ANALYSIS OF AN AI BASED CODE DISTINGUISHER

QUENTIN LAVIT, JULIEN SEINTURIER, AND PASCAL VERON

This is a preprint of a paper to be published in the 20th IMA International Conference on Cryptography and Coding proceedings.

Abstract. Recently, *DeepDistinguisher* [1] demonstrated the capability of transformer-like architectures to accurately distinguish Goppa codes from linear codes randomly generated, outperforming traditional distinguishers in certain settings. The purpose of our research is to examine the working mechanism of the mentioned machine learning algorithm. In particular, we explore the performance of the distinguisher using saliency maps and attention visualization techniques. We first identify a low-weight-row artifact present in the random instances used during training and show that the model partially relies on this feature. After removing this artifact, the distinguisher still achieves significant accuracy, suggesting the use of more structural properties. Inspired by the constraints induced by the minimum distance of Goppa codes, we proposed a rule based on pairwise XOR operation between the rows of the generator matrix. Our experiments have shown that the predictions made by *DeepDistinguisher* and XOR-based rule are highly correlated especially for high-rate codes. Furthermore, when evaluated on random matrices satisfying both the minimum distance and XOR constraints, the neural distinguisher loses most of its predictive power. This suggests that transformer-based architectures exploit general combinatorial properties of a code rather than some hidden underlying algebraic structure.

1. Introduction

In 1978, the seminal work of Robert J. McEliece [2] laid the foundations of code based cryptography. Despite its strong security guarantees and decades of resistance against cryptanalytic attacks, code-based cryptography remained for a long time relatively marginal within the broader cryptographic landscape. This situation changed suddendly with the emergence of the quantum threat. In response to this threat, the National Institute of Standard and Technology launched in 2016 the post quantum standardization process whose goal was to identify and standardize cryptographic schemes resistant to quantum attacks. In this context, code based cryptography has become one of the concrete post quantum alternative, alongside with lattice based, multivariates and hash based approaches.

Among the many hard computational problems arising in code-based cryptography, the *Code Distinguishing Problem* occupies a particularly important place. Informally, this problem consists in determining whether a given generator matrix corresponds to a highly structured family of error-correcting codes (such as Goppa codes) or whether it is simply the generator matrix of a random linear code. The hardness of distinguishing structured codes from random ones is fundamental for the security of several cryptographic constructions, especially those relying on hidden algebraic structure.

The Goppa Distinguisher (GD) problem was formally introduced in [**?**] and was

initially considered computationally hard. Because of this presumed hardness, it was used as a foundational assumption in some security proof in code based cryptography. In the last decade, a variety of approaches were suggested for solving the GD problem. A major breakthrough came with the FGOPT distinguisher proposed by J.C. Faugère, E. Gautier, A. Otmani, L. Perret, and J.-P. Tillich [3]. They proposed a polynomial time distinguisher for codes whose rate $R = \frac{k}{n}$ is close to 1. In 2023, Couvreur, Mora and Tillich [4] improved this approach and obtained a distinguisher for code rate $R \in [\frac{2}{3}, 1]$. The complexity of their algorithm is subexponential as long as the code dimension k grows between linear and quadratic with respect to the codimension $n - k$. Asymptotically, the limitation on the rate R has been dropped in 2025 by H. Randriambololona [5] with a distinguisher which can solve GD for a large range of parameters and whose complexity is subexponential in the error-correcting capacity of the code. Nevertheless, for practical finite-size parameters, the rate still remains a critical limiting factor. These distinguishers are able to find in structured codes certain invariants capable of revealing hidden algebraic dependencies absent in random codes.

More recently, machine learning and artificial intelligence techniques have opened a new research direction in cryptanalysis. In a pioneering work, M. Malhou, L. Perret, and K. Lauter introduced *DeepDistinguisher* [1], a neural-network-based distinguisher capable of identifying some statistical patterns separating structured codes from random linear codes. It achieves high accuracy in distinguishing Goppa codes, and outperforms traditional distinguishers in certain settings.

This work relies on an analysis of *DeepDistinguisher* to identify the criteria that leads to the prediction such as hidden algebraic structure in the code to classify or generic combinatorial constraints of a code. To validate our conclusions about the behavior of *DeepDistinguisher*, we introduce two combinatorial distinguishers, referred to as *XOR-based* distinguishers, and evaluate the agreement between their decisions and those produced by *DeepDistinguisher*.

This paper is organized as follows : Section 2 recalls the necessary background in coding theory. Section 3 presents *DeepDistinguisher* and Section 4 proposes its analysis using gradient based saliency and attention maps. In Section 5 we present our *XOR-based distinguishers* and compare them with *Deep Distinguisher*. All the code developped for this paper is avalaible at : `https://github.com/apzaED0/DeepDistinguisher-analysis`.

2. Background in coding theory

In this section, we review the minimal coding-theoretic concepts required for the understanding of this paper.

Definition 1. *A binary (n, k) linear code $\mathcal{C}$ is a k-dimensional vector subspace of $\mathbb{F}_2^n$. A $k \times n$ matrix G is called a generator matrix of $\mathcal{C}$ if its rows form a basis of $\mathcal{C}$. The elements of $\mathcal{C}$ are called* codewords. *The* rate *of the code is the quantity k/n. It represents the proportion of information symbols among the n symbols of a codeword, the remaining symbols providing redundancy for error correction.*

Definition 2. *Up to a permutation of its columns, a generator matrix of a binary (n, k) linear code can be written in the form $G = (I_k \mid M)$. Such a matrix is called a systematic generator matrix.*

Definition 3. *Let $x = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$. The Hamming weight of x, denoted by $w_H(x)$, is the number of its nonzero coordinates: $w_H(x) = card(\{\, i \in \{1, \ldots, n\} \mid x_i \neq \})$.*

Definition 4. *The minimum weight of a linear code $\mathcal{C}$ is defined as : $w_{\min}(\mathcal{C}) = \min(\{w_H(x), x \in \mathcal{C} \setminus \{0\}\})$. The error correction capacity of a binary linear code $\mathcal{C}$ is the value $t = \lfloor \frac{w_{\min}-1}{2} \rfloor$. It means that for any $y \in \mathbb{F}_2^n$, there exists at most one codeword x, such that $y = x + e$, with $x \in \mathcal{C}$, $e \in \mathbb{F}_2^n$ and $w_H(e) \leqslant t$.*

Proposition 1 (General Decoding Problem (GDP))**.** *Let $y \in \mathbb{F}_2^n$, and $\mathcal{C}$ a t-error correcting linear binary code. Decide whether there exists a codeword $x \in \mathcal{C}$ and $e \in \mathbb{F}_2^n$ such that $y = x + e$ with $w_H(e) \leqslant t$ is an NP-complete problem* [6].

While the decoding of a random linear binary code is generally an NP-complete problem, there exists classes of codes for which the problem can be solved in polynomial time. In this paper we will focus on such a class : Goppa codes [7].

Definition 5. *Let $g(z) \in \mathbb{F}_{2^m}[z]$ and $L = \{\alpha_1, \ldots, \alpha_n\} \subset \mathbb{F}_{2^m}$, such that $\forall i$, $g(\alpha_i) \neq 0$, the binary Goppa code $\Gamma(L, g)$ is defined as the set of words $(c_1, \ldots, c_n) \in \mathbb{F}_2^n$ such that : $\sum_{i=1}^{n} \frac{c_i}{z-\alpha_i} \equiv 0 \pmod{g(z)}$.*

Proposition 2. *Let $\Gamma(L, g)$ be a binary Goppa code, then :*

- *$n = card(L)$,*
- *$k \geqslant n - m \deg g(z)$,*
- *$w_{\min}(\Gamma(L, g)) \geqslant \deg \bar{g}(z) + 1$,*
- *$t \geqslant \lfloor \frac{\deg \bar{g}(z)}{2} \rfloor$, where $\bar{g}(z)$ is the square-free polynomial of smallest degree such that $g(z)$ divides $\bar{g}(z)$.*

In practice, the dimension is most often equal to the designed dimension, namely $k = n - m \deg g(z)$. For cryptographic applications, the Goppa polynomial $g(z)$ is usually chosen to be irreducible, or at least square-free over $\mathbb{F}_{2^m}$, so that $\bar{g}(z) = g(z)^2$ and so $w_{\min}(\Gamma(L, g)) \geqslant 2 \deg(z) + 1$ and $t \geqslant \deg g(z)$.

