 decodable syndrome out of 2^{216}.

We need to:

- either change $\mathcal{MD}$ so that it returns decodable syndromes
- or perform complete decoding

Complete decoding

To perform complete decoding we will need to increase this ratio

$\Rightarrow$ It will be greater for codes correcting less errors, as a Goppa code can correct approximately 1 syndrome out of $t!$

$$\overbrace{\binom{n}{t}}^{\text{decodable syndromes}} \cdot \frac{1}{\underbrace{2^{mt}}_{\text{total syndromes}}} \simeq \frac{n^t}{t!\ \underbrace{2^{mt}}_{n}} \simeq \frac{1}{t!}$$

$$\begin{aligned} 9! &= 362\,880 \\ 10! &= 3\,628\,800 \\ 11! &= 39\,916\,800 \end{aligned}$$

If we want the signature time to stay reasonnable we will need a t not greater than 10

INRIA

Secure parameters

We have a small t but still want a good security (about 2^{80} CPU operations)
$\Rightarrow$ n will be large

Number of binary operations for an attack
(based on the attack by A. Canteaut and F. Chabaud [CC98])

n	$t=9$	$t=10$
2^{13}	$2^{69.3}$	$2^{72.3}$
2^{14}	$2^{74.0}$	$2^{77.4}$
2^{15}	$2^{78.8}$	$2^{87.4}$
2^{16}	$2^{83.7}$	$2^{90.9}$
2^{17}	$2^{88.2}$	$2^{94.6}$

$$\begin{cases} t = 10 \text{ and } n = 15 \\ t = 9 \text{ and } n = 16 \quad \longleftarrow \text{ 10 times faster} \end{cases}$$

Signature algorithm

We use the following algorithm:

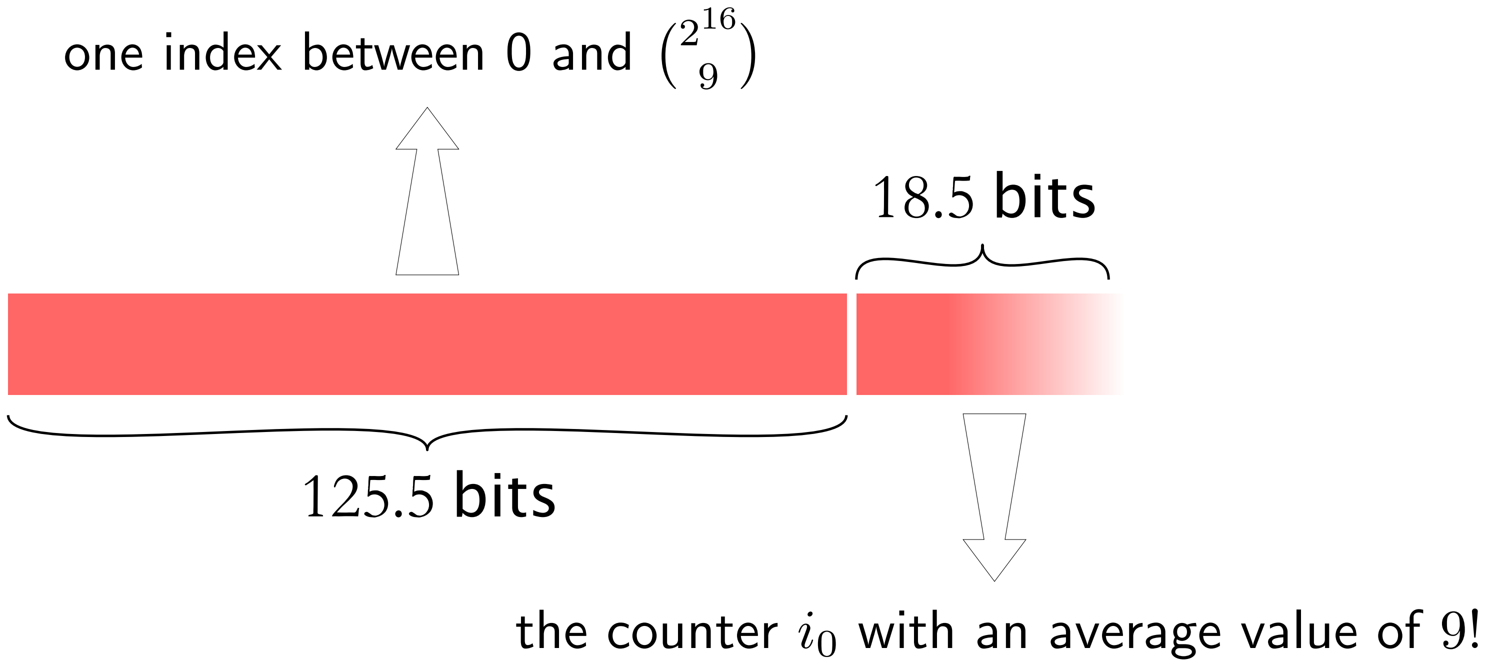
- add a counter i to the document
- apply $\mathcal{MD}$ to the document and i at the same time to obtain a syndrome s_i
- try to decode s_i with $\mathcal{T}$
- if it does not work, increment i and try again

We call i_0 the smallest value of i for which the decoding is possible.

$\Rightarrow$ The signature will have to contain both $\mathcal{T}(s_{i_0})$ and i_0 for the verification

Signature size

We index all the words of weight 9 and length 2^{16}.



$\Rightarrow$ The counter must be present for verification and can be made of constant length if necessary

$\Rightarrow$ Signature is in average 144 bits long.

Reducing the signature size. . .

Verification is very fast (summing 9 columns of H and hashing one file)

$\Rightarrow$ The signature can be shortened by omitting some information: verificator will then try all possible values

$\Rightarrow$ Signature will contain less than t positions

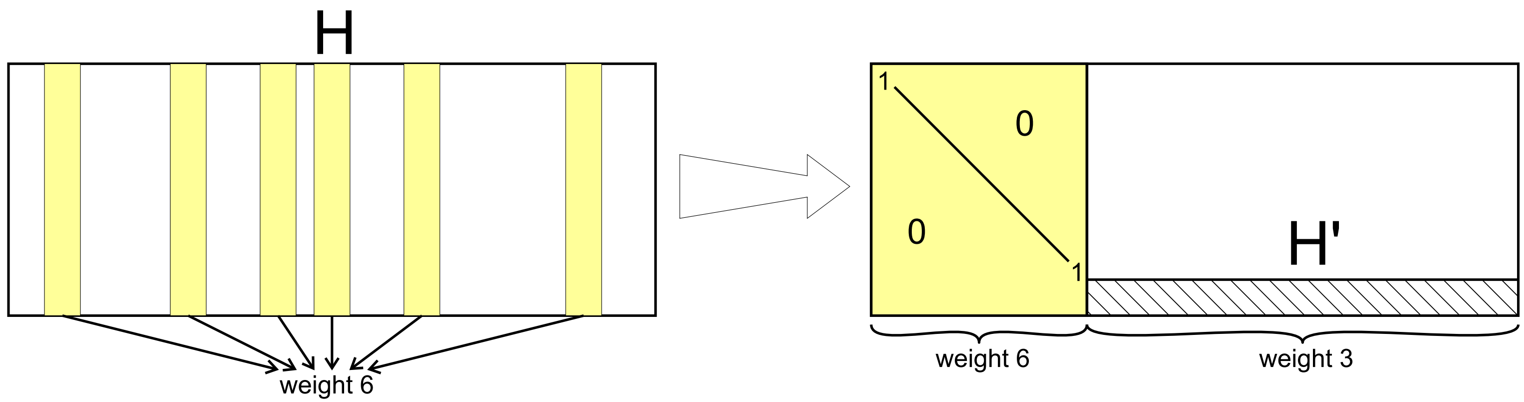
omitted positions	signature length		verification	
	partial	total	WF	time
0	125.5	144	9	$\sim \mu$s
1	112.7	131	9	$\sim \mu$s
2	99.7	118	2^{14}	$\sim$ ms
3	86.5	105	2^{27}	$\sim$ 30s
4	73.1	92	2^{40}	—
5	59.4	77	2^{54}	—

We can verify a signature of 105 bits in about 30 seconds.

INRIA

Reducing more

We can reduce the signature size even more by giving only approximate positions
$\Rightarrow$ group the columns in small clusters of 16 columns



$\Rightarrow$ We decode 3 errors in a shortened code. The parity check matrix H' of this code is obtained by applying a Gaussian elimination to H (about 2^{24} column operations).

$\Rightarrow$ We can get signatures of 81 bits.

Scalability

$\Rightarrow$ The signature algorithm is easily scalable. For one omitted position we have the following asymptotic values:

signature cost	$t!t^2m^3$
signature length	$(t-1)m + \log_2 t$
verification cost	t^2m
public key size	$tm2^m$
cost of best decoding attack	$2^{tm(1/2+o(1))}$

$\Rightarrow$ Security increases much faster than any other parameter

Conclusion

- Signature using McEliece is possible!
- The algorithm obtained is polymorphic. It gives:
  - either very short signatures of 81 bits
  - or short signatures (131 or 118 bits) with a faster verification
- the signature time is long (about 1 minute)
- the public key is large (1MB)
- its security relies on well known hard problems
- it is easily scalable