Lopez-Garcia and Canto-Navarro
EURASIP Journal on Information Security (2025) 2025:30
https://doi.org/10.1186/s13635-025-00218-2

EURASIP Journal on Information Security

RESEARCH Open Access

A fast method for finding separable Goppa polynomials used in post-quantum McEliece-based cryptography

Mariano Lopez-Garcia[1*†] and Enrique Canto-Navarro[2†]

Abstract

This paper introduces a simple and efficient method for generating Goppa polynomials used in post-quantum cryptography based on any variant of the McEliece algorithm. The approach demonstrates that such polynomials can be constructed more rapidly by multiplying several low-degree polynomials that satisfy specific properties. It is also proven that employing these polynomials does not compromise the code's error-correcting capability or overall security. The proposed method is especially advantageous when high-order Goppa polynomials are required. As a proof of concept, we present an application for user identification that combines cryptography and iris biometrics. In this system, encrypted versions of iris templates are securely stored. Using the homomorphic property of McEliece, recognition can be performed within the encrypted domain, ensuring that biometric data remains confidential throughout the entire process.

Keywords Post-quantum cryptography, McEliece, Biometrics, Iris recognition

1 Introduction

Goppa binary codes were introduced in the early 1970s as a particular family of linear codes [1]. Robert J. McEliece utilized this type of error-correcting codes as the foundation of his original public-key cryptosystem [2, 3]. In this algorithm, the plaintext is transformed into a codeword using a scrambled and permuted version of a generator matrix of size $(k \times n)$. The codeword is then masked with an error vector of length n, which randomly flips some bits to produce the final ciphertext. The decryption process involves recovering the original codeword by removing the error vector introduced during encryption. The error-correction capability of the code is determined by the degree t of the Goppa polynomial, which corresponds to the maximum number of non-zero bits (Hamming weight) that can be included in the error vector. The parameters of the McEliece cryptosystem are represented by the tuple (n, k, t).

An interesting variation of this algorithm was introduced by Niederreiter, who used Reed-Solomon codes (GRS) [4]. This variant offers faster software and hardware implementations, along with a shorter key size. However, in the early 1990s, Sidelnikov and Shestakov demonstrated that Niederreiter's proposal was insecure when using GRS codes [5]. Other studies proposed replacing Goppa codes with algebraic-geometric, Reed-Muller, Gabidulin, or even low-density parity-check codes [6–9]. Nevertheless, all these variants proved to be either inefficient or less secure than the original McEliece proposal [10–13].

†Mariano Lopez-Garcia and Enrique Canto-Navarro contributed equally to this work.

*Correspondence:
Mariano Lopez-Garcia
mariano.lopez@upc.edu
[1] Electronic Engineering Department, Universitat Politècnica de Catalunya, Avda. Victor Balaguer 1, 08800 Vilanova i la Geltrú, Spain
[2] Electrical, Automatic and Electronic Engineering Department, Universitat Rovira i Virgili, Avda. Països Catalans 26, 43007 Tarragona, Spain

Unlike other cryptographic algorithms, McEliece has demonstrated its security over time and remains unbroken when certain parameters are properly chosen. A summary of the main non-quantum attacks proposed by various authors over the past 40 years can be found in [14]. Considering some key-size optimizations, it can be noted that, after all these efforts, a security level of 2^b can be achieved with a key size of approximately $(c_0 + o(1))b^2(\lg b)^2$ bits.

Quantum computers have become a significant threat to classical cryptography. Aware of this issue, in 2016, the NIST (National Institute of Standards and Technology) issued a call to the international community to submit proposals for a new quantum-resistant public-key cryptographic standard [15]. In the first round, three proposals used McEliece, or its dual variant Niederreiter, as the basis for developing post-quantum algorithms: Classic McEliece [14], LEDACrypt [16], and NIST-KEM [17] (NTS-KEM and Classic McEliece merged after the second round). This convergence is not surprising, since McEliece remains secure even against quantum attacks based on Grover's algorithm [18, 19]. This algorithm reduces the attack complexity to roughly its square root, meaning that executing such an attack on a hypothetical quantum computer can achieve a security level of 2^b if the key size is increased to approximately $(4c_0 + o(1))b^2(\lg b)^2$ bits. The NIST call for proposals defines several security strength categories [15]. For example, in Classic McEliece, parameters like $n = 8192$, $k = 6528$, and $t = 128$ are chosen to provide maximum security (category 5); for the same category, NTS-KEM proposes parameters such as $n = 8192$, $k = 6424$, and $t = 136$. Both algorithms utilize binary Goppa codes, with the degree-t Goppa polynomial being a key part of the secret key. Codes based on McEliece have been subjected to nearly all known attacks, demonstrating their robustness and consistency to date. Therefore, selecting an appropriate Goppa polynomial is one of the most critical steps in the key-generation process.

Post-quantum cryptographic algorithms were primarily designed as general-purpose key-encapsulation mechanisms (KEM), intended for securely exchanging or distributing cryptographic keys over insecure communication channels. However, McEliece and its dual variant Niederreiter have found applications beyond KEM. For example, Arjona et al. proposed a post-quantum biometric authentication system that uses Classic McEliece to encrypt facial recognition features [20]. Additionally, Canto-Navarro et al. presented an FPGA implementation of a very fast version of the original McEliece algorithm, utilizing parameters such as $n = 32768$, $k = 16388$, and $t = 1092$ [21]. As demonstrated in these publications, the choice of cryptosystem parameters is influenced not only by the desired security level but also by other factors, such as the length of biometric features or their encoding.

This paper presents a method for finding Goppa polynomials based on the multiplication of N low-degree irreducible polynomials. The main contribution is that this method reduces the computational cost by a factor of N^2 compared to the previous proposal made in the Classic McEliece post-quantum algorithm. Additionally, when compared to NTS-KEM, a significant improvement is achieved with a reduction factor that depends on the value of N, m, and t. This is especially useful in applications where the number of errors to be corrected (the value of t) is high. The paper also analyzes the security of the proposed method and demonstrates that, for the commonly chosen values of (n, k, t), the security level is not compromised.

This paper is organized as follows. Section 2 reviews the previous methods used to find Goppa polynomials. Sections 3 and 4 introduce the foundation of our proposed approach and a discussion about its security, advantages, and disadvantages. In Sec. 5, we present an application based on iris recognition that incorporates the McEliece cryptosystem as part of an identity verification process. Finally, Sect. 6 reports and discusses the experimental results.

2 Previous works and problem statement

2.1 Basic definitions

This section introduces the fundamental concepts and basic theory related to Goppa binary codes, as well as the methods proposed in the literature for finding Goppa polynomials.

Definition 1 A Goppa polynomial $G(x)$ is defined as a degree-t polynomial over the extension field $\mathbb{F}_{p^m}$ (*or* $GF(p^m)$), being p prime, and a set of points (so-called support) $L = \{\alpha_1, ..., \alpha_n\} \subseteq \mathbb{F}_{p^m}$ such that $G(\alpha_i) \neq 0$ for all $1 \leq i \leq n$. As we are interested in binary codes then $p = 2$. Hereafter, we denote the Goppa polynomial $G(x)$ as:

$$G(x) = g_0 + g_1 x + g_2 x^2 + ... + g_{t-1} x^{t-1} + x^t = \sum_{i=0}^{t} g_i x^i \tag{1}$$

with $g_i \in \mathbb{F}_{2^m}$ and $g_t = 1$ to ensure that the polynomial is monic.

Definition 2 For every vector $c = \{c_1, ..., c_n\} \in \mathbb{F}_2^n$ the syndrome polynomial of c is defined as:

$$S_c(x) = \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} \tag{2}$$

Definition 3 A binary Goppa code $\Gamma(L, G(x))$, defined by a Goppa polynomial $G(x)$ and its support L, is the set of all vectors c such that:

$$\mathbb{C} = \{(c_1, ..., c_n) \in \mathbb{F}_2^n \ : \ S_c(x) = 0 \ (mod \ G(x))\} \tag{3}$$

This code has a dimension $k = n - mt$, being $n = 2^m$, and a minimal distance $d = 2t + 1$. This means it can correct up to $t = \lfloor \frac{d-1}{2} \rfloor$ errors.

In the Classic McEliece cryptosystem, and in some hardware-software implementations of the original scheme, the Goppa polynomial $G(x)$ is chosen to be irreducible over the field $GF(2^m)$ [14, 21, 22]. However, this constraint is not strictly necessary, since in general it is enough if $G(x)$ is separable without linear factors [17, 23]. In such case, we say that $\Gamma(L, G(x))$ is a binary *separable* Goppa code.

Definition 4 A polynomial $G(x)$ over a finite field $\mathbb{E}$ is separable if all its roots are distinct in an algebraic closure of $\mathbb{E}$.

1 Theorem 1

Let $\Gamma(L, G(x))$ be a binary Goppa code in which the degree of $G(x)$ is t. Thus, if $G(x)$ is a separable polynomial without linear factors then the minimum distance d of such a code is $2t + 1$.