Important remark : As a consequence, for all codeword $c \in \Gamma(L, g) \setminus \{0\}$, $w_H(c) \geqslant 2 \deg g(z) + 1$ and this will be an important point used in the sequel of this paper. Among the "difficult problems" in coding theory, the Goppa Distinguisher problem is believed to be computationally hard to solve [**?**].

Definition 6 (Goppa Distinguisher Problem)**.** *A distinguisher $\mathcal{D}$ is an algorithm that takes as input a $k \times n$ matrix G and returns a bit. $\mathcal{D}$ solves the GD problem if it wins the following game :*

- *b is randomly chosen in $\{0, 1\}$,*
- *If $b = 0$ then G is chosen as a generator matrix of a (n, k) Goppa code else G is a random $k \times n$ binary matrix,*
- *If $\mathcal{D}(G) = b$ then $\mathcal{D}$ wins else $\mathcal{D}$ loses.*

3. DeepDistinguisher

3.1. **Code distinguishability task.** *DeepDistinguisher* is a Neural Network (NN) based algorithm that uses supervised learning introduced by M. Malhou, L. Perret and K. Lauter [1]. Given a generator matrix $G \in \mathcal{M}_{k \times n}(\mathbb{F}_q)$ in systematic form, the model classifies G as either a code generating matrix from a structured family (Goppa, alternant, MDPC or QC-MDPC), or a random matrix. In this work,

we focus on binary Goppa codes, which means $q = 2$. We first describe the general pipeline of *DeepDistinguisher*, then study its behavior experimentally, before presenting our own contribution.

3.2. **General pipeline of *DeepDistinguisher*.** The input is a generator matrix in systematic form. Each row of this matrix is first mapped to a real-valued vector through an embedding layer. The resulting sequence of row representations is then processed by a Transformer encoder, following the self-attention architecture introduced in [8]. The output representations are aggregated by a max-pooling operation, and a final linear classification layer produces a scalar value, called a logit. Applying a sigmoid function to this logit gives the predicted probability that the input matrix belongs to the structured code family. We now describe the main steps of this architecture.

Embedding Layer. Let $G \in \mathcal{M}_{k\times n}(\mathbb{F}_2)$ be the input generator matrix. The embedding layer transforms the binary matrix G into a real matrix $X^{(0)} \in \mathcal{M}_{k\times d}(\mathbb{R})$ where d is the embedding dimension of the Transformer. This step should be understood as a learned change of representation: instead of working directly with binary rows, the model works with continuous vectors whose coordinates are optimized during training.

Transformer encoder. The matrix $X^{(0)}$ is then processed by a sequence of Transformer blocks. Each block contains two main components: a multi-head self-attention mechanism and a feed-forward neural network.

The role of self-attention is to update the representation of each row by allowing it to interact with the representations of all other rows.

Let $X \in \mathcal{M}_{k\times d}(\mathbb{R})$ denote the matrix of row representations at the input of one Transformer block. The purpose of self-attention is to update the representation of each row by allowing it to interact with the representations of all other rows.

In the multi-head attention mechanism, this interaction is not computed only once. Instead, the model uses several attention heads in parallel. Each head processes the whole sequence of rows, but through its own learned projected representation. Therefore, the different heads do not split the rows of the matrix, they split the representation space. More precisely, let h be the number of heads used. For each head $a \in 1, \ldots, h$, the input matrix X is linearly projected into three matrices: $Q^{(a)} = XW_Q^{(a)}$, $K^{(a)} = XW_K^{(a)}$, $V^{(a)} = XW_V^{(a)}$. The matrices $W_Q^{(a)}, W_K^{(a)}, W_V^{(a)}$ are trainable real-valued parameters of the neural network. They are not derived from the generator matrix G, nor from any explicit algebraic description of the code. They are initialized randomly at the beginning of training and then adjusted by backpropagation and gradient-based optimization in order to reduce the classification loss. For a given head a, the matrices $Q^{(a)}$, $K^{(a)}$, and $V^{(a)}$ play different roles. The query matrix $Q^{(a)}$ describes what each row is looking for in the other rows. The key matrix $K^{(a)}$ describes how rows are compared to one another. The value matrix $V^{(a)}$ contains the information that will be combined once the attention weights have been computed. For each head a, the first step is to compare the query vector of each row with the key vectors of all rows. This is done by computing the matrix of scalar products $S^{(a)} = Q^{(a)} \times {}^tK^{(a)}$. This produces a matrix of similarity scores. After normalization by $\sqrt{d}$ and application of the row-wise softmax function, these scores become attention weights: $A^{(a)} = \text{softmax}\left(\frac{S^{(a)}}{\sqrt{d}}\right)$. The entry $A_{ij}^{(a)}$ measures

the similarity, in head a, between the query vector associated with row i and the key vector associated with row j. Thus, the i-th row of $A^{(a)}$ contains the scores used to decide which rows are relevant for updating the representation of row i. The output of head a is then $O^{(a)} = A^{(a)}V^{(a)}$. Thus, for each head, every row representation is replaced by a weighted combination of the value vector associated with all rows. Since the attention weights depend on the input matrix, the combinations are adaptive and may vary from one generator matrix to another. This mechanism allows the model to detect relations between rows, rather than treating them independently. The outputs of the h attention heads are then concatenated: $O = \text{Concat}\left(O^{(1)}, \ldots, O^{(h)}\right)$, and projected back to the model dimension through another trainable matrix: $\text{MHSA}(X) = OW_O$. The matrix W_O is also learned during training. This final projection mixes the information extracted by the different heads.

The use of several heads gives the model the capacity to combine several statistical views of the same generator matrix.

After the multi-head self-attention step, a feed-forward network is applied independently to each row representation. This network acts as a nonlinear transformation of the features computed by the attention mechanism. The successive Transformer blocks therefore alternate between two operations: exchanging information between rows through multi-head attention, and transforming the resulting row-wise representations through nonlinear maps.

After b Transformer blocks, the model outputs a matrix $H^{(b)} \in \mathcal{M}_{k\times d}(\mathbb{R})$, whose rows are contextual representations of the rows of the original generator matrix. Here, "contextual" means that the representation of a given row no longer depends only on this row, but also on its learned relations with the other rows of the matrix. *DeepDistinguisher* applies four transformer blocks successively, i.e. $b = 4$: $H^{(4)} = \text{Block}_4 \circ \text{Block}_3 \circ \text{Block}_2 \circ \text{Block}_1(X^{(0)})$.

Aggregation and classification. The Transformer encoder produces one vector for each row of the input matrix. To obtain a single representation of the whole code, *DeepDistinguisher* applies a max-pooling operation over the row dimension. This produces a vector $u \in \mathbb{R}^d$, where each coordinate u_j is defined as : $u_j = \max_{i=1,\ldots,k} H^{(4)}_{i,j}$. This pooling step aggregates the information extracted from the rows into a global descriptor of the input matrix. A final linear layer then maps this descriptor to a scalar logit $\ell = \langle w, u\rangle + b$, where w and b are learned parameters. The predicted probability is obtained by applying the sigmoid function: $p = \sigma(\ell) = \frac{1}{1+e^{-\ell}}$. The input matrix is classified as structured when p is above a fixed threshold, usually $1/2$, and as random otherwise.

3.3. **Reported results from *DeepDistinguisher*.** The original experiments of *DeepDistinguisher* show that transformer-based models can distinguish Goppa generator matrices from random generator matrices with high accuracy on several small and medium-size parameter sets (see Fig. 18 and 19 in Annex B reproduced from [1]).

