SURF: A new code-based signature scheme [⋆]

Thomas Debris-Alazard[1,2], Nicolas Sendrier[2], and Jean-Pierre Tillich[2]

[1] Sorbonne Universités, UPMC Univ Paris 06
[2] Inria, Paris
{thomas.debris,nicolas.sendrier,jean-pierre.tillich}@inria.fr

Abstract. We present here a new code-based digital signature scheme. This scheme uses $(U, U+V)$ codes where both U and V are random. We show that the distribution of signatures is uniform by suitable rejection sampling. This is one of the key ingredients for our proof that the scheme achieves *existential unforgeability under adaptive chosen message attacks* (EUF-CMA) in the random oracle model (ROM) under two assumptions from coding theory, both strongly related to the hardness of decoding in a random linear code. Another crucial ingredient is the proof that the syndromes produced by $(U, U+V)$ codes are statistically indistinguishable from random syndromes. Note that these two key properties are also required for applying a recent and generic proof for code-based signature schemes in the QROM model [CD17]. As noticed there, this allows to instantiate the code family which is needed and yields a security proof of our scheme in the QROM. Our scheme also enjoys an efficient signature generation and verification. For a (classical) security of 128 bits, the signature size is less than one kilobyte. Contrarily to a current trend in code-based or lattice cryptography which reduces key sizes by using structured codes or lattices based on rings, we avoid this here and still get reasonable public key sizes (less than 2 megabytes for the aforementioned security level). Our key sizes compare favorably with TESLA-2, which is an (unstructured) lattice based signature scheme that has also a security reduction in the QROM model. This gives the first practical signature scheme based on binary codes which comes with a security proof and which scales well with the security parameter: for a security level of 2^λ, the signature size is of order $O(\lambda)$, public key size is of size $O(\lambda^2)$, signature generation cost is of order $O(\lambda^3)$, and signature verification cost is of order $O(\lambda^2)$.

Keywords: code-based cryptography, digital signature scheme, decoding algorithm, security proof.

1 Introduction

Code-based signature schemes. It is a long standing open problem to build an efficient and secure signature scheme based on the hardness of decoding a linear code which could compete in all respects with DSA or RSA. Such schemes could indeed give a quantum resistant signature for replacing in practice the aforementioned signature schemes that are well known to be broken by quantum computers. A first partial answer to this question was given in [CFS01]. It consisted in adapting the Niederreiter scheme [Nie86] for this purpose. This requires a linear code for which there exists an efficient decoding algorithm for a non-negligible set of inputs. This means that if $\mathbf{H}$ is an $r \times n$ parity-check matrix of the code, there exists for a non-negligible set of elements $\mathbf{s}$ in $\mathbb{F}_2^r$ an efficient way to find a word $\mathbf{e}$ in $\mathbb{F}_2^n$ of smallest Hamming weight such that $\mathbf{H}\mathbf{e}^T = \mathbf{s}^T$. In such a case, we say that $\mathbf{s}$, which is generally called a syndrome in the literature, can be decoded. To sign a message $\mathbf{m}$, a hash function h is used to produce a sequence $\mathbf{s}_0, \dots, \mathbf{s}_\ell$ of elements of $\mathbb{F}_2^r$. For instance $\mathbf{s}_0 = h(\mathbf{m})$ and $\mathbf{s}_i = h(\mathbf{s}_0, i)$ for $i > 0$. The first $\mathbf{s}_i$ that can be decoded defines the signature of $\mathbf{m}$ as the word $\mathbf{e}$ of smallest Hamming weight such that

$$\mathbf{H}\mathbf{e}^T = \mathbf{s}_i^T.$$

[⋆] This work was supported in part by the Commission of the European Communities through the Horizon 2020 program under project number 645622 PQCRYPTO.

The CFS signature scheme. The authors of [CFS01] noticed that very high rate Goppa codes are able to fulfill this task, and their scheme can indeed be considered as the first step towards a solution of the aforementioned problem. Moreover they gave a security proof of their scheme relying only on the assumption that two problems were hard, namely (i) decoding a generic linear code and (ii) distinguishing a Goppa code from a random linear code with the same parameters. However, afterwards it was realized that the parameters proposed in [CFS01] can be attacked by an unpublished attack of Bleichenbacher. The significant increase of parameters needed to thwart the Bleichenbacher attack was fixed by a slight variation [Fin10]. However, this modified scheme is not able to fix two other worrying drawbacks of the CFS scheme, namely

(i) a lack of security proof in light of the distinguisher of high rate Goppa codes found in [FGO+11] (see also [FGO+13] for more details) which shows that the hypotheses used in [CFS01] to give a security proof of the signature scheme were not met,
(ii) poor scaling of the parameters when security has to be increased. It can be readily seen that the complexity S of the best known attack scales only polynomially as $S \approx K^{t/2}$ where K is the key size in bits and t is some parameter that has to be kept very small (say smaller than 12 in practice), since the number of syndromes $\mathbf{s}_i$ that have to be computed before finding one that can be decoded is roughly $t!$.

Other code-based signature schemes. Other signature schemes based on codes were also given in the literature such as for instance the KKS scheme [KKS97, KKS05] or its variants [BMS11, GS12]. But they can be considered at best to be one-time signature schemes in the light of the attack given in [COV07] and great care has to be taken to choose the parameters of these schemes as shown by [OT11] which broke all the parameters proposed in [KKS97, KKS05, BMS11].

There has been some revival of the CFS strategy [CFS01], by choosing other code families. The new code families that were used are LDGM codes in [BBC+13], i.e. codes with a Low Density Generator Matrix, or (essentially) convolutional codes [GSJB14]. There are still some doubts that there is a way to choose the parameters of the scheme [GSJB14] in order to avoid the attack [LT13] on the McEliece cryptosystem based on convolutional codes [LJ12] and the LDGM scheme was broken in [PT16].

A last possibility is to use the Fiat-Shamir heuristic to turn a zero-knowledge authentication scheme into a signature scheme. When based on the Stern authentication scheme [Ste93] this gives a code-based signature scheme. However this approach leads to really large signature sizes (of the order of hundreds of thousands of bits). This represents a complete picture of code-based signature schemes based on the Hamming metric. There has been some recent progress in this area for another metric, namely the rank metric [GRSZ14] with the RankSign scheme. This scheme enjoys remarkably small key sizes, it is of order tens of thousands bits for 128 bits of security. It also comes with a partial security proof showing that signatures do not leak information when the number of available signatures is smaller than some bound depending on the code alphabet, but ensuring this condition represents a rather strong constraint on the parameters of RankSign. Moreover there is no overall reduction of the security to well identified problems in (rank metric) coding theory. Irrespective of the merits of this signature scheme, it is certainly desirable to also have a signature scheme for the Hamming metric due to the general faith in the hardness of decoding in it.

Moving from error-correcting codes to lossy source codes. It can be argued that the main problem with the CFS approach is to find a family of linear codes that are at the same time (i) indistinguishable from a random code and (ii) that have a non-negligible fraction of syndromes that can be decoded. There are not so many codes for which (ii) can be achieved and this is probably too much to ask for. However if we relax a little bit what we ask for the code, namely just a code such that the equation (in $\mathbf{e}$)

$$\mathbf{H}\mathbf{e}^T = \mathbf{s}^T \tag{1}$$

admits for most of the $\mathbf{s}$'s a solution $\mathbf{e}$ of small enough weight, then there are many more codes that are able to fulfill this task. This kind of codes are not used in error-correction but can be found in lossy source coding or source-distortion theory where the problem is to find codes with an associated decoding algorithm which can approximate *any* word of the ambient space by a close enough codeword. In the case of linear codes, this means a code and a associated decoding algorithm that can find for any syndrome $\mathbf{s}$ a vector $\mathbf{e}$ of small enough weight satisfying (1) where $\mathbf{H}$ is a parity-check matrix of the code.

Solving (1) is the basic problem upon which all code-based cryptography relies. This problem has been studied for a long time and despite many efforts on this issue [Pra62, Ste88, Dum91, Bar97, MMT11, BJMM12, MO15, DT17] the best algorithms for solving this problem [BJMM12, MO15] are exponential in the weight w of $\mathbf{e}$ as long as $w = (1 - \epsilon)r/2$ for any $\epsilon > 0$. Furthermore when w is sublinear in n, the exponent of the best known algorithms has not changed [CTS16] since the Prange algorithm [Pra62] dating back to the early sixties. Moreover, it seems very difficult to lower this exponent by a multiplicative factor smaller than $\frac{1}{2}$ in the quantum computation model as illustrated by [Ber10, KT17].

Our contribution: a new signature scheme based on $(U, U + V)$ codes. Convolutional codes, LDGM and polar codes come with a decoding algorithm which is polynomial for weights below $r/2$. They could theoretically be used in this context. However in the light of the key attacks [LJ12, PT16, BCD$^+$16] performed on related schemes, it seems very difficult to propose parameters which avoid those attacks. We are instead introducing a new class of codes in this context namely $(U, U + V)$ codes. A $(U, U + V)$ code is just a way of building a code of length n when we have two codes U and V of length $n/2$. It consists in

$$(U, U + V) \stackrel{\triangle}{=} \{(\mathbf{u}, \mathbf{u} + \mathbf{v}) : \mathbf{u} \in U, \mathbf{v} \in V\}.$$

Generalized $(U, U+V)$ codes have already been proposed in the cryptographic context for building a McEliece encryption scheme [MCT16a]. However, there it was suggested to take U and V to be codes that have an efficient decoding algorithm (this is mandatory in the encryption context). In the signature context, when we just need to find a small enough solution of (1) this is not needed. In our case, we can afford to choose *random* codes for U and V. It turns out that if we choose U and V random with the right choice of the dimension of U and V, then a suitable use of the Prange algorithm on the code U and the code V provides an advantage in this setting. It allows to solve (1) for weights w that are significantly below $r/2$, that is in the range of weights for which the best decoding algorithms are exponential.

Moreover, by tweaking a little bit the output of the Prange algorithm in our case and performing an appropriate rejection sampling, it turns out that the signatures are indistinguishable from a random word of weight w. Furthermore we also show that syndromes $\mathbf{He}^T$ associated to this kind of codes are statistically indistinguishable from random syndromes. These are the two key properties that allow to give a tight security proof of our signature scheme which relies only on two problems:

P1: Solving the decoding problem (1) when w is sufficiently below $r/2$
P2: Deciding whether a linear code is permuted $(U, U + V)$ code or not.

Interestingly enough some recent work [CD17] has shown that these two properties (namely statistical indistinguishability of the signatures and the syndromes associated to the code family chosen in the scheme) are also enough to obtain a tight security proof in the quantum random oracle model (QROM) for generic code-based signatures under the assumption that Problem P1 stays hard against a quantum computer and that the code family used is computationally indistinguishable from generic linear codes. In other words, as noticed in [CD17], this can be used to give a tight security proof of our $(U, U + V)$ codes in the QROM.

Problem P1 is the problem upon which all code-based cryptography relies. Here we are in a case where there are multiple solutions of (1) and the adversary may produce any number of instances of (1) with the same matrix $\mathbf{H}$ and various syndromes $\mathbf{s}$ and is interested in solving only one of them. This relates to the, so called, Decoding One Out of Many (DOOM) problem. This problem was first considered in [JJ02]. It was shown there how to modify slightly the known algorithms for decoding a linear code in order to solve this modified problem. This modification was later analyzed in [Sen11]. The parameters of the known algorithms for solving (1) can be easily adapted to this scenario where we have to decode simultaneously multiple instances which all have multiple solutions.

Problem P2 might seem at first sight to be an ad-hoc problem. However, even in the case when the permutation is restricted to leave globally stable the right and left part, detecting whether the resulting code is a permuted $(U, U+V)$-code is already an NP-complete problem (see Problem 4 and Theorem 3 in Subsection 7.5). Moreover, we are really in a situation where the resulting permuted $(U, U+V)$ code is actually very close to a random code. The only different behavior that can be found seems to be in the weight distribution for small weights. In this case, the permuted $(U, U+V)$ code has some codewords of a weight slightly smaller than the minimum distance of a random code of the same length and dimension. It is very tempting to conjecture that the best algorithms for solving Problem P2 come from detecting such codewords. This approach can be easily thwarted by choosing the parameters of the scheme in such a way that the best algorithms for solving this task are of prohibitive complexity. Notice that the best algorithms that we have for detecting such codewords are in essence precisely the generic algorithms for solving Problem P1. In some sense, it seems that we might rely on the very same problem, even if our proof technique does not show this.

All in all, this gives the first practical signature scheme based on binary codes which comes with a security proof and which scales well with the parameters: it can be shown that if one wants a security level of 2^λ, then signature size is of order $O(\lambda)$, public key size is of order $O(\lambda^2)$, signature generation is of order $O(\lambda^3)$, whereas signature verification is of order $O(\lambda^2)$. It should be noted that contrarily to the current thread of research in code-based or lattice-based cryptography which consists in relying on structured codes or lattices based on ring structures in order to decrease the key-sizes we did not follow this approach here. This allows for instance to rely on the NP-complete problem P1 which is generally believed to be hard on average rather that on decoding in quasi-cyclic codes for instance whose status is still unclear with a constant number of circulant blocks. Despite the fact that we did not use the standard approach for reducing the key sizes relying on quasi-cyclic codes for instance, we obtain acceptable key sizes (less than 2 megabytes for 128 bits of security) which compare very favorably to unstructured lattice-based signature schemes such as TESLA-2 for instance [ABB+17]. This is due in part to the tightness of our security reduction.

Organization of the paper. The paper is organized as follows, we present our scheme in §3, in §4 we prove it is secure under *existential unforgeability under an adaptive chosen message attack* (EUF-CMA) in the ROM, in relation with this proof we respectively examine in §5, §6, and §7, how to produce uniformly distributed signatures as well as the best message and key attacks. Finally we give some set of parameters on par with the security reduction and with the current state-of-the-art for decoding techniques.

2 Notation

We provide here some notation that will be used throughout the paper.

General notation. The notation $x \overset{\triangle}{=} y$ means that x is defined to be equal to y. We denote by $\mathbb{F}_2$ the finite field with 2 elements and by S_w the subset of $\mathbb{F}_2^n$ of words of weight w.

Vector notation. Vectors will be written with bold letters (such as $\mathbf{e}$) and uppercase bold letters are used to denote matrices (such as $\mathbf{H}$). Vectors are in row notation. Let $\mathbf{x}$ and $\mathbf{y}$ be two vectors, we will write $(\mathbf{x}, \mathbf{y})$ to denote their concatenation. For a vector $\mathbf{x} = (x_i)_{1 \leq i \leq n}$ and a permutation π of length n we denote by $\pi(\mathbf{x})$ the vector $(x_{\pi(i)})_{1 \leq i \leq n}$. We also denote for a subset I of positions of the vector $\mathbf{x} = (x_i)_{1 \le i \le n}$ by $\mathbf{x}_I$ the vector whose components are those of $\mathbf{x}$ which are indexed by I, i.e.

$$\mathbf{x}_I = (x_i)_{i \in I}.$$

We define the support of $\mathbf{x}$ as

$$\operatorname{Supp}(\mathbf{x}) \stackrel{\triangle}{=} \{i \in \{1, \cdots, n\} \text{ such that } x_i \neq 0\}$$

The Hamming weight of $\mathbf{x}$ is denoted by $|\mathbf{x}|$. By some abuse of notation, we will use the same notation to denote the size of a finite set: $|S|$ stands for the size of the finite set S. It will be clear from the context whether $|\mathbf{x}|$ means the Hamming weight or the size of a finite set. Note that

$$|\mathbf{x}| = |\operatorname{Supp}(\mathbf{x})|.$$

Probabilistic notation. Let S be a finite set, then $x \hookleftarrow S$ means that x is assigned to be a random element chosen uniformly at random in S. For a distribution $\mathcal{D}$ we write $\xi \sim \mathcal{D}$ to indicate that the random variable ξ is chosen according to $\mathcal{D}$. The uniform distribution on a certain discrete set is denoted by $\mathcal{U}$. The set will be specified in the text. We denote the uniform distribution on S_w by $\mathcal{U}_w$. When we have probability distributions $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$ over discrete sets $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$, we denote by $\mathcal{D}_1 \otimes \mathcal{D}_2 \otimes \cdots \otimes \mathcal{D}_n$ the product probability distribution, i.e $\mathcal{D}_1 \otimes \cdots \otimes \mathcal{D}_n(x_1, \ldots, x_n) \stackrel{\triangle}{=} \mathcal{D}_1(x_1) \ldots \mathcal{D}_n(x_n)$ for $(x_1, \ldots, x_n) \in \mathcal{E}_1 \times \cdots \times \mathcal{E}_n$. The n-th power product of a distribution $\mathcal{D}$ is denoted by $\mathcal{D}^{\otimes n}$, i.e. $\mathcal{D}^{\otimes n} \stackrel{\triangle}{=} \underbrace{\mathcal{D} \otimes \cdots \otimes \mathcal{D}}_{n \text{ times}}$.