1 *Proof*

The proof of this theorem can be found in [23].

Since irreducible polynomials are a subset of the set of separable polynomials without linear factors, all algorithms that use $G(x)$ as irreducible are perfectly valid. In fact, in the original McEliece, as well as in Classic McEliece and almost all proposals made in the past, this was the most common choice.

2.2 Finding Goppa polynomials

There are several proposals in the literature for finding monic Goppa polynomials, depending on whether the goal is to find an irreducible or a separable polynomial without linear factors. This section provides a summary of these methods and describes the algorithms included in the proposed approaches.

2.2.1 *Random irreducible polynomial*

Basically, this method involves creating a polynomial $G(x)$ over $GF(2^m)$ with coefficients chosen at random. Afterwards, the irreducibility of such a polynomial is checked. If the test fails, a new polynomial with different random coefficients is generated, and the process is repeated until a successful result is obtained. This method is employed by cryptographic software packages available in [24, 25], and it has also been used in several papers proposing FPGA implementations of McEliece [21, 22, 26]. The algorithm used to test the irreducibility of $G(x)$ is based on the IEEE Standard Specification for Public-Key Cryptography (see Algorithm 1) [27].

Algorithm 1 Check if polynomial G(x) is irreducible

Require: Input: polynomial $G(x)$ with coefficients in $GF(2^m)$
Ensure: Output: true if $G(x)$ is irreducible; otherwise return false
1: $d \Leftarrow$ degree of $G(x)$
2: $u(x) \Leftarrow x$
3: **for** $i = 1$ to $\lfloor d/2 \rfloor$ **do**
4: **for** $j = 1$ to m **do**
5: $u(x) \Leftarrow u(x)^2 \bmod G(x)$
6: **end for**
7: $f(x) \Leftarrow GCD(u(x) + x, G(x))$ (GDC ≡ Greatest Common Divisor)
8: **if** $f(x) \neq 1$ **then**
9: **return** False
10: **end if**
11: **end for**
12: **return** True

As mentioned in [28, 29], the probability δ of randomly selecting a monic irreducible polynomial of degree t is roughly $\delta = 1/t$. This implies that, on average, about t trials are needed to find a suitable polynomial (see (5) in Sect. 4.1.1). However, as Algorithm 1 shows, this test involves complex operations, including the computation of the greatest common divisor, which results in a high computational cost. In fact, for large values of t, this method should be avoided if feasible execution time is required.

2.2.2 *Minimal irreducible polynomial*

This method is described in detail in [30] and is the algorithm used in Classic McEliece [14]. Essentially, it is based on a deterministic approach that allows for finding a monic irreducible polynomial of degree t with an extremely high probability of success and at a relatively low computational cost. Additionally, the paper presents

a very fast FPGA implementation of the complete key-generation algorithm. An extended FPGA implementation of the entire Classic McEliece cryptosystem can also be found in [31–33].

Definition 5 Let $\mathbb{F}$ be a finite field and let $\mathbb{E}|\mathbb{F}$ be a field extension. Then, $\alpha \in \mathbb{E}$ is algebraic on $\mathbb{F}$ if and only if it exists a polynomial $g(x) \in \mathbb{F}[x]$ such that $g(\alpha) = 0$.

Definition 6 Let $\mathbb{F}$ be a finite field, $\mathbb{E}|\mathbb{F}$ a field extension and $\alpha \in \mathbb{E}$ an algebraic element on $\mathbb{F}$. Then, the minimal polynomial of α is the polynomial of lowest degree $g(x)$, with coefficients in $\mathbb{F}$, such that $g(\alpha) = 0$. The minimal polynomial is irreducible and any other polynomial that has α as root is multiple of $g(x)$.

Let $H(x) = \sum_{i=0}^{t} h_i x^i$ be a monic degree-t irreducible polynomial. By Definition 6, the minimal polynomial $G(x)$ of a random element $\beta(x) = \sum_{i=0}^{t-1} \beta_i x^i$ (note that β has degree $(t-1)$) over an extension field $GF(2^m)[x]/H(x)$, is defined as the monic non-zero polynomial of lowest degree having coefficients in $GF(2^m)$ such that $G(\beta(x)) = 0$. The minimal polynomial is always of degree t and irreducible if it exists [30]. The method consists of three steps:

1. Find t random values $\beta_i \in GF(2^m)$ and create the random element $\beta(x) = \sum_{i=0}^{t-1} \beta_i x^i$.
2. Since $G(\beta(x)) = 0$, a system of t linear equations and t unknowns (the coefficients g_i, $0 \leq i \leq (t-1)$) can be created.
3. Solve the system by Gaussian elimination and find the t coefficients (unknowns) g_i, $0 \leq i \leq (t-1)$ of $G(x)$.

The advantage of this method is that it uses very simple operations based on linear algebra and it provides a random monic degree-t irreducible polynomial with a probability close to 1 [34].

2.2.3 Separable Goppa polynomial

In [17], instead of requiring irreducibility as a necessary condition, the authors proposed using of monic separable polynomials (i.e., square-free) without linear factors. Therefore, to be $G(x)$ a valid Goppa separable polynomial, the following two properties must be met (see Algorithm 2):

Algorithm 2 Find a separable Goppa polynomial G(*x*)

Require: Input: field $GF(2^m)$, degree t of the polynomial $G(x)$.
Ensure: Output: $G(x)$
START:
for $i = 0$ to $t-1$ **do**
  $G(i) = ((short)random())$ & ($\underbrace{00..00}_{16\text{-m}}\underbrace{11..11}_{\text{m}}$) $(G(i) = g_i)$
end for
$G(t) = 1$ $(G(t) = g_t)$
$\bar{F} = additive\_FFT(G(x), GF(2^m))$ being $\bar{F} = \{G(\alpha_1), G(\alpha_2), ..., G(\alpha_n)\}$
for $i = 0$ to $n-1$ **do**
  if $F(i) = 0$ **then**
    goto START;
  end if
end for
SetZero(D(x))
for $i = 0$ to $\lceil t/2 \rceil - 1$ **do**
  $D(2i) = G(2i+1)$
end for
$f(x) \Leftarrow GCD(G(x), D(x))$
if $deg(f(x)) < 1$ **then**
  return $G(x)$
else
  goto START
end if

1. $G(x)$ has no roots in the field $GF(q)$ $(q = 2^m)$, i.e., for all the elements of its support vector $L = \{\alpha_1, ..., \alpha_n\} \subseteq \mathbb{F}_{2^m}$ it must be satisfied that $G(\alpha_i) \neq 0$. An efficient and fast method to evaluate this condition is by means of the additive FFT algorithm [30, 35, 36]. This algorithm evaluates a polynomial at all elements in the field $GF(2^m)$ with an asymptotic time complexity of $\mathcal{O}(2^m log_2(t))$.
2. $G(x)$ has no repeated roots. Then, it holds that such polynomial and its derivative are relative prime, so that the greatest common divisor between them is 1, i.e., $GCD(G(x), \frac{d}{dx}G(x)) = 1$. An interesting advantage is that the derivative of $G(x)$ only contains even powers. As shown in [17], this property is because the field has characteristic 2, so that the derivative can be directly estimated by simply applying $D(x) = \frac{d}{dx}G(x) = \sum_{i=0}^{\lceil t/2 \rceil - 1} g_{2i+1} x^{2i}$.

In [34] it is shown that, when t is not very small, the chance of non-divisibility of a random monic degree-t polynomial can be roughly estimated by $\xi = (1 - \frac{1}{q})^q (1 - \frac{1}{q^4})^{(q^2-q)/2} (1 - \frac{1}{q^6})^{(q^3-q)/3} \ldots$. As $\lim_{q\to\infty} (1 - \frac{1}{q})^q = 1/e$ and $\lim_{q\to\infty} (1 - \frac{1}{q^{2k}})^{(q^k-q)/k} = 1$, then, if q is enough large, the expression of ξ is dominated by the first factor, so that it can be approximated by $1/e$. Some interesting conclusions, corroborated experimentally, can be drawn from this result:

1. The first term of ξ, i.e., $(1-\frac{1}{q})^q \approx e^{-1} = 0.367879$, corresponds to the probability that there are no linear factors dividing a uniform-random monic degree-t polynomial. Note that this property is tested by means of the additive FFT function (see Algorithm 2), which checks if $G(\alpha_i) \neq 0$. Thus, in average, the additive FFT should be executed 2.7183 times before passing to the second step in which the GCD is used.
2. As the rest of the terms of ξ tend to 1, it means that the GCD algorithm should be executed only once with a probability extremely close to 1.

Therefore, as the additive FFT can be run very quickly, the whole algorithm is quite efficient offering a relative good performance in terms of execution time.

3 Separable polynomials implemented by the product of irreducible polynomials

3.1 Theoretical foundations

We propose a new algorithm for obtaining separable polynomials without linear factors merging the two methods presented in Sects. 2.2.2 and 2.2.3.