For binary Goppa codes with parameters $m = 6$ and $n = 64$, the reported accuracies are close to perfect for small values of the Goppa degree t. The model reaches more than 98% accuracy for $2 \leq t \leq 5$, and still obtains about 96.7% for $t = 6$. Notably, this latter case corresponds to a code rate of about 0.44, substantially lower than the rate threshold of 0.67 reported in [4]. The performance

decreases for larger degrees, which suggests that the task becomes harder as mt increases, or equivalently as the code rate decreases.

These results motivate the rest of this report. Since *DeepDistinguisher* achieves high classification accuracy on small binary Goppa parameters, a natural question is whether the model has learned genuinely structural properties of such Goppa codes, or whether it exploits simpler statistical artifacts in the data generation process. In the following sections, we investigate this question using gradient-based saliency maps and controlled modifications of the random distribution.

4. Saliency and Attention Analysis

Following the authors' suggestions on explainability we apply gradient-based saliency and attention-map visualization to both random and Goppa code matrices. Before proceeding, we discuss the representation of matrices given as input to *DeepDistinguisher*, as well as the generation of random matrices.

4.1. **Representation.** The original approach uses the full systematic generator matrix $G = (I_k \mid A)$ as input to the model. However, once the generator matrices are put in systematic form, the identity block I_k is common to both Goppa and random instances. Therefore, this block is not expected to contain direct information about the difference between the two distributions.

Hence, we trained the model using only the redundancy block A. We obtained the same results as those obtained with the full generator matrix G (see Appendix B, Fig. 20), but with a moderately reduced computational cost, depending on the code parameters. Therefore, in the following experiments, all models are trained on the redundancy-only representation rather than on the full generator matrix.

4.2. **Low-weight-row artifact.** In a cryptographic context, for a Goppa polynomial of degree t, the minimum distance of the associated binary Goppa code satisfies $d \geq 2t + 1$. Hence, for $G = (I_k \mid A)$, every row of A must have weight at least $2t$. For a uniformly random binary matrix $A \in \mathcal{M}_{k\times(n-k)}(\mathbb{F}_2)$, this condition is not automatically satisfied. If the rows of A are sampled uniformly and independently, then the Hamming weight of one row A_i follows a binomial distribution: $w_H(A_i) \sim \mathrm{Bin}(n-k, 1/2)$. Hence, the probability that a fixed row satisfies the Goppa row-weight constraint $w_H(A_i) \geq 2t$ is $\Pr[w_H(A_i) \geqslant 2t] = \sum_{j=2t}^{n-k} \binom{n-k}{j} 2^{-(n-k)}$. Since the rows are sampled independently, the probability that at least one row violates the constraint is $1 - \left(\sum_{j=2t}^{n-k} \binom{n-k}{j} 2^{-(n-k)}\right)^k$. The original random dataset used to train *DeepDistinguisher* contains matrices with rows that could not occur in the A-part of a binary Goppa generator matrix.

Definition 7. *Let A be a random $(n - mt) \times mt$ matrix. A row A_i is said to be* Goppa-compatible *if $w_H(A_i) \geq 2t$. Otherwise, i.e., if $w_H(A_i) < 2t$, the row is said to be* Goppa-incompatible.

A natural question is then whether the model can distinguish Goppa matrices from random matrices by exploiting this simple low-weight-row artifact.

4.3. **Gradient-based saliency visualization.** On the first hand, we conduct gradient-based saliency, to asses whether these rows are particularly informative for the model.

Let $X \in \mathcal{M}_{(n-mt)\times mt}(\mathbb{F}_2)$ denote the matrix given to the model, and let $z(X)$ be the logit produced by the distinguisher before the sigmoid activation. For each entry $X_{i,j}$, we compute $\nabla_{i,j}(X) = \frac{\partial z(X)}{\partial X_{i,j}}$. Since the inputs are binary, we use the following centered gradient attribution: $S_{i,j} = \nabla_{i,j}\left(X_{i,j} - \frac{1}{2}\right)$. This quantity gives a signed local attribution for each entry of the matrix. A positive value of $S_{i,j}$ means that the current value of the entry tends to increase the logit, hence to push the prediction towards class 1. In our setting, class 1 corresponds to Goppa codes, while class 0 corresponds to random codes. Conversely, a negative value indicates that the current value tends to decrease the logit, and therefore pushes the prediction towards the random class.

To determine the importance on the prediction of each row, we aggregate the gradient-based attributions at the row level. Given a saliency map S, we define the row attribution score as $R_i = \frac{1}{n-k}\sum_j |S_{i,j}|$. We then compare the average value of R_i for rows violating the constraint $w_H(A_i) \geq 2t$ with the average value obtained for rows satisfying the constraint (see Fig. 1). We observe that these rows are partic-

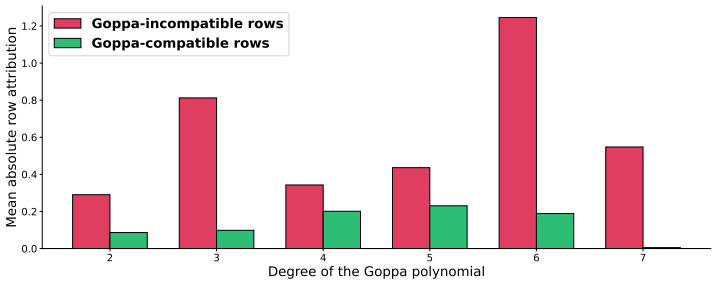

FIGURE 1. Mean absolute row attribution for Goppa-compatible and Goppa-incompatible rows in random generator matrices, for codes of length $n = 64$ and dimension $k = n - 6t$, as a function of the Goppa polynomial degree t.

ularly informative in comparison of Goppa-compatible rows. To determine whether the model actually uses these rows as an indication of the random distribution, we then compute the signed row-attribution score as follows: $R_i = \frac{1}{n-k}\sum_j S_{i,j}$. Figure 2 shows that the signed attribution of Goppa-incompatible rows is negative on average. Since class (1) corresponds to Goppa codes and class (0) to random

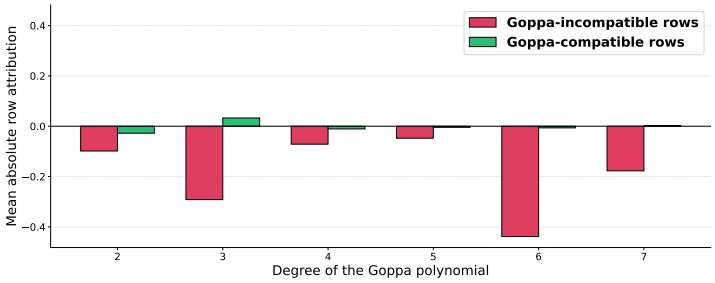

FIGURE 2. Mean signed row attribution for Goppa-compatible and Goppa-incompatible rows in random generator matrices, for codes of length $n = 64$ and dimension $k = n - 6t$, as a function of the Goppa polynomial degree t.

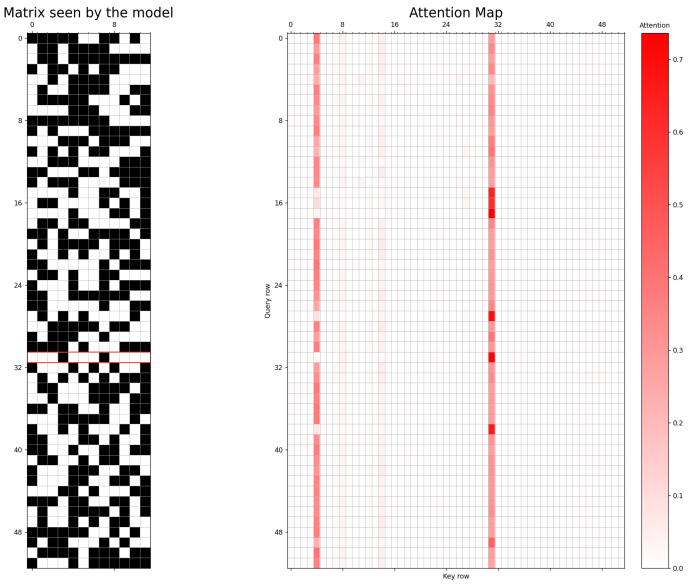

FIGURE 3. Attention map visualization after the 4th and last layer for a random binary generator matrix associated with a random linear $[n, k] = [64, 52]$ code. The code has the same length and dimension as the Goppa code with parameters $(n, m, t) = (64, 6, 2)$. Entries equal to 1 are represented in black, while entries equal to 0 are represented in white.