Sometimes when we wish to emphasize on which probability space the probabilities or the expectations are taken, we denote by a subscript the random variable specifying the associated probability space over which the probabilities or expectations are taken. For instance the probability $\mathbb{P}_X(\mathcal{E})$ of the event $\mathcal{E}$ is taken over Ω the probability space over which the random variable X is defined, *i.e.* if X is for instance a real random variable, X is a function from a probability space Ω to $\mathbb{R}$, and the aforementioned probability is taken according to the probability chosen for Ω.

Coding theory. A binary linear code $\mathcal{C}$ of length n and dimension k is a subspace of $\mathbb{F}_2^n$ of dimension k and is usually defined by a parity-check matrix $\mathbf{H}$ of size $r \times n$ as

$$\mathcal{C} = \left\{\mathbf{x} \in \mathbb{F}_2^n : \mathbf{H}\mathbf{x}^T = \mathbf{0}\right\}.$$

When $\mathbf{H}$ is of full rank (which is usually the case) we have $r = n - k$. The rate of this code (that we denote by R) is defined as $R \stackrel{\triangle}{=} \frac{k}{n}$. In this case we say that $\mathcal{C}$ is a $[n, k]$-code.

3 The $(U, U+V)$-signature Scheme

3.1 The general scheme $\mathcal{S}_{\text{code}}$

Our scheme can be viewed as a probabilistic version of the full domain hash (FDH) signature scheme as defined in [BR96] which is similar to the probabilistic signature scheme introduced in [Cor02] except that we replace RSA with a trapdoor function based upon the hardness of Problem P1. Let $\mathcal{C}$ be a binary linear code of length n defined by a parity-check matrix $\mathbf{H}$. The one way function $f_{\mathbf{H},w}$ we consider is given by

$$\begin{aligned} f_{\mathbf{H},w} : S_w &\longrightarrow \mathbb{F}_2^{n-k} \\ \mathbf{e} &\longmapsto \mathbf{e}\mathbf{H}^T \end{aligned}$$

Inverting this function on an input $\mathbf{s}$ amounts to solve Problem P1. We are ready now to give the general scheme we consider. We assume that we have a family of codes which is defined by a set

$\mathcal{F}$ of parity-check matrices of size $(n-k)\times n$ such that for all $\mathbf{H}_{\text{sec}} \in \mathcal{F}$ we have an algorithm $D_{\mathbf{H}_{\text{sec}},w}$ which on input $\mathbf{s}$ computes $\mathbf{e} \in f^{-1}_{\mathbf{H}_{\text{sec}},w}(\mathbf{s})$. Then we pick uniformly at random $\mathbf{H}_{\text{sec}} \in \mathcal{F}$, an $n\times n$ permutation matrix $\mathbf{P}$, a non-singular matrix $\mathbf{S} \in \mathbb{F}_2^{(n-k)\times(n-k)}$ which define the secret and public key as:

$$sk \leftarrow (\mathbf{H}_{\text{sec}}, \mathbf{P}, \mathbf{S}) \ ; \ pk \leftarrow \mathbf{H}_{\text{pub}} \text{ where } \mathbf{H}_{\text{pub}} \overset{\triangle}{=} \mathbf{SHP}$$

Remark 1. Let $\mathcal{C}_{\text{sec}}$ be the code defined by $\mathbf{H}_{\text{sec}}$, then $\mathbf{H}_{\text{pub}}$ defines the following code:

$$\mathcal{C}_{\text{pub}} = \{\mathbf{cP} : \mathbf{c} \in \mathcal{C}_{\text{sec}}\}.$$

We also select a cryptographic hash function $h : \{0,1\}^* \to \mathbb{F}_2^{n-k}$ and a parameter λ_0 for the random salt $\mathbf{r}$. The algorithms $\texttt{Sgn}^{\text{sk}}$ and $\texttt{Vrfy}^{\text{pk}}$ are defined as follows

```
Sgn^sk(m):                              | Vrfy^pk(m, (e', r)):
  $\mathbf{r} \hookleftarrow \{0,1\}^{\lambda_0}$                   |   $\mathbf{s} \leftarrow h(\mathbf{m},\mathbf{r})$
  $\mathbf{s} \leftarrow h(\mathbf{m},\mathbf{r})$                      |   if $\mathbf{H}_{\text{pub}}\mathbf{e}'^T = \mathbf{s}^T$ and $|\mathbf{e}'| = w$ return 1
  $\mathbf{e} \leftarrow D_{\mathbf{H}_{\text{sec}},w}(\mathbf{S}^{-1}\mathbf{s}^T)$         |   else return 0
  return$(\mathbf{eP},\mathbf{r})$                    |
```

Remark 2. We add a salt in the scheme in order to have a tight security proof.

The correction of the verification step (i.e. that the pair $(\mathbf{eP},\mathbf{r})$ passes the verification step) follows from the fact that by definition of $D_{\mathbf{H}_{\text{sec}},w}(\mathbf{S}^{-1}\mathbf{s}^T)$ we have $\mathbf{H}_{\text{sec}}\mathbf{e}^T = \mathbf{S}^{-1}\mathbf{s}^T$. Therefore $\mathbf{H}_{\text{pub}}(\mathbf{eP})^T = (\mathbf{H}_{\text{pub}}\mathbf{P}^T)\mathbf{e}^T = \mathbf{SH}_{\text{sec}}\mathbf{e}^T = \mathbf{SS}^{-1}\mathbf{s}^T = \mathbf{s}^T$. We also have $|\mathbf{eP}| = |\mathbf{e}| = w$.

To summarize, a valid signature of a message $\mathbf{m}$ consists of a pair $(\mathbf{e},\mathbf{r})$ such that $\mathbf{H}_{\text{pub}}\mathbf{e}^T = h(\mathbf{m},\mathbf{r})^T$ with $\mathbf{e}$ of Hamming weight w.

3.2 Source-distortion codes and decoders

Source-distortion theory is a branch of information theory which deals with obtaining a family of codes, with an associated set of parity-check matrices $\mathbf{H} \in \mathcal{F}$, of the smallest possible dimension which can be used in our setting (*i.e.* for which we can invert $f_{\mathbf{H},w}$). Recall that a linear code is a vector space and the dimension of the code is defined as the dimension of this vector space. For a linear code specified by a full rank parity-check matrix of size $r \times n$, the dimension k of the code is equal to $n-r$. It is essential to have the smallest possible dimension in our cryptographic application, since this makes the associated problem P1 harder: the smaller $n-r$ is, the bigger r is and the further away w can be from $r/2$ (where solving P1 becomes easy). This kind of codes is used for performing lossy coding of a source. Indeed assume that we can perform this task, then this means that for every binary word $\mathbf{y}$, we compute $\mathbf{s}^T \overset{\triangle}{=} \mathbf{Hy}^T$, we find $\mathbf{e}$ of Hamming weight w such that $\mathbf{He}^T = \mathbf{s}^T$ which leads to deduce a codeword $\mathbf{c} \overset{\triangle}{=} \mathbf{y} - \mathbf{e}$ which is at distance w from $\mathbf{y}$. The word $\mathbf{y}$ is compressed with a compact description of $\mathbf{c}$. Since the dimension of the code is $n-r$ we just need $n-r$ bits to store a description of $\mathbf{c}$. We have replaced here $\mathbf{y}$ with a word which is not too far away from it. Of course, the smaller $n-r$ is, the smaller the compression rate $\frac{n-r}{n}$ is. There is some loss by replacing $\mathbf{y}$ by $\mathbf{c}$ since we are in general close to $\mathbf{y}$ but not equal to it.

In this way, finding a close codeword $\mathbf{c}$ of a given word $\mathbf{y}$ is equivalent to find for the syndrome $\mathbf{Hy}^T$ a low weight "error" $\mathbf{e}$ such that $\mathbf{He}^T = \mathbf{Hy}^T$. For our purpose it will be more convenient to adopt the error and syndrome viewpoint than the codeword viewpoint. To stress the similarity with error-correction we will call the function which associates to a syndrome $\mathbf{s}$ such an $\mathbf{e}$ a source-distortion decoder.

Definition 1 (Source Distortion Decoder). *Let n, $k \le n$ be integers and let $\mathcal{F}$ be a family of parity-check matrices (which define binary linear codes of length n and dimension k). A source distortion decoder for $\mathcal{F}$ is a probabilistic algorithm D:*

$$\begin{aligned} D : \mathcal{F} \times \mathbb{F}_2^{n-k} &\longrightarrow \mathbb{F}_2^n \\ (\mathbf{H},\mathbf{s}) &\longmapsto \mathbf{e} \end{aligned}$$

such that $\mathbf{H}\mathbf{e}^T = \mathbf{s}^T$*. When the weight of the error is fixed, we call it a decoder of fixed distortion* w *and we denote it by* D_w*. We say that the distortion* w *is achievable if there exists a family of codes with a decoder of fixed distortion* w*.*

This discussion raises a first question: for given n and k, what is the minimal distortion w which is achievable? We know from Shannon's rate-distortion theorem that the minimal w is given by the Gilbert-Varshamov bound $d_{\mathrm{GV}}(n,k)$ which follows:

Definition 2 (Gilbert-Varshamov's bound). *For given integers* n *and* k *such that* $k \leq n$*, the Gilbert-Varshamov bound* $d_{\mathrm{GV}}(n,k)$ *is given by:*

$$d_{\mathrm{GV}}(n,k) \overset{\triangle}{=} nh^{-1}\left(1 - k/n\right)$$

where h *denotes the binary entropy:* $h(x) = -x\log_2 x - (1-x)\log_2(1-x)$ *and* h^{-1} *its inverse defined on* $[0,1]$ *and whose range is* $[0, \frac{1}{2}]$.

Achieving distortion $w = (n-k)/2$ with the Prange technique. The study of random codes shows that they achieve the Gilbert-Varshamov source-distortion bound in average. Nevertheless we do not know for them an efficient source-distortion algorithm. However, as the following proposition shows, it is not the case when the distortion w is higher. When $w = (n-k)/2$ there is a very efficient decoder using the Prange technique [Pra62] for decoding. To explain it consider a parity-check matrix $\mathbf{H}$ which defines a linear code $\mathcal{C}$ of dimension k and length n. We want to find for a given $\mathbf{s} \in \mathbb{F}_2^{n-k}$ an error $\mathbf{e}$ of low weight such that $\mathbf{H}\mathbf{e}^T = \mathbf{s}^T$. $\mathbf{H}$ is a full-rank matrix and it therefore contains an invertible submatrix $\mathbf{A}$ of size $(n-k) \times (n-k)$. We choose a set of positions I of size $n-k$ for which $\mathbf{H}$ restricted to these positions is a full rank matrix. For simplicity assume that this matrix is in the first $n-k$ positions: $\mathbf{H} = \big(\mathbf{A}|\mathbf{B}\big)$. We look for an $\mathbf{e}$ of the form $\mathbf{e} = (\mathbf{e}', \mathbf{0})$ where $\mathbf{e}' \in \mathbb{F}_2^{n-k}$. We should therefore have $\mathbf{s}^T = \mathbf{H}\mathbf{e}^T = \mathbf{A}{\mathbf{e}'}^T$, that is ${\mathbf{e}'}^T = \mathbf{A}^{-1}\mathbf{s}^T$. The expected weight of $\mathbf{e}'$ is $\frac{n-k}{2}$ and it is easily verified that by randomly picking a random set I of size $n-k$ we have to check a polynomial number of them until finding an ${\mathbf{e}'}^T$ of weight exactly $(n-k)/2$.

Notation. We denote by $D^{\mathrm{Prange}}_{(n-k)/2}$ this fixed distortion decoder and by D^{Prange} the decoder which picks a random subset until finding one for which $\mathbf{H}$ restricted to the columns corresponding to I is invertible and computes $\mathbf{e}'$ as explained above. D^{Prange} does not necessarily output an error of weight $(n-k)/2$.

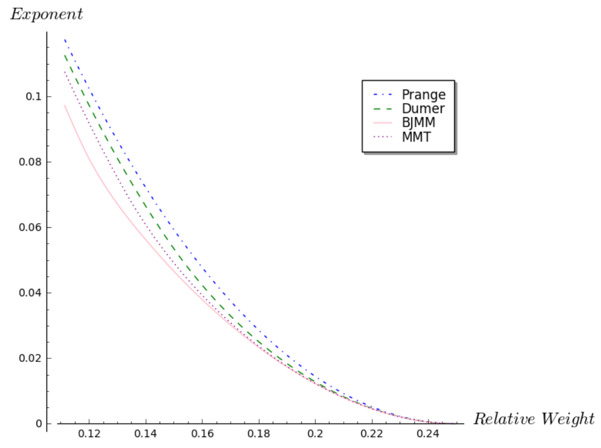
From the previous discussion we easily obtain

Proposition 1 (Generic Source Distortion Decoder).
The decoder $D^{\mathrm{Prange}}_{(n-k)/2}$ *runs in polynomial time on average over full rank* $(n-k) \times n$ *matrices.*

When we consider in general the family of random parity-check matrices (which define random linear codes) we speak about generic source-distortion decoders as there is no structure, except linearity of the code they define. In contrast to the distortion $(n-k)/2$, the only algorithms we know for linear codes for smaller values of w are all exponential in the distortion. This is illustrated by Figure 1 where we give the exponents (divided by the length n) of the complexity in base 2 as a function of the distance, for the fixed rate $R = k/n = 0.5$, of the best generic fixed-w source-distortion decoders. As we see, the normalized exponent is 0 for distortion $(n-k)/2$ and the difficulty increases as w approaches the Gilbert-Varshamov bound (which is equal approximately to $0.11n$ in this case).

Decoding Errors and Erasures Simultaneously. In the following problem, the word $\mathbf{x}$, more precisely its support, is called the *erasure pattern.*

Fig. 1. Normalized exponents in base 2 of the best generic fixed-w source distortion decoders.



Problem 1 (Decoding Errors and Erasures).
Instance: $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_2^{n-k}$, $\mathbf{x} \in \mathbb{F}_2^n$, ν integer
Output: $\mathbf{e} \in \mathbb{F}_2^n$ such that $\mathbf{H}\mathbf{e}^T = \mathbf{s}^T$ and $|\mathrm{Supp}(\mathbf{e}) \setminus \mathrm{Supp}(\mathbf{x})| = \nu$

In fact, the weight of the solution $\mathbf{e}$ is constrained outside of the erasure pattern $\mathbf{x}$. Within the erasure pattern the coordinates of $\mathbf{e}$ can take any value. For the sake of simplicity, we will overload the notation and denote $D_\nu(\mathbf{H}, \mathbf{s}, \mathbf{x})$ a solution of the above problem whereas $D_\nu(\mathbf{H}, \mathbf{s})$ denotes the (erasure-less) decoding of ν errors. The problem of erasure decoding appears very naturally in coding theory, including in source-distortion problem. We may reduce the error and erasure decoding to an error only decoding in a smaller code.

Proposition 2. *Let* $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ *and* $\mathbf{x} \in \mathbb{F}_2^n$ *be such that the* $|\mathbf{x}| = \rho$ *columns of* $\mathbf{H}$ *indexed by* $\mathrm{Supp}(\mathbf{x})$ *are independent. For any* $\mathbf{s} \in \mathbb{F}_2^{n-k}$ *we can derive* $\mathbf{e} = D_\nu(\mathbf{H}, \mathbf{s}, \mathbf{x})$ *from* $\mathbf{e}'' = D_\nu(\mathbf{H}'', \mathbf{s}'')$ *in polynomial time where*

(i) $\mathbf{H}'' \in \mathbb{F}_2^{(n-k-\rho)\times(n-\rho)}$ *can be derived in polynomial time from* $\mathbf{H}$ *and* $\mathbf{x}$*,*
(ii) $\mathbf{s}'' \in \mathbb{F}_2^{n-k-\rho}$ *can be derived in polynomial time from* $\mathbf{H}$*,* $\mathbf{x}$*, and* $\mathbf{s}$*.*

Proof. Without loss of generality, we assume that the '1's in $\mathbf{x}$ come first, $\mathbf{x} = (1\cdots 1, 0\cdots 0)$. A Gaussian elimination on $\mathbf{H}$ using the first ρ positions as pivots yields

$$\mathbf{S}\mathbf{H} = \left(\begin{array}{c|c} \mathbf{I}_\rho & \mathbf{H}' \\ \hline \mathbf{0} & \mathbf{H}'' \end{array}\right)$$

for some non-singular matrix $\mathbf{S}$. Let $(\mathbf{s}', \mathbf{s}'') = \mathbf{s}\mathbf{S}^T$ with $\mathbf{s}' \in \mathbb{F}_2^\rho$ and $\mathbf{s}'' \in \mathbb{F}_2^{n-k-\rho}$, $\mathbf{e}'' = D_\nu(\mathbf{H}'', \mathbf{s}'')$, $\mathbf{e}' = \mathbf{s}' + \mathbf{e}''\mathbf{H}'^T$, and $\mathbf{e} = (\mathbf{e}', \mathbf{e}'')$. We easily check that $\mathbf{H}\mathbf{e}^T = \mathbf{s}^T$ and $|\,\mathrm{Supp}(\mathbf{e}) \setminus \mathrm{Supp}(\mathbf{x})| = |\,\mathrm{Supp}(\mathbf{e}'')| = \nu$, thus $\mathbf{e} = D_\nu(\mathbf{H}, \mathbf{s}, \mathbf{x})$. All operations, except possibly the call to D_ν, are polynomial time. □

The reduction of the above proposition applies to D^{Prange}. Given $(\mathbf{H}, \mathbf{s}, \mathbf{x})$ and using the notation of the proof, we set $\mathbf{e}'' = D^{\mathrm{Prange}}(\mathbf{H}'', \mathbf{s}'')$ and we denote $\mathbf{e} = (\mathbf{e}', \mathbf{e}'') = D^{\mathrm{Prange}}(\mathbf{H}, \mathbf{s}, \mathbf{x})$ the corresponding error. With fixed distortion we have $\mathbf{e}'' = D^{\mathrm{Prange}}_{(n-k-|\mathbf{x}|)/2}(\mathbf{H}'', \mathbf{s}'')$ and we denote $\mathbf{e} = D^{\mathrm{Prange}}_{(n-k-|\mathbf{x}|)/2}(\mathbf{H}, \mathbf{s}, \mathbf{x})$.