1 Theorem 2

Let β be an algebraic element over a field $\mathbb{F}$, and let $g_i(x), i = 1, 2, \ldots, M$ be a set of M monic irreducible polynomials that have β as common root. Then, $g_1(x) = g_2(x) = \ldots = g_M(x)$

1 *Proof*

Let $h(x) \in \mathbb{F}[x]$ be a monic irreducible polynomial. Let's assume h(x) as the minimal polynomial of β. Then, by Definition of minimal polynomial stated in Definition 6, h(x) divides any polynomial in $\mathbb{F}[x]$ that has β as root. In particular, h(x) divides all polynomials $g_i(x)$, $1 \leq i \leq M$. As all $g_i(x)$ are irreducible, then $h(x) = g_1(x) = g_2(x) = \ldots = g_M(x)$. □

Definition 7 Let $\mathbb{F}_{2^m}$ be a finite field. If $g_i(x)$ is an irreducible polynomial over $\mathbb{F}_{2^m}$ then it is separable.

1 Theorem 3

Let $g_i(x)$ be a set of N different minimal polynomials over a field $\mathbb{F}_{2^m}$, each one related to a different algebraic element β_i, $i = 1, 2, \ldots, N$. Then, the polynomial $G(x) = \prod_{i=1}^{N} g_i(x)$ is a monic separable polynomial without linear factors.

1 *Proof*

(⇒) By Definition 6, $g_i(x)$ are irreducible monic polynomials and uniques for each β_i. As stated in Definition 7, these polynomials are separable, since $g_i(x) \in \mathbb{F}_{2^m}[x]$, being $\mathbb{F}_{2^m}$ a finite field. Then, the polynomials $g_i(x)$ are irreducible, separable and different.

(⇐) Let's assume the opposite assumption, that is $G(x) = \prod_{i=1}^{N} g_i(x)$ is not a separable polynomial. Then, it means that $G(x)$ has repeated roots, so that the individual polynomials $g_i(x)$ must have common roots in the algebraic closure of $\mathbb{F}_{2^m}$. By Theorem 3, there exist polynomials such that $g_k = g_j$, $k \neq j$, which is a contradiction since all the polynomials $g_i(x)$ are different. □

Note that, as all polynomials $g_i(x)$ described in Theorem 3 are irreducible, then $G(x)$ become a monic polynomial without linear factors nor repeated roots (i.e., separable).

3.2 Proposed method

The method proposed in this paper is based on finding the Goppa polynomial $G(x)$ by using Theorem 3. Note that, each of the N irreducible polynomials $g_i(x)$ has a degree-t_i, $1 \leq i \leq N$, being $\sum_{i=1}^{N} t_i = t$, where it is possible that $t_k \neq t_j$, $k \neq j$. Although it is not strictly necessary, in order to make the method simpler we take $t_i = \frac{t}{N}$, being t an integer multiple of N. The steps to follow are the following:

1. Generate (t/N) random values $\beta_p \in GF(2^m)$, $0 \leq p \leq (\frac{t}{N} - 1)$ and create the first algebraic element $\beta_1(x) = \sum_{p=0}^{\frac{t}{N}-1} \beta_p x^p$.
2. Repeat the first step $(N-1)$ times, in order to generate the remaining set of algebraic elements $\beta_i(x) = \sum_{p=0}^{\frac{t}{N}-1} \beta_p x^p$, $2 \leq i \leq N$. For each of the $(N-1)$ iterations, new random values $\beta_p \in GF(2^m)$ are generated.
3. For each iteration i, being $i > 1$, check that $\beta_j(x) \neq \beta_i(x)$, $\forall\, j \mid 1 \leq j < i$. If not, find a new algebraic element $\beta_i(x)$ until this condition is met.
4. For each algebraic element $\beta_i(x)$, apply the algorithm described in Sect. 2.2.2 and find the set of N monic degree-(t/N) irreducible polynomials $g_i(x)$, $1 \leq i \leq N$.
5. Find the Goppa separable polynomial without linear factors as the product of all $g_i(x)$, i.e., $G(x) = \prod_{i=1}^{N} g_i(x)$.

Figure 1 shows a flow diagram presenting all the steps involved to generate the t-degree separable Goppa polynomial without linear factors.

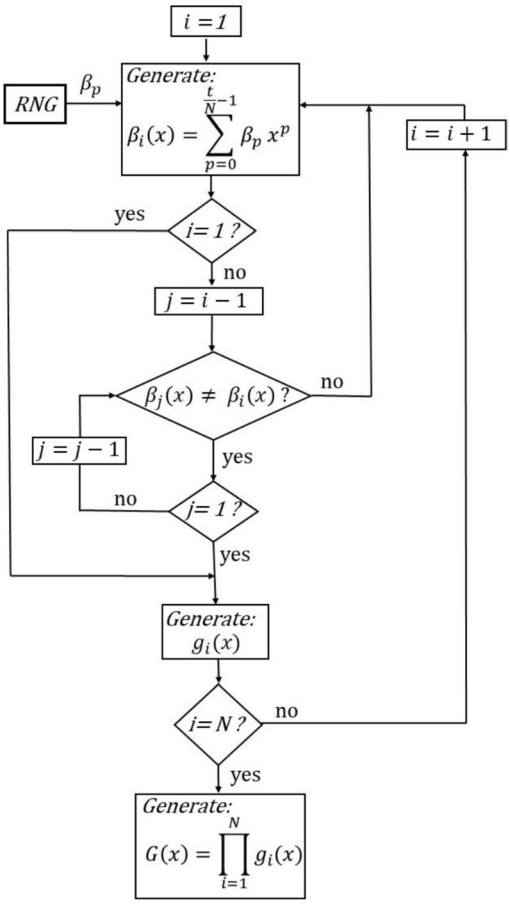

Fig. 1 Flow diagram of the proposed method

3.3 Performance of the proposed method

The main advantage of our proposal is that the execution time required to find the Goppa polynomial is significantly reduced compared to the three methods described in Sect. 2.2. Although the results depend on the values of n and t, they are generally quite similar across methods, except for the approach based on finding random irreducible polynomials, which tends to be slower.

For a degree-t polynomial, the t linear equations arising from the condition $G(\beta(x)) = 0$ are obtained by performing $(t-1)$ multiplications between two polynomials over the extension field $GF(2^m)/h(x)$, where $h(x)$ is typically chosen as a monic degree-t irreducible polynomial. Usually, and when possible, $h(x)$ is selected as a binary trinomial or pentanomial to reduce computational complexity. A comprehensive list of such binary polynomials can be found in [37].

Furthermore, since multiplying two degree-$(t-1)$ polynomials has a computation complexity $\mathcal{O}(t-1)^2$, it follows that the total execution time for these multiplications is proportional to $\propto (t-1)^3$ (here $\propto$ means "proportional to"). In our proposed method, these multiplications are performed on the extension field $GF(2^m)/p_i(x)$, where $p_i(x)$ is a degree-$(\frac{t}{N})$ irreducible polynomial (again, preferably chosen as a binary trinomial or pentanomial). Since this process is repeated N times, once for each $g_i(x)$, then the total execution time scales proportionally to $\propto (N(\frac{t}{N}-1)^3) = (t-N)^3/N^2$.

Additionally, solving a linear system of t equations via Gaussian elimination takes time proportional to $\propto t^3$. It is simple to prove that N linear systems, each one consisting in t/N equations, can be resolved in an execution time $\propto N(t/N)^3 = t^3/N^2$.

Finally, we must account for the extra time needed to multiply all the polynomials $g_i(x)$. To optimize this process, polynomial multiplications are ordered from lowest to highest degree. If N is a power of two and t is not very small, then the total execution time is approximately proportional to $\propto \frac{t^2}{N}\sum_{i=1}^{\log_2 N} 2^{i-2}$ if $N \geq 2$.

Figure 2 presents a simulation for different values of N when $t = 512$. As observed, for $N \leq 4$ —the typical case— the total execution time is primarily dominated by the Gaussian elimination algorithm. Since this algorithm's complexity is approximately proportional to t^3/N^2, then the proposed method could theoretically reduce the execution time by a factor of about N^2 compared to the method described in Sect. 2.2.2.

4 Discussion about the proposed method

4.1 Security analysis

4.1.1 Irreducible versus separable polynomials

This section analyzes the security based on the number of possible Goppa polynomials that can be obtained using any of the methods proposed in Sects. 2 and 3.

The number $N_{qi}(t)$ of monic irreducible polynomials of degree t over a finite field $GF(q)$ is given by Gauss's formula [38]:

$$N_{qi}(t) = \frac{1}{t}\sum_{d|t} q^d \mu\left(\frac{t}{d}\right) = \frac{1}{t}\sum_{d|t} \mu(d) q^{(t/d)} \tag{4}$$

where $q = 2^m$ and $\mu : \mathbb{N} \to \{-1, 0, 1\}$ is the Möbius function defined by $\mu(d) = (-1)^r$ if d is a product of r distinct primes, and $\mu(d) = 0$ otherwise ($\mu(1) = 1$). Here, d runs over the set of all positive divisors of t, including 1 and t. Without loss of generality, this formula is usually approximated by :

$$N_{qi}(t) \approx \frac{q^t}{t} = \frac{2^{mt}}{t} \tag{5}$$

Note that, since there are 2^{mt} monic polynomials of degree t, the probability of randomly finding an monic irreducible polynomial is $1/t$.