codes, this indicates that Goppa-incompatible rows tend to decrease the Goppa logit and therefore push the prediction towards the random class. This supports the conjecture that the neural distinguisher exploits Goppa-incompatible rows as a random-like signal. As it can be seen on Figure 2, Goppa-compatible rows seem to have no effect on the final decision. Attention-maps visualization is used to highlight the interactions between the rows performed by the Transformer. For each Transformer layer, we extract the attention weights computed during the forward pass. Therefore, the coefficient $A_{i,j}$ measures how much row i attends to row j when updating its representation. The resulting matrices are visualized as heatmaps. The two attention-map visualizations reveal different attention patterns for the random baseline and the Goppa instance. In the Goppa case, the attention map remains comparatively diffuse, although some vertical structures are still visible (see Fig. 4). In the random case, the attention map contains more pronounced vertical bands, meaning that many query rows attend to the same key rows (see Fig. 3). Some of these key rows correspond to Goppa-incompatible rows in the matrix representation, (see line 31 highlighted in red in Fig. 3). This provides qualitative evidence that the model has learned to focus on specific rows carrying a strong statistical signal, rather than relying uniformly on the whole generator matrix. The attribution and attention-map analyses both seem to indicate that Goppa-incompatible rows play a central role in the decision process of the neural distinguisher. Since these Goppa-incompatible rows provide a strong and easily exploitable signal for distinguishing random matrices from Goppa matrices, we introduce a modified random distribution in which such rows are removed. The goal

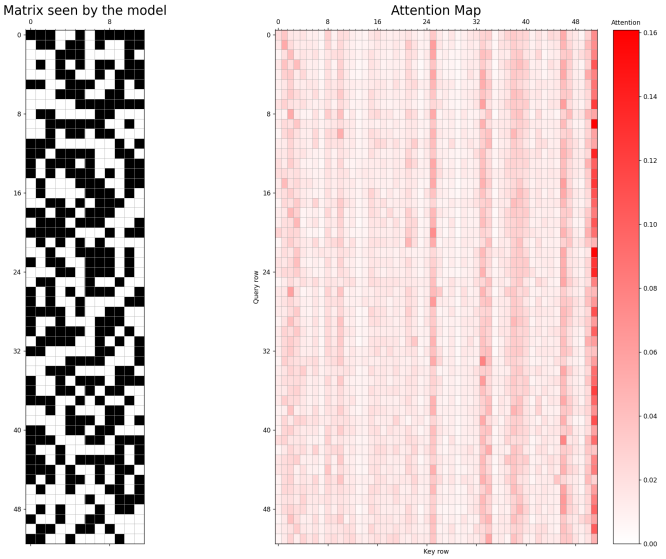

FIGURE 4. Attention map visualization after the 4th and last layer, for a generator matrix of a $(n, m, t) = (64, 6, 2)$ Goppa code. Entries equal to 1 are represented in black, while entries equal to 0 are represented in white.

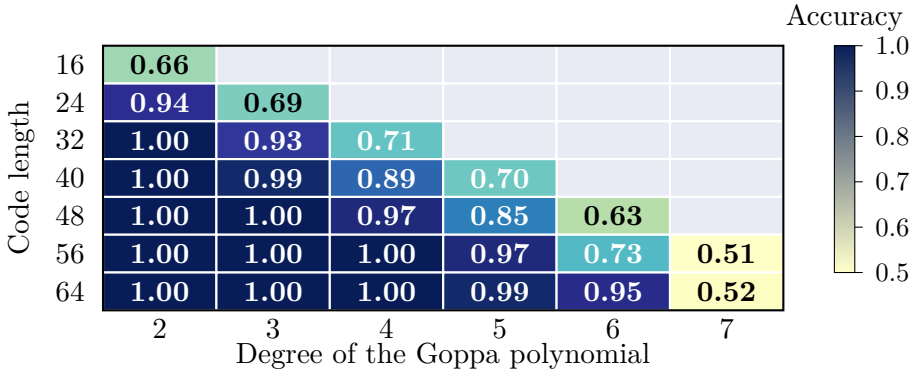

FIGURE 5. Results of *DeepDistinguisher* trained on the original distribution when evaluated on the corrected random distribution.

is to suppress this simple statistical cue and thereby make the distinguishing task more challenging.

4.4. **Corrected random distribution.** To remove this artifact, we modify the random matrix generation procedure by rejecting every random matrix A containing a Goppa-incompatible row. The distribution containing a balanced amount of random samples with only Goppa-compatible row and Goppa samples is called "corrected random distribution".

The first experiment conducted was to evaluate *DeepDistinguisher* on this new corrected random distribution. Although the accuracy decreases for some parameters (see Fig. 5), *DeepDistinguisher* remains able to distinguish Goppa matrices

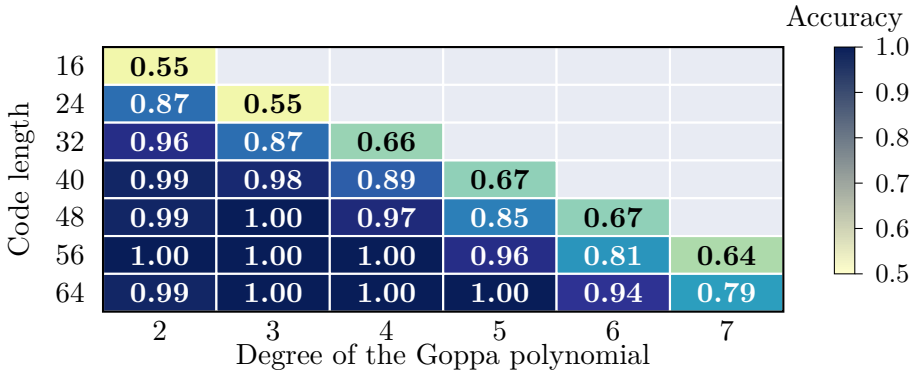

FIGURE 6. Results of *DeepDistinguisher* when trained on the corrected random distribution and evaluated on it.

from corrected random matrices. This suggests that the low-weight-row artifact is not the only signal exploited by the model.

Seeing that *DeepDistinguisher* is still able to distinguish with high accuracy between random code generator matrices with only Goppa-compatible rows and Goppa code generator matrices, we trained *DeepDistinguisher* on a balanced dataset with random samples from the corrected random distribution. When retrained, *DeepDistinguisher* obtains similar results compare to previously, with some drops of accuracy for some parameters, especially low code length and low degree of Goppa polynomial (see Fig. 6).

We therefore apply attention-map visualization again to the retrained *DeepDistinguisher*. The two attention-map visualizations still exhibit distinct attention patterns for the random baseline and the Goppa instance. In the Goppa case, the attention weights are mainly concentrated along the diagonal, with only weak off-diagonal interactions (see Fig. 7). This suggests that, at this layer, each row mostly attends to itself rather than to other rows. Such a pattern can be interpreted as a lack of strong cross-row signal captured by the attention mechanism. In the random case, the attention map contains pronounced vertical bands: many query rows attend to the same key rows (see Fig. 8). Compared with the previous visualization, more rows appear to be involved, but a small subset of rows still acts as attention attractors. These attractive rows may carry features used by the model to classify the matrix as coming from the random distribution. The corrected random distribution removes the most direct row-wise artifact: all rows now satisfy the individual Hamming-weight constraint imposed by the minimum distance of binary Goppa codes. However, *DeepDistinguisher* still distinguishes Goppa matrices from corrected random matrices with high accuracy. The attention maps suggest that the remaining signal may involve interactions between rows rather than isolated rows. This motivates the search for another deterministic constraint, in the same spirit as the previous low-weight-row constraint, but applied to combinations of rows. We investigate this possibility in the next section through a XOR-based deterministic distinguisher.