Finally, let us point out that if $\mathbf{H}$ is the parity check matrix of a binary linear $[n, k]$-code $\mathcal{C}$, the matrix $\mathbf{H}''$ that appears in Proposition 2 is the parity-check matrix of the punctured code in $I = \mathrm{Supp}(\mathbf{x})$ as defined below:

Definition 3 (Punctured code). *Consider a code $\mathcal{C}$ of length n. The punctured code* $\mathrm{Punc}_I(\mathcal{C})$ *in a set of positions $I \subset \{1,\dots,n\}$ is a code of length $n-|I|$ defined as*

$$\underset{I}{\mathrm{Punc}}(\mathcal{C}) \overset{\triangle}{=} \{\mathbf{c}_{\bar{I}} : \mathbf{c} \in \mathcal{C}\}$$

where $\bar{I} = \{1,\dots,n\} \setminus I$.

Therefore, what Proposition 2 says is that decoding ν errors and ρ erasures in an $[n,k]$-code is essentially the same thing as decoding ν errors in an $[n-\rho,k]$-code.

3.3 The $(U, U+V)$ Code Family and Its Decoding

Source-distortion theory has found over the years several families of codes with an efficient source-distortion algorithm which achieves asymptotically the Gilbert-Varshamov source-distortion bound, one of the most prominent ones being probably the Arikan polar codes [Arı09] (see [Kor09]). The naive way would be to build our signature on such a code-family and hoping that permuting the code positions and publishing a random parity-check matrix of the permuted code would destroy all the structure used for decoding. All known families of codes used in this context have low weight codewords and this can be used to mount an attack. We will proceed differently here and introduce in this setting the $(U, U+V)$ codes mentioned in the introduction. The point is that they (i) have very little structure, (ii) have a very simple source-distortion decoder which is more powerful than the generic source decoder, (iii) they do not suffer from low weight codewords as was the case with the aforementioned families. It will be useful to recall here that

Definition 4 ($(U, U+V)$-Codes). *Let U, V be linear binary codes of length $n/2$ and dimension k_U, k_V. We define the subset of $\mathbb{F}_2^n$:*

$$(U, U+V) \overset{\triangle}{=} \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) \text{ such that } \mathbf{u} \in U \text{ and } \mathbf{v} \in V\}$$

which is a linear code of length n and dimension $k = k_U + k_V$. The resulting code is of minimum distance $\min(2d_U, d_V)$ where d_U is the minimum distance of U and d_V is the minimum distance of V. A parity-check matrix of such a code is given by

$$\begin{pmatrix} \mathbf{H}_U & \mathbf{0} \\ \mathbf{H}_V & \mathbf{H}_V \end{pmatrix}$$

where $\mathbf{H}_U \in \mathbb{F}_2^{(n/2-k_U)\times n/2}$ (resp. $\mathbf{H}_V \in \mathbb{F}_2^{(n/2-k_V)\times n/2}$) is a parity-check matrix of U (resp. V).

We are now going to present a source-distortion for a $(U, U+V)$ code.

We can use the generic source-distortion decoder of Proposition 1 for source distortion decoding a $(U, U+V)$ code. Assume that we have a $(U, U+V)$ code of length n of parity-check matrix $\mathbf{H}_{\text{sec}} \overset{\triangle}{=} \begin{pmatrix} \mathbf{H}_U & \mathbf{0} \\ \mathbf{H}_V & \mathbf{H}_V \end{pmatrix}$ where $\mathbf{H}_U, \mathbf{H}_V$ are random and a syndrome $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2) \in \mathbb{F}_2^{n/2-k_U} \times \mathbb{F}_2^{n/2-k_V}$ that we want to decode. Let us first remark that, for $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2) \in \mathbb{F}_2^{n/2} \times \mathbb{F}_2^{n/2}$,

$$\mathbf{H}_{\text{sec}}\mathbf{e}^T = \begin{pmatrix} \mathbf{H}_U\mathbf{e}_1^T \\ \mathbf{H}_V(\mathbf{e}_1+\mathbf{e}_2)^T \end{pmatrix} = \begin{pmatrix} \mathbf{s}_1^T \\ \mathbf{s}_2^T \end{pmatrix} \iff \mathbf{H}_U\mathbf{e}_1^T = \mathbf{s}_1^T \text{ and } \mathbf{H}_V(\mathbf{e}_1+\mathbf{e}_2)^T = \mathbf{s}_2^T$$

In this way, we first decode $\mathbf{s}_2$ in V to find $\mathbf{e}_V \overset{\triangle}{=} \mathbf{e}_1 + \mathbf{e}_2$. That is $\mathbf{e}_V = D^{\text{Prange}}_{(n/2-k_V)/2}(\mathbf{H}_V, \mathbf{s}_2)$ with Prange's polynomial time fixed distortion algorithm. We next decode $\mathbf{s}_1$ in U using $\mathbf{e}_V$ as an erasure pattern. The idea here is that $\mathbf{e}_V$ covers a large part of $\mathbf{e}_1$ leaving us with an error which is, hopefully, easier to find. We compute $\mathbf{e}_U = D^{\text{Prange}}_{(n/2-k_U-|\mathbf{e}_V|)/2}(\mathbf{H}_U, \mathbf{s}_1, \mathbf{e}_V)$ with Prange's polynomial time fixed distortion algorithm. We claim that $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2) = (\mathbf{e}_U, \mathbf{e}_V + \mathbf{e}_U)$ verifies $\mathbf{H}_{\text{sec}}\mathbf{e}^T = \mathbf{s}^T$ and has weight $n/2 - k_U$. The procedure is described in Algorithm 1.

Algorithm 1 UV–**sddV**1 : $(U, U+V)$–**Source Distortion Decoder**

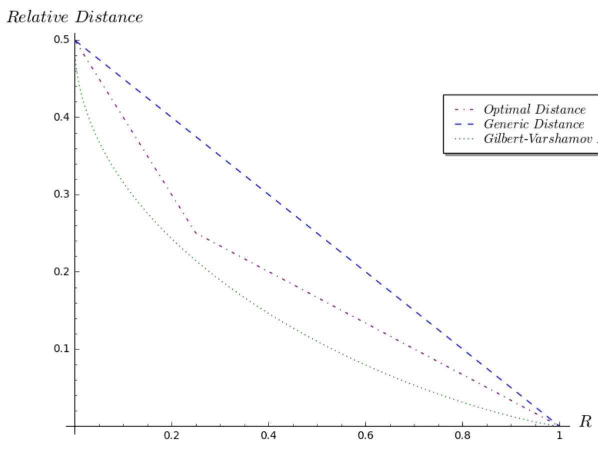
```
Parameter: a $(U,U+V)$ code of length $n$ and dimension $k = k_U + k_V$
Input: $(\mathbf{s}_1, \mathbf{s}_2)$ with $\mathbf{s}_1 \in \mathbb{F}_2^{n/2-k_U}$, $\mathbf{s}_2 \in \mathbb{F}_2^{n/2-k_V}$
Output: $\mathbf{e} \in \mathbb{F}_2^n$
Assumes: $2k_U - k_V \leq n/2$.
1: $\mathbf{e}_V \leftarrow D^{\text{Prange}}_{(n/2-k_V)/2}(\mathbf{H}_V, \mathbf{s}_2)$
2: $\nu \leftarrow (n/2 - k_U - |\mathbf{e}_V|)/2$
3: $\mathbf{e}_U \leftarrow D^{\text{Prange}}_{\nu}(\mathbf{H}_U, \mathbf{s}_1, \mathbf{e}_V)$
4: return $(\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V)$
```

Proposition 3. *The algorithm* UV-**sddV**1 *is a fixed-*$(n/2 - k_U)$ *source-distortion decoder which works in polynomial average-time when* $2k_U - k_V \leq n/2$.

Proof. First remark that both calls to D^{Prange} are made for a distortion level that is achieved in polynomial time. It only remains to prove that the output has the expected weight. We have $|\mathbf{e}_V| = (n/2 - k_V)/2$. The word $\mathbf{e}_U$ splits in two disjoint parts $\mathbf{e}'_U$ whose support is $\text{Supp}(\mathbf{e}_U) \cap \text{Supp}(\mathbf{e}_V)$ and $\mathbf{e}''_U$ whose support is $\text{Supp}(\mathbf{e}_U) \setminus \text{Supp}(\mathbf{e}_V)$. By construction, the second call to Prange corrects exactly $\nu \stackrel{\triangle}{=} (n/2 - k_U - |\mathbf{e}_V|)/2$ errors, this is also the weight of $\mathbf{e}''_U$. Finally we can write $\mathbf{e}_1 = \mathbf{e}_U = \mathbf{e}'_U + \mathbf{e}''_U$ and $\mathbf{e}_2 = \mathbf{e}_U + \mathbf{e}_V = (\mathbf{e}'_U + \mathbf{e}_V) + \mathbf{e}''_U$ with $\text{Supp}(\mathbf{e}'_U) \subset \text{Supp}(\mathbf{e}_V)$ and $\text{Supp}(\mathbf{e}''_U) \cap \text{Supp}(\mathbf{e}_V) = \emptyset$. We derive that $|\mathbf{e}_1| = |\mathbf{e}'_U| + |\mathbf{e}''_U|$, $|\mathbf{e}_2| = |\mathbf{e}'_U + \mathbf{e}_V| + |\mathbf{e}''_U|$, and $|\mathbf{e}'_U + \mathbf{e}_V| + |\mathbf{e}'_U| = |\mathbf{e}_V|$. And finally $|\mathbf{e}| = |\mathbf{e}_1| + |\mathbf{e}_2| = |\mathbf{e}_V| + 2\nu = n/2 - k_U$. □

We can now choose the parameters k_U and k_V in order to minimize the distortion $n/2 - k_U$ for a fixed dimension $k = k_U + k_V$ of the code. Figure 2 compares the distance that we obtain with this algorithm to $(n-k)/2$ which corresponds to what is achieved by the generic decoder and to the optimal distance (Gilbert-Varshamov bound) where R denotes the rate of the code. As we see there is a non-negligible gain. Nevertheless, UV-**sddV**1 approximates to a fixed distance in each step of its execution which leads to correlations between some bits that can be used to recover the structure of the secret key. In order to fix this problem and as it is asked in our proof of security, we will present a modified version of UV-**sddV**1 in §5 which uses a rejection sampling method to simulate uniform outputs. This comes at the price of slightly increasing the weight of the error output by the decoder.

Fig. 2. Comparison of the Optimal Signature Distance, the Gilbert-Varshamov Bound and Generic Distance



3.4 $(U, U+V)$ codes and cryptography

It is not the first time that $(U, U+V)$ codes are suggested for a cryptographic use. This was already considered for constructing a McEliece cryptosystem in [KKS05, p.225-228] or more recently in [PMIB17] (it is namely a particular case of a generalized concatenated code). However both papers did not consider the improvement in the error correction performance that comes with the $(U, U+V)$-construction if a decoder that uses soft information is used. For instance [KKS05] studies only the case of hard-decision decoder for Goppa codes and concludes that the obtained code has a worse error correction capability than the original Goppa code and therefore worse public-key sizes. This is actually a situation which really depends on the code family (call it $\mathcal{F}$) and its decoder. If $\mathcal{F}$ is the family of (generalized) Reed-Solomon codes with the Koetter-Vardy decoder which is able to cope with soft information on the symbols, then the results go the other way round (at least in a certain range of rates) [MCT16b]. In this case, a $(U, U+V)$ code based on generalized Reed-Solomon codes, has for certain rates better error-correction capacity when decoded with the Koetter-Vardy decoder than a generalized Reed-Solomon of the same rate decoded with the same decoder. This allows in principle to decrease the public key-size. Our signature scheme where $\mathcal{F}$ is the family of linear codes and the associated (source-distortion) decoder is the Prange decoder is another example of this kind.

In order to understand what soft-decoding has to do in this setting, it is helpful to recall a few points from coding theory. What we are going to review here is how a $(U, U+V)$ code is decoded in the polar code construction [Arı09] or in the case of Reed-Muller codes [DS06]. Consider a binary linear code of length n defined by a parity-check matrix $\mathbf{H}$. In hard decoding, we aim at recovering the error $\mathbf{e}$ of minimum weight satisfying a given syndrome $\mathbf{H}\mathbf{e}^T = \mathbf{s}^T$. In soft decoding, we know all probabilities $\mathbb{P}(e_i = 1)$ and want to find the error which maximizes $\mathbb{P}(\mathbf{e}|\mathbf{H}\mathbf{e}^T = \mathbf{s}^T)$. Both decoding perform the same task in the case of a binary symmetric channel of crossover probability $p \leq \frac{1}{2}$ (this means that $\mathbb{P}(e_i = 1) = p$ for all i).

It turns out that soft-decoding is the natural scenario for decoding a $(U, U+V)$ code when we decode the V component first and then the U component, even if one wants to perform hard decoding of the whole $(U, U+V)$ code. This really amounts to find the error of minimum weight $\mathbf{e} = (\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V)$ such that

$$\begin{pmatrix} \mathbf{H}_U & 0 \\ \mathbf{H}_V & \mathbf{H}_V \end{pmatrix} \begin{pmatrix} \mathbf{e}_U^T \\ \mathbf{e}_U^T + \mathbf{e}_V^T \end{pmatrix} = \begin{pmatrix} \mathbf{s}_1^T \\ \mathbf{s}_2^T \end{pmatrix}. \tag{2}$$

We will assume that the error model is a binary symmetric channel of crossover probability p, meaning that

$$\mathbb{P}(\mathbf{e}_U(i) = 1) = p \tag{3}$$
$$\mathbb{P}(\mathbf{e}_U(i) + \mathbf{e}_V(i) = 1) = p \tag{4}$$

where $\mathbf{e}_U(i)$ and $\mathbf{e}_V(i)$ denote the i-th coordinate of $\mathbf{e}_U$ and $\mathbf{e}_V$ respectively. Recall that hard decoding $\mathbf{e}$ really amounts to soft-information decoding $\mathbf{e}$ with respect to this error model. Decoding can now be done through the following steps.

Step 1. We observe that (2) implies that $\mathbf{H}_V\mathbf{e}_V^T = \mathbf{s}_2^T$. Recovering the V component amounts here to hard decode $\mathbf{e}_V$. The rationale behind this is that the channel model of $\mathbf{e}_V(i)$ is a binary symmetric channel of crossover probability $2p(1-p)$. This can be verified by observing that $\mathbf{e}_V = \mathbf{e}_U + (\mathbf{e}_U + \mathbf{e}_V)$ with $\mathbf{e}_U$ and $\mathbf{e}_U + \mathbf{e}_V$ being independent random variables whose components are i.i.d. with probability distributions given by (3) and (4). From this we deduce that the components $\mathbf{e}_V(i)$ are i.i.d. with $\mathbb{P}(\mathbf{e}_V(i) = 1) = 2p(1-p)$. Let us now assume that we have decoded $\mathbf{e}_V$ correctly.

Step 2. Recovering $\mathbf{e}_U$ can in principle be done in two different ways. The first one uses (2) directly from which we deduce

$$\mathbf{H}_U\mathbf{e}_U^T = \mathbf{s}_1^T. \tag{5}$$

Now that we know $\mathbf{e}_V$ we could also notice that

$$\mathbf{H}_U(\mathbf{e}_U + \mathbf{e}_V)^T = \mathbf{H}_U\mathbf{e}_U^T + \mathbf{H}_U\mathbf{e}_V^T = \mathbf{s}_1^T + \mathbf{H}_U\mathbf{e}_V^T \tag{6}$$

and we know here the right-hand term. There are therefore two ways to recover $\mathbf{e}_U$

Method 1 We perform hard decoding of the syndrome $\mathbf{s}_1^T$ and find the $\mathbf{e}_U$ of minimum weight satisfying (5).

Method 2 We perform hard decoding of the syndrome $\mathbf{s}_1^T + \mathbf{H}_U\mathbf{e}_V^T$ by finding the vector $\mathbf{x}$ of minimum weight satisfying $\mathbf{H}_U\mathbf{x}^T = \mathbf{s}_1^T + \mathbf{H}_U\mathbf{e}_V^T$ and let $\mathbf{e}_U = \mathbf{x} + \mathbf{e}_V$.