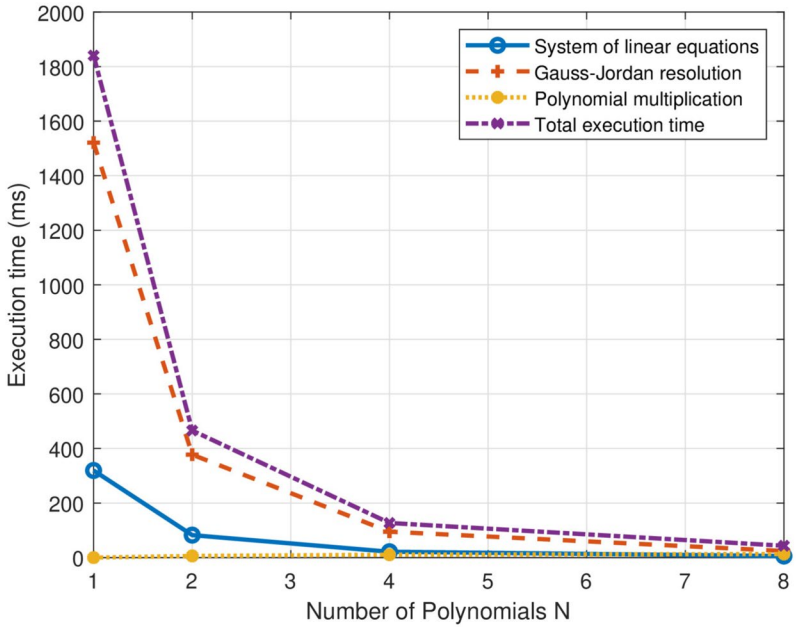

Fig. 2 Execution time when finding a monic separable Goppa polynomial using different values of N (number of polynomials) and $t = 512$

As demonstrated in [34], when t is not very small, the use of monic squarefree polynomials without linear factors increases the security level by $\log_2(\frac{t}{e}) \approx \log_2(t) - 1.44$ bits. Therefore, defining $N_{qs}(t)$ as the number of monic degree-t squarefree polynomials without linear factors, the following expression can be deduced using (5):

$$N_{qs}(t) \approx \frac{N_{qi}(t) \cdot t}{e} = \frac{q^t}{e} = \frac{2^{mt}}{e} \tag{6}$$

being e the Euler's constant. Note that $N_{qs}(t) > N_{qi}(t)$, so there are more squarefree polynomials without linear factors than irreducible polynomials given that (assuming $t > 2$):

$$\frac{N_{qs}(t)}{N_{qi}(t)} \approx \frac{t}{e} > 1 \tag{7}$$

If, as described in Section 3.2, our proposal is simplified by imposing that all polynomials $g_i(x)$ have the same degree, i.e., $t_i = \frac{t}{N}$, and assuming that the random values β_p used to generate each $g_i(x)$ are independent of one another, then the number of distinct monic squarefree polynomials $N_{qsr}(t)$ of degree t without linear factors that can be obtained using our approach is given by:

$$\begin{aligned} N_{qsr}(t) &= \prod_{i=0}^{N-1} \left(\frac{2^{\frac{mt}{N}}}{\frac{t}{N}} - i \right) = \left(\frac{2^{\frac{mt}{N}}}{\frac{t}{N}} \right) \left(\frac{2^{\frac{mt}{N}}}{\frac{t}{N}} - 1 \right) \cdots \left(\frac{2^{\frac{mt}{N}}}{\frac{t}{N}} - N + 1 \right) \\ &\approx 2^{mt} \left(\frac{N}{t} \right)^N \end{aligned} \tag{8}$$

Then,

$$\frac{N_{qsr}(t)}{N_{qi}(t)} \approx t \left(\frac{N}{t} \right)^N \tag{9}$$

and

$$\frac{N_{qsr}(t)}{N_{qs}(t)} \approx e \left(\frac{N}{t} \right)^N \tag{10}$$

Usually, $t > N$ and $t > e$, so $N_{qsr}(t)$ is smaller than both $N_{qi}(t)$ and $N_{qs}(t)$. Table 1 shows some results for $N_{qsr}(t)$, $N_{qs}(t)$, and $N_{qi}(t)$ using different values of m, t and N. In

Table 1 Number of irreducible and separable polynomials without linear factors for different values of N

	$m=13, t=128$			$m=13, t=256$			$m=14, t=512$		
Value of N	**2**	**4**	**8**	**2**	**4**	**8**	**2**	**4**	**8**
$N_{qsr}(t) = 2^{mt}\left(\frac{N}{t}\right)^N$	2^{1652}	2^{1644}	2^{1632}	2^{3314}	2^{3304}	2^{3288}	2^{7152}	2^{7140}	2^{7120}
$N_{qs}(t) = \frac{2^{mt}}{e}$	$2^{1662.56}$			$2^{3326.56}$			$2^{7166.56}$		
$N_{qi}(t) = \frac{2^{mt}}{t}$	2^{1657}			2^{3320}			2^{7159}		

all cases, the security against a brute-force attack aimed at finding the polynomial $G(x)$ is at least 1632 bits, which is significantly higher than the 256-bit security provided by the entire McEliece post-quantum algorithm. Therefore, requiring all polynomials $g_i(x)$ to have the same degree could be a reasonable constraint when standard McEliece parameters are used.

4.1.2 Fault injection and side-channel analysis attacks

As mentioned before, McEliece is resistant to cryptanalysis attacks aimed at finding the input plaintext from the ciphertext, given only the public key, and assuming the input message is chosen randomly. However, physical attacks target the physical implementation of cryptographic devices in order to obtain the private key or the plaintext [39, 40]. These attacks fall into two main categories: fault injection attacks (FI), which intentionally cause errors in the normal operation of the device through lasers, electromagnetic pulses, or voltage glitches, and side-channel analysis (SCA) attacks, which observe power consumption or electromagnetic emissions to extract secret information. There are a large number of publications on FI and SCA attacks applied to the McEliece cryptosystem, targeting the key generation, encapsulation (incl. encrypting), and decapsulation (incl. decrypting) stages [41, 42].

Brinkmann et al. proposed in [43] an SCA attack to recover the private key during the computation of the parity-check matrix $\mathbf{H}$ using the Gaussian elimination algorithm. The attack is performed on Classic McEliece running on an ARM microprocessor. As a countermeasure, it was proposed to increase the value of m, as well as t, to maintain overall security. However, this also increases the size of the public key, making this approach impractical. A more effective countermeasure involves randomly changing the invertible matrix $\mathbf{S}$ before computing the public key. This way, the Gaussian elimination is not applied solely to $\mathbf{H}$, but also to $\mathbf{S}\cdot\mathbf{H}$, so it is necessary that the multiplication of both matrices does not leak any relevant information.

Recent publications have focused on attacks targeting the encapsulation stage to recover the input message. In [44], a laser fault injection is used during the syndrome computation to corrupt specific instructions. The message can be recovered in a few seconds if the matrix multiplication is computed in $\mathbb{N}$ instead of $\mathbb{F}_2$. However, this attack it not successful if the encryption algorithm is optimized to make optimal use of the machine word capacity. The same authors proposed an improved version resistant to SCA attacks, which analyzes power consumption traces using machine learning techniques [45]. Additionally, they proposed a countermeasure based on masking the error vector to change its Hamming weight.

Most physical attacks focus on the decapsulation stage. Lahr et al. presents in [46] an attack that exploits the leakage information from the electromagnetic field (EM) generated in an FPGA implementation of Classic McEliece. The described attack requires, on average, fewer than 600 measurements to recover the plaintext. Similarly, Guo et al. showed in [47] both an FPGA and ARM Cortex M4 implementation of the same algorithm. The power consumption is analyzed during the computation of the additive FFT used to evaluate the error locator polynomial, and only 800 traces are necessary to recover the key. Recently, Gan et al. presented in [48] a fully protected implementation of Classic McEliece against both SCA and FI attacks. The proposal is intended to secure vulnerable operations, such as additive FFT and Gaussian elimination. This goal is achieved by including Gaussian noise generators, randomized clock modules, redundancy countermeasures, and inserting random delays in the input signals.

However, to the best of our knowledge, none of these attacks have focused on the generation of the Goppa polynomial. Note that all the methods described in Sect. 2 rely on the use of random numbers that change each time the process is executed. Therefore, extracting relevant information from the analysis of the correlation between traces and consumption models becomes difficult. However, note that $G(x)$ is defined as the product of N irreducible polynomials $g_i(x)$ of degree $\left(\frac{t}{N}\right)$, which are computed by solving a linear system of $\left(\frac{t}{N}\right)$ equations using Gaussian elimination. Since this operation has been the target of several attacks in previous publications, a future research line could focus on including some of the countermeasures cited in [48], and evaluating their effectiveness in different hardware implementations of our proposed method.