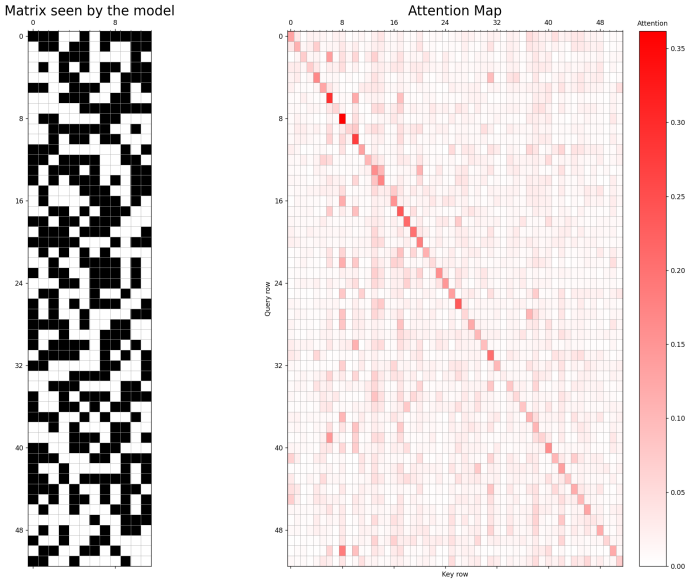

FIGURE 7. Attention map visualization after the 4th and last layer, for Goppa generator matrix of code $(n, m, t) = (64, 6, 2)$. Entries equal to 1 are represented in black, while entries equal to 0 are represented in white.

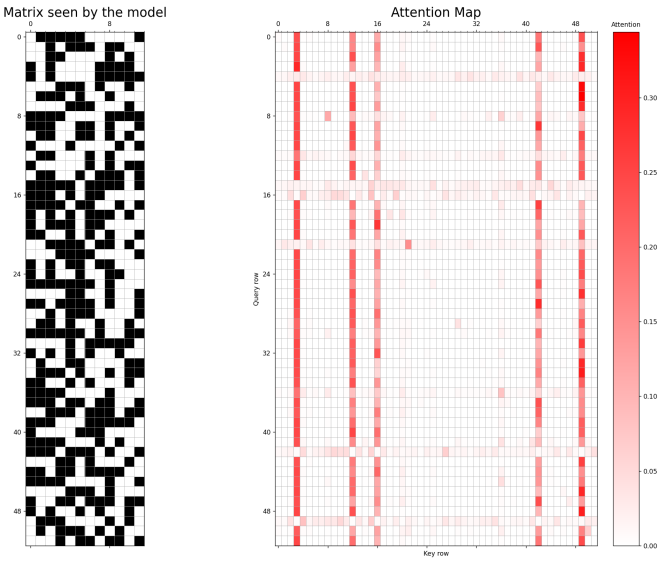

FIGURE 8. Attention map visualization after the 4th and last layer for a random binary generator matrix from the corrected distribution associated with a random linear $[n, k] = [64, 52]$ code. The code has the same length and dimension as the Goppa code with parameters $(n, m, t) = (64, 6, 2)$. Entries equal to 1 are represented in black, while entries equal to 0 are represented in white.

5. XOR-based deterministic distinguisher

5.1. **XOR rule definition.** Let $G = (I_k \mid A)$ be a generator matrix of a Goppa code. For any two distinct rows A_i and A_j of the non-systematic block A, we must have $w_H(A_i + A_j) \geqslant 2t - 1$. This motivates a simple deterministic XOR-based algorithm that we use as a "naive" distinguisher. Given the block A, compute all pairwise XORs $A_i + A_j$, for $i < j$, and predict "random" if at least one of these vectors has Hamming weight smaller than $2t - 1$. Otherwise, predict "Goppa".

For a matrix $A \in \mathcal{M}_{k\times mt}(\mathbb{F}_2)$, with $k = n-mt$, the XOR distinguisher computes the Hamming weight of $A_i + A_j$ for all pairs $i < j$. There are $\binom{k}{2}$ such pairs, and each XOR/Hamming-weight computation costs $O(mt)$. Hence, the naive time complexity is $\left(\binom{k}{2}mt\right) = O(k^2(n-k))$.

However, the relevant limitation of this distinguisher is not its runtime but its statistical power. For a uniformly random binary matrix A, the entries of two distinct rows A_i and A_j are independent Bernoulli variables of parameter $1/2$. Hence, for each coordinate ℓ, $(A_i + A_j)_\ell$ is equal to 1 with probability

$$\Pr[(A_i)_\ell \neq (A_j)_\ell] = \Pr[(A_i)_\ell = 1, (A_j)_\ell = 0] + \Pr[(A_i)_\ell = 0, (A_j)_\ell = 1] = \frac{1}{2}.$$

Thus $A_i + A_j$ is again distributed as a uniformly random binary vector of length mt, and its Hamming weight satisfies $w_H(A_i + A_j) \sim \mathrm{Bin}(mt, 1/2)$. Therefore, a lower bound on the probability that the XOR distinguisher detects at least one violation is $1-(\Pr[w_H(A_i + A_j) \geqslant 2t - 1])^{\binom{k}{2}}$. This expression gives a lower bound since the pairwise XORs $A_i + A_j$ are not mutually independent. For small parameters and high-rate codes, this probability can be close to one, which makes the rule highly informative. For larger cryptographic parameters, the violation event becomes

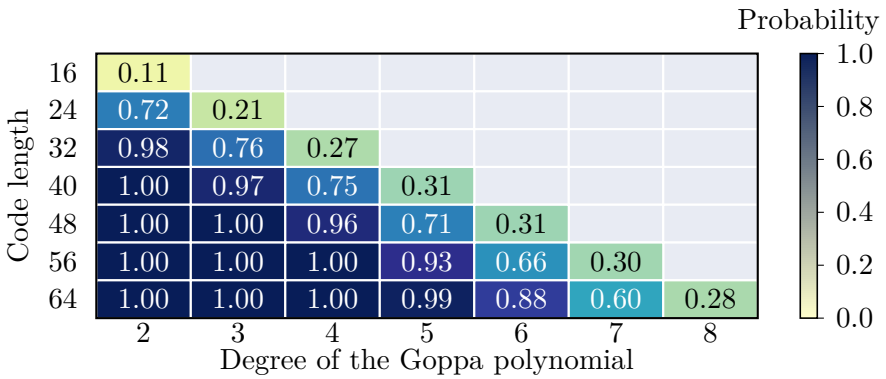

FIGURE 9. Probability that the XOR of at least one pair of row of a random $(n, n - 6t)$ matrix isn't of correct Hamming weight as a function of the code length n and the degree of the Goppa polynomial t.

rare, and the XOR rule is no longer expected to provide a useful distinguisher, even though it remains computationally cheap. We experimentally confirm this results by evaluating our XOR-distinguisher on a balanced dataset with no restriction on random samples.

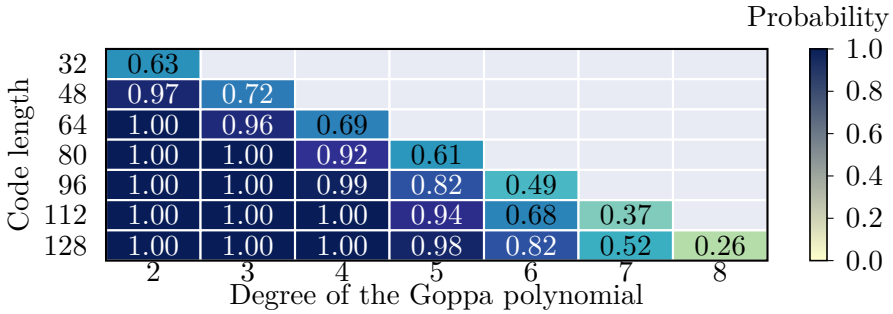

FIGURE 10. Probability that the XOR of at least one pair of row of a random $(n, n - 7t)$ matrix isn't of correct Hamming weight as a function of the code length n and the degree of the Goppa polynomial t.

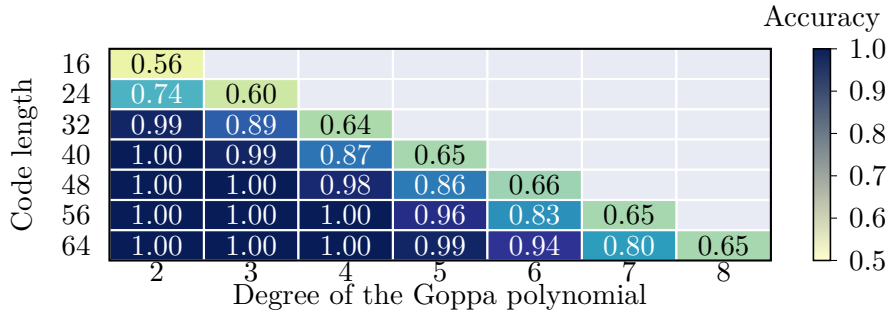

FIGURE 11. Classification accuracy of XOR-distinguisher on $(n, n - 6t)$ codes when computing all XOR-pairs as a function of code length n and the parameter t.

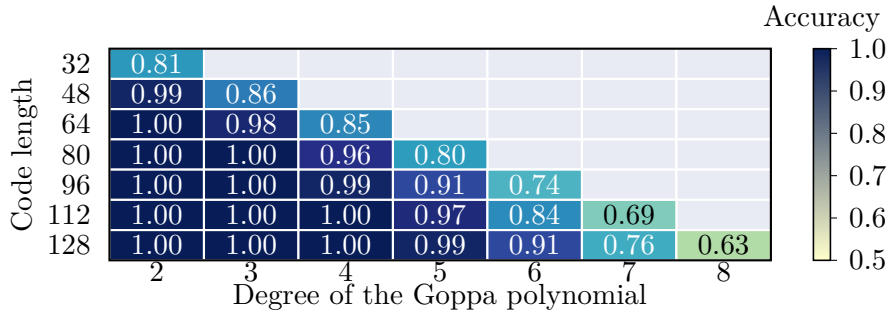

FIGURE 12. Classification accuracy of XOR-distinguisher on $(n, n - 7t)$ codes when computing all XOR-pairs as a function of code length n and the parameter t.

5.2. **Experimental results.** The results obtained by the XOR-based distinguisher are close to the accuracy predicted from the violation probability (see Fig. 9, 10, 11 and 12). Since the datasets are balanced, all Goppa matrices are correctly classified by construction, while a random matrix is correctly classified whenever at least one

XOR violation is detected. Hence,

$$\text{Accuracy(XOR)} \approx \frac{1 + P_{\text{viol}}}{2},$$

where P_{viol} is the probability that a random matrix contains at least one pair of rows satisfying $w_H(A_i + A_j) < 2t - 1$. This agreement supports the probabilistic estimate derived above for the appearance of pairwise XOR violations. The XOR-based distinguisher obtains slightly better results than *DeepDistinguisher*. Notably, for $m = 6$, we obtain an accuracy of about 80% for a code rate of approximately 0.35.

5.3. **Improvement.** We then tried improving the accuracy of the decision-rule by computing all XOR-pairs and XOR-triplets.

This improvement causes a significant increase in computational cost. Indeed, for a matrix $A \in \mathcal{M}_{k\times mt}(\mathbb{F}_2)$, with $(k = n - mt)$, the complexity of checking all two and three rows XORs is $O\Big(\big(\binom{k}{2} + \binom{k}{3}\big)mt\Big) = O(k^3(n-k))$.

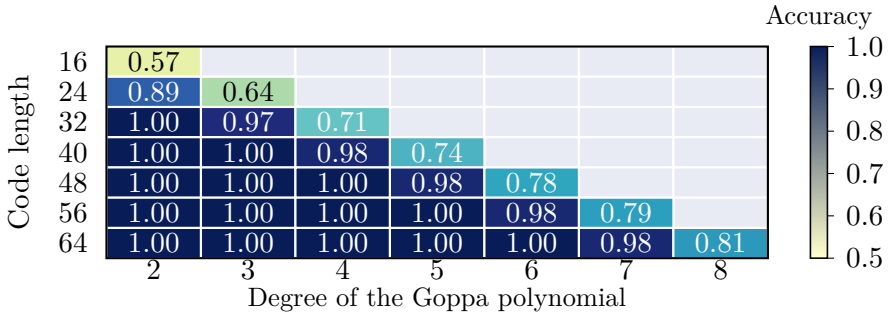

FIGURE 13. Classification accuracy of XOR-distinguisher for $(n, n - 6t)$ codes when computing all XOR-pairs and XOR-triplets, as a function of code length n and the parameter t.

For extension degree $m = 6$, our results are close to those of *DeepDistinguisher* for small values of t, especially $t = 2$ and $t = 3$ (see Fig. 13). For larger values of t, the XOR-based distinguisher gives significantly better results. For example, when

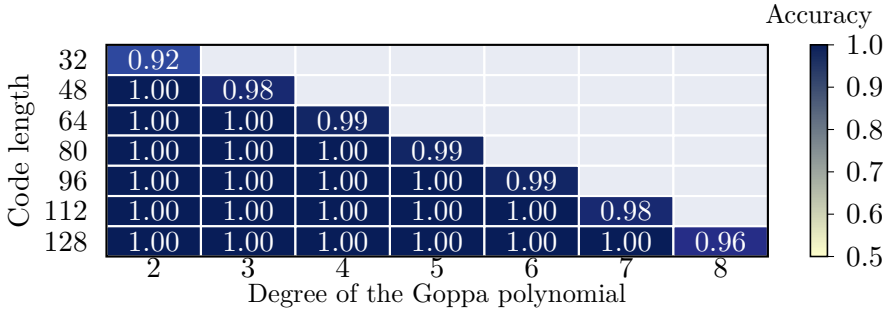

FIGURE 14. Classification accuracy of XOR-distinguisher for $(n, n - 7t)$ codes when computing all XOR-pairs and XOR-triplets, as a function of code length n and the parameter t.

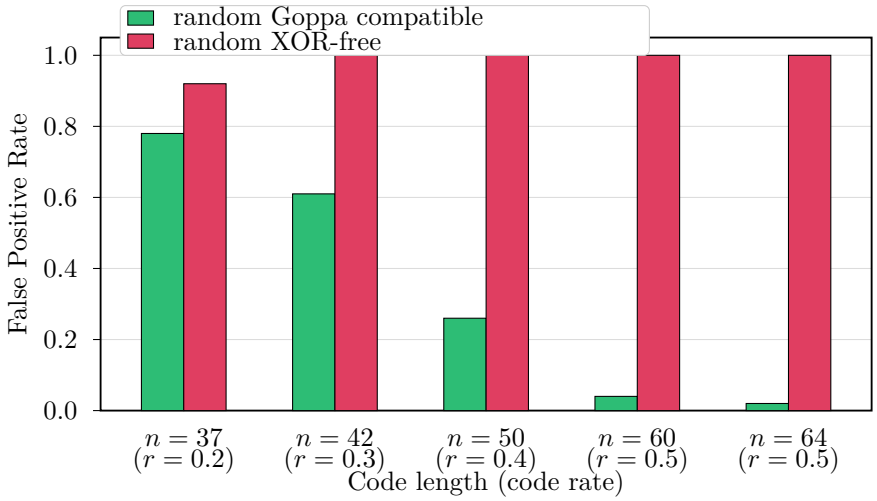

FIGURE 15. False positive rate for the prediction of *DeepDistinguisher*, for Goppa-compatible and XOR-free random distributions. Model is trained with 1M training samples from Goppa distribution and 1M from random distribution for sizes : [37,42,50,60,64]. A high value in this figure should not be interpreted as high accuracy.

$n = 64$ and $t = 8$, it reaches approximately 80% accuracy, compared with about 57% for *DeepDistinguisher* and this corresponds to a code rate of 0.25.

For extension degree $m = 7$, results of XOR-distinguisher are close to 100% for every considered parameters, with its worse prediction performances being 92% when $n = 32$ and $t = 2$, corresponding to a rate ≈ 0.56 (see Fig. 14). These high performances are expected when compared to the probabilty table.