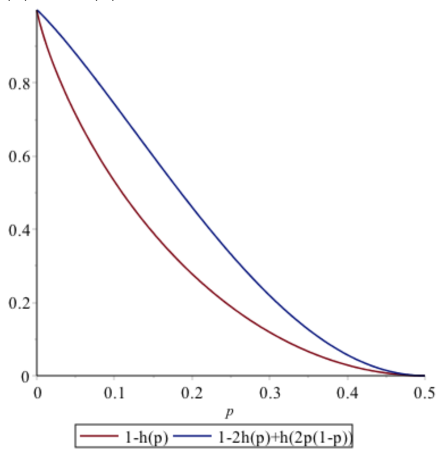
In [KKS05] it is suggested to perform both decodings and to choose for computing $\mathbf{e}_U$ the decoding which gives the smallest error weight (the decoding is not explained in terms of syndromes there, but expressing their decoding in terms of syndrome decoding amounts to the decision rule that we have just given). It is clear that some amount of information is lost during this process. This can be seen by noticing that once we know $\mathbf{e}_V$, we have a much finer knowledge on $\mathbf{e}_U$. It is readily seen that we can now use $\mathbb{P}(\mathbf{e}_U(i) = 1|\mathbf{e}_V(i))$ instead of $\mathbb{P}(\mathbf{e}_U(i) = 1)$. This calculation follows from the fact that $\mathbf{e}_U(i)$ and $\mathbf{e}_U(i) + \mathbf{e}_V(i)$ are independent and we know the distribution of these two random variables. A straightforward calculation leads to

$$\mathbb{P}(\mathbf{e}_U(i) = 1|\mathbf{e}_V(i) = 0) = \frac{p^2}{(1-p)^2 + p^2} \tag{7}$$

$$\mathbb{P}(\mathbf{e}_U(i) = 1|\mathbf{e}_V(i) = 1) = \frac{1}{2}. \tag{8}$$

In other words, when $\mathbf{e}_V(i) = 0$, we can view $\mathbf{e}_U(i)$ as an error originating from a binary symmetric channel of crossover probability $\frac{p^2}{(1-p)^2+p^2}$ (which is much smaller than p) and when $\mathbf{e}_V(i) = 1$ we may consider that the position has just been erased. A decoder for U which uses this soft information has potentially much better performances than the previous hard decoder. In fact, in this case we just need a decoder which decodes errors and erasures. When the alphabet is non binary, the channel model is slightly more complicated: this is why the Koetter-Vardy soft decoder is used in [MCT16b] and not just an error and erasure decoder of generalized Reed-Solomon codes. By using the noise model corresponding to the probability computations (7) and (8) we obtain a much less noisy model than the original binary symmetric channel. This can be checked by a capacity calculation which is in a sense a measure of the noise of transmission channel (the capacity is a decreasing function of the noise level in some sense). The capacity of the binary symmetric channel of crossover probability p is $1 - h(p)$ whereas it is $1 - 2h(p) + h(2p(1-p))$ for the noise model corresponding to the probability computations (7) and (8). We have represented these two capacities in Figure 3 and it can be verified there that the new noise model has a much larger capacity than the original channel.

Fig. 3. Capacity of the original binary symmetric channel vs. capacity of the channel model corresponding to the probability computations (7) and (8).



This discussion explains why in the binary setting we would really like to use a decoder for U which is able to correct errors on the positions where $\mathbf{e}_V(i) = 0$ and erasures on the positions where $\mathbf{e}_V(i) = 1$. In our context where we perform source-distortion decoding the situation is actually similar. Our strategy works here because the Prange decoder has a natural and powerful extension to the error/erasure scenario. It is natural to expect that a family of codes and associated decoders which are powerful in the erasure/error scenario behave better when used in a $(U, U+V)$ construction and decoded as above, than the original family of codes. Our strategy for obtaining a signature scheme really builds upon this approach : a $(U, U+V)$ code decoded as above with the Prange decoder has better distortion than the Prange decoder used directly on a linear code with the same length and dimension as the $(U, U+V)$-code. The trapdoor here for obtaining the better distortion is only the $(U, U+V)$ structure, but we can afford to have random linear codes for U and V.

4 Security Proof

We give in this section a security proof of the signature scheme $\mathcal{S}_{\text{code}}$. This proof is in the spirit of the security proof of the FDH signatures in the random oracle model (see [BR93]). However in order to have a tight security reduction we were inspired by the proof of [Cor02]. Our main result is to reduce the security to two major problems in code-based cryptography.

4.1 Basic tools

Basic definitions. A function $f(n)$ is said to be negligible if for all polynomials $p(n)$, $|f(n)| < p(n)^{-1}$ for all sufficiently large n. The statistical distance between two discrete probability distributions over a same space $\mathcal{E}$ is defined as:

$$\rho(\mathcal{D}^0, \mathcal{D}^1) \stackrel{\triangle}{=} \frac{1}{2} \sum_{x \in \mathcal{E}} |\mathcal{D}^0(x) - \mathcal{D}^1(x)|.$$

We will need the following well known property for the statistical distance which can be easily proved by induction.

Proposition 4. *Let $(\mathcal{D}_1^0, \dots, \mathcal{D}_n^0)$ and $(\mathcal{D}_1^1, \dots, \mathcal{D}_n^1)$ be two n-tuples of discrete probability distributions where $\mathcal{D}_i^0$ and $\mathcal{D}_i^1$ are distributed over a same space $\mathcal{E}_i$. We have for all positive integers n:*

$$\rho\left(\mathcal{D}_1^0 \otimes \dots \otimes \mathcal{D}_n^0, \mathcal{D}_1^1 \otimes \dots \otimes \mathcal{D}_n^1\right) \leq \sum_{i=1}^{n} \rho(\mathcal{D}_i^0, \mathcal{D}_i^1).$$

A *distinguisher* between two distributions $\mathcal{D}^0$ and $\mathcal{D}^1$ over the same space $\mathcal{E}$ is a randomized algorithm which takes as input an element of $\mathcal{E}$ that follows the distribution $\mathcal{D}^0$ or $\mathcal{D}^1$ and outputs $b \in \{0, 1\}$. It is characterized by its advantage:

$$Adv^{\mathcal{D}^0, \mathcal{D}^1}(\mathcal{A}) \stackrel{\triangle}{=} \mathbb{P}_{\xi \sim \mathcal{D}^0}\left(\mathcal{A}(\xi) \text{ outputs } 1\right) - \mathbb{P}_{\xi \sim \mathcal{D}^1}\left(\mathcal{A}(\xi) \text{ outputs } 1\right).$$

We call this quantity the *advantage* of $\mathcal{A}$ against $\mathcal{D}^0$ and $\mathcal{D}^1$.

Definition 5 (Computational Distance and Indistinguishability). *The computational distance between two distributions $\mathcal{D}^0$ and $\mathcal{D}^1$ in time t is:*

$$\rho_c\left(\mathcal{D}^0, \mathcal{D}^1\right)(t) \stackrel{\triangle}{=} \max_{|\mathcal{A}| \leq t} \left\{ Adv^{\mathcal{D}^0, \mathcal{D}^1}(\mathcal{A}) \right\}$$

where $|\mathcal{A}|$ denotes the running time of $\mathcal{A}$ on its inputs.

The ensembles $\mathcal{D}^0 = (\mathcal{D}_n^0)$ and $\mathcal{D}^1 = (\mathcal{D}_n^1)$ are computationally indistinguishable in time (t_n) if their computational distance in time (t_n) is negligible in n.

In other words, the computational distance is the best advantage that any adversary could get in bounded time.

Digital signature and games. Let us recall the concept of signature schemes, the security model that will be considered in the following and to recall in this context the paradigm of games in which we give a security proof of our scheme.

Definition 6 (Signature Scheme). *A signature scheme $\mathcal{S}$ is a triple of algorithms* `Gen`*,* `Sgn`*, and* `Vrfy` *which are defined as:*

- *The key generation algorithm* `Gen` *is a probabilistic algorithm which given 1^λ, where λ is the security parameter, outputs a pair of matching public and private keys (pk, sk);*
- *The signing algorithm is probabilistic and takes as input a message $\mathbf{m} \in \{0,1\}^*$ to be signed and returns a signature $\sigma = \texttt{Sgn}^{\text{sk}}(\mathbf{m})$;*
- *The verification algorithm takes as input a message $\mathbf{m}$ and a signature σ. It returns $\texttt{Vrfy}^{\text{pk}}(\mathbf{m}, \sigma)$ which is 1 if the signature is accepted and 0 otherwise. It is required that $\texttt{Vrfy}^{\text{pk}}(\mathbf{m}, \sigma) = 1$ if $\sigma = \texttt{Sgn}^{\text{sk}}(\mathbf{m})$.*

For this kind of scheme, one of the strongest security notion is *existential unforgeability under an adaptive chosen message attack* (EUF-CMA). In this model the adversary has access to all signatures of its choice and its goal is to produce a valid forgery. A valid forgery is a message/signature pair $(\mathbf{m}, \sigma)$ such that $\texttt{Vrfy}^{\text{pk}}(\mathbf{m}, \sigma) = 1$ whereas the signature of $\mathbf{m}$ has never been requested by the forger. More precisely, the following definition gives the EUF-CMA security of a signature scheme:

Definition 7 (EUF-CMA Security). *Let $\mathcal{S}$ be a signature scheme.*
A forger $\mathcal{A}$ is a $(t, q_{\text{hash}}, q_{\text{sign}}, \varepsilon)$-adversary in EUF-CMA *against $\mathcal{S}$ if after at most q_{hash} queries to the hash oracle, q_{sign} signatures queries and t working time, it outputs a valid forgery with probability at least ε. We define the* EUF-CMA *success probability against $\mathcal{S}$ as:*

$$Succ_{\mathcal{S}}^{\text{EUF-CMA}}(t, q_{\text{hash}}, q_{\text{sign}}) \stackrel{\triangle}{=} \max\left(\varepsilon | \textit{it exists a } (t, q_{\text{hash}}, q_{\text{sign}}, \varepsilon)\textit{-adversary}\right).$$

The signature scheme $\mathcal{S}$ is said to be $(t, q_{\text{hash}}, q_{\text{sign}})$-secure in EUF-CMA *if the above success probability is a negligible function of the security parameter λ.*

The game associated to our code-based signature scheme. The modern approach to prove the security of cryptographic schemes is to relate the security of its primitives to well-known problems that are believed to be hard by proving that breaking the cryptographic primitives provides a mean to break one of these hard problems. In our case, the security of the signature scheme is defined as a game with an adversary that has access to hash and sign oracles. It will be helpful here to be more formal and to define more precisely the games we will consider. They are games between two players, an *adversary* and a *challenger*. In a game G, the challenger executes three kind of procedures:

- an initialization procedure `Initialize` which is called once at the beginning of the game.
- oracle procedures which can be requested at the will of the adversary. In our case, there will be two, `Hash` and `Sign`. The adversary $\mathcal{A}$ which is an algorithm may call `Hash` at most q_{hash} times and `Sign` at most q_{sign} times.
- a final procedure `Finalize` which is executed once $\mathcal{A}$ has terminated. The output of $\mathcal{A}$ is given as input to this procedure.

The output of the game G, which is denoted $G(\mathcal{A})$, is the output of the finalization procedure (which is a bit $b \in \{0,1\}$). The game G with $\mathcal{A}$ is said to be successful if $G(\mathcal{A}) = 1$. The standard approach for obtaining a security proof in a certain model is to construct a sequence of games such that the success of the first game with an adversary $\mathcal{A}$ is exactly the success against the model of security, the difference of the probability of success between two consecutive games is negligible until the final game where the probability of success is the probability for $\mathcal{A}$ to break one of the problems which is supposed to be hard. In this way, no adversary can break the claim of security with non-negligible success unless it breaks one of the problems that are supposed to be hard.

Definition 8 (challenger procedures in the EUF-CMA Game). *The challenger procedures for the* EUF-CMA *Game corresponding to $\mathcal{S}_{code}$ are defined as:*

`proc Initialize`(λ)	`proc Hash`$(\mathbf{m},\mathbf{r})$	`proc Sign`$(\mathbf{m})$	`proc Finalize`$(\mathbf{m},\mathbf{e},\mathbf{r})$
$(pk, sk) \leftarrow \mathtt{Gen}(1^\lambda)$	`return` $h(\mathbf{m},\mathbf{r})$	$\mathbf{r} \hookleftarrow \{0,1\}^{\lambda_0}$	$\mathbf{s} \leftarrow \mathtt{Hash}(\mathbf{m},\mathbf{r})$
$\mathbf{H}_{\mathtt{pub}} \leftarrow pk$		$\mathbf{s} \leftarrow \mathtt{Hash}(\mathbf{m},\mathbf{r})$	`return`
$(\mathbf{H}_{\mathtt{sec}}, \mathbf{P}, \mathbf{S}) \leftarrow sk$		$\mathbf{e} \leftarrow D_{\mathbf{H}_{\mathtt{sec}},w}(\mathbf{S}^{-1}\mathbf{s}^T)$	$\mathbf{H}_{\mathtt{pub}}\mathbf{e}^T = \mathbf{s}^T \wedge \lvert\mathbf{e}\rvert = w$
`return` $\mathbf{H}_{\mathtt{pub}}$		`return` $(\mathbf{eP},\mathbf{r})$	

4.2 Code-Based Problems

We introduce in this subsection the code-based problems that will be used in the security proof. The first is Decoding One Out of Many (DOOM) which was first considered in [JJ02] and later analyzed in [Sen11]. We will come back to the best known algorithms to solve this problem as a function of the distance w in §6.

*Problem 2 (*DOOM *– Decoding One Out of Many).*

Instance:	$\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$, $\mathbf{s}_1, \cdots, \mathbf{s}_q \in \mathbb{F}_2^{n-k}$, w integer
Output:	$(\mathbf{e}, i) \in \mathbb{F}_2^n \times [\![1,q]\!]$ such that $\lvert\mathbf{e}\rvert = w$ and $\mathbf{He}^T = \mathbf{s}_i^T$.

Definition 9 (One-Wayness of DOOM). *We define the success of an algorithm $\mathcal{A}$ against* DOOM *with the parameters n, k, q, w as:*

$$Succ^{n,k,q,w}_{\text{DOOM}}(\mathcal{A}) = \mathbb{P}\big(\mathcal{A}(\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_q) \text{ solution of DOOM}\big)$$

where $\mathbf{H}$ is chosen uniformly at random in $\mathbb{F}_2^{(n-k)\times n}$, the $\mathbf{s}_i$'s are chosen uniformly at random in $\mathbb{F}_2^{n-k}$ and the probability is taken over these choices of $\mathbf{H}$, the $\mathbf{s}_i$'s and the internal coins of $\mathcal{A}$.

The computational success in time t of breaking DOOM *with the parameters n, k, q, w is then defined as:*

$$Succ^{n,k,q,w}_{\text{DOOM}}(t) = \max_{|\mathcal{A}|\le t}\left\{Succ^{n,k,q,w}_{\text{DOOM}}(\mathcal{A})\right\}.$$

Another problem will appear in the security proof: distinguish random codes from a code drawn uniformly at random in the family used for public keys in the signature scheme. We will denote in the rest of the article by $\mathbf{H}_{\text{pub}}$ the random matrix chosen as the public parity-check matrix of our scheme. Let us recall that it is obtained as

$$\mathbf{H}_{\text{pub}} = \mathbf{SH}_{\text{sec}}\mathbf{P} \text{ with } \mathbf{H}_{\text{sec}} = \begin{pmatrix} \mathbf{H}_U & \mathbf{0} \\ \mathbf{H}_V & \mathbf{H}_V \end{pmatrix}, \tag{9}$$

where $\mathbf{S}$ is chosen uniformly at random among the invertible binary matrices of size $(n-k)\times(n-k)$, $\mathbf{H}_U$ is chosen uniformly at random among the binary matrices of size $(n/2-k_U)\times n/2$, $\mathbf{H}_V$ is chosen uniformly at random among the binary matrices of size $(n/2-k_V)\times n/2$ and $\mathbf{P}$ is chosen uniformly at random among the permutation matrices of size $n\times n$. The distribution of the random variable $\mathbf{H}_{\text{pub}}$ is denoted by $\mathcal{D}_{\text{pub}}$. On the other hand $\mathcal{D}_{\text{rand}}$ will denote the uniform distribution over the parity-check matrices of all $[n,k]$-codes with $k = k_U + k_V$.