4.2 Pros and cons of the proposed method

An interesting discussion is presented in [34], regarding the pros and cons of using irreducible versus separable monic degree-t polynomials without linear factors. In general, all the advantages and disadvantages mentioned in that paper are applicable to the method proposed in this document. The primary argument for continuing to use irreducible polynomials is the high level of security they provide, supported by numerous security analyses conducted over time on the original McEliece cryptosystem. Another advantage is that finding an irreducible polynomial is straightforward, as it involves computing minimal polynomials via linear algebra. The only significant benefit cited for using separable polynomials is the larger number of available square-free polynomials without linear factors, which can yield approximately

$\log_2(t) - 1.44$ (non constrained case) bits of additional security.

Since our method relies on separable monic degree-t polynomials, it shares, in principle, the same advantages and disadvantages discussed above. However, two additional benefits can be highlighted: first, our approach is clearly faster than any of the previous methods, which can be a substantial improvement when working with large values of t. Second, our proposal is as simple as the method described in [30] for finding irreducible polynomials, since it involves repeatedly computing minimal polynomials of lower degree using basic linear algebra techniques.

Most FPGA hardware implementations use a Gaussian systemizer to generate the Goppa polynomial and the public key through the parity check matrix $\mathbf{H}$ [26, 30]. An optimal implementation of this module is essential, as it consumes a significant portion of the logical resources required by the key generation stage. Note that Gaussian systemization is applied in step 4 of our algorithm to find the coefficients g_i of each monic irreducible polynomial $g_i(x)$. The implementations proposed in [26, 30] are based on an array of processors that perform simple operations on rows and columns. The number of processors in the array depends on the size of the matrix to be systemized, i.e., it depends on the degree of the polynomial $g_i(x)$. Note that, using our proposal, this degree is $t_i = t/N$. Therefore, it is expected that the size of the Gaussian systemizer (FPGA resources) can be reduced compared to an equivalent implementation that directly computes the Goppa polynomial $G(x)$ of degree t (method proposed in Sect. 2.2.2, case $N = 1$).

However, there are certain applications where the use of our method might present limitations when the product $(m \cdot t)$ is not very high. For example, the implementation of Classic McEliece that provides lower security (category 1) uses the parameters $m = 12$, $n = 3488$, and $t = 64$ [14]. If the value of N is chosen to be 8, the irreducible polynomials $g_i(x)$ will have a degree $t_i = 8$. Note that there are only 2^{93} irreducible polynomials of that degree. Therefore, to preserve security above 256-bits, the value of N should be smaller, leading to an execution time similar to that provided by some of the methods described in Sect. 2.2.

5 Application on biometric cryptosystems

5.1 Background

One of the main objectives of biometric cryptosystems is to generate cryptographic keys of arbitrary length from biometric features. The primary challenge lies in managing the intra-class variation that naturally occurs between two samples of the same biometric trait. Since biometric data are inherently noisy, a perfect match is quite rare and unlikely. The robustness of many cryptographic functions depends on the accuracy of key reproduction, as even a single bit change can produce a completely different output. Therefore, linking cryptographic keys to noisy biometric features remains an open problem, with ongoing research proposing various improvements.

These issues have been extensively studied in numerous publications, with the architecture proposed in [49] being one of the most widely accepted schemes. The fuzzy commitment scheme combines error-correcting codes with cryptography: the former addresses the noisy nature of biometric data, while the latter ensures that the biometric information keeps the key secret. An improved version of this scheme was introduced in [50], which uses iris biometrics as an example. The iris is encoded into a 2048-bit string generated using the well-documented algorithm developed by Daugman [51–54].

Differences between two iris samples from the same eye can generally be categorized into two types: first, random bits caused by camera noise or iris distortion; second, burst errors resulting from specular reflections, undetected eyelashes, or eyelids. To address these error types, the fuzzy commitment scheme employs a two-layer error correction network [50]. Random errors are corrected using a Hadamard code, while burst errors are handled with a Reed-Solomon code. This scheme is illustrated in Fig. 3 and operates as follows:

1. The error-correcting encoding network consists in a Reed-Solomon and Hadamard codes, whereas in the error-correcting decoding the order is reversed, including a Hadamard and Reed-Solomon code.

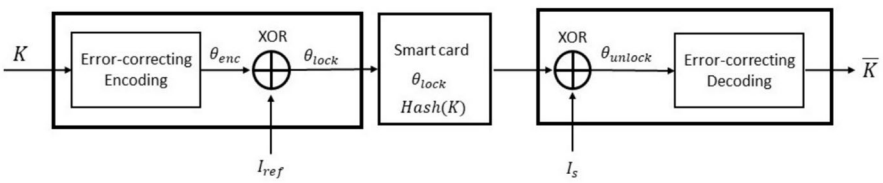

Fig. 3 Fuzzy scheme proposed in [50] for biometric key generation

2. A random key K is generated. This key is encoded with the error-correcting network that adds some redundancy, obtaining $\theta_{enc} = Encoding(K)$.
3. Then, θ_{enc} is Xored with a sample I_{ref} of the iris code of an specific user. Note that, the key is now protected and cannot be recovered without knowing the iris code.
4. The result θ_{lock}, and a *hash* of K, are stored in a smartcard.
5. During the decoding phase, the user presents a fresh iris sample I_s and provides the data stored in the smart card. This new sample is Xored with θ_{lock}, obtaining $\theta_{unlock} = \theta_{enc} \oplus I_{ref} \oplus I_s$
6. Note that, $I_{ref} \oplus I_s$ is a vector in which those bits that disagree between the two samples are set to one. If the error correcting capability of the code is well selected, then it will be able to remove these bits, providing the value of θ_{enc}.
7. Afterwards, the error-correcting mechanism is applied in reverse mode in order to obtain $\bar{K}$.
8. Finally, the $hash(\bar{K})$ is calculated and, if it matches the stored $hash(K)$, the derived key is considered correct.

Using Hadamard and Reed-Solomon codes, it is possible to correct the approximately 10–20% of bits that typically differ between iris codes from the same eye. Conversely, the disagreement rate between iris codes from different eyes is around 40–60%. However, in some cases, the statistical distributions of bits that disagree for the same eye versus different eyes overlap [53], making the selection of an appropriate error-correcting threshold (or parameter) a critical issue. In the scheme proposed in [50], a false rejection rate (FRR) of 0.47% and a false acceptance rate (FAR) of 0% are achieved with a key length of 140 bits. This key length may be insufficient to meet current security standards required by cryptographic protocols. Moreover, the performance of this scheme depends not only on the number of errors but also on how well the Hadamard and Reed-Solomon codes are designed—specifically, whether their error-correction capabilities are appropriately chosen to handle both random and burst errors.

An alternative approach was presented in [20], which uses facial recognition as the biometric feature. During user enrollment, the facial biometric is encoded using a Classic McEliece post-quantum encryption algorithm, and the encrypted feature is stored in a database. During verification, a new facial sample is captured and encrypted with the public key. The encrypted data from enrollment and verification are then XORed. The error-correction capability of Classic McEliece (determined by the degree t of the Goppa polynomial) is set so that if both facial features originate from the same person, the number of differing bits is less than or equal to t, allowing correct identification via the secret key. Otherwise, the user is considered an impostor. Since facial features are encoded into a bit-string of 348 bits, additional bits are appended to form an extended vector—by adding $(n - 348)$ bits—and a secret permutation is included to enhance security. The appended bits ensure a Hamming weight (HW) higher than t, providing protection against impersonation attempts. With these parameters, the resulting recognition rates (FAR and FRR) are quite meaningful. This scheme primarily targets user identification rather than generating biometric keys. If facial features are replaced with iris codes, however, the value of t must be significantly increased to accommodate higher error rates.

5.2 Proposed scheme

Our proposal is based on a fuzzy commitment scheme that enables user identification while also generating a cryptographic key derived from the biometric feature. This represents a significant difference compared to [50], where the biometric data is used solely to reveal the key. The goal is to demonstrate an application that requires a high value of t. Therefore, having a fast method for constructing the Goppa polynomial is highly beneficial to reduce the overall execution time when generating McEliece keys. Let the private key be denoted as $\Theta = (G(x), \alpha_1, \alpha_2, \ldots, \alpha_n)$, where $G(x)$ is the Goppa polynomial and $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ is its support vector. The parity check matrix in systematic form is represented as $\hat{H} = (I_{n-k}|\Psi)$, with the public key being Ψ. The scheme is illustrated in Fig. 4 and operates as follows:

1. The enconding and decoding subroutines are based on the post-quantum Classic McEliece algorithm with parameters $n = 16384$, $m = 14$, $k = 6864$, and $t = 680$.
2. During the enrollment phase, an iris image is captured and its 2048-bit iriscode I_{ref} is extracted. Then, I_{ref} is randomly permuted using a permutation secret vector P, creating a new error-vector $\hat{I}_{ref}$ of length 16,384 bits. This permutation simply redistributes the bits set as 1. Note that $HW(I_{ref}) = HW(\hat{I}_{ref})$. The Hash of vector $\hat{I}_{ref}$ represents the biometric key $K = Hash(\hat{I}_{ref})$. The use of P is important, because then K can be replaced if it is suspected that such a key was compromised.
3. Next, $\hat{I}_{ref}$ is encoded using the parity check matrix $\hat{H}$ (public key). The result $C_0 = \hat{H}\hat{I}_{ref}$ and the $Hash(\hat{I}_{ref})$ are stored in a database or in a smart card. Note that full privacy and protection are provided for every user, as only encrypted information about the iris code is stored.

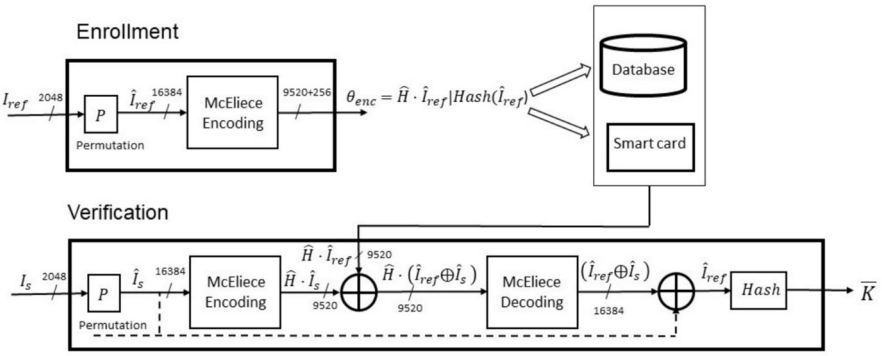

Fig. 4 Cryptobiometric system for identification and biometric key generation

4. At the verification phase, a fresh iris image is captured and its iriscode I_s is extracted. This iriscode is permuted using the vector P, generating the error-vector $\hat{I}_s$ and, using the public key Ψ, the codeword $C'_0 = \hat{H}\hat{I}_s$ is computed.
5. Now the Homomorphic property of McEliece is used. Depending on whether the result of the enrollment process was stored in a database or a smart card, C'_0 is Xored with C_0, obtaining $C = \hat{H}\hat{I}_s \oplus \hat{H}\hat{I}_{ref} = \hat{H}(\hat{I}_s \oplus \hat{I}_{ref})$.
6. In the simplest version of Daugman's algorithm, the user is accepted or rejected depending on the result of the hamming distance HD defined as:

$$HD = \frac{||I_s \oplus I_{ref}||}{2048} = \frac{||\hat{I}_s \oplus \hat{I}_{ref}||}{2048} \tag{11}$$

   HD represents the fraction of disagreeing bits between two irises [51]. An usual threshold for HD is about $HD_{Thd} = 0.332$, i.e., those iriscodes that disagree in less than 680 bits are considered authentic; otherwise are labeled as impostors. Thus, as the value of $t = 680$ was selected to match a HD of 0.332 ($t = 2048 \cdot HD_{Thd}$), after decrypting C using the private key Θ, only authentic users will be decoded correctly. Once $(\hat{I}_s \oplus \hat{I}_{ref})$ is recovered, using $\hat{I}_s$ it is easy to obtain $\hat{I}_{ref}$ and straightforwardly $\bar{K} = Hash(\hat{I}_{ref})$. Note that it is not possible to decrypt $\hat{I}_{ref}$ using the private key without knowing $\hat{I}_s$. This is because, as mentioned by Daugman in [55], the average Hamming weight of an iris code is 1024, which is higher than the correction capability t of the Goppa code. Therefore, the iris template stored in the database or on a smart card remains completely secure, even if the private key is compromised.
7. Finally, it is checked if $\bar{K} = K$, and depending on the result the user is identified as authentic (and the key $\bar{K}$ is correct) or rejected as impostor.

An interesting characteristic of the proposed scheme is its ability to generate cryptographic keys of any length. A Shake-256 hash function was selected for this purpose, but any other standard hash function could be used. An additional advantage over the scheme presented in [50] is that the positions of bits that disagree between iris codes can be distributed arbitrarily within the error vector. Consequently, the location of random bits or the length of bit sequences corresponding to burst errors does not influence the false acceptance rate (FAR) or false rejection rate (FRR).

6 Experimental results

6.1 Database

A synthetic database of iris codes was generated using the hidden Markov model (HMM) proposed in [55]. The statistical distributions of cross-comparisons between these iris codes closely match the HD scores obtained from actual iris images. The database comprises 2000 distinct 2048-bit iris codes, which were used to evaluate our proposed cryptobiometric system.

6.2 Performance

Experimental results were obtained using an Intel Core i7-8700 CPU clocked at 3.2 GHz and 16 GB of RAM. The implementation was developed by coding all algorithms in ANSI C. Additionally, all functions related to operations over the field $GF(2^m)$ were programmed to accept m, n, and t as parameters. In other words, these operations were not optimized for specific parameter values, as is typically done in publicly available software for Classic McEliece or NTS-KEM. This approach allows multiple results to be obtained by running the same software with different parameters, at the cost of increased execution time.

All benchmarks were executed and compiled on Windows 11 using MinGW-W64-builds-4.3.5, with compilation options set to -march=native -O3. Since the code does not utilize vector instructions, optimizations such as -msse2 or -mavx2 were not included. Execution

Table 2 Goppa polynomial execution time (ms), $m = 13, n = 8192$

	$t=128$			$t=256$			$t=384$		
Value of N	**1**	**2**	**4**	**1**	**2**	**4**	**1**	**2**	**4**
Proposed method	29.99	7.82	**2.30**	235.309	60.45	**16.81**	796.31	202.90	**57.02**
Classic McEliece[a]	29.99			235.31			796.31		
NTS-KEM[b]	16.53			100.59			324.82		
Random search	11,853			195,100			2,427,874		

[a] Described in [30]

[b] Described in [17]

times were measured by reading the timestamp counter via the RDTSC and RDTSCP assembly instructions. To prevent out-of-order execution from distorting measurements, the CPUID instruction was serialized before reading the timestamp.

First, the execution time (in milliseconds) for generating the Goppa polynomial was measured using methods from NTS-KEM, Classic McEliece, random search, and our proposed approach. Table 2 presents these results for various values of N and t. The McEliece parameters are chosen to ensure at least maximum security (category 5, with $m = 13$ and $n = 8192$). Each measurement was averaged over 1000 runs of each algorithm.

As observed, for this specific set of parameters, the algorithm for finding separable Goppa polynomials described in NTS-KEM is faster than the method proposed in Classic McEliece. This performance gap becomes slightly more pronounced as t increases. However, when $N \geq 2$, our proposed method is clearly faster for all values of t. Specifically, for $N = 4$, the speed-up factors are approximately $\times 7.18$, $\times 5.98$, and $\times 5.69$ for $t = 128$, $t = 256$, and $t = 384$, respectively. It is worth noting that, compared to Classic McEliece, the execution time of our method is reduced by roughly a factor of N^2, as expected. Finally, this table also shows the execution time when an irreducible polynomial is found at random. In the best case, this time is on the order of seconds, which aligns with the theoretical predictions discussed in Sect. 2.2.1. Therefore, since on average the complex GCD algorithm must be run approximately $1/t$ times, this approach becomes impractical for real-world applications.

Table 3 presents similar results using the parameter set required by the iris cryptobiometric system shown in Fig. 4 ($t = 680$, $n = 16384$, $m = 14$, and $k = 6864$). Once again, our proposed method yields the best performance when $N \geq 2$, achieving an acceleration factor of approximately $\times 4.84$ compared to NTS-KEM. Similarly, when compared to the method proposed in Classic McEliece ($N = 1$), this factor is around $\times 14.39$, which is very close to the theoretical value of $N^2 = 16$.

Table 3 Execution time (in clock cycles) when computing the Goppa polynomial for $t = 680$

Method	$N = 1$	$N = 2$	$N = 4$
Proposed method	$1.12533 \cdot 10^{10}$	$2.88163 \cdot 10^{9}$	$7.81986 \cdot 10^{8}$
Classic McEliece[a]	$1.12533 \cdot 10^{10}$		
NTS-KEM[b]	$3.78807 \cdot 10^{9}$		

[a] Described in [30]

[b] Described in [17]

Table 4 provides detailed results for the main processes involved in encryption and decryption algorithms.

Figure 5 illustrates the distribution of Hamming distance (HD) scores between unrelated (different eyes) and related (same eye) iris codes, including seven rotations. As discussed in [53] and [55], these distributions can be well-modeled by fractional binomial distributions, represented by the solid curve shown in the figure. The resulting false rejection rate (FRR) and false acceptance rate (FAR) depend on the threshold value chosen for HD when using expression (11). It is important to note that since t and HD are fully correlated, selecting a value for t is equivalent to setting the threshold HD_{Thd}. This

Table 4 Execution time (in clock cycles) for encryption and decryption. Parameters: $t = 680, m = 14, n = 16384, k = 6864$, and $N = 4$

McEliece encryption	
Product $C_0 = \hat{H}\hat{I}_{ref}$	260593582 (81.435 *ms*)
McEliece decryption	
Product $\hat{H}\hat{I}_s$ and $\rightarrow C = \hat{H}(\hat{I}_s \oplus \hat{I}_{ref})$	19606185 (6.127 *ms*)
Extended binary parity check matrix $\rightarrow \hat{H}_{ext}$	263631440 (82.385 *ms*)
Syndrome computation $\hat{H}_{ext}(C\|0)$	19606185 (6.127 *ms*)
Berlekamp algorithm	19638590 (6.137 *ms*)
Error Locator (additive FFT)	19638590 (5.334 *ms*)
Total time decryption	320170762 (100.053 *ms*)

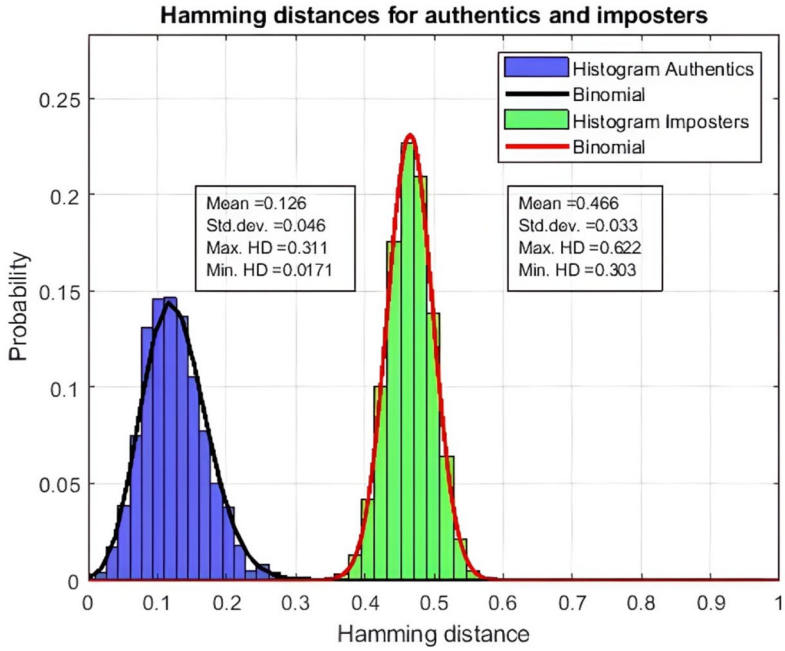

Fig. 5 Hamming distances for authentics and imposters users, allowing seven different degrees of eye or head tilt

relationship is demonstrated in Table 5, which shows the corresponding FRR and FAR rates for different values of t. Therefore, the use of McEliece does not affect the overall accuracy of the recognition process, so that the same FRR and FAR rates are obtained when compared with the non-protected scheme.

7 Conclusion

In this paper, we present a novel method for obtaining monic degree-t separable Goppa polynomials without linear factors. Such polynomials form part of the secret key in all cryptographic algorithms based on McEliece or its variants. Our main contribution is that the proposed method is faster than any previous approach, achieving speed-up factors greater than $\times 7.1$ when using the parameters for n and t specified in Classic McEliece. Additionally, experimental results corroborate that this factor is proportional to the square of the number N of polynomials used in the computation. When compared to NTS-KEM, a substantial enhancement is also observed. This improvement becomes particularly significant as the value of t increases. Furthermore, we demonstrated an application based on an iris biometric cryptosystem. The system employs a fuzzy commitment scheme, enabling the creation of biometric keys of arbitrary length and verifying user identity through iris codes. Moreover, biometric templates can be stored securely, thereby enhancing user privacy and protecting against attacks aimed at stealing confidential biometric data.

Table 5 False rejection rate (FRR) and false acceptance rate (FAR) depending on the threshold value HD_{Thd} chosen for HD, being $t = 2048 \cdot HD_{Thd}$. Values obtained using the theoretical binomial distributions shown in Fig. 5

HD_{Thd}	$t = 2048 \cdot HD_{Thd}$	FRR (%)	FAR (%)
0.3203	656	$1.19 \cdot 10^{-2}$	$1.87 \cdot 10^{-3}$
0.3242	664	$7.35 \cdot 10^{-3}$	$1.35 \cdot 10^{-3}$
0.3281	672	$4.41 \cdot 10^{-3}$	$2.44 \cdot 10^{-3}$
0.3320	680	$1.58 \cdot 10^{-3}$	$3.58 \cdot 10^{-3}$
0.3359	688	$9.32 \cdot 10^{-4}$	$5.17 \cdot 10^{-3}$
0.3398	696	$7.80 \cdot 10^{-4}$	$5.69 \cdot 10^{-3}$
0.3437	704	$5.36 \cdot 10^{-4}$	$7.20 \cdot 10^{-3}$

Acknowledgements
Our sincere recognition to the anonymous reviewers, whose constructive feedback has enhanced the quality of this manuscript.

Authors' contributions
M.L.G. wrote the main manuscript and prepared figures. E.C.N. prepared the main simulations. All authors reviewed the manuscript.

Funding
This work is supported by Ministerio de Ciencia e Innovación, Spain, grants PID2019-107274RB-100 (MCIN/AEI/10.13039/501100011033) and PID2023-148649OB-I00.

Data availability
No datasets were generated or analyzed during the current study.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 23 April 2025 Accepted: 26 September 2025
Published online: 24 October 2025

References

1. V. Goppa, A new class of linear error-correcting codes. Probl. Peredachi Inf. **6**, 24–60 (1970)
2. R.J. McEliece. A public-key cryptosystem based on algebraic coding theory. (1978). Preprint at https://tmo.jpl.nasa.gov/progress_report/42-44/44N.PDF
3. N. Sendrier, in *Encyclopedia of Cryptography and Security*, ed. by H.C.A. Van Tilborg. McEliece public key cryptosystem (Springer US, Boston, 2011), pp. 767–768
4. H. Niederreiter, Knapsack-type cryptosystems and algebraic coding theory prob. Contr. Inform. Theory **15**, 159–166 (1986)
5. V.M. Sidelnikov, O. Shestakov, On insecurity of cryptosystems based on generalized reed-solomon codes. J. Discret. Math. Appl. **2**(4), 439–444 (1992). https://doi.org/10.1515/dma.1992.2.4.439
6. H. Janwa, O. Moreno, Mceliece public key cryptosystems using algebraic-geometric codes. Des. Codes Crypt. **8**, 293–307 (1996). https://doi.org/10.1023/A:1027351723034
7. V.M. Sidelnikov, A public-key cryptosystem based on binary reed-muller codes. J. Discret. Math. Appl. **4**(3), 191–208 (1994). https://doi.org/10.1515/dma.1994.4.3.191
8. K. Gibson, The security of the Gabidulin public key cryptosystem. Advances in Cryptology — EUROCRYPT '96 (1996), pp. 212–223. https://doi.org/10.1007/3-540-68339-9_19
9. M. Baldi, M. Bianchi, F. Chiaraluce, Security and complexity of the McEliece cryptosystem based on quasi-cyclic low-density parity-check codes. IET Inf. Secur. **7**(3), 212–220 (2013). https://doi.org/10.1049/iet-ifs.2012.0127
10. A. Couvreur, I. Marquez-Corbella, R. Pellikaan, Cryptanalysis of mceliece cryptosystem based on algebraic geometry codes and their subcodes. IEEE Trans. Inf. Theory **63**(8), 5405–5418 (2017). https://doi.org/10.1109/TIT.2017.2712636
11. L. Minder, A. Shokrollahi, *Advances in Cryptology - EUROCRYPT 2007*, Cryptanalysis of the Sidelnikov cryptosystem (2007), pp. 347–360. https://doi.org/10.1007/978-3-540-72540-4_20
12. M.A. Borodin, I.V. Chizhov, Effective attack on the mceliece cryptosystem based on reed-muller codes. Discret. Math. Appl. **24**(5), 273–280 (2014). https://doi.org/10.1515/dma-2014-0024
13. T. Fabšič, V. Hromada, *International Workshop on Post-Quantum Cryptography*, A reaction attack on the QC-LDPC McEliece cryptosystem (2017), pp. 51–68. https://doi.org/10.1007/978-3-319-5
14. M. Albrecht, D. Bernstein, Classic McEliece: conservative code-based cryptography. (2020). Preprint at https://classic.mceliece.org/nist/mceliece-20201010.pdf
15. NIST, Post-quantum cryptography standardization: call for proposals. (2016). Preprint at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/call-for-proposals
16. M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, P. Santini, Ledacrypt specification v3.0. (2020). Preprint at https://nts-kem.io/
17. M. Albrecht, C. Cid, K.G. Paterson, C. Tjhai, M. Tomlinson, Nts-kem a code-based key-encapsulation mechanism submitted to NIST post-quantum cryptography standardization process. (2019). Preprint at https://nts-kem.io/
18. L.K. Grover, A fast quantum mechanical algorithm for database search (1996). https://doi.org/10.1145/237814.237866
19. D. Bernstein, Grover vs. McEliece, International Workshop on Post-Quantum Cryptography (2010), pp. 73–80. https://doi.org/10.1007/978-3-642-12929-2_6
20. R. Arjona, P. López-González, P. Román, I. Baturone, Post-quantum biometric authentication based on homomorphic encryption and classic McEliece. Appl. Sci. **13**(2:757), 1–19 (2023). https://doi.org/10.3390/app13020757
21. E. Cantó-Navarro, M. López-García, AXI hardware accelerator for McEliece on FPGA embedded systems. IEEE Trans. Dependable Secure Comput. 1–14 (2024). https://doi.org/10.1109/TDSC.2024.3445181
22. M. López-García, E. Cantó-Navarro, Proceedings of the 2020 future of information and communication conference (ficc). Hardware-Software Implementation of a McEliece Cryptosystem for Post-quantum Cryptography. **3**, 814–825 (2020). https://doi.org/10.1007/978-3-030-39442-4_60
23. E. Jochemsz, Goppa codes and the McEliece cryptosystem, Vrije Universiteit Amsterdam. (2002). Preprint at https://pdfcoffee.com/goppa-codes-and-the-mceliece-cryptosystem-pdf-free.html
24. R. Safieddine, A. Desmarais, Comparison of different decoding algorithms for binary Boppa codes (University of Lyon, Hubert Curien Laboratory. pp. 1–19 (2014). Preprint at https://www.cayrel.net/?Implementation-of-Goppa-codes
25. Flexiprovider: JCryptool Cryptogtaphy for everybody. (2016). Preprint at https://www.cryptool.org/en/
26. A. Shoufan, T. Wink, H.G. Molter, S.A. Huss, E. Kohnert, A novel cryptoprocessor architecture for the mceliece public-key cryptosystem. IEEE Trans. Comput. **59**(11), 1533–1546 (2010). https://doi.org/10.1109/TC.2010.115
27. IEEE Standard Specifications for Public-Key Cryptography, IEEE Std **1363–2000**, 1–228 (2000). https://doi.org/10.1109/IEEESTD.2000.92292
28. E.R. Berlekamp, *Algebraic Coding Theory* (McGraw-Hill, New York, 1968)
29. K.C. Sunil, J. Minac, Counting irreducible polynomials over finite fields using the inclusion-exclusion principle. Math. Mag. **84**(5), 369–371 (2011). arXiv:1001.0409
30. W. Wang, J. Szefer, R. Niederhagen, FPGA-based Key Generator for the Niederreiter Cryptosystem Using Binary Goppa Codes, in *Cryptographic Hardware and Embedded Systems - CHES 2017*. ed. by W. Fischer, N. Homma (Springer International Publishing, Cham, 2017), pp.253–274
31. W. Wang, J. Szefer, R. Niederhagen, FPGA-Based Niederreiter Cryptosystem Using Binary Goppa Codes, in *Post-Quantum Cryptography*. ed. by T. Lange (Springer International Publishing, Cham, 2018), pp.77–98
32. M. Chen, T. Chou, Classic McEliece on the ARM Cortex-M4. IACR Trans. Cryptographic Hardw. Embed. Syst. **2021**(3), 125–148 (2021). https://doi.org/10.46586/tches.v2021.i3.125-148
33. P. Chen, T. Chou, S. Deshpande, N. Lahr, R. Niederhagen, J. Szefer, W. Wang, Complete improved FPGA implementation of classic McEliece. IACR Trans. Cryptographic Hardw. Embed. Syst. **2022**(3), 71–113 (2022). https://doi.org/10.46586/tches.v2022.i3.71-113
34. Classic McEliece vs NTS-KEM Classic McEliece Comparison Task Force. (2018). Preprint at https://classic.mceliece.org/nist/vsntskem-20180629.pdf
35. S. Gao, M. Todd, Complete improved fpga implementation of classic mceliece. Addit. Fast Fourier Transforms Over Finite Fields **56**(12), 6265–6272 (2010). https://doi.org/10.1109/TIT.2010.2079016
36. D. Bernstein, T. Chou, P. schwabe, in *Cryptographic Hardware and Embedded Systems - CHES 2013*, ed. by G. Bertoni, J.S. Coron. McBits: fast constant-time code-based sryptography, (Springer, Berlin Heidelberg, Berlin, Heidelberg, 2013), pp.250–272
37. G. Seroussi, Table of low-weight binary irreducible polynomials. (1998). Preprint at https://api.semanticscholar.org/CorpusID:18027797
38. C. Gauss, *Untersuchungen über höhere Arithmetik* (Chelsea Publishing Company, New York, 1981)
39. D. Owens, R. Khatib, M. Bisheh-Niasar, R. Azarderakhsh, M.M. Kermani, Efficient and side-channel resistant ed25519 on arm cortex-m4. IEEE Trans. Circuits Syst. I Regul. Pap. **71**(6), 2674–2686 (2024). https://doi.org/10.1109/TCSI.2024.3384414
40. R. Elkhatib, B. Koziel, R. Azarderakhsh, M.M. Kermani, Cryptographic engineering a fast and efficient SIKE in FPGA. ACM Trans. Embed. Comput. Syst. **23**(2), 1–25 (2024). https://doi.org/10.1145/3584919
41. P. S., J. Geier, J. Danner, D. Mueller-Gritschneder, A. Wachter-Zeh, Key-recovery fault injection attack on the classic McEliece KEM. (2022). Preprint at https://eprint.iacr.org/2022/1529
42. S. Bitzer, J. Delvaux, E. Kirshanova, A. Maaßen, S. May, A. Wachter-Zeh, How to lose some weight - a practical template syndrome decoding attack. (2024). Preprint at https://eprint.iacr.org/2024/621
43. M. Brinkmann, C. Chuengsatiansup, A. May, J. Nowakowski, J. Yarom, Leaky McEliece: secret key recovery from highly erroneous side-channel information. (2023). Preprint at https://eprint.iacr.org/2023/1536

44. P. Cayrel, B. Colombier, V. Dragoi, A. Menu, L. Bossuet, Message-recovery laser fault injection attack on the classic McEliece cryptosystem. (2020). Preprint at https://eprint.iacr.org/2020/900
45. B. Colombier, V. Dragoi, P. Cayrel, V. Grosso, Profiled side-channel attack on cryptosystems based on the binary syndrome decoding problem. IEEE Trans. Inf. Forensics Secur. **17**, 3407–3420 (2022). https://doi.org/10.1109/TIFS.2022.3198277
46. N. Lahr, R. Niederhagen, R. Petri, S. Samardjiska, Side channel information set decoding using iterative chunking. (2019). Preprint at https://eprint.iacr.org/2019/1459
47. Q. Guo, A. Johansson, T. Johansson, A key-recovery side-channel attack on classic McEliece implementations. IACR Trans. Cryptographic Hardw. Embed. Syst. **2022**(4), 800-–827 (2022). https://doi.org/10.46586/tches.v2022.i4.800-827
48. P. Gan, P. Ravi, K. Raj, A. Baksi, A. Chattopadhyay, Classic McEliece hardware implementation with enhanced side-channel and fault resistance. (2024). Preprint at https://eprint.iacr.org/2024/1828
49. A. Juels, A. Wattenberg, in *Proceedings of the 6th ACM Conference on Computer and Communications Security*, A fuzzy commitment scheme (Association for Computing Machinery, New York, 1999), pp. 28–36
50. F. Hao, R. Anderson, J. Daugman, Combining crypto with biometrics effectively. IEEE Trans. Comput. **55**(9), 1081–1088 (2006). https://doi.org/10.1109/TC.2006.138
51. J. Daugman, High confidence visual recognition of persons by a test of statistical independence. IEEE Trans. Pattern Anal. Mach. Intell. **15**(11), 1148–1161 (1993). https://doi.org/10.1109/34.244676
52. J. Daugman, The importance of being random: statistical principles of iris recognition. Pattern Recogn. **36**(2), 279–291 (2003). https://doi.org/10.1016/S0031-3203(02)00030-4
53. J. Daugman, New methods in iris recognition. IEEE Trans. Syst. Man Cybern. B. **37**(5), 1167–1175 (2007). https://doi.org/10.1109/TSMCB.2007.903540
54. M. López-García, J. Daugman, E. Cantó-Navarro, Hardware–software co-design of an iris recognition algorithm. IET Inf. Secur. **5**(1), 60–68 (2011). https://doi.org/10.1049/iet-ifs.2009.0267
55. J. Daugman, Information theory and the iriscode. IEEE Trans. Inf. Forensics Secur. **11**(2), 400–409 (2016). https://doi.org/10.1109/TIFS.2015.2500196

Publisher's Note