5.4. **Behaviour comparison with *DeepDistinguisher*.** We call a random matrix *XOR-free* if it has been sampled under the following constraints: $\forall i, \quad w_H(A_i) \geq 2t$, and $\forall i, \ \forall j > i, \quad w_H(A_i + A_j) \geq 2t - 1$. The first condition guarantees that all the rows are Goppa-compatible (see Def. 7). The second condition removes the pairwise XOR artifact, since a full pairwise row sum has weight $2 + w_H(A_i + A_j)$.

To test whether the model has learned the XOR-based rule, we perform two experiments:

- we evaluate *DeepDistinguisher* trained on the original data distribution on *Goppa-compatible* random matrices,
- we evaluate *DeepDistinguisher* trained on the original data distribution on *XOR-free* random matrices, i.e. matrices constructed so that neither individual rows nor pairwise row XORs violate the corresponding Goppa minimum-distance constraint.

Figure 15 shows the ratio of false positives prediction of *DeepDistinguisher*. Since all samples are random, it indicates that the model incorrectly classifies many XOR-free random matrices as Goppa (in fact 100% of wrong classification for code rate greater than 0.3), which strongly supports the hypothesis that the removed XOR feature was used by the model. In Figures 16 and 17, we compare the agreement between *DeepDistinguiser* and our XOR-based distinguisher using for

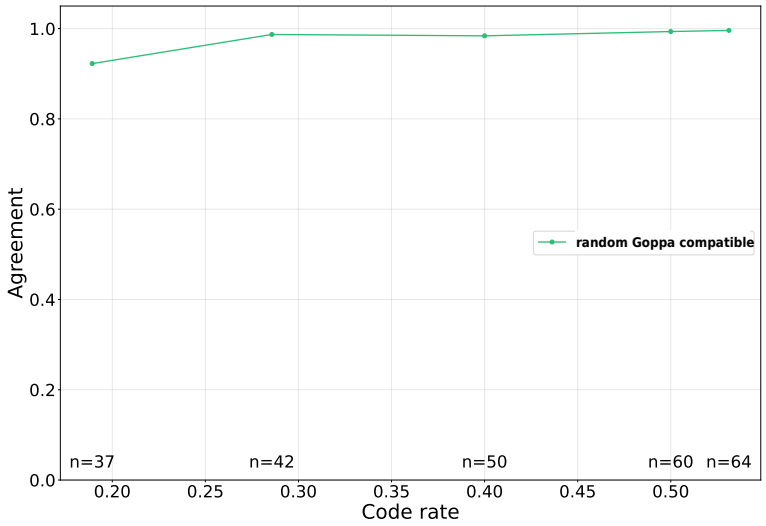

FIGURE 16. Agreement between *DeepDistinguisher* and XOR rule on random Goppa-compatible distribution. Computed on a balanced dataset with Goppa samples and Goppa-compatible random samples.

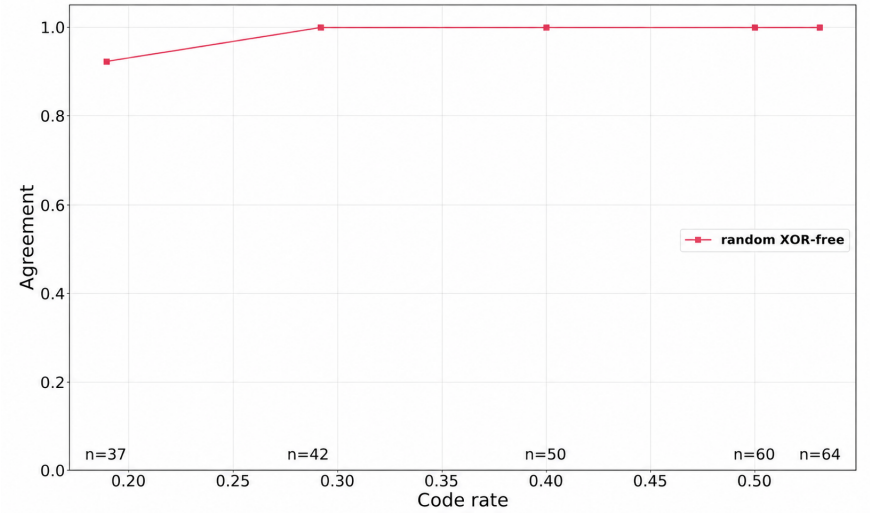

FIGURE 17. Agreement between *DeepDistinguisher* and XOR rule on XOR-free distribution. Computed on a balanced dataset with Goppa samples and random XOR-free samples.

random matrices a Goppa-compatible distribution and a XOR-free distribution. Since our XOR-based distinguisher only checks a constraint on the weight of the sum of any pair of rows, it always outputs a wrong classification for XOR-free random matrices. We observe a similar agreement pattern between *DeepDistinguisher* and our XOR-based distinguisher for both random distributions. The predictions of *DeepDistinguisher* and of the XOR-based distinguisher are highly correlated,

reaching almost 100% agreement for most code rate. The agreement decreases for lower-rate codes, suggesting that *DeepDistinguisher* may then rely on additional or more complex features not captured by the pairwise XOR rule. Thus, the XOR rule explains a significant part of the model's behavior for code rate greater than 0.3, but it does not fully characterize the learned distinguisher across all parameter ranges. To conclude, when evaluated on XOR-free random matrices, *DeepDistinguisher* loses most of its distinguishing power. Moreover, on several parameter sets, its predictions agree strongly with those of the XOR-based rule. This suggests that the model has learned a rule close to the pairwise XOR constraint.

We then tried to train *DeepDistinguisher* on the XOR-free distribution with the same hyperparameter on code parameters $n = 64$, $m = 6$ and $t = 5$. After 100 epoch, corresponding to 1.5M training steps, the training loss converges close to 0.005, while the test accuracy remains around 52%. This indicates severe overfitting: the model memorizes the training distribution but does not generalize to unseen XOR-free samples.

6. Conclusion

By applying gradient-based saliency and attention-map visualization, we identified a generation artifact in the random samples that the model was able to exploit to distinguish Goppa generator matrices from random ones. This artifact is related to the existence, in the random distribution, of rows whose Hamming weight is incompatible with the minimum-distance constraint of binary Goppa codes. However, this artifact does not fully explain the model's decision rule. Indeed, after evaluating the model on corrected random samples where this low-weight row artifact was removed, the model was still able to distinguish Goppa matrices from random matrices with high accuracy. This led us to investigate other features, in particular features involving interactions between rows.

Motivated by the minimum-distance constraint on sums of codewords, we introduced a simple XOR-based distinguisher. This distinguisher computes all pairwise XOR of rows of the non-systematic block A, and detects violations of the Hamming-weight constraint that should hold for binary Goppa codes. When we compared this rule with the predictions of *DeepDistinguisher*, we observed a high agreement for several parameter sets, especially in the high-rate regime.

Moreover, when this XOR-related feature was removed from the random samples by generating XOR-free random matrices, the model largely lost its ability to distinguish them from Goppa matrices. This suggests that the pairwise XOR constraint is an important feature used by the model in these parameter ranges. Nevertheless, the XOR rule does not fully characterize the learned distinguisher. For lower-rate codes, the agreement between the XOR-based rule and *DeepDistinguisher* slightly decreases. This indicates that *DeepDistinguisher* may rely on additional or more complex structural features that are not captured by the simple XOR rule. Thus, the XOR distinguisher should not be interpreted as a complete description of *DeepDistinguisher* behavior.

References

[1] M. Malhou, L. Perret, and K. Lauter, "AI for code-based cryptography," in *Selected Areas in Cryptography – SAC 2025*, ser. Lecture Notes in Computer Science, C. Boura, A. Mashatan, and A. Miri, Eds., vol. 16207. Cham: Springer, 2026, pp. 489–518.