We will discuss about the difficulty of the task to distinguish $\mathcal{D}_{\text{pub}}$ and $\mathcal{D}_{\text{rand}}$ in §7. It should be noted that the syndromes associated to matrices $\mathbf{H}_{\text{pub}}$ are indistinguishable in a very strong sense from random syndromes as the following proposition shows

Proposition 5. *Let $\mathcal{D}_w^{\mathbf{H}}$ be the distribution of the syndromes $\mathbf{He}^T$ when $\mathbf{e}$ is drawn uniformly at random among the binary vectors of weight w and $\mathcal{U}$ be the uniform distribution over the syndrome space $\mathbb{F}_2^{n-k}$. We have*

$$\mathbb{E}_{\mathbf{H}_{\text{pub}}}\left(\rho(\mathcal{D}_w^{\mathbf{H}_{\text{pub}}}, \mathcal{U})\right) \le \frac{1}{2}\sqrt{\varepsilon}$$

with

$$\varepsilon = \frac{2^{n-k}}{\binom{n}{w}} + \frac{2^{n/2-k_U}\binom{n/2}{w/2}}{\binom{n}{w}} + \sum_{\substack{j\in\{0,\dots,w\} \\ j\equiv w \pmod 2}} \frac{2^{2j+n/2-k_V}\binom{n/2}{(w-j)/2}^2\binom{n/2-(w-j)/2}{j}^2}{\binom{n/2}{j}\binom{n}{w}^2}.$$

Remark 3. In the paradigm of code-based signatures we have w greater than the Gilbert-Varshamov bound, which gives $2^{n-k} \ll \binom{n}{w}$ and for the set of parameters we present in §8, $\varepsilon \lll \frac{1}{2^\lambda}$ with λ the security parameter.

4.3 EUF-CMA Security Proof

This subsection is devoted to our main theorem and its proof. Let us first introduce some notation that will be used. We will denote by $\mathcal{D}_w$ the distribution $\left\{D_{\mathbf{H}_{\text{sec}},w}(\mathbf{s}) : \mathbf{s} \hookleftarrow \mathbb{F}_2^{n-k}\right\}$ where $D_{\mathbf{H}_{\text{sec}},w}$ is the source-distortion decoder used in the signature scheme. Recall that $\mathcal{U}_w$ is the uniform distribution over S_w (which is the set of words of weight w in $\mathbb{F}_2^n$), $\mathcal{D}_{\text{pub}}$ is the distribution of public keys, $\mathcal{D}_{\text{rand}}$ is the uniform distribution over parity-check matrices of all $[n,k]$-codes and $\mathcal{S}_{\text{code}}$ is our signature scheme defined in §3.1 with the family of $(U, U+V)$ codes.

Theorem 1 (Security Reduction). *Let q_{hash} (resp. q_{sign}) be the number of queries to the hash (resp. signing) oracle. We assume that $\lambda_0 = \lambda + 2\log_2(q_{\text{sign}})$ where λ is the security parameter of the signature scheme. We have in the random oracle model* (ROM) *for all time t:*

$$Succ_{\mathcal{S}_{\text{code}}}^{\text{EUF-CMA}}(t, q_{\text{hash}}, q_{\text{sign}}) \leq 2Succ_{\text{DOOM}}^{n,k,q_{\text{hash}},w}(t_c) + \frac{1}{2}q_{\text{hash}}\sqrt{\varepsilon} + q_{\text{sign}}\rho\left(\mathcal{D}_w, \mathcal{U}_w\right) + \rho_c\left(\mathcal{D}_{\text{rand}}, \mathcal{D}_{\text{pub}}\right)(t_c) + \frac{1}{2^\lambda}$$

where $t_c = t + O\left(q_{\text{hash}} \cdot n^2\right)$ and ε given in Proposition 5.

Proof. Let $\mathcal{A}$ be a $(t, q_{\text{sign}}, q_{\text{hash}}, \varepsilon)$-adversary in the EUF-CMA model against $\mathcal{S}_{\text{code}}$ and let $(\mathbf{H}_0, \mathbf{s}_1, \cdots, \mathbf{s}_{q_{\text{hash}}})$ be drawn uniformly at random among all instances of DOOM for parameters n, k, q_{hash}, w. We stress here that syndromes $\mathbf{s}_j$ are random and independent vectors of $\mathbb{F}_2^{n-k}$. We write $\mathbb{P}(S_i)$ to denote the probability of success for $\mathcal{A}$ of game G_i. Let

Game 0 is the EUF-CMA game for $\mathcal{S}_{\text{code}}$.

Game 1 is identical to Game 0 unless the following failure event F occurs: there is a collision in a signature query (*i.e.* two signatures queries for a same message $\mathbf{m}$ lead to the same salt $\mathbf{r}$). By using the difference lemma (see for instance [Sho04, Lemma 1]) we get:

$$\mathbb{P}(S_0) \leq \mathbb{P}(S_1) + \mathbb{P}(F).$$

The following lemma (see A.2 for a proof) shows that in our case as $\lambda_0 = \lambda + 2\log_2(q_{\text{sign}})$, the probability of the event F is negligible.

Lemma 1. *For $\lambda_0 = \lambda + 2\log_2(q_{\text{sign}})$ we have:*

$$\mathbb{P}(F) \leq \frac{1}{2^\lambda}.$$

Game 2 is modified from Game 1 as follows:

```
proc Hash($\mathbf{m},\mathbf{r}$)
if $\mathbf{r} \in L_{\mathbf{m}}$
    $\mathbf{e}_{\mathbf{m},\mathbf{r}} \hookleftarrow S_w$
    return $\mathbf{e}_{\mathbf{m},\mathbf{r}}\mathbf{H}_{\text{pub}}^T$
else
    $j \leftarrow j+1$
    return $\mathbf{s}_j$
```

```
proc Sign($\mathbf{m}$)
$\mathbf{r} \leftarrow L_{\mathbf{m}}$.next()
$\mathbf{s} \leftarrow$ Hash($\mathbf{m},\mathbf{r}$)
$\mathbf{e} \leftarrow D_{\mathbf{H}_{\text{sec}},w}(\mathbf{S}^{-1}\mathbf{s}^T)$
return ($\mathbf{eP},\mathbf{r}$)
```

To each message $\mathbf{m}$ we associate a list $L_{\mathbf{m}}$ containing q_{sign} random elements of $\mathbb{F}_2^{\lambda_0}$. It is constructed the first time it is needed. The call $\mathbf{r} \in L_{\mathbf{m}}$ returns true if and only if $\mathbf{r}$ is in the list. The call $L_{\mathbf{m}}$.`next()` returns elements of $L_{\mathbf{m}}$ sequentially. The list is large enough to satisfy all queries.

The Hash procedure now creates the list $L_{\mathbf{m}}$ if needed, then, if $\mathbf{r} \in L_{\mathbf{m}}$ it returns $\mathbf{e}_{\mathbf{m},\mathbf{r}}\mathbf{H}_{\text{pub}}^T$ with $\mathbf{e}_{\mathbf{m},\mathbf{r}} \hookleftarrow S_w$. This leads to a valid signature $(\mathbf{e}_{\mathbf{m},\mathbf{r}}, \mathbf{r})$ for $\mathbf{m}$. The error value is stored. If $\mathbf{r} \notin L_{\mathbf{m}}$ it outputs one of $\mathbf{s}_j$ of the instance $(\mathbf{H}_0, \mathbf{s}_1, \dots, \mathbf{s}_{q_{\text{hash}}})$ of the DOOM problem. The Sign procedure is unchanged, except for $\mathbf{r}$ which is now taken in $L_{\mathbf{m}}$. The global index j is set to 0 in proc Initialize.

We can relate this game to the previous one through the following lemma.

Lemma 2.
$$\mathbb{P}(S_1) \leq \mathbb{P}(S_2) + \frac{q_{\text{hash}}}{2}\sqrt{\varepsilon} \text{ where } \varepsilon \text{ is given in Proposition 5.}$$

The proof of this lemma is given in Appendix A.3 and relies among other things on the following points:

- Proposition 4;
- Syndromes produced by matrices $\mathbf{H}_{\text{pub}}$ with errors of weight w have average statistical distance from the uniform distribution over $\mathbb{F}_2^{n-k}$ at most $\frac{1}{2}\sqrt{\varepsilon}$ (see Proposition 5). This follows from a lemma which is a variation of the leftover hash lemma (see [BDK+11]) and which can be expressed as follows.
- **Lemma 3.** *Consider a finite family $\mathcal{H} = (h_i)_{i \in I}$ of functions from a finite set E to a finite set F. Denote by ε the bias of the collision probability, i.e. the quantity such that*
$$\mathbb{P}_{h,e,e'}(h(e) = h(e')) = \frac{1}{|F|}(1 + \varepsilon)$$
where h is drawn uniformly at random in $\mathcal{H}$, e and e' are drawn uniformly at random in E. Let $\mathcal{U}$ be the uniform distribution over F and $\mathcal{D}(h)$ be the distribution of the outputs $h(e)$ when e is chosen uniformly at random in E. We have
$$\mathbb{E}_h \{\rho(\mathcal{D}(h), \mathcal{U})\} \leq \frac{1}{2}\sqrt{\varepsilon}.$$
Remark 4. In the leftover hash lemma, there is the additional assumption that $\mathcal{H}$ is a universal family of hash functions, meaning that for any e and e' distinct in F, we have $\mathbb{P}_h(h(e) = h(e')) = \frac{1}{|F|}$. This assumption allows to have a general bound on the bias ε. In our case, where the h's are hash functions defined as $h(e) = \mathbf{H}_{\text{pub}}\mathbf{e}^T$, $\mathcal{H}$ does not form a universal family of hash functions (essentially because the distribution of the $\mathbf{H}_{\text{pub}}$'s is not the uniform distribution over $\mathbb{F}_2^{(n-k)\times n}$). However in our case we can still bound ε by a direct computation. This lemma is proved in Appendix §A.3.

Game 3 differs from Game 2 by changing in proc Sign calls "$\mathbf{e} \leftarrow D_{\mathbf{H}_{\text{sec}},w}(\mathbf{S}^{-1}\mathbf{s}^T)$" by "$\mathbf{e} \leftarrow \mathbf{e}_{\mathbf{m},\mathbf{r}}$" and "return $(\mathbf{eP}, \mathbf{r})$" by "return $(\mathbf{e}, \mathbf{r})$". Any signature $(\mathbf{e}, \mathbf{r})$ produced by proc Sign is valid. The error $\mathbf{e}$ is drawn according to the uniform distribution $\mathcal{U}_w$ while previously it was drawn according to the source distortion decoder distribution, that is $\mathcal{D}_w$. By using Proposition 4 it follows that
$$\mathbb{P}(S_2) \leq \mathbb{P}(S_3) + q_{\text{sign}}\rho(\mathcal{U}_w, \mathcal{D}_w).$$

Game 4 is the game where we replace the public matrix $\mathbf{H}_{\text{pub}}$ by $\mathbf{H}_0$. In this way we will force the adversary to build a solution of the DOOM problem. Here if a difference is detected between games it gives a distinguisher between distributions $\mathcal{D}_{\text{rand}}$ and $\mathcal{D}_{\text{pub}}$:
$$\mathbb{P}(S_3) \leq \mathbb{P}(S_4) + \rho_c(\mathcal{D}_{\text{pub}}, \mathcal{D}_{\text{rand}})(t_c).$$

We show in appendix how to emulate the lists $L_{\mathbf{m}}$ in such a way that list operations cost, including its construction, is at most linear in the security parameter λ. Since $\lambda \leq n$, it follows that the cost to a call to proc Hash cannot exceed $O(n^2)$ and the running time of the challenger is $t_c = t + O\left(q_{\text{hash}} \cdot n^2\right)$.

Game 5 differs in the finalize procedure.

```
proc Finalize($\mathbf{m},\mathbf{e},\mathbf{r}$)
s ← Hash($\mathbf{m},\mathbf{r}$)
$b \leftarrow \mathbf{H}_{\mathtt{pub}}\mathbf{e}^T = \mathbf{s}^T \wedge |\mathbf{e}| = w$
return $b \wedge \mathbf{r} \notin L_{\mathbf{m}}$
```

We assume the forger outputs a valid signature $(\mathbf{e},\mathbf{r})$ for the message $\mathbf{m}$. The probability of success of Game 5 is the probability of the event "$S_4 \wedge (\mathbf{r} \notin L_{\mathbf{m}})$".

If the forgery is valid, the message $\mathbf{m}$ has never been queried by `Sign`, and the adversary never had access to any element of the list $L_{\mathbf{m}}$. This way, the two events are independent and we get:

$$\mathbb{P}(S_5) = (1-2^{-\lambda_0})^{q_{\text{sign}}}\mathbb{P}(S_4).$$

As we assumed $\lambda_0 = \lambda + 2\log_2(q_{\text{sign}}) \geq \log_2(q^2_{\text{sign}})$, we have:

$$\left(1-2^{-\lambda_0}\right)^{q_{\text{sign}}} \geq \left(1-\frac{1}{q^2_{\text{sign}}}\right)^{q_{\text{sign}}} \geq \frac{1}{2}.$$

Therefore

$$\mathbb{P}(S_5) \geq \frac{1}{2}\mathbb{P}(S_4). \tag{10}$$

The probability $\mathbb{P}(S_5)$ is then exactly the probability for $\mathcal{A}$ to output $\mathbf{e}_j \in S_w$ such that $\mathbf{H}_0\mathbf{e}_j^T = \mathbf{s}_j^T$ for some j which gives

$$\mathbb{P}(S_5) \leq \ Succ^{n,k,q_{\text{hash}},w}_{\text{DOOM}}(t_c). \tag{11}$$

(10) together with (11) imply that

$$\mathbb{P}(S_4) \leq 2 \cdot \ Succ^{n,k,q_{\text{hash}},w}_{\text{DOOM}}(t_c).$$

This concludes the proof of Theorem 1 by combining this together with all the bounds obtained for each of the previous games.

5 Achieving the Uniform Distribution of the Outputs

5.1 Rejection Sampling Method

In our security proof, we use the fact that the distribution of the outputs of the $(U, U+V)$ decoder is close to the uniform distribution on the words of weight w. We will show how to modify a little bit the decoder by performing some moderate rejection sampling in order to meet this property. Note that ensuring such a property is actually not only desirable for the security proof, it is also more or less necessary since there is an easy way to attack the signature when it is based on the decoder UV-**sddV**1. Indeed, it is readily verified that with this decoder the probability $\mathbb{P}(e_i = 1, e_j = 1)$ we have on the output $\mathbf{e}$ of the decoder for certain i and j is larger than the same probability for a random word $\mathbf{e}$ of weight w. The pairs (i,j) which have this property correspond to the image by the permutation $\mathbf{P}$ of pairs of the form $(x, x+n/2)$ or $(x+n/2, x)$. In other words, signatures leak information in this case and this can be used to recover completely the permuted $(U, U+V)$ structure of the code.

To explain the rejection method, let us introduce some notation. Let $\mathbf{e} \in \mathbb{F}_2^n$,

$$w_1(\mathbf{e}) \stackrel{\triangle}{=} \left|\left\{i \in \{1,\cdots,n/2\} \ : \ e_i \neq e_{i+n/2}\right\}\right|,$$
$$w_2(\mathbf{e}) \stackrel{\triangle}{=} \left|\left\{i \in \{1,\cdots,n/2\} \ : \ e_i = e_{i+n/2} = 1\right\}\right|.$$

The problem is that algorithm UV-**sddV**1 outputs errors $\mathbf{e}$ for which $w_1(\mathbf{e})$ and $w_2(\mathbf{e})$ are constant: $w_1(\mathbf{e}) = (n/2 - k_V)/2$ and $w_2(\mathbf{e}) = (n/2 - k_U - w_1(\mathbf{e}))/2 = n/8 - k_U/2 + k_V/4$. Obviously uniformly distributed errors $\mathbf{e}$ in S_w do not have this behavior. Our strategy to attain this uniform distribution on the outputs $\mathbf{e}$ is to change a little bit the source-distortion decoder for V in order to attain variable weight errors which are such that the weight of $\mathbf{e}_V$ (which corresponds

Algorithm 2 UV**-sddV2** : $(U, U+V)-$**Source Distortion Decoder**

Parameter: a $(U, U+V)$ code of length n
Inputs: · $(\mathbf{s}_1, \mathbf{s}_2)$ with $\mathbf{s}_1 \in \mathbb{F}_2^{n/2-k_U}$, $\mathbf{s}_2 \in \mathbb{F}_2^{n/2-k_V}$
· no-rejection probability vector $\mathbf{x} = (x_i)_{0 \leq i \leq n-k_V} \in [0,1]^{n-k_V}$
Output: $\mathbf{e} \in \mathbb{F}_2^n$ with $|\mathbf{e}| = w$.
Assumes: $2k_U - k_V \leq n/2$.

```
1: repeat
2:     $\mathbf{e}_V \leftarrow D(\mathbf{H}_V, \mathbf{s}_2)$
3:     $p \hookleftarrow [0,1]$
4: until $|\mathbf{e}_V| \leq w$, $w - |\mathbf{e}_V| \equiv 0 \pmod 2$ and $p \leq x_{|\mathbf{e}_V|}$
5: $\mathbf{e}_U \leftarrow D_{(w-|\mathbf{e}_V|)/2}(\mathbf{H}_U, \mathbf{s}_1, \mathbf{e}_V)$
6: return $(\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V)$
```

to $w_1(\mathbf{e})$) have the same distribution as $w_1(\mathbf{e}')$ where $\mathbf{e}'$ is a random error of weight w which is uniformly distributed. This can be easily done by rejection sampling as in Algorithm 2. Recall that D is a Source Distortion Decoder (see Definition 1 in §3.2).

From now on we consider two random variables : $\mathbf{e}$ which is the output of Algorithm 2 and $\mathbf{e}'$ which is a uniformly distributed error of weight w. It is easily verified that $w_1(\mathbf{e}) = |\mathbf{e}_V|$ and $w_2(\mathbf{e}) = (w - |\mathbf{e}_V|)/2$. Moreover, it turns out that it is not only necessary in order to achieve uniform distribution on the output to enforce that $w_1(\mathbf{e})$ follows the same law as $w_1(\mathbf{e}')$, this is also sufficient. To check this, let us introduce some additional notation. For $i \in \{1,2\}$ we define the quantities

$$p_i^{sdd}(j) \stackrel{\triangle}{=} \mathbb{P}_{\mathbf{e}}\left(w_i(\mathbf{e}) = j\right) \quad ; \quad p_i^{u}(j) \stackrel{\triangle}{=} \mathbb{P}_{\mathbf{e}'}\left(w_i(\mathbf{e}') = j\right)$$

We will also say that a source distortion decoder D *behaves uniformly* for a parity-check matrix $\mathbf{H}$ if $\mathbb{P}_{\mathbf{s},\theta}\left(\mathbf{e} = D(\mathbf{H},\mathbf{s})\right)$ only depends on the weight $|\mathbf{e}|$ (here θ denotes the internal randomness of algorithm D).

In such a case, the no-rejection vector $\mathbf{x}$ can be chosen so that the output of Algorithm 2 is uniformly distributed as shown by the following theorem.

Theorem 2. *If the source decoder D used in Algorithm 2 behaves uniformly for $\mathbf{H}_V$ and uniformly for $\mathbf{H}_U''$ which is obtained from $(\mathbf{H}_U, \mathbf{e}_V)$ in Proposition 2 (see §3.2) for all error patterns $\mathbf{e}_V$ obtained as $\mathbf{e}_V = D(\mathbf{H}_V, \mathbf{s}_2)$, we have:*

$$\rho\left(\mathcal{D}_w, \mathcal{U}_w\right) = \rho\left(p_1^{sdd}, p_1^{u}\right)$$

where $\mathcal{D}_w$ is the output distribution of Algorithm 2. Then, output of Algorithm 2 is the uniform distribution over S_w if in addition two executions of D are independent and the no-rejection probability vector $\mathbf{x}$ is chosen for any i in $\{0,\dots,w\}$ as

$$x_i = \frac{1}{M_{rs}} \frac{p_1^u(i)}{p(i)} \text{ if } w \equiv i \pmod 2 \quad x_i = 0 \text{ otherwise}$$

with $p(i) \stackrel{\triangle}{=} \mathbb{P}_{\mathbf{s},\theta}(|D(\mathbf{H}_V,\mathbf{s})| = i)$ and $M_{rs} \stackrel{\triangle}{=} \sup\limits_{\substack{0 \leq i \leq w \\ i \equiv w \pmod 2}} \frac{p_1^u(i)}{p(i)}$.

5.2 Application to the Prange source distortion decoder.

The Prange source decoder (defined in §3.2) is extremely close to behave uniformly for almost all linear codes. To keep this paper within a reasonable length we just provide here how the relevant distribution $p(i)$ is computed.

Proposition 6 (Weight Distribution of the Prange Algorithm).
Let $p(i) = \sum_{\mathbf{e}:|\mathbf{e}|=i} \mathbb{P}_{\mathbf{s},\theta}\left(\mathbf{e} = D^{\text{Prange}}(\mathbf{H},\mathbf{s})\right)$. For all $w,k,n \in \mathbb{N}$ with $k \leq n$, $w \leq n-k$, all parity-check matrices of size $(n-k) \times n$, we have $p(w) = \frac{\binom{n-k}{w}}{2^{n-k}}$.

By using Theorem 2 with this distribution p we can set up the no-rejection probability vector $\mathbf{x}$ in Algorithm 2. To have an efficient algorithm it is essential that the parameter M_{rs} is as small as possible (it is readily verified that the average number of calls in Algorithm 2 to $D^{\mathrm{Prange}}(\mathbf{H}_V, \mathbf{s}_2)$ is M_{rs}). Let $\mathbf{e}$ be an error of weight w chosen uniformly at random. This average number of calls can be chosen to be small by imposing that the distributions of $w_1(\mathbf{e})$ and $|D(\mathbf{H}_V, \mathbf{s}_2)|$ to have the same expectation. The expectation of $w_1(\mathbf{e})$ is approximately $w\left(1-\frac{w}{n}\right)$ and the expectation of $|D(\mathbf{H}_V, \mathbf{s}_2)|$ is $(n/2-k_V)/2$. We choose therefore k_V such that

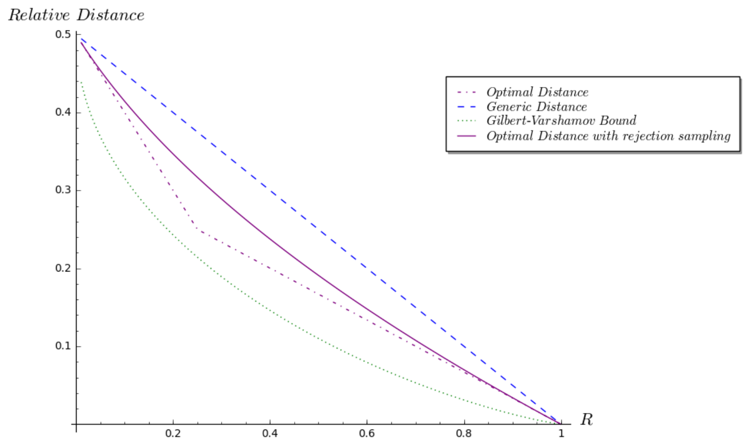
$$(n/2-k_V)/2 \approx w\left(1-\frac{w}{n}\right).$$

Thanks to this property, k_V is chosen to "align" both distributions and in this way M_{rs} is small. This rejection sampling method comes at the price of slightly increasing the weight the decoder can output as it is shown in Figure 4. It is easy to see that the optimal choice of the parameters k_U, k_V, w minimizing M_{rs} for given n, $R \overset{\triangle}{=} k/n$ leads to the following choice:

$$w \approx n\frac{3-\sqrt{1+8R}}{4}, \quad k_U = n/2 - w, \quad k_V \approx n/2 - 2w\left(1-\frac{w}{n}\right).$$

For instance for $n = 2000$, $k = 1000$, we have $w = 382$, $k_U = 618$, $k_V = 382$ and $M_{\mathrm{rs}} \approx 2.54$. Figure 4 gives $\omega \overset{\triangle}{=} w/n$ as a function of R. For instance with $R = 0.5$ we have $\omega = 0.1909$.

Fig. 4. Comparison of the Optimal Signature Distortion with or without the Rejection Sampling Method, the Gilbert-Varshamov Bound and the Generic Distortion



6 Best Known Algorithms for Solving the DOOM Decoding Problem

We consider here the best known techniques for solving Problem 2.

Problem 2. [DOOM – Decoding One Out of Many]

Instance:	$\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$, $\mathbf{s}_1, \cdots, \mathbf{s}_q \in \mathbb{F}_2^{n-k}$, w integer
Output:	$(\mathbf{e}, i) \in \mathbb{F}_2^n \times [\![1, q]\!]$ such that $|\mathbf{e}| = w$ and $\mathbf{H}\mathbf{e}^T = \mathbf{s}_i^T$.

When $q = 1$, Problem 2 is known as the Syndrome Decoding (SD) problem. Information Set Decoding (ISD) is the best known technique to solve SD, it can be traced back to Prange [Pra62]. It has been improved in [Ste88, Dum91] by introducing a birthday paradox. The current state-of-the-art can be found in [MMT11, BJMM12, MO15]. The DOOM problem was first considered in [JJ02] then analyzed in [Sen11] for Dumer's variant of ISD.

Existing literature usually assumes that there is a unique solution to the problem. This is true when w is smaller than the Gilbert-Varshamov bound (see Definition 2). When w is larger, as it is the case here, we speak of *source-distortion decoding*, the number of solutions grows as $M = \binom{n}{w}/2^{n-k}$ and the cost analysis must be adapted. Considering multiple instances, as in DOOM above, also alters the cost analysis.

From this point and till the end of this section, the parameters n, k, w are fixed.

6.1 Why Does DOOM Strengthen the Security Proof?

An attacker may produce many, say q, favorable messages and hash them to obtain $\mathbf{s}_1, \ldots, \mathbf{s}_q$ submitted to a solver of Problem 2 together with the public key $\mathbf{H}$ and the signature weight w. The output of the solver will produce a valid signature for one of the q messages. In the security reduction, the assumption related to DOOM is precisely the same, that is assuming key indistinguishability and a proper distribution of the signatures, the adversary has to solve an instance of DOOM as described above and the reduction is tight in this respect.

The usual Full Domain Hash (FDH) proof for existential forgery would use SD rather than DOOM and to guaranty a security parameter λ, the cost of SD, denoted WF, has to be at least $q2^{\lambda}$ where $q \leq 2^{\lambda}$ is the number of hash queries. This would require code parameters (n, k, w) such that $\mathrm{WF} \geq 2^{2\lambda}$. Instead we only require the cost of DOOM to be at least 2^{λ}, and even though DOOM is easier than SD, this will provide a tighter bound and allow smaller parameters.

We denote by $\mathrm{WF}^{1-\delta}$ the workfactor of DOOM when q can be as large as allowed. It is shown in [Sen11] that solving DOOM with ISD with q instances cannot cost less than $\mathrm{WF}/\sqrt{q}$, corresponding to $\delta = 0.33$ and a choice of parameters such that $\mathrm{WF} \geq 2^{1.5\lambda}$. In practice, the situation is more favorable. When decoding codes of rate $k/n = 1/2$ at the Gilbert-Varshamov bound ($w = 0.11\,n$), for Dumer's variant of ISD we get $\delta \leq 0.25$ and $\mathrm{WF} \geq 2^{1.32\lambda}$. When w grows, the situation is even better, for a rate $1/2$ and $w = 0.19\,n$ (the signature parameters) we get $\delta \approx 0.07$ and $\mathrm{WF} \geq 2^{1.08\lambda}$.

Finally, this means that using state-of-the-art solutions for DOOM, we only need to increase the code size by 8% compared with SD's requirement, whereas the usual proof would require to double the parameters. The rest of this section is devoted to a detailed analysis leading to this conclusion.

6.2 ISD – Information Set Decoding

The ISD algorithm for solving DOOM is sketched in Algorithm 3. In all variants of ISD, the

Algorithm 3 (generalized) ISD

```
1: input: $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}, \mathbf{s}_1, \ldots, \mathbf{s}_q \in \mathbb{F}_2^{n-k}, w$ integer
2: loop
3:     pick an $n \times n$ permutation matrix $\mathbf{P}$
4:     perform partial Gaussian elimination on $\mathbf{HP}$

       $\mathbf{UHP} = \begin{array}{|c|c|l} \hline \mathbf{I}_{n-k-\ell} & \mathbf{H}'' & \updownarrow n-k-\ell \\ \hline 0 & \mathbf{H}' & \updownarrow \ell \\ \hline \end{array} \qquad \mathbf{Us}_i^T = \begin{array}{|c|} \hline \mathbf{s}_i''^T \\ \hline \mathbf{s}_i'^T \\ \hline \end{array} \quad i = 1, \ldots, q$

5:     compute $\mathcal{E} = \{(\mathbf{e}', i) \in \mathbb{F}_2^{k+\ell} \times [\![1, q]\!] \mid \mathbf{H}'\mathbf{e}'^T = \mathbf{s}_i', |\mathbf{e}'| = p\}$, $\mathbf{H}' \in \mathbb{F}_2^{\ell\times(k+\ell)}$
6:     for all $(\mathbf{e}', i) \in \mathcal{E}$ do
7:         $\mathbf{e}'' \leftarrow \mathbf{e}'\mathbf{H}''^T + \mathbf{s}_i''$ ; $\mathbf{e} \leftarrow (\mathbf{e}'', \mathbf{e}')\mathbf{P}^T$
8:         if $|\mathbf{e}| = w$ then return $(\mathbf{e}, i)$
```

computation of the set $\mathcal{E}$ (Instruction 5) dominates the cost of one loop of Algorithm 3, we denote it by $C_q(p,\ell)$. As it is described, the loop is repeated until a solution is found. The standard version corresponds to a single instance, that is $q = 1$. Below we explain how the cost estimate of the algorithm varies in various situations: when we have a single instance and a single solution, when the number of solutions increases and when the number of instances (q) increases. For each value of n, k, w and q, the algorithm is optimized over the parameters p and ℓ. The optimal values of p and ℓ will change with the number of solutions and the number of instances.

Single Instance and Single Solution. We consider a situation where we wish to estimate the cost of the algorithm for producing one specific solution of Problem 2 with $q = 1$. In that case, even when w is large and there are multiple solutions, the solution we are looking for, say $\mathbf{e}$, is returned if and only if the permutation $\mathbf{P}$ is such that $|\mathbf{e}'| = p$ and $|\mathbf{e}''| = w - p$ where $(\mathbf{e}'', \mathbf{e}') \leftarrow \mathbf{eP}$. This will happen with probability $\mathcal{P}(p,\ell)$ leading to the workfactor $\mathrm{WF}^{(1)}$

$$\mathcal{P}(p,\ell) = \frac{\binom{n-k-\ell}{w-p}\binom{k+\ell}{p}}{\binom{n}{w}}, \mathrm{WF}^{(1)} = \min_{p,\ell} \frac{C_1(p,\ell)}{\mathcal{P}(p,\ell)},$$

which is obtained by solving an optimization problem over p and ℓ. The exact expression of $C_1(p,\ell)$ depends on the variant, for instance, for Dumer's algorithm [Dum91] we have $C_1(p,\ell) = \max\left(\sqrt{\binom{k+\ell}{p}}, \binom{k+\ell}{p}2^{-\ell}\right)$ up to a small polynomial factor. For more involved variants [BJMM12, MO15], the value of $C_1(p,\ell)$ is, for each (p,ℓ), the solution of another optimization problem.

Single Instance and Multiple Solutions. We now consider a situation where there are M solutions to a syndrome decoding problem ($q = 1$). If w is larger than the Gilbert-Varshamov bound we expect $M = \binom{n}{w}/2^{n-k}$ else $M = 1$. Assuming each of the M solutions can be independently produced, the probability that one particular iteration produces (at least) one of the solutions becomes $\mathcal{P}_M(p,\ell) = 1 - (1 - \mathcal{P}(p,\ell))^M$. The corresponding workfactor is

$$\mathrm{WF}^{(M)} = \min_{p,\ell} \frac{C_1(p,\ell)}{\mathcal{P}_M(p,\ell)}.$$

Let (p_0,ℓ_0) be the optimal value of the pair (p,ℓ) for a single solution.

Case 1: $\mathcal{P}(p_0,\ell_0) \le 1/M$. We have $\mathcal{P}_M(p_0,\ell_0) = 1-(1-\mathcal{P}(p_0,\ell_0))^M \ge \mathrm{e}^{-1}M\mathcal{P}(p_0,\ell_0)$ and thus

$$\mathrm{WF}^{(M)} = \min_{p,\ell} \frac{C_1(p,\ell)}{\mathcal{P}_M(p,\ell)} \le \frac{C_1(p_0,\ell_0)}{\mathcal{P}_M(p_0,\ell_0)} \le \frac{\mathrm{e}}{M}\frac{C_1(p_0,\ell_0)}{\mathcal{P}(p_0,\ell_0)} = \frac{\mathrm{e}}{M}\mathrm{WF}^{(1)}.$$

Also remark that $\mathcal{P}_M(p,\ell) \le M\mathcal{P}(p,\ell)$ and thus $\mathrm{WF}^{(M)} \ge \mathrm{WF}^{(1)}/M$. In other words, up to a small constant factor, the workfactor for multiple solutions is simply obtained by dividing the single solution workfactor by the number of solutions.

Case 2: $\mathcal{P}(p_0,\ell_0) > 1/M$. In this case the success probability $\mathcal{P}_M(p_0,\ell_0) < M\mathcal{P}(p_0,\ell_0)$ and the pair (p,ℓ) that minimizes the workfactor is going to be different. We observe that the gain is much less than the factor M of Case 1.

In practice, and for the parameters we consider in this work, we are always in Case 2. In fact, for $k/n = 0.5$, with Dumer's algorithm Case 1 only applies when $w/n < 0.150$, while the Gilbert-Varshamov bound corresponds to $w/n = 0.110$. With BJMM's algorithm, Case 1 only happens when $w/n \le 0.117$. In our signature scheme we have $w/n \approx 0.19$ and we always fall in Case 2, even with a single instance.

Multiples Instances with Multiple Solutions. We now consider the case where the adversary has access to q instances for the same matrix $\mathbf{H}$ and various syndromes. This is the Problem 2 that appears in the security reduction. For each instance, we expect $M = \max\left(1, \binom{n}{w}/2^{n-k}\right)$ solutions.

As before, the cost is dominated by Instruction 5, we denote it by $C_q(p,\ell)$, and the probability of success is $\mathcal{P}_{qM}(p,\ell) = 1-(1-\mathcal{P}(p,\ell))^{qM}$. The overall cost has to be minimized over p and ℓ

$$\mathrm{WF}_q^{(M)} = \min_{p,\ell} \frac{C_q(p,\ell)}{\mathcal{P}_{qM}(p,\ell)}.$$

Indeed how to compute $\mathcal{E}$, and thus the value of $C_q(p,\ell)$, is not specified in Algorithm 3. This is in fact what [JJ02, Sen11] are about. For instance with Dumer's algorithm, we have [Sen11]

$$C_q(p,\ell) = \max\left(\sqrt{q\binom{k+\ell}{p}}, \frac{q\binom{k+\ell}{p}}{2^\ell}\right),\ q \le \binom{k+\ell}{p}$$

up to a small polynomial factor. Introducing multiple instances in advanced variants of ISD has not been done so far and is an open problem. We give in Table 1 the asymptotic exponent for various decoding distances and for the code rate 0.5. The third column gives the largest useful value of q. It is likely that BJMM's algorithm will have a slightly lower exponent when addressing multiple instances. Note that for Dumer's algorithm in this range of parameters, the improvement from $\mathrm{WF}^{(M)}$ (single instance) to $\mathrm{WF}_q^{(M)}$ (multiple instances) is relatively small, there is no reason to expect a much different behavior for BJMM.

		Dumer			BJMM
w/n	$\frac{1}{n}\log_2 M$	$\frac{1}{n}\log_2 q$	$\frac{1}{n}\log_2 \mathrm{WF}_q^{(M)}$	$\frac{1}{n}\log_2 \mathrm{WF}^{(M)}$	$\frac{1}{n}\log_2 \mathrm{WF}^{(M)}$
0.11	0.0000	0.0872	0.0872	0.1152	0.1000
0.15	0.1098	0.0448	0.0448	0.0535	0.0486
0.19	0.2015	0.0171	0.0171	0.0184	0.0175

Table 1. Asymptotic Exponent for Algorithm 3 for $k/n = 0.5$

Finally, let us mention that the best asymptotic exponent among all known decoding techniques was proposed in [MO15]. However it is penalized by a big polynomial overhead which makes it more expensive at this point for the sizes considered here.

6.3 Other Decoding Techniques.

As mentioned in [CJ04, FS09], the Generalized Birthday Algorithm (GBA) [Wag02] is a relevant technique to solve decoding problems, in particular when there are multiple solutions. However, it is competitive only when the ratio k/n tends to 1, and does not apply here. We refer the reader to [MS09] for more details on GBA and its usage.

7 Distinguishing a permuted $(U, U+V)$ code

We discuss in this section how hard it is to decide whether a given linear code is a permuted $(U, U+V)$-code or not and give the best algorithm we have found to perform this task. This algorithm is based on a series of works on related problems [OT11, LT13, GHPT17].

7.1 Main idea used in the algorithms distinguishing or recovering the structure of a $(U, U+V)$-code we present here

A $(U, U+V)$ code where U and V are random seems very close to a random linear code. There is for instance only a very slight difference between the weight distribution of a random linear code and the weight distribution of a random $(U, U+V)$-code of the same length and dimension. This slight difference happens for small and large weights and is due to codewords of the form $(\mathbf{u},\mathbf{u})$ where $\mathbf{u}$ belongs to U or codewords of the form $(\mathbf{0},\mathbf{v})$ where $\mathbf{v}$ belongs to V. More precisely, we have the following proposition

Proposition 7. *Assume that we choose a $(U, U+V)$ code by picking the parity-check matrices of U and V uniformly at random among the binary matrices of size $(n/2-k_U)\times n/2$ and $(n/2-k_V)\times n/2$ respectively. Let $a_{(U,U+V)}(w)$, $a_{(U,U)}(w)$ and $a_{(0,V)}(w)$ be the expected number of codewords of weight w that are respectively in the $(U, U+V)$ code, of the form $(\mathbf{u},\mathbf{u})$ where $\mathbf{u}$ belongs to U and of the form $(\mathbf{0},\mathbf{v})$ where $\mathbf{v}$ belongs to V. These numbers are given for even w in $\{0,\dots,n\}$ by*

$$a_{(U,U+V)}(w) = \frac{\binom{n/2}{w/2}}{2^{n/2-k_U}} + \frac{\binom{n/2}{w}}{2^{n/2-k_V}} + \frac{1}{2^{n-k_U-k_V}}\left(\binom{n}{w} - \binom{n/2}{w} - \binom{n/2}{w/2}\right)$$

$$a_{(U,U)}(w) = \frac{\binom{n/2}{w/2}}{2^{n/2-k_U}} \quad ; \quad a_{(0,V)}(w) = \frac{\binom{n/2}{w}}{2^{n/2-k_V}}$$

and for odd w in $\{0,\dots,n\}$ by

$$a_{(U,U+V)}(w) = \frac{\binom{n/2}{w}}{2^{n/2-k_V}} + \frac{1}{2^{n-k_U-k_V}}\left(\binom{n}{w} - \binom{n/2}{w}\right)$$

$$a_{(U,U)}(w) = 0 \quad ; \quad a_{(0,V)}(w) = \frac{\binom{n/2}{w}}{2^{n/2-k_V}}$$

On the other hand, when we choose a code of length n with a random parity-check matrix of size $(n-k_U-k_V)\times n$ chosen uniformly at random, then the expected number $a(w)$ of codewords of weight $w>0$ is given by

$$a(w) = \frac{\binom{n}{w}}{2^{n-k_U-k_V}}.$$

Remark 5. When the $(U, U+V)$ code is chosen in this way, its dimension is k_U+k_V with probability $1-O\left(\max(2^{k_U-n/2}, 2^{k_V-n/2})\right)$. This also holds for the random codes of length n.

We have plotted in Figure 5 the normalized logarithm of the density of codewords of the form $(\mathbf{u},\mathbf{u})$ and $(\mathbf{0},\mathbf{v})$ of relative *even* weight $x \overset{\triangle}{=} \frac{w}{n}$ against x in the case U is of rate $\frac{k_U}{n/2}=0.6$ and V is of rate $\frac{k_V}{n/2}=0.4$. These two relative densities are defined respectively by

$$\alpha_{(U,U)}(w/n) = \frac{\log_2(a_{(U,U)}(w)/a(w))}{n} \quad ; \quad \alpha_{(0,V)}(w/n) = \frac{\log_2(a_{(0,V)}(w)/a(w))}{n}$$

We see that for a relative weight w/n below approximately 0.18 almost all the codewords are of the form $(\mathbf{0},\mathbf{v})$ in this case.

Since the weight distribution is invariant by permuting the positions, this slight difference also survives in the permuted version of $(U, U+V)$. These considerations lead to the best attack we have found for recovering the structure of a permuted $(U, U+V)$ code. It consists in applying known algorithms aiming at recovering low weight codewords in a linear code. We run such an algorithm until getting at some point either a permuted $(\mathbf{u},\mathbf{u})$ codeword where $\mathbf{u}$ is in U or a permuted $(\mathbf{0},\mathbf{v})$ codeword where $\mathbf{v}$ belongs to V. The rationale behind this algorithm is that the density of codewords of the form $(\mathbf{u},\mathbf{u})$ or $(\mathbf{0},\mathbf{v})$ is bigger when the weight of the codeword gets smaller.

Once we have such a codeword we can bootstrap from there very similarly to what has been done in [OT11, Subs. 4.4]. Note that this attack is actually very close in spirit to the attack that was devised on the KKS signature scheme [OT11]. In essence, the attack against the KKS scheme really amounts to recover the support of the V code. The difference with the KKS scheme is that the support of V is much bigger in our case. As explained in the conclusion of [OT11] the attack against the KKS scheme has in essence an exponential complexity. This exponent becomes really prohibitive in our case when the parameters of U and V are chosen appropriately as we will now explain.

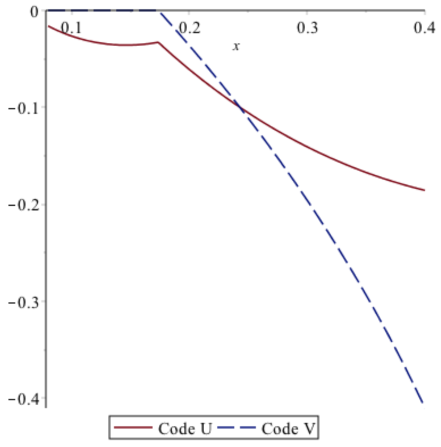


Fig. 5. $\alpha_{(U,U)}(w/n)$ and $\alpha_{(0,V)}(w/n)$ against $x \stackrel{\triangle}{=} \frac{w}{n}$.

7.2 Recovering the V code up to a permutation

The aforementioned attack recovers V up to some permutation of the positions. In a first step it recovers a basis of

$$V' \stackrel{\triangle}{=} (0, V)\mathbf{P} = \{(\mathbf{0}, \mathbf{v})\mathbf{P} : \mathbf{v} \in V\}.$$

Once this is achieved, the support $\text{Supp}(V')$ of V' can be obtained. Recall that this is the set of positions for which there exists at least one codeword of V' that is non-zero in this position. This allows to recover the code V up to some permutation. The basic algorithm for recovering the support of V' and a basis of V' is given in Algorithm 4.

Algorithm 4 ComputeV: algorithm that computes a set of independent elements in V'.

```
Parameters: (i) $\ell$ : small integer ($\ell \leqslant 40$),
(ii) $p$ : very small integer (typically $1 \leqslant p \leqslant 10$).
Input: (i) $\mathcal{C}_{\text{pub}}$ the public code used for verifying signatures.
(ii) $N$ a certain number of iterations
Output: an independent set of elements in $V'$
 1: function ComputeV($\mathcal{C}_{\text{pub}}$,$N$)
 2:     for $i = 1, \ldots, N$ do
 3:         $B \leftarrow \emptyset$
 4:         Choose a set $I \subset \{1, \ldots, n\}$ of size $n - k - \ell$ uniformly at random
 5:         $\mathcal{L} \leftarrow$ Codewords($\text{Punc}_I(\mathcal{C}_{\text{pub}}), p$)
 6:         for all $\mathbf{x} \in \mathcal{L}$ do
 7:             $\mathbf{x} \leftarrow$ Complete($\mathbf{x}, I, \mathcal{C}_{\text{pub}}$)
 8:             if CheckV($\mathbf{x}$) then
 9:                 add $\mathbf{x}$ to $B$ if $\mathbf{x} \notin < B >$
10:     return $B$
```

It uses other auxiliary functions

- Codewords($\text{Punc}_I(\mathcal{C}_{\text{pub}}), p$) which computes all (or a big fraction of) codewords of weight p of the punctured public code $\text{Punc}_I(\mathcal{C}_{\text{pub}})$. All modern [Dum91, FS09, MMT11, BJMM12, MO15] algorithms for decoding linear codes perform such a task in their inner loop.
- Complete($\mathbf{x}, I, \mathcal{C}_{\text{pub}}$) which computes the codeword $\mathbf{c}$ in $\mathcal{C}_{\text{pub}}$ such that its restriction outside I is equal to $\mathbf{x}$.
- CheckV($\mathbf{x}$) which checks whether $\mathbf{x}$ belongs to V'.

Choosing N appropriately. Let us first analyze how we have to choose N such that ComputeV returns $\Omega(1)$ elements. This is essentially the analysis which can be found in [OT11, Subsec 5.2]. This analysis leads to

Proposition 8. *The probability P_{succ} that one iteration of the for loop (Instruction 2) in Algorithm 4 adds elements to the list B is lower-bounded by*

$$P_{succ} \geq \sum_{w=0}^{n/2} \frac{\binom{n/2}{w}\binom{n/2}{n-k-\ell-w}}{\binom{n}{n-k-\ell}} f\left(\binom{n/2-w}{p} 2^{k_V+w-n/2}\right) \tag{12}$$

where f is the function defined by $f(x) \overset{\triangle}{=} \max\left(x(1-x/2), 1-\frac{1}{x}\right)$. Algorithm 4 returns a non zero list with probability $\Omega(1)$ when N is chosen as $N = \Omega\left(\frac{1}{P_{succ}}\right)$.

Proof. It will be helpful to recall [OT11, Lemma 3]

Lemma 4. *Choose a random code $\mathcal{C}_{rand}$ of length n from a parity-check matrix of size $r \times n$ chosen uniformly at random in $\mathbb{F}_2^{r\times n}$. Let X be some subset of $\mathbb{F}_2^n$ of size m. We have*

$$\mathbb{P}(X \cap \mathcal{C}_{rand} \neq \emptyset) \geq f\left(\frac{m}{2^r}\right).$$

To lower-bound the probability P_{succ} that an iteration is successful, we bring in the following random variables

$$I' \overset{\triangle}{=} I \cap \text{Supp}(I'') \quad \text{and} \quad W \overset{\triangle}{=} |I'|$$

where I'' is the set of positions that are of the images of the permutation $\mathbf{P}$ of the $n/2$ last positions. ComputeV outputs at least one element of V' if there is an element of weight p in $\text{Punc}_{I'}(V')$. Therefore the probability of success P_{succ} is given by

$$P_{\text{succ}} = \sum_{w=0}^{n/2} \mathbb{P}(W = w)\mathbb{P}\left(\exists \mathbf{x} \in V' : |\mathbf{x}_{\bar{I}'}| = p \mid W = w\right) \tag{13}$$

where $\bar{I}' \overset{\triangle}{=} \text{Supp}(V') \setminus I'$. On the other hand, by using Lemma 4 with the set

$$X \overset{\triangle}{=} \left\{\mathbf{x} = (x_j)_{j \in \text{Supp}(V')} : |\mathbf{x}_{\bar{I}'}| = p\right\}$$

which is of size $\binom{n/2-w}{p}2^w$, we obtain

$$\mathbb{P}\left(\exists \mathbf{x} \in V' : |\mathbf{x}_{\bar{I}'}| = p | W = w\right) \geq f(x). \tag{14}$$

with

$$x \overset{\triangle}{=} \frac{\binom{n/2-w}{p}2^w}{2^{n/2-k_V}} = \binom{n/2-w}{p} 2^{k_V+w-n/2}$$

The first quantity is clearly equal to

$$\mathbb{P}(W = w) = \frac{\binom{n/2}{w}\binom{n/2}{n-k-\ell-w}}{\binom{n}{n-k-\ell}}. \tag{15}$$

Plugging in the expressions obtained in (14) and (15) in (13) we have an explicit expression of a lower bound on P_{succ}

$$P_{\text{succ}} \geq \sum_{w=0}^{n/2} \frac{\binom{n/2}{w}\binom{n/2}{n-k-\ell-w}}{\binom{n}{n-k-\ell}} f\left(\binom{n/2-w}{p} 2^{k_V+w-n/2}\right) \tag{16}$$

The claim on the number N of iterations follows directly from this. □

Complexity of recovering a permuted version of V. The complexity of a call to ComputeV can be estimated as follows. The complexity of computing the list of codewords of weight p in a code of length $k+\ell$ and dimension k is equal to $C_1(p,\ell)$ (this quantity is introduced in §6). It depends on the particular algorithm used here [Dum91, FS09, MMT11, BJMM12, MO15]. This is the complexity of the call Codewords$(\text{Punc}_I(\mathcal{C}_{\text{pub}}), p)$ in Step 5 in Algorithm 4. The complexity of ComputeV and hence the complexity of recovering a permuted version of V is clearly lower bounded by $\Omega\left(\frac{C_1(p,\ell)}{P_{\text{succ}}}\right)$. It turns out that the whole complexity of recovering a permuted version of V is actually of this order, namely $\Theta\left(\frac{C_1(p,\ell)}{P_{\text{succ}}}\right)$. This can be done by a combination of two techniques

- Once a non-zero element of V' has been identified, it is much easier to find other ones. This uses one of the tricks for breaking the KKS scheme (see [OT11, Subs. 4.4]). The point is the following: if we start again the procedure ComputeV, but this time by choosing a set I on which we puncture the code which contains the support of the codeword that we already found, then the number N of iterations that we have to perform until finding a new element is negligible when compared to the original value of N.
- The call to CheckV can be implemented in such a way that the additional complexity coming from all the calls to this function is of the same order as the N calls to Codewords. The strategy to adopt depends on the values of the dimensions k and k_V. In certain cases, it is easy to detect such codewords since they have a typical weight that is significantly smaller than the other codewords. In more complicated cases, we might have to combine a technique checking first the weight of $\mathbf{x}$, if it is above some prescribed threshold, we decide that it is not in V', if it is below the threshold, we decide that it is a suspicious candidate and use then the previous trick. We namely check whether the support of the codeword $\mathbf{x}$ can be used to find other suspicious candidates much more quickly than performing N calls to CheckV.

To keep the length of this paper within some reasonable limit we avoid here giving the analysis of those steps and we will just use the aforementioned lower bound on the complexity of recovering a permuted version of V.

7.3 Recovering the U code up to permutation

We consider here the permuted code

$$U' \stackrel{\triangle}{=} (U,U)\mathbf{P} = \{(\mathbf{u},\mathbf{u})\mathbf{P} : \mathbf{u} \in U\}.$$

The attack in this case consists in recovering a basis of U'. Once this is done, it is easy to recover the U code up to permutation by matching the pairs of coordinates which are equal in U'. The algorithm for recovering U' is the same as the algorithm for recovering V'. We call the associated function ComputeU though since they differ in the choice for N. The analysis is slightly different indeed.

Choosing N appropriately. As in the previous subsection let us analyze how we have to choose N in order that ComputeU returns $\Omega(1)$ elements of U'. We have in this case the following result.

Proposition 9. *The probability P_{succ} that one iteration of the for loop (Instruction 2) in ComputeU adds elements to the list B is lower-bounded by*

$$P_{succ} \geq \sum_{w=0}^{n/2} \frac{\binom{n/2}{w}\binom{n/2-w}{k+\ell-2w}2^{k+\ell-2w}}{\binom{n}{k+\ell}} \max_{i=0}^{\lfloor p/2 \rfloor} f\left(\frac{\binom{k+\ell-2w}{p-2i}\binom{w}{i}}{2^{\max(0,k+\ell-w-k_U)}}\right) \tag{17}$$

where f is the function defined by $f(x) \stackrel{\triangle}{=} \max\left(x(1-x/2), 1-\frac{1}{x}\right)$. ComputeU returns a non zero list with probability $\Omega(1)$ when N is chosen as $N = \Omega\left(\frac{1}{P_{succ}}\right)$.

Proof. Here the crucial notion is the concept of *matched positions*. We say that two positions i and j are matched if and only if $c_i = c_j$ for every $\mathbf{c} \in U'$. There are clearly $n/2$ pairs of matched positions. W will now be defined by the number of matched pairs that are included in $\{1, \dots, n\} \setminus I$. We compute the probability of success as before by conditioning on the values taken by W:

$$P_{\text{succ}} = \sum_{w=0}^{n/2} \mathbb{P}(W = w) \mathbb{P}\left(\exists \mathbf{x} \in U' : |\mathbf{x}_{\bar{I}}| = p \mid W = w\right) \tag{18}$$

where $\bar{I} \overset{\triangle}{=} \{1, \dots, n\} \setminus I$. Notice that we can partition $\bar{I}$ as $\bar{I} = J_1 \cup J_2$ where J_2 consists in the union of the matched pairs in $\bar{I}$. Note that $|J_2| = 2w$. We may further partition J_2 as $J_2 = J_{21} \cup J_{22}$ where the elements of a matched pair are divided into the two sets. In other words, neither J_{21} nor J_{22} contains a matched pair. We are going to consider the codes

$$U" \overset{\triangle}{=} \underset{I}{\text{Punc}}(U')$$
$$U''' \overset{\triangle}{=} \underset{I \cup J_{22}}{\text{Punc}}(U')$$

The last code is of length $k + \ell - w$. The point of defining the first code is that

$$\mathbb{P}\left(\exists \mathbf{x} \in U' : |\mathbf{x}_{\bar{I}}| = p \mid W = w\right)$$

is equal to the probability that $U"$ contains a codeword of weight p. The problem is that we can not apply Lemma 4 to it due to the matched positions it contains. This is precisely the point of defining U'''. In this case, we can consider that it is a random code whose parity-check matrix is chosen uniformly at random among the set of matrices of size $\max(0, k + \ell - w - k_U) \times (k + \ell - w)$. We can therefore apply Lemma 4 to it. We have to be careful about the words of weight p in $U"$ though, since they do not have the same probability of occurring in $U"$ due to the possible presence of matched pairs in the support. This is why we introduce for i in $\{0, \dots, \lfloor p/2 \rfloor\}$ the sets X_i defined as follows

$$X_i \overset{\triangle}{=} \{\mathbf{x} = (x_i)_{i \in \bar{I} \setminus J_{22}} \mathbb{F}_2^{k+\ell-w} : |\mathbf{x}_{J_1}| = p - 2i,\ |\mathbf{x}_{J_{21}}| = i\}$$

A codeword of weight p in $U"$ corresponds to some word in one of the X_i's by puncturing it in J_{22}. We obviously have the lower bound

$$\mathbb{P}\left\{\exists \mathbf{x} \in U' : |\mathbf{x}_{\bar{I}}| = p \mid W = w\right\} \geq \max_{i=0}^{\lfloor p/2 \rfloor} \left\{\mathbb{P}(X_i \cap U''' \neq \emptyset)\right\} \tag{19}$$

By using Lemma 4 we have

$$\mathbb{P}(X_i \cap U''' \neq \emptyset) \geq f\left(\frac{\binom{k+\ell-2w}{p-2i}\binom{w}{i}}{2^{\max(0, k+\ell-w-k_U)}}\right). \tag{20}$$

On the other hand, we may notice that $\mathbb{P}(W = w) = \mathbb{P}(w_2(\mathbf{e}) = w)$ when $\mathbf{e}$ is drawn uniformly at random among the binary words of weight $k + \ell$ and length n. By using Proposition 10 we deduce

$$\mathbb{P}(W = w) = \frac{\binom{n/2}{w}\binom{n/2-w}{k+\ell-2w}2^{k+\ell-2w}}{\binom{n}{k+\ell}}.$$

These considerations lead to the following lower bound on P_{succ}

$$P_{\text{succ}} \geq \sum_{w=0}^{n/2} \frac{\binom{n/2}{w}\binom{n/2-w}{k+\ell-2w}2^{k+\ell-2w}}{\binom{n}{k+\ell}} \max_{i=0}^{\lfloor p/2 \rfloor} f\left(\frac{\binom{k+\ell-2w}{p-2i}\binom{w}{i}}{2^{\max(0, k+\ell-w-k_U)}}\right) \tag{21}$$

□

Complexity of recovering a permuted version of $\boldsymbol{U}$. As for recovering the permuted V code, the complexity for recovering the permuted U is of order $\Omega\left(\frac{C_1(p,\ell)}{P_{\text{succ}}}\right)$.

7.4 Distinguishing a $(\boldsymbol{U}, \boldsymbol{U}+\boldsymbol{V})$ code

It is not clear in the first case that from the single knowledge of V' and a permuted version of V we are able to find a permutation of the positions which gives to the whole code the structure of a $(U, U+V)$-code. However in both cases as single successful call to ComputeV (resp. ComputeU) is really distinguishing the code from a random code of the same length and dimension. In other words, we have a distinguishing attack whose complexity is given by $\min(O(C_U), O(C_V))$ where

$$C_U \overset{\triangle}{=} \frac{C_1(p,\ell)}{\sum_{w=0}^{n/2} \frac{\binom{n/2}{w}\binom{n/2-w}{k+\ell-2w}2^{k+\ell-2w}}{\binom{n}{k+\ell}} \max_{i=0}^{\lfloor p/2 \rfloor} f\left(\frac{\binom{k+\ell-2w}{p-2i}\binom{w}{i}}{2^{\max(0,k+\ell-w-k_U)}}\right)}$$

$$C_V \overset{\triangle}{=} \frac{C_1(p,\ell)}{\sum_{w=0}^{n/2} \frac{\binom{n/2}{w}\binom{n/2}{n-k-\ell-w}}{\binom{n}{n-k-\ell}} f\left(\binom{n/2-w}{p}2^{k_V+w-n/2}\right)}$$

and $f(x) \overset{\triangle}{=} \max\left(x(1-x/2), 1-1/x\right)$. As for the decoders of §6 the above numbers are minimized (independently) over p and ℓ.

We end this section by remarking that the dual of a code $(U, U+V)$ is $(U^\perp + V^\perp, V^\perp)$ thus we have the same attack with the dual. With $k/n = 0.5$, these two attacks have the same complexity as $C_U = C_{V^\perp}$ and $C_V = C_{U^\perp}$.

7.5 NP-completeness

Algorithm 4 does not recover the complete structure of a $(U, U+V)$-code. However it reveals the support of the V-code. This allows to reorder the positions of the public code such that the first half is a permutation of the first $n/2$ positions of the $(U, U+V)$-code whereas the second half is a permutation of the $n/2$ last positions of the $(U, U+V)$-code. Note that we just have to reorder the second half so that it corresponds to a valid $(U, U+V)$ code (where the new U-code is a permutation of the old U-code). In other words the problem we have to solve is the following.

Problem 3. Let U and V be two binary linear codes of length $n/2$ and let π be a permutation of length $n/2$. Let

$$(U, \pi(U)+V) \overset{\triangle}{=} \{(\mathbf{u}, \pi(\mathbf{u})+\mathbf{v}), \mathbf{u} \in U, \mathbf{v} \in V\}.$$

Find a permutation π' acting on the right-hand part which gives to $(U, \pi(U)+V)$ the structure of a $(U, U+V)$-code, i.e. find π' permutation of length $n/2$ such that $(U, \pi'(\pi(U)+V)) = (U, U+V')$ for a certain binary linear code V', where

$$(U, \pi'(\pi(U)+V)) \overset{\triangle}{=} \{(\mathbf{u}, \pi'(\pi(\mathbf{u})+\mathbf{v})), \mathbf{u} \in U, \mathbf{v} \in V\}.$$

Remark 6. $\pi' = \pi^{-1}$ is a solution to this problem but there might be other solutions of course.

The decision problem which is related to this search version is the following.

Problem 4 (Problem P2': weak $(U, U+V)$-distinguishing). Consider a binary linear code $\mathcal{C}$ of length n where n is even. Do there exist two binary linear codes U and V of length $n/2$ and a permutation π of length $n/2$ such that

$$(U, U+V) = \{(\mathbf{x}, \pi(\mathbf{y})) \ : \ (\mathbf{x}, \mathbf{y}) \in \mathcal{C}, \mathbf{x} \in \mathbb{F}_2^{n/2}, \mathbf{y} \in \mathbb{F}_2^{n/2}\}.$$

This problem can be viewed as Problem P1 where we have some side information available where we have been revealed the split of the support of the $(U, U+V)$ construction in the left and the right part, but the left part and the right part have been permuted "internally". It turns out that this decision problem is already NP-complete

Theorem 3. *The weak-$(U, U+V)$ distinguishing Problem P2' is an NP-complete problem.*

The proof of this theorem is given in the appendix.

8 Parameter Selection

In the light of the security proof in §4 and the rejection sampling method in §5, we need to derive parameters which lead to negligible success for the two following problems:

1. Solve a syndrome decoding problem with multiple instances (DOOM) for parameters n, k, w and an arbitrarily large number of instances.
2. Distinguish public matrices of the code family $(U, U+V)$ from random matrices of same size.

In the security proof we required a salt size $\lambda_0 = \lambda + 2\log_2(q_{\text{sign}})$ where q_{sign} is the number of signature queries allowed to the adversary. Since $q_{\text{sign}} \leq 2^\lambda$ (λ the security parameter) we choose a conservative $\lambda_0 = 3\lambda$. We gave in §6 and §7 state-of-the-art algorithms for the two problems mentioned above. This served as a basis for the parameters proposed in Table 2. For the key security, the estimates C_U and C_V are derived from the formulas at the end of §7. In those formulas the $C_1(p, \ell)$ term derives from Dumer's algorithm. Using more involved techniques [MMT11, BJMM12, MO15] will reduce the key security but will leave it above the security claims. For the message security ($\log_2 \mathrm{WF}$), it is based on the DOOM variant of Dumer's algorithm, which is the current state-of-the-art. Algorithmic improvements, like adapting DOOM to BJMM, may lower the message security and require an adjustment of the sizes.

λ (security)	80	128	256
n	4800	7700	15400
$k = k_U + k_V$	2400	3850	7700
k_V	916	1470	2940
w	916	1470	2940
Signature length (bits)	4940	8084	16168
Public key size (MBytes)	0.720	1.853	7.411
Secret key size (MBytes)	0.347	0.887	3.525
$\log_2(q_{\text{hash}}\sqrt{\varepsilon})$ (Prop. 5 §4.2)	-208	-334	-668
$\log_2 C_V$ (§7)	171	275	550
$\log_2 C_U$ (§7)	250	401	803
$\log_2 \mathrm{WF}$ (§6)	80	128	256

Table 2. Proposed Parameters for the $(U, U+V)$ Signature Scheme

Key Sizes. The public key is a parity-check matrix, given in systematic form $\mathbf{H}_{\text{pub}} = (\mathbf{I} \mid \mathbf{R})$ which requires $k(n-k)$ bits. The secret key consists of a non-singular matrix $\mathbf{S}$, a secret parity-check matrix $\mathbf{H}_{\text{sec}}$ and a permutation matrix $\mathbf{P}$ where

$$\mathbf{H}_{\text{pub}} = \mathbf{S}\mathbf{H}_{\text{sec}}\mathbf{P} \text{ with } \mathbf{H}_{\text{sec}} = \begin{pmatrix} \mathbf{H}_U & \mathbf{0} \\ \mathbf{H}_V & \mathbf{H}_V \end{pmatrix},$$

and $\mathbf{H}_U$ and $\mathbf{H}_V$ are parity-check matrices of U and V. Here, we not not need to include $\mathbf{S}$ in the secret key, it is used in the signature to compute $\mathbf{S}^{-1}\mathbf{s}^T$ which can be derived from $\mathbf{H}_{\text{sec}}$ and $\mathbf{P}$ when the public key is systematic. We have

$$\mathbf{S}^{-1}\mathbf{s}^T = \mathbf{S}^{-1}(\mathbf{I} \mid \mathbf{R}) \begin{pmatrix} \mathbf{s}^T \\ \mathbf{0} \end{pmatrix} = \mathbf{S}^{-1}\mathbf{H}_{\text{pub}} \begin{pmatrix} \mathbf{s}^T \\ \mathbf{0} \end{pmatrix} = \mathbf{H}_{\text{sec}}\mathbf{P} \begin{pmatrix} \mathbf{s}^T \\ \mathbf{0} \end{pmatrix}.$$

The secret thus consists of $\mathbf{P}$, that is stored with $\log(n!) \leq n\ln(n)$ bits, and $\mathbf{H}_U, \mathbf{H}_V$ in systematic form, that is stored with $k_U(n/2 - k_U) + k_V(n/2 - k_V)$ bits.

Implementation. In Table 2 the ratio w/n is chosen close to 0.191 to minimize the rejection probability (see §5). For the three security levels we need to perform on average 27, 37, or 75 Gaussian eliminations to produce a signature. Most of those Gaussian elimination are performed on parity-check matrices of shortened codes. Finally, let us mention that the signature length ($n + 3\lambda$ in the table) can be reduced (by about 30%) by choosing a compact representation of the sparse error vector.

9 Concluding remarks

We have presented the first code-based signature scheme whose security parameter scales polynomially in key size. By code-based scheme, we mean here the restricted case of the Hamming metric for expressing the decoding problem. This setting presents the advantage that we are in the case where the decoding problem has been thoroughly studied for many decades and where it can be considered that the complexity of the best known attacks has not dramatically changed since the early sixties.

Comparison with TESLA-2. Contrarily to the overwhelming majority of lattice-based or code-based proposals during the last decade, we avoided here the use of structured codes based on a ring structure (such as quasi-cyclic codes). The entropy of our public key is a constant fraction of the entropy of a random matrix of the same size. Note that the entropy loss in our case is much lower than the one observed for the aforementioned structured cases. This is the first code-based signature scheme where such a low entropy loss in the public key has been obtained. With a parameter selection matching tightly with the security reduction, the public key size stays reasonable: less than 2 megabytes (MB) for 128 bits of classical security and less than 6 MB with the QROM reduction of [CD17] for 128 bits of quantum security. This is strongly related to the tightness of our security reduction. There are no other (Hamming metric) code-based to compare with our scheme. The closest scheme we can compare with is TESLA-2 [ABB$^+$17] which is an unstructured lattice based scheme that has a quantum security reduction in the QROM too. The public key sizes are much bigger in their case: almost 22 MB for the same level of quantum security or more than 11 MB for the same level of classical security.

Problem P2: the $(U, U+V)$-code distinguishing problem. The second problem on which our security relies is indeed very clean and simple. It consists in deciding whether there exists for a given linear code, a permutation of its positions that makes it a $(U, U+V)$ code. This simplicity makes the problem really appealing in a cryptographic context. It departs from the actual trend in code-based or lattice-based cryptography which relies solely on the difficulty of decoding or deciding whether or not a code/lattice contains codewords/lattice elements of low weight/norm. Despite its simplicity, there are strong reasons to believe in the hardness of this problem. Even weak versions of this problem are NP-complete. For instance, even the restricted problem of deciding whether there exists a permutation of the second half of the code positions which makes the code to be a $(U, U+V)$-code is NP-complete (see Theorem 3). The fact that $(U, U+V)$ codes seem to depart from random codes solely by their lowest/highest weight codewords and the fact that the best algorithms to date for solving this problem use low weight codeword finding algorithms might indicate that there might be a tight connection of this problem to the problem of finding low weight codewords in a code. This is a tantalizing connection which is worth investigating.

Acknowledgments

We would thank André Chailloux for a careful reading of a preliminary version of this paper [DST17], for finding an error in the security proof and for suggesting Proposition 5 to fix this error.

References

[ABB$^+$17] Erdem Alkim, Nina Bindel, Johannes A. Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Krämer, and Filip Pawlega. Revisiting TESLA in the quantum random oracle model. In *Post-Quantum Cryptography 2017*, volume 10346 of *LNCS*, pages 143–162, Utrecht, The Netherlