Selected Topics on Security and Cryptography 2005

Codes in Cryptography

Matthieu Finiasz

Outline

I	Introduction to linear error-correcting codes
II	Some famous linear codes
III	The McEliece public key cryptosystem
IV	Other cryptographic constructions relying on hard coding problems
V	Other applications where codes can be useful...

Part I

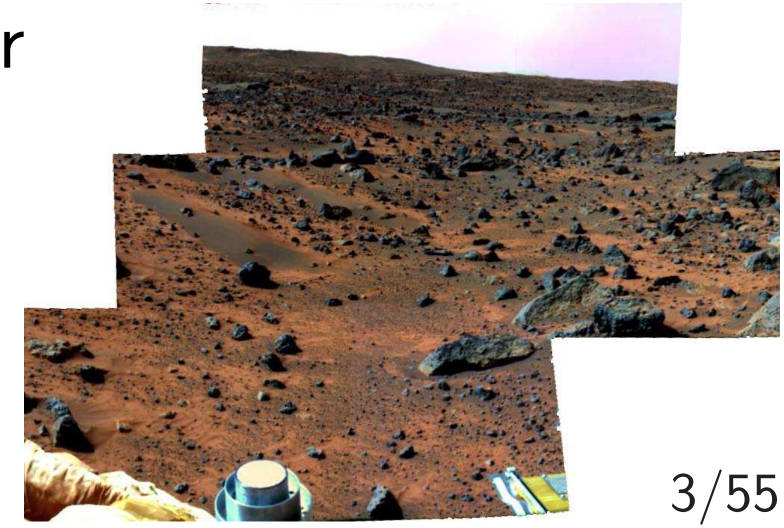
Introduction to linear error-correcting codes

What are error-correcting codes?

- They make possible the correction of errors when communicating over a noisy channel.
  - Add redundancy to the transmitted information.
  - Correct errors when the received data is corrupted.
- Stronger than a simple CRC or checksum: these can only detect errors.

Where are they used?

- DVD, CD: reduce the effect of dust and scratches
- cell-phones: improve communication quality
- Mars Pathfinder: save energy when sending pictures to Earth.
  - for a same final error probability, it is cheaper to emit longer with less power
- cryptography...



What are linear codes?

- The most widely used kind of error-correcting codes,
  - tend to be replaced by convolutional codes...
- Error-correcting codes for which the redundancy depends linearly of the information.
- Can be defined by a generator matrix $\mathcal{G}$:

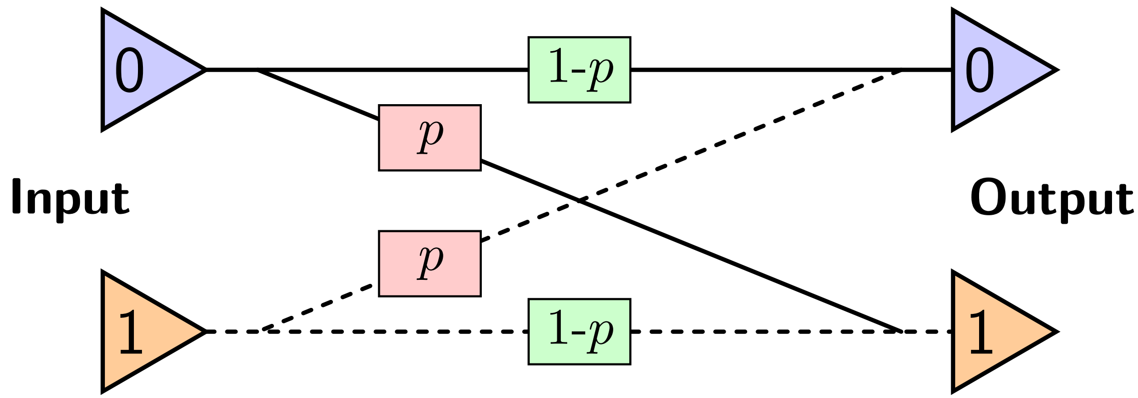
$$c = m \times \underbrace{\left[\begin{array}{ccc|c} 1 & & 0 & \\ & \ddots & & \\ 0 & & 1 & \end{array}\right]}_{\mathcal{G}}$$

- The generator matrix $\mathcal{G}$ may not be given in systematic form, but is always of maximal rank.
- The code $\mathcal{C}$ is the vectorial subspace of dimension k defined by $\mathcal{G}$
  - there is not a unique generator matrix.
- The length n of the code is the length of a code word.
  - the matrix $\mathcal{G}$ is of size $k \times n$.
- The ratio $r = \frac{k}{n}$ is the transmission rate of the code.

- The transmitter sends $c = m\mathcal{G}$, but the receiver will get $c' = c + e$.
  - Decoding consists in recovering c from c'.

- Most often, we want maximum likelihood decoding:
  - find the code word which had the best probability of giving the received word.
  - This will depend on the channel/noise.

The binary symmetric channel



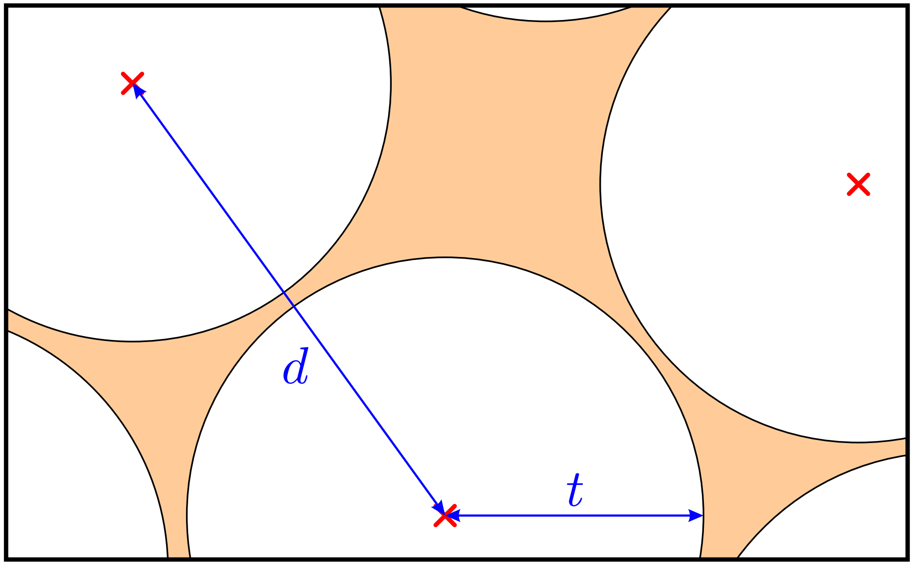
- The Hamming weight of a word c is it's number of non-zero coordinates.
  - Most probable errors are those of lower weight.
- Decoding c' consists in finding the closest (for the Hamming distance) code word.

Minimal distance

- The minimal distance d of a code is the minimum of the Hamming distance between two code words.
  - It is also the smallest possible weight for a non-zero code word.
- For any code $d \leq n - k + 1$.
  - If $d = n - k + 1$ the code is called Maximum Distance Separable (MDS).
- We note $[n, k, d]$ a code of length n, dimension k and minimal distance d.

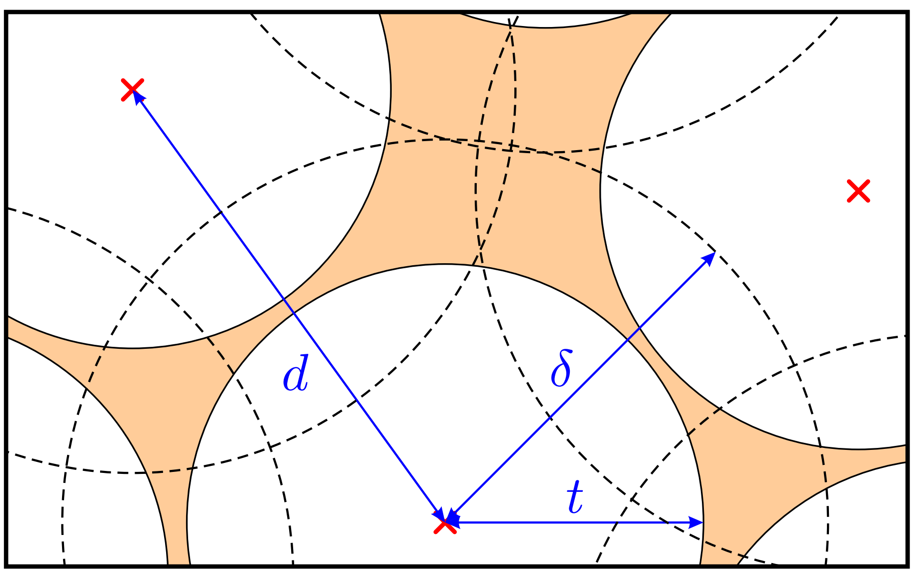
Bounded decoding

- Maximum likelihood decoding is often hard to achieve.
- We restrict to bounded decoding up to the distance t:
  - find any code word at distance less or equal to t.
  - If $t \leq \frac{d-1}{2}$ decoding is always unique.



Bounded decoding

- Maximum likelihood decoding is often hard to achieve.
- We restrict to bounded decoding up to the distance t:
  - find any code word at distance less or equal to t.
  - If $t \leq \frac{d-1}{2}$ decoding is always unique.



δ is the covering radius.
⚠ Bounded decoding up to δ is not unique.

- **Error exhaustive search**: choose e of small weight, calculate $c' - e$ and check if it is in the code.
- **Code word exhaustive search**: calculate $c' - m\mathcal{G}$ for all possible m and check its weight.
- **Information Set Decoding**: choose k coordinates of c' and reconstruct $c'' = (c'\mathcal{G}^{-1})\mathcal{G}$ for these coordinates. Check the weight of $c' - c''$.
  - $c'' = c$ if there is no error among the k coordinates.
    - check $\binom{n-k}{t}$ error patterns at a time.

The parity check matrix $\mathcal{H}$ is orthogonal to $\mathcal{G}$:

- it is a $(n-k) \times n$ matrix.
- the code $\mathcal{C}$ is the kernel of $\mathcal{H}$.
  - $c \in \mathcal{C}$ if and only if $\mathcal{H}c = 0$.
- $\mathcal{S} = \mathcal{H}c' = \cancel{\mathcal{H}c} + \mathcal{H}e$ is the syndrome of the **error**.

- Syndrome decoding consists in finding a low weight linear combination of columns of $\mathcal{H}$ summing to $\mathcal{S}$.
  - The same methods apply: information set decoding...

Part II

Some famous linear codes

The repetition code

- Each bit is simply reapeated d times:
  - 00100 is coded 000 000 111 000 000.
- This code is a $[d, 1, d]$ code.
  - it is MDS!

- Transmission rate is too small.
- Only usefull for very high noise level in a memoryless channel.

The Hamming code

- It is a binary $[2^\ell - 1, 2^\ell - 1 - \ell, 3]$ code. Its parity check matrix contains all the different ℓ bit columns.

For $\ell = 3$ it looks like:

$$\mathcal{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

- The minimal distance d is 3.
  - No code words of weight 1 or 2.
- Syndrome decoding can correct exactly one error.
- These are perfect codes: any word can be decoded.

Reed-Solomon codes

[Reed-Solomon 1960]

Evaluation codes over $\mathbb{F}_q$ (usually $\mathbb{F}_{2^m}$).

- The support $\mathcal{L}$ of the code is a list of n elements of $\mathbb{F}_q$.
- The RS code of support $\mathcal{L}$ and dimension k contains the evaluations (on $\mathcal{L}$) of all polynomials of degree $< k$.

For $\mathcal{L} = (\alpha_1, ..., \alpha_n)$, and a message $m = (m_0, ..., m_{k-1})$:

- we define $P(X) = \displaystyle\sum_{i=0}^{k-1} m_i X^i$,
- we get the code word $c = (P(\alpha_1), ..., P(\alpha_n))$.

- If P_1 and P_2 coincide on k points of $\mathcal{L}$ they are equal.
  - The minimal distance of a RS code is $d = n - k + 1$.
  - RS codes are always MDS!
- Decoding can be done very efficiently:
  - uniquely up to $t = \frac{n-k}{2}$ (Berlekamp-Massey).
  - list decoding up to $t = n - \sqrt{nk}$ (Sudan).

⚠ These codes are very convenient, but n has to be smaller or equal to q.

- Using a binary transmission, RS codes will work better correcting burst errors.

Gilbert-Varshamov lower bound:
A $[n, k, d]$ code over $\mathbb{F}_q$ exists if:

$$\sum_{i=0}^{d-2} \binom{n-1}{i} (q-1)^i < q^{n-k}.$$

- In $\mathbb{F}_2$ it gives: $\sum_{i=0}^{d-2} \binom{n-1}{i} < 2^{n-k}$.
  - Simplifying things a lot you get $n^d \lesssim 2^{n-k}$ and:

$$\boxed{d \lesssim \frac{n-k}{\log_2 n}.}$$

Goppa codes
[V.D. Goppa 1970]

Goppa codes are codes on $\mathbb{F}_p$ build from codes on $\mathbb{F}_{p^m}$.

- ▷ choose a support $\mathcal{L} \subset \mathbb{F}_{p^m} = (\alpha_1, \ldots, \alpha_n)$, and a primitive polynomial g of degree t.
- ▷ build a parity check matrix $\mathcal{H}$ of size $t \times n$ in $\mathbb{F}_{p^m}$.
- ▷ extend $\mathcal{H}$ to a $mt \times n$ parity check matrix on $\mathbb{F}_p$.
- ▶ The code $\Gamma(\mathcal{L}, g)$ has a minimal distance $\geq t + 1$.
- ▶ When $p = 2$, $\Gamma(\mathcal{L}, g^2) = \Gamma(\mathcal{L}, g)$ and has a minimal distance of $2t + 1$.
  - ▷ Decode t errors uniquely (Berlekamp-Massey).

Random codes

A random code is defined by a random $k \times n$ generator matrix $\mathcal{G}$ of rank k.

- Random codes are good codes!
  - In average the minimal distance meets the GV bound.
- Decoding in a random linear code is a NP-complete problem.
- Finding the minimal distance of a random linear code is a NP-complete problem.

Part III

The McEliece public key cryptosystem

[McEliece 1978]

The basic idea

- Generate a code and its generator matrix $\mathcal{G}$.
  - This is the private key.
- Scramble $\mathcal{G}$ to obtain $\mathcal{G}'$ which looks like random.
  - This is the public key.
- Encode a message m by computing:

  $$c' = m\mathcal{G}' + e \qquad \text{with } e \text{ a random error.}$$

- Only the person knowing the underlying structure in $\mathcal{G}'$ can decode and recover m.

Using binary Goppa codes

- A Goppa parity check matrix has a structure in $\mathbb{F}_{2^m}$.
  - Once projected on $\mathbb{F}_2$ this structure is spread over different lines.
- Take a Goppa code $\Gamma(\mathcal{L}, g)$, its generator matrix $\mathcal{G}$, a permutation $\mathcal{P}$ and an invertible matrix $\mathcal{Q}$.
  - Compute $\mathcal{G}' = \mathcal{Q} \times \mathcal{G} \times \mathcal{P}$
- Distinguishing $\mathcal{G}'$ from a random binary matrix is believed to be a hard problem.

Key generation

- Choose some parameters n, t, m
  - make sure $n \leq 2^m$ and $2mt \leq n$
- Choose a subset $\mathcal{L} \subset \mathbb{F}_{2^m}$ of size n and a primitive polynomial g of degree t on $\mathbb{F}_{2^m}$.
- Build $\Gamma(\mathcal{L}, g)$ and a generator matrix $\mathcal{G}$
- Choose random matrices $\mathcal{P}$ and $\mathcal{Q}$.
- Compute $\mathcal{G}' = \mathcal{Q} \times \mathcal{G} \times \mathcal{P}$

- $\mathcal{G}'$ is the public key, $(\mathcal{L}, g, \mathcal{P}, \mathcal{Q})$ are the private key.

Encryption

Using the public key

- ◇ Split the message in blocks of length $k = n - 2mt$
- ◇ Encrypt each block b_i independently
  - Compute $c_i = b_i \times \mathcal{G}'$.
  - Choose a random error e of weight t.
  - Compute $c'_i = c_i + e$.
- ◇ Send the encrypted message $(c'_0 || c'_1 || \ldots)$.

▶ The encrypted message is longer than the original message by a ratio $\frac{1}{r} = \frac{n}{k}$.

Decryption

Using the private key

- For each received block c_i'
  - Compute $c_i'\mathcal{P}^{-1} = (m_i\mathcal{Q})\mathcal{G} \times \cancel{\mathcal{P}\mathcal{P}^{-1}} + e\mathcal{P}^{-1}$.
  - $e\mathcal{P}^{-1}$ is of weight t and $(m_i\mathcal{Q})\mathcal{G} \in \Gamma(\mathcal{L}, g)$.
    - ▷ Using $\mathcal{L}$ and g, decode and recover $m_i\mathcal{Q}$.
  - Compute $(m_i\mathcal{Q})\mathcal{Q}^{-1}$ to obtain m_i.
- Rebuild the original message $(m_0||m_1||...)$.

A public key cryptosystem always relies on two problems:

- ◇ Recovering the private key from the public key.
  - ▷ For RSA: factorization of $n = pq$.
- ◇ Decrypting without knowing the private key.
  - ▷ For RSA: e^{th} root extraction modulo n.

- ▶ For McEliece the problems are:
  - ▷ Distinguishing $\mathcal{G}'$ from a random matrix.
  - ▷ Decoding in a random code (NP-complete).

- Structural attacks: recovering $\Gamma(\mathcal{L}, g)$ from $\mathcal{G}'$.
  - Testing code equivalence is hard in theory, but easy in practice (*support splitting algorithm* [Sendrier 2000]).
  - Test the equivalence between $\mathcal{G}'$ and all Goppa codes.

$$\text{Complexity:} \quad \mathcal{O}\left(mt2^{m(t-2)}\right)$$

- Decoding attacks: decode considering $\mathcal{G}'$ as random.
  - Many information set decoding algorithms.
  - The best one is by A. Canteaut and F. Chabaud.

$$\text{Complexity:} \quad \mathcal{O}\left(2^{mt(\frac{1}{2}+o(1))}\right)$$

The re-encryption problem

Sending twice the same message block b with the same key is dangerous:

- ▷ If one sends $c_0 = b\mathcal{G}' + e_0$ and $c_1 = b\mathcal{G}' + e_1$,
- ▷ the sum $c_0 + c_1 = e_0 + e_1$ is of weight $2t < n - k$.
  - ▷ One can get k coordinates with no errors and decode.

⚠ Using a random e can be dangerous

- ▷ Maybe $e = \text{hash}(b)$ can be more secure.
- ▷ Or add some randomness inside the k bits of message.

The Niederreiter variant
[Niederreiter 1986]

- Consists in putting the information in the error instead of the code word.
  - Send a syndrome of this error.
- The public key is a scrambled parity check matrix:
  - $\mathcal{H}' = \mathcal{Q} \times \mathcal{H} \times \mathcal{P}$.
- The private key is still $(\mathcal{L}, g, \mathcal{P}, \mathcal{Q})$.

Encryption/Decryption

- Encryption:
  - Convert the data into e of length n and weight t.
  - Compute $\mathcal{S} = \mathcal{H}'e$ (sum of t columns of $\mathcal{H}'$).
  - $\mathcal{S}$ is the ciphertext.
- Decryption:
  - Compute $\mathcal{Q}^{-1}\mathcal{S} = \cancel{\mathcal{Q}^{-1}\mathcal{Q}}\mathcal{H}(\mathcal{P}e)$.
  - $\mathcal{P}e$ is of weight t and can be decoded.
  - Reconvert e into the clear text.

McEliece vs. Niederreiter

Which is better?

McEliece	Niederreiter
▷ Transmission rate: $k/n \simeq 0.82$	For $(n = 2048, m = 11, t = 33)$ $\log_2 \binom{n}{t} / mt \simeq 0.66$
▷ Block size: $k = 1685$	$\log_2 \binom{n}{t} \simeq 240$
▷ Encryption cost (per bit): $\mathcal{O}(n)$	$\log_2 \binom{n}{t} \simeq \log_2 \frac{n^t e^t}{t^t} \simeq t(m - \log_2 t)$ $\mathcal{O}(t)+$ error encoding
▷ Decryption cost: syndrome + decoding + inversion	decoding + error de-encoding
▷ Re-encryption problem: Yes	No

Constant weight encoding

Preparing Niederreiter input

- Problem: how can I transform binary data in a word of length n and weight t?
- Exact conversion: index words with $\log_2 \binom{n}{t}$ bit integers.
  - Error e has non zero bits at positions $(i_1, ..., i_t)$:
$$I_e = \binom{i_1}{1} + \binom{i_2}{2} + ... + \binom{i_t}{t}.$$
- Regular words: build t words of weight 1 and length $\frac{n}{t}$.
  - e will have one non zero position per block of $\frac{n}{t}$.
  - Only $t \log_2 \frac{n}{t}$ bits per word.
  - What about security? Is it still hard to decode?

Constant weight encoding

Using source coding techniques

Use the binary data to code the distance between the non-zero positions of e.

▷ A bit complicated to be explained here...

- Very fast constant weight encoding.
- Covers $\approx 99\%$ of possible errors e.
  ▷ No security issues.

- The amount of data needed to code e is not constant.

Fast public key encryption

Tweaking Niederreiter's parameters

- When $t \ll n$ the best attacks on Niederreiter have a complexity of $\mathcal{O}\left(\text{Poly}(mt) \times 2^{\frac{mt}{2}}\right)$.
  - We need $mt \geq 144$.
- We can choose $m = 16, t = 9$ and $n = 2^{16} = 65536$.
  - The size of $\mathcal{H}'$ is 144×65536 (9 Mbits).
  - Encryption is the XOR of 9 columns of 144 bits.
- Using the source coding constant weight encoding it is possible to reach throughputs of 50Mbits/s in software (10 times faster than RSA-1024 with a light e).

Part IV

Other cryptographic constructions relying on hard coding problems

McEliece digital signature

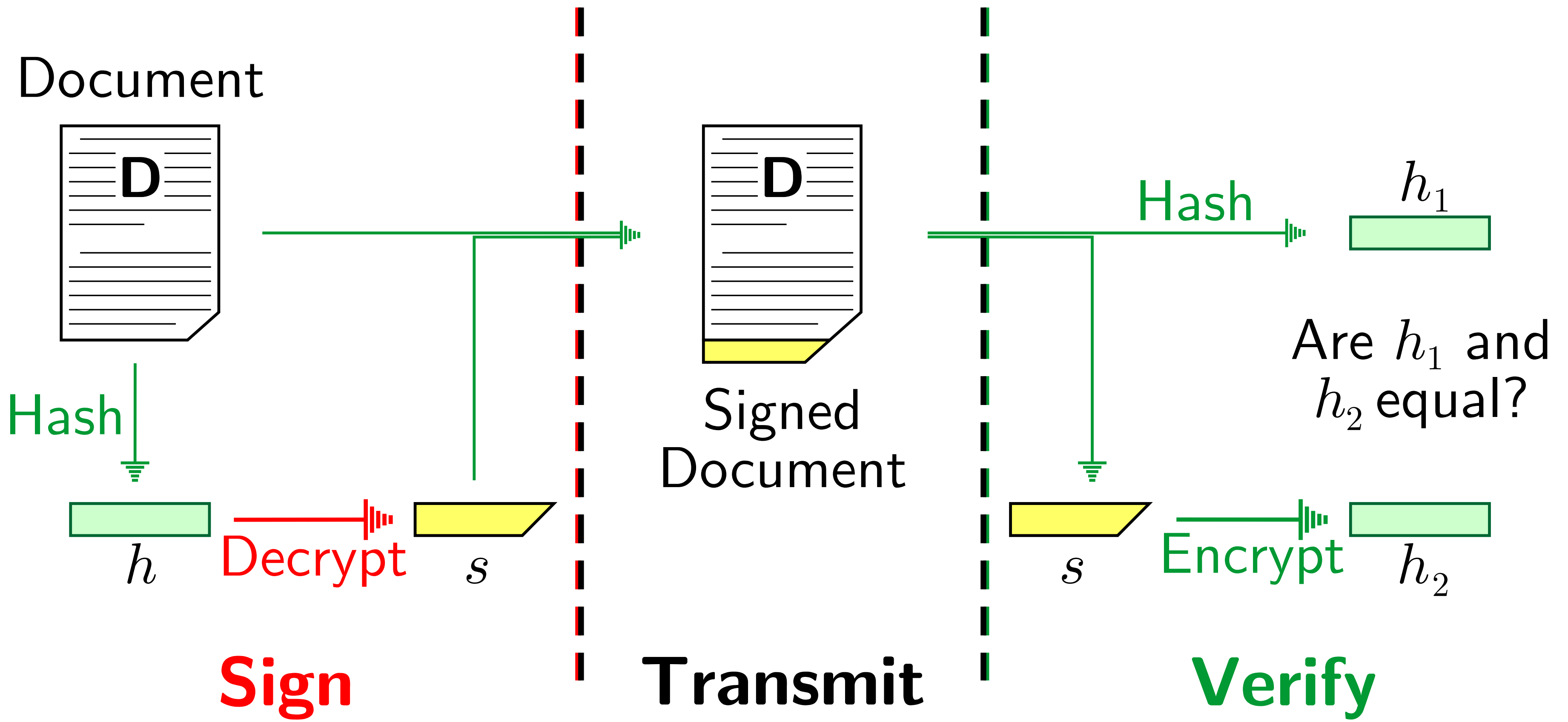
[Courtois, Finiasz, Sendrier 2001]

- Usually, any public key cryptosystem can be transformed in a signature scheme in a straightforward way.
  - It only requires a suitable hash function.

- For McEliece or Niederreiter this is not so easy:
  - this is due to the message expansion.

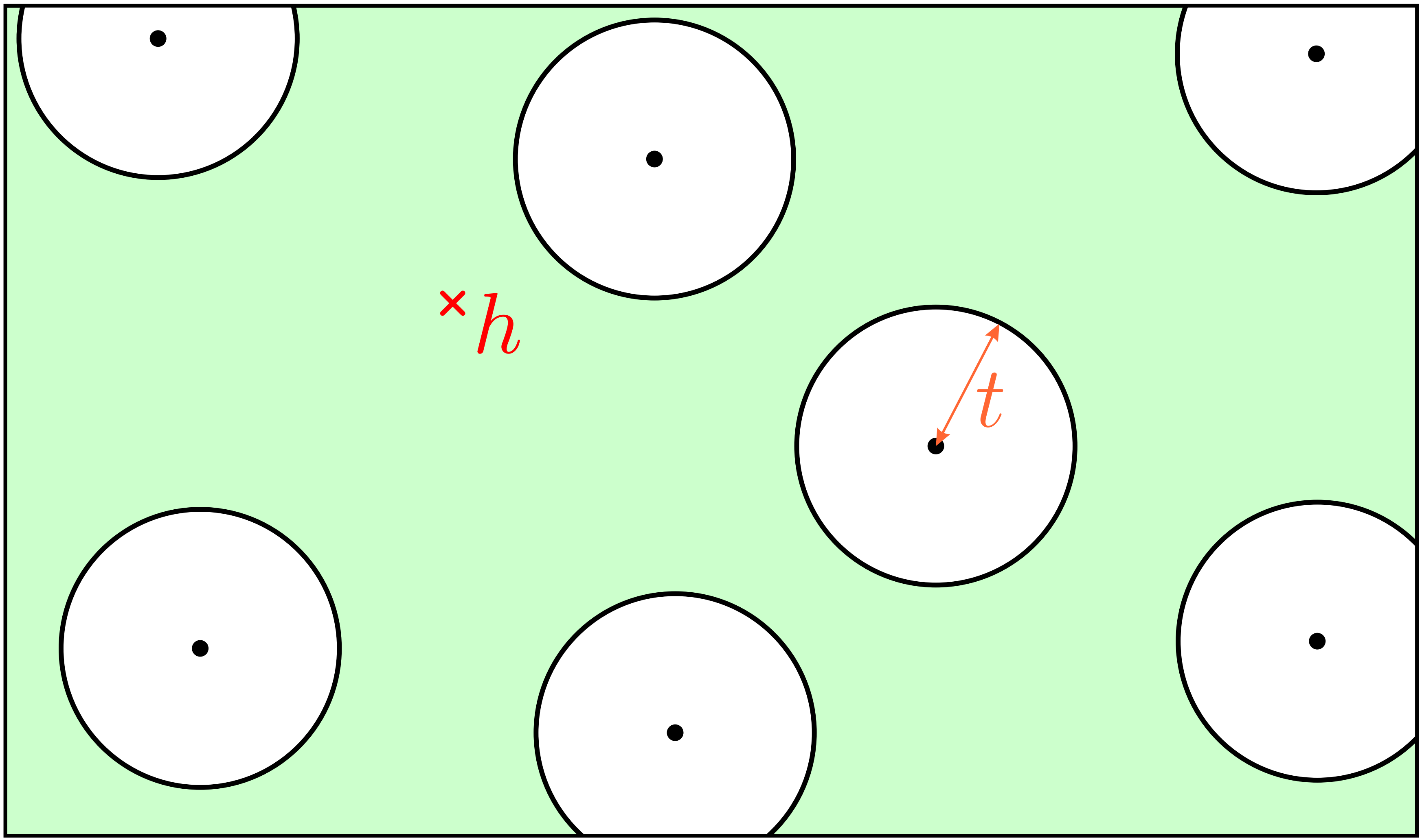
Digital signature

Generic construction

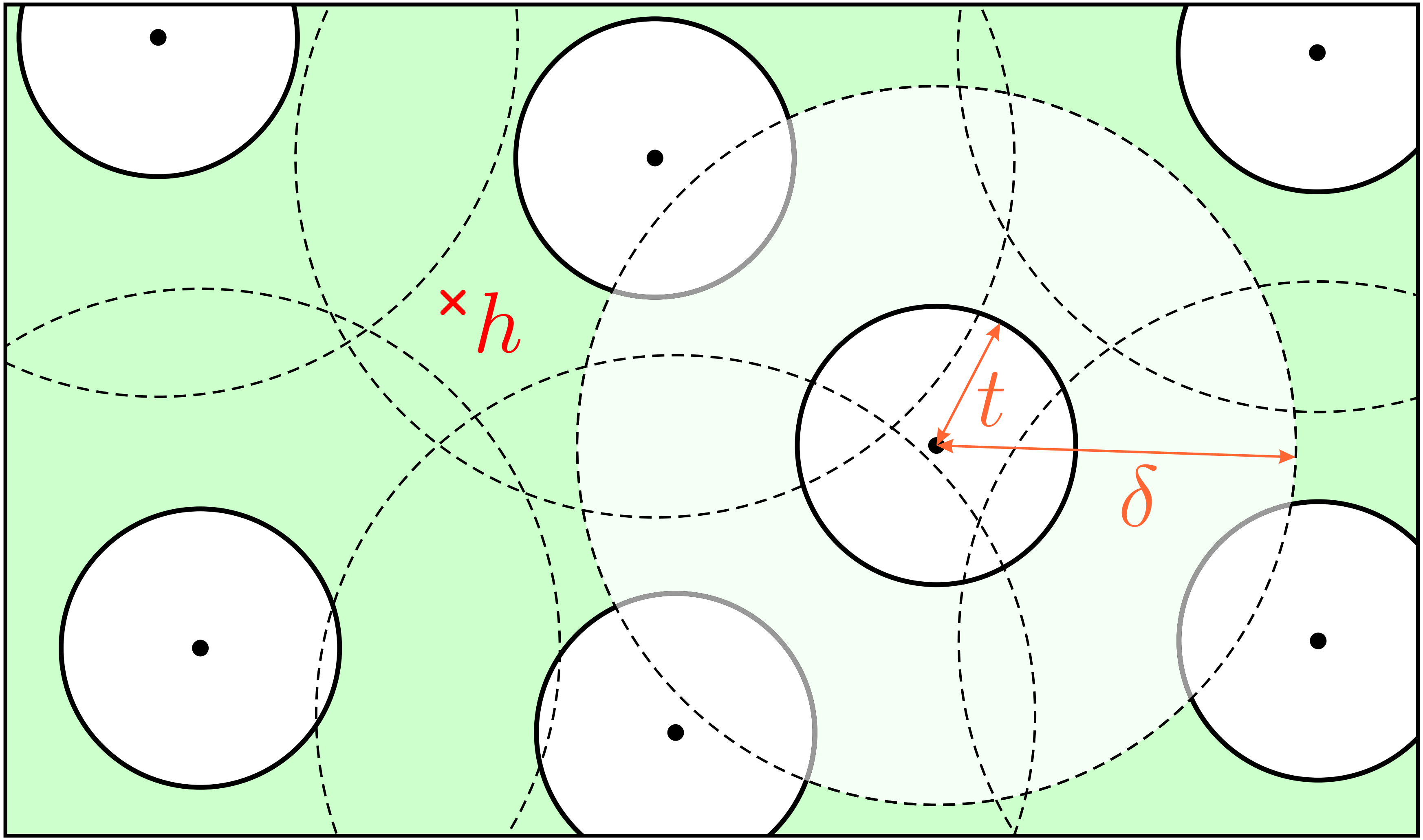


- The ciphertext h is obtained by hashing:
  - requires to decrypt a “random” ciphertext.
- In a Goppa code one can decode up to t errors.
  - The probability $\mathcal{P}_{\leq t}$ that a random word is at distance less or equal to t from a code word is very low.
  - For $(n = 2048, m = 11, t = 33)$ we have $\mathcal{P}_{\leq t} \simeq 2^{-123}$.

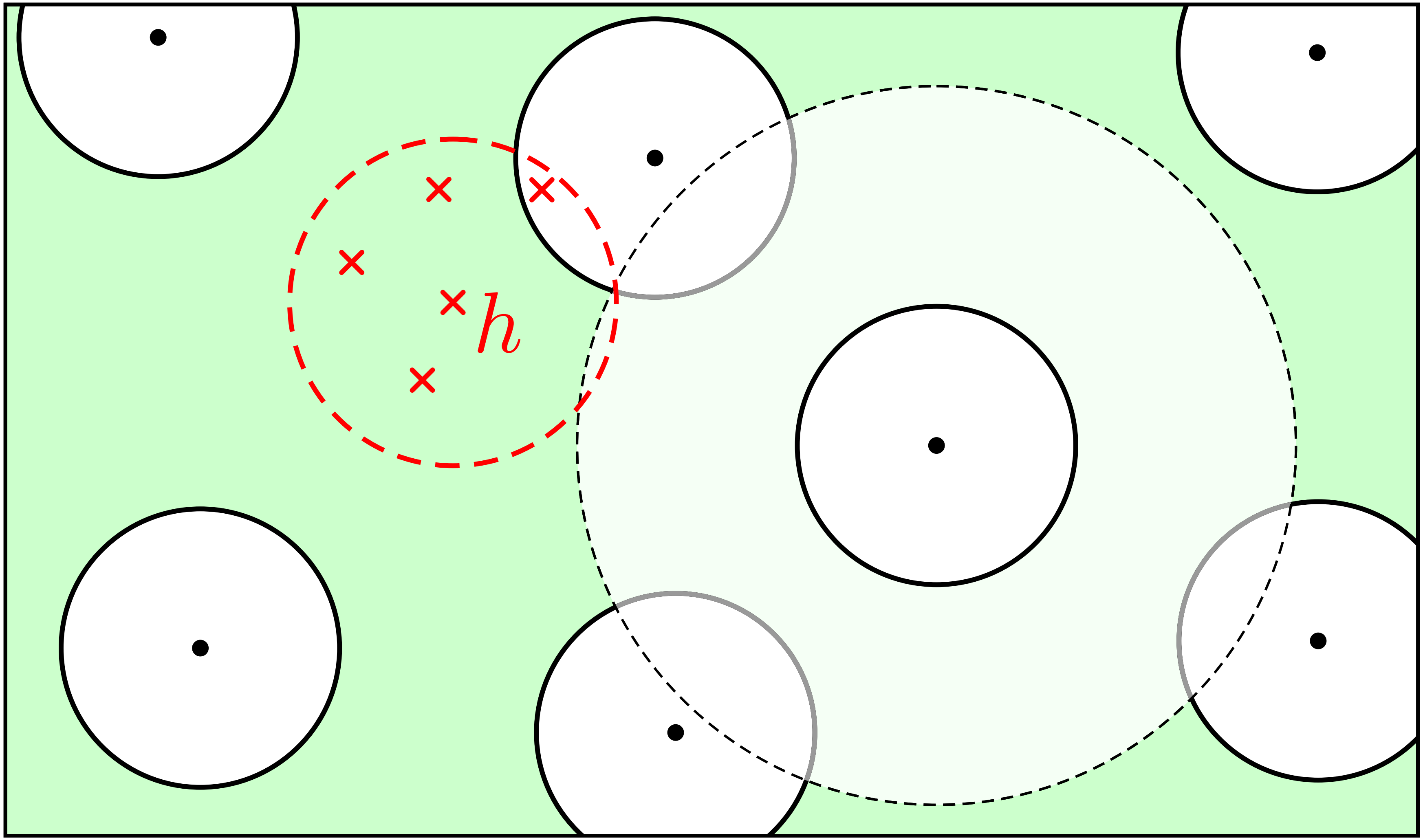
- Two solutions:
  - either we can perform complete decoding.
  - or we need to hash into a decodable word.



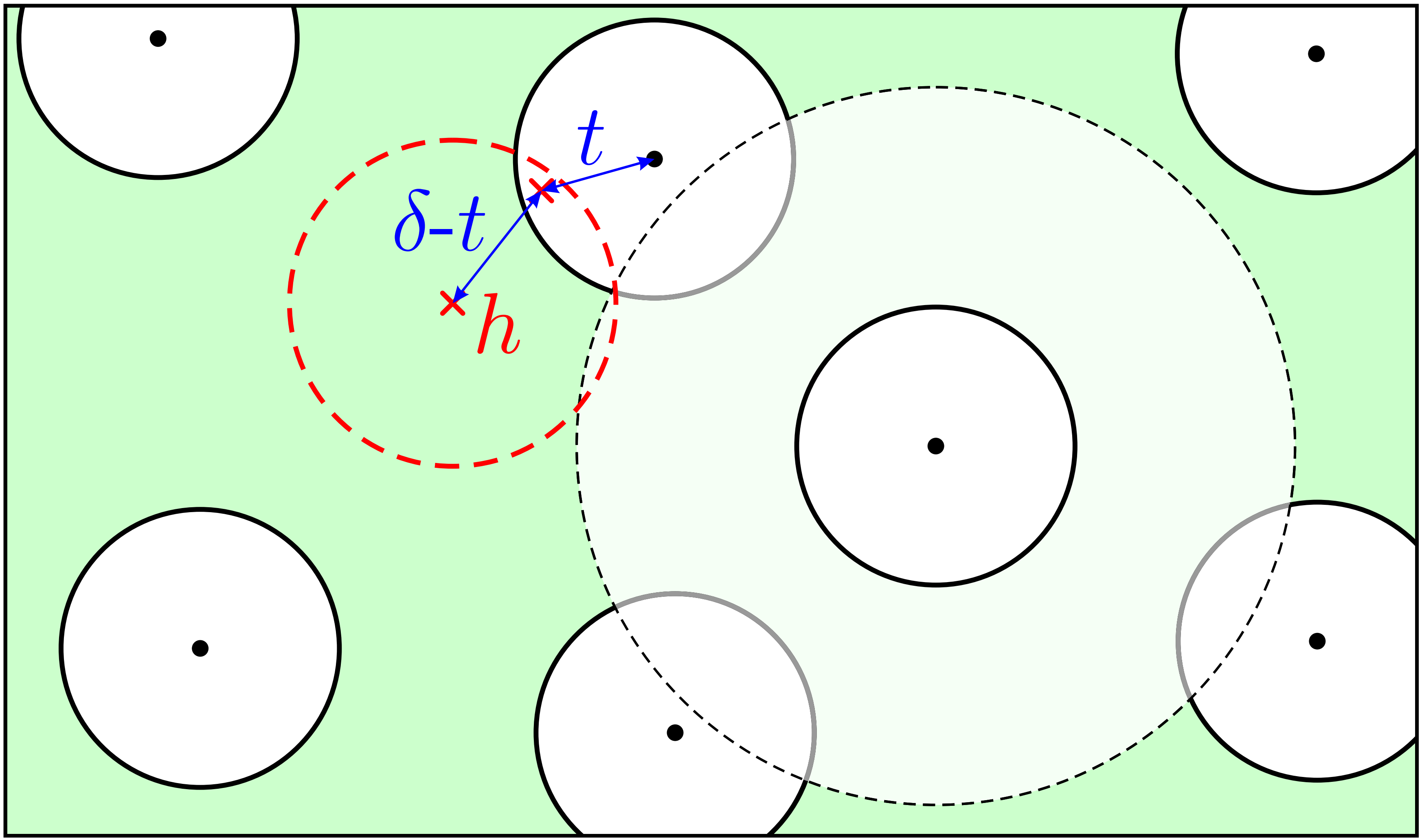
h
t



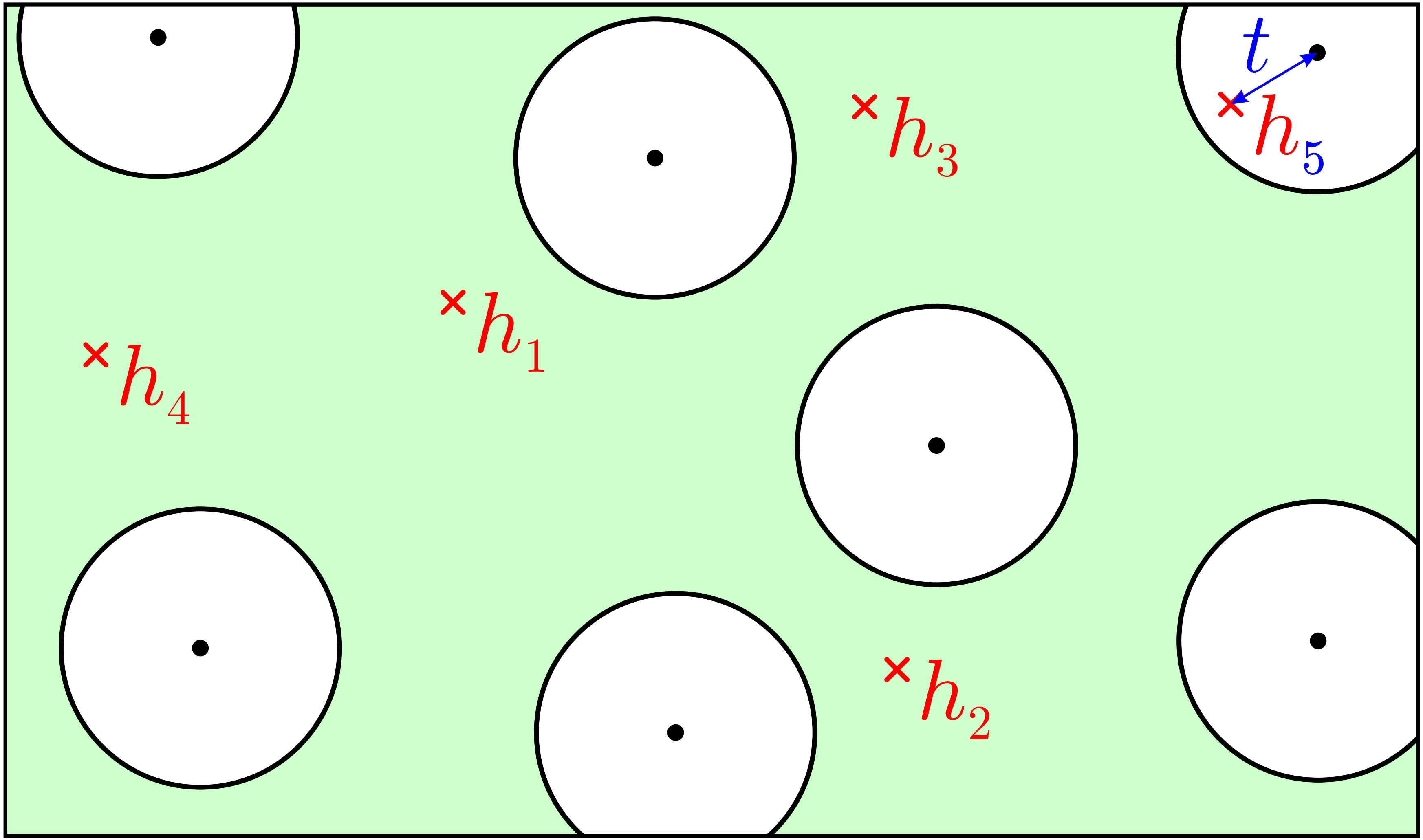
h
t
δ



h



t
δ-t
h



t
h_3
h_5
h_1
h_4
h_2

Choosing suitable parameters

- For both solutions we need about $t!$ tries.
  - choose the smallest possible t.
- We suggest the parameters $(n = 2^{16}, m = 16, t = 9)$.
  - Signing requires $9! = 362880$ decodings.
  - This takes about 10 seconds on a Pentium 4 at 2Ghz.
  - On FPGA it takes a fraction of second.
  - Verification is very fast: hash $+ 9 \times 144$ bit XORs.
- In both cases signatures are about 150 bit long.

Reducing the signature length

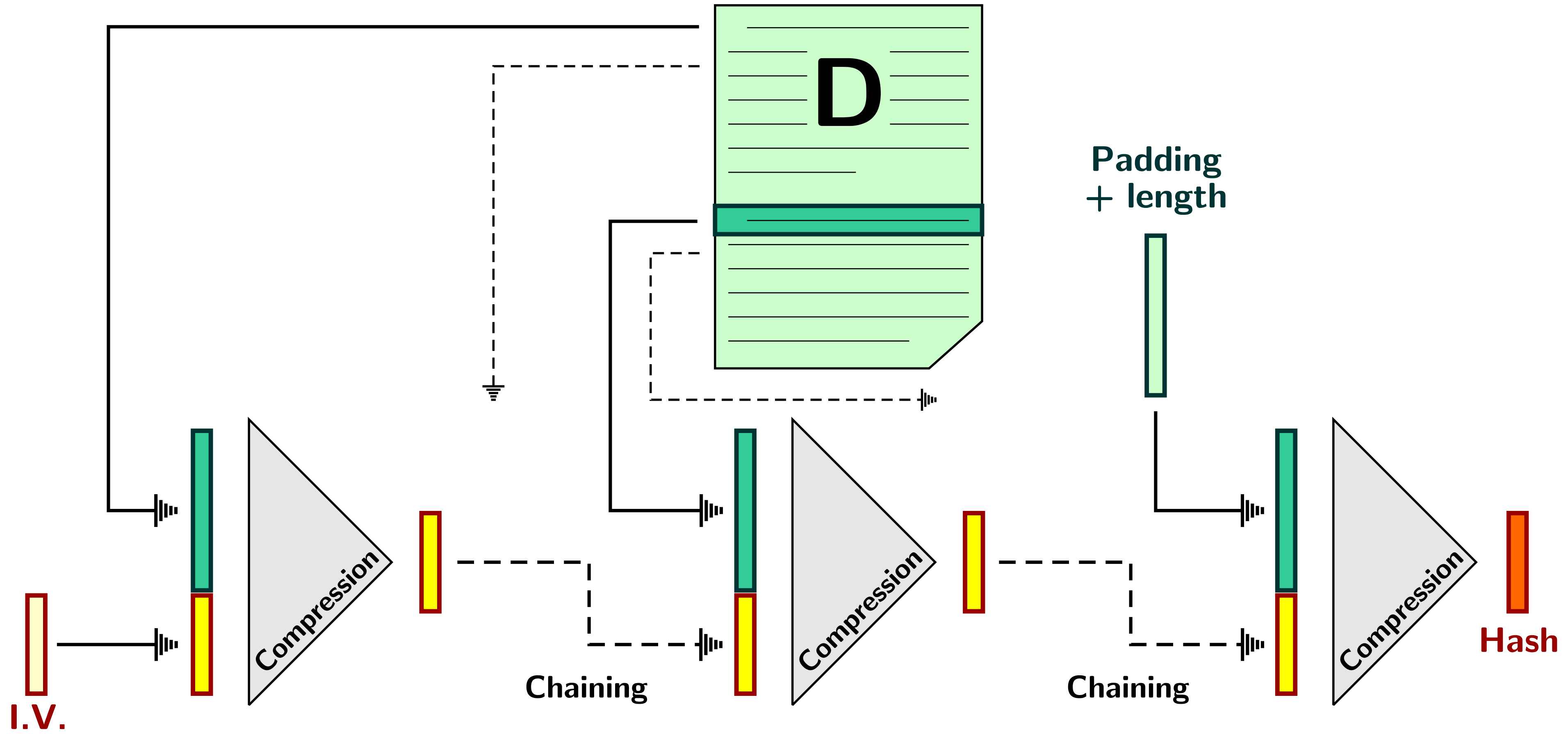
- One can shorten a signature by omitting a few bits:
  - the verifier has to test all possible values.
  - Omitting ℓ bits will require 2^ℓ verifications.
  - This doesn't affect the security of the signature!
- In our case the signature is a word of weight t:
  - we can omit some positions.
  - Verification can be done more efficiently than exhaustive search.
- Multiplying the verification time by 2^{27} only (about 30 seconds), we obtain signatures of 81 bits in average.

A provably secure hash function
[Augot, Finiasz, Sendrier ??]

- Hash functions are designed to be the fastest possible:
  - it is impossible to perform complex operations.
  - it is hard to evaluate their security.
- Some provably secure hash functions exist:
  - they use public key encryption techniques,
  - they are very slow.
- We wanted to build a fast provably secure function using Niederreiter like techniques.

Generic hash function construction

[Damgård, Merkle 1989]



Security of this construction

A hash function is secure if these problems are hard:

- ◇ inversion: given h, find X such that $Hash(X) = h$.
- ◇ second pre-image:
  given Y, find X such that $Hash(X) = Hash(Y)$.
- ◇ collision:
  find X and Y such that $Hash(X) = Hash(Y)$.

▶ Security of the compression function suffices to prove the security of the whole chain.

The compression function

We take a random parity check matrix $\mathcal{H}$ of size $r \times n$.

- The input is a word of low weight w.
- The output is its syndrome by $\mathcal{H}$ of length r.

⚠ We need $r < \log_2 \binom{n}{w}$ to compress.

- Security:
  - Inversion: syndrome decoding.
  - Collision: find a code word of weight $\leq 2w$.

Implementation and parameter choice

- We use regular words for constant weight encoding.
  - Very fast, but less input bits (more rounds to do).
  - Attacking is still a NP-complete problem.
  - Wagner's generalized birthday paradox can be used to find collisions.
- Security of 2^{80} against collision can be obtained with $(n = 21760, r = 400, w = 85)$.
  - The matrix is of 8.3Mbits.
  - Throughput is around 70Mbits/s in software.

Part V

Other applications where codes can be useful...

MDS matrices for optimal diffusion

- Block ciphers are usually built as a cascade of diffusion and confusion layers.
  - Confusion consists in applying small S-boxes in parallel.
  - Diffusions mixes the S-box outputs together.

- Diffusion doesn't have to add confusion, so a basic linear transformation can be enough.

MDS matrices for optimal diffusion

Using linear diffusion

Say the input of the diffusion layer is $I \in (\mathbb{F}_{2^m})^p$ (the output of p S-boxes on m bits) and its output $O \in (\mathbb{F}_{2^m})^q$.

The diffusion layer can be a $p \times q$ matrix $\mathcal{G}$ in $\mathbb{F}_{2^m}$ with:

$$O = I \times \left[\; \mathcal{G} \; \right].$$

- Diffusion is good if small variations on I yield large variations on O.
  - The different concatenated $(I||O)$ have to be distant from each other.

MDS matrices for optimal diffusion

- We build the following generator matrix:

$$\mathcal{G}' = \begin{array}{c|c} \overbrace{}^{p} & \overbrace{}^{q} \\ \begin{matrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{matrix} & \mathcal{G} \end{array}$$

- Then: $I||O = I \times \mathcal{G}'$.
  - Diffusion will be best when the code defined by $\mathcal{G}'$ has a large minimal distance d.
  - If $\mathcal{G}'$ is MDS ($d = q + 1$), diffusion is optimal.
- Ciphers like FOX or AES use square diffusion matrices $\mathcal{G}$ taken from MDS matrices $\mathcal{G}'$.

MDS matrices for optimal diffusion

Limitations of this technique

- Depending on the parameters it is not always possible to build a MDS matrix:
  - if $n = p + q > 2^m$ such code certainly doesn't exist.
- Diffusion among blocks is good, but not at the bit level:
  - there are $m(p+q)$ input/output bits and the minimal bit distance is also $q + 1$.
- For diffusion among 4 or 8 blocks of 8 bits like in AES and FOX, these are perfect.

MDS matrices for optimal diffusion

For an optimal 4×4 matrix on $\mathbb{F}_{2^8}$ one needs a $[8, 4, 5]$ code.

- It is possible to build a $[16, 8, 9]$ code on $\mathbb{F}_{2^4}$.
  - This yields an optimal 8×8 matrix on $\mathbb{F}_{2^4}$.

This matrix will be as efficient for block level diffusion, but will be better for sub-blocks (of size 4) diffusion.

- It is not used because it is much slower...

Threshold Secret Sharing

We want to share a secret among S users in such a way that any coalition of T users can recover it, but no coalition of $T-1$ can get any information about it.

- We build an MDS code of length $n = S + 1$ and dimension $k = T$ on $\mathbb{F}_q$ and make it public.
- We choose a secret $x_1 \in \mathbb{F}_q$ and build a code word $\boldsymbol{x} = (x_1, ..., x_n)$ from random $x_2, ..., x_k$.
- Each user gets a share x_i for $i \in [2..n]$.

- A coalition of $T = k$ users knows k coordinates of $\boldsymbol{x}$: this is an information set.
  - They can recover the whole code word, including x_1.
- A coalition of $T - 1 = k - 1$ users only know $k - 1$ coordinates of $\boldsymbol{x}$.
  - Whatever the value of x_1 there exists a code word interpolating with x_1 and their coordinates.
  - They don't get any information at all.

Other examples

- Threshold problems:
  - Digital fingerprinting.
  - Traitor tracing.

  Requires the use of multiple codes.
- Building resilient boolean functions.
- Cryptanalysis:
  - Stream ciphers: finding low weight multiples of a polynomial.
  - Block ciphers: finding biased combinations for linear cryptanalysis.

Part VI

Conclusion

Conclusion

- Error correcting codes are used in many domains of cryptography: design as well as cryptanalysis.
- Some cryptographic schemes rely on codes:
  - very fast for public key constructions,
  - they usually use a lot of memory.
- Codes might be a solution for some devices with small computational power...

A few references

[1] Matthieu Finiasz. *Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef publique*. PhD thesis, INRIA - École Polytechnique, 2004. [pdf]

More difficult to read:

[2] James L. Massey. *Some Applications of Coding Theory in Cryptography*. [pdf]

[3] Designs, Codes and Cryptography, *Journal*, Springer (rather look at recent issues) [link]