[2] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *Jet Propulsion Laboratory DSN Progress Report*, vol. 4244, no. 1978, pp. 114–116, 1978.
[3] J.-C. Faugère, V. Gauthier-Umaña, A. Otmani, L. Perret, and J.-P. Tillich, "A distinguisher for high-rate McEliece cryptosystems," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6830–6844, 2013.
[4] A. Couvreur, R. Mora, and J.-P. Tillich, "A new approach based on quadratic forms to attack the McEliece cryptosystem," in *Advances in Cryptology – ASIACRYPT 2023*, ser. Lecture Notes in Computer Science, J. Guo and R. Steinfeld, Eds., vol. 14441. Singapore: Springer, 2023, pp. 3–38.
[5] H. Randriambololona, "The syzygy distinguisher," in *Advances in Cryptology – EUROCRYPT 2025*, ser. Lecture Notes in Computer Science, S. Fehr and P.-A. Fouque, Eds., vol. 15606. Cham: Springer, 2025, pp. 324–354.
[6] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Information Theory*, vol. 24, no. 3, pp. 384–386, 1978.
[7] E. Berlekamp, "Goppa codes," *IEEE Trans. Inf. Theor.*, vol. 19, no. 5, p. 590–592, Sep. 2006.
[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

Appendix A. Experimental setup

To make the experiments reproducible, we list below the hyperparameters that are kept fixed unless stated otherwise.

All experiments were performed with the following hyperparameters:

- Batch size(train / val) : 512/2000
- Optimizer : Adam with linear learning-rate warmup and L2 weight decay.
  - Learning rate : 1×10^{-5}
  - Warmup steps : 1000
  - Weight decay : 1×10^{-3}
- Gradient clipping norm : 5
- Encoder architecture : 4 layers, 4 heads, 1024-d embeddings
- Positional timescale : 40
- Precision : float16
- q : 2
- Dropout : 0
- Attention Dropout : 0
- Extension degree m : 6

The classification task is balanced: class 1 corresponds to Goppa matrices and class 0 to random matrices with the same parameters.

All models used in this article are trained on several code length values for one value of t and extension degree m is fixed at 6. More specifically, for a fixed value of t, the model is trained on all code lengths

$$n \in \{8t, 8t + 8, 8t + 16, \ldots, 64\}.$$

For example, a model trained on Goppa code of parameters $(n, m, t) = (64, 6, 4)$ is trained on generator matrices for code of length $\in [32, 40, 48, 56, 64]$. The number of samples is equal for every code length. The total dataset size is $8.4M$ samples for t $\in [2, 5]$ and $42M$ samples for $t > 5$. The total number of samples includes both classes. For each value of t, 5% of the samples are used for evaluation and are therefore excluded from the training set. The evaluation set is balanced and disjoint from the training set. Number of epochs performed for each parameters are listed in Table 1. The same setup is used for *DeepDistinguisher* models trained on the original random distribution and on the row-weight-corrected random distribution.

t	**Total samples**	**Epochs**
2	8.4M	2
3	8.4M	3
4	8.4M	4
5	8.4M	5
6	42M	5
7	42M	10

TABLE 1. Dataset size and number of epochs performed during training for each Goppa degree t.

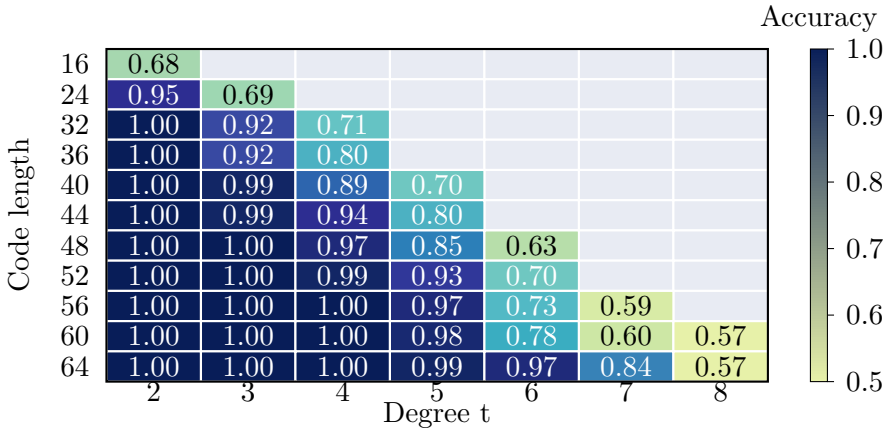

FIGURE 18. Heatmap of the classification accuracy on test sets for $q = 2$, $m = 6$ as a function of code length n and degree parameter t. In this experiment, one model is trained per value of t only. Meaning the model is trained on codes with varying lengths n, allowing the model to interpolate well to unseen code lengths. Reproduced from [1].

APPENDIX B. HEATMAPS

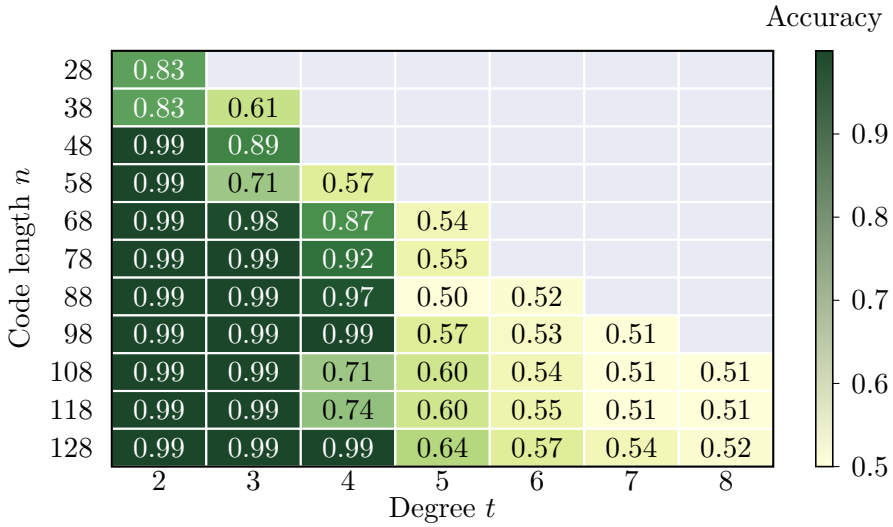

FIGURE 19. Heatmap of the classification accuracy on test sets for $q = 2$, $m = 7$ as a function of code length n and degree parameter t. In this experiment, one model is trained per value of t only. Meaning the model is trained on codes with varying lengths n, allowing the model to interpolate well to unseen code lengths. Reproduced from [1].

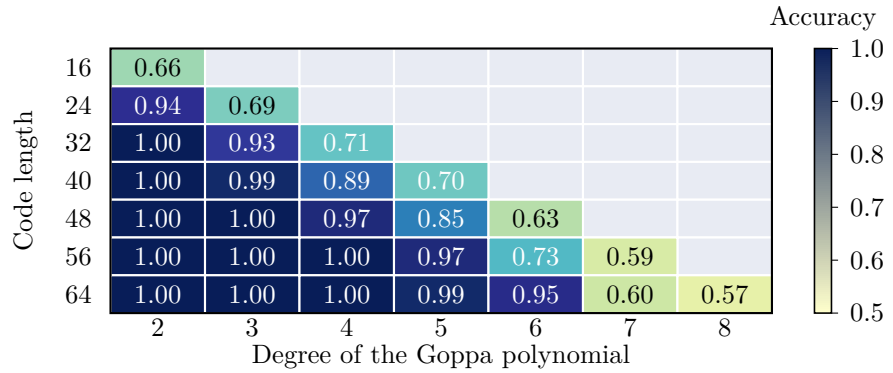

FIGURE 20. Heatmap of the classification accuracy on the test sets for $q = 2$ and $m = 6$ as a function of the code length n and the degree parameter t. In this experiment, a separate model is trained for each value of t using only the representation A (the redundancy matrix), rather than G (the full generator matrix).

CNRS,INRIA,LORIA, UNIVERSITÉ DE LORRAINE, FRANCE
Email address: quentin.lavit@loria.fr

LABORATOIRE LIS, UNIV TOULON, FRANCE
Email address: julien.seinturier@univ-tln.fr

LABORATOIRE IMATH, UNIV TOULON, FRANCE
Email address: pascal.veron@univ-tln.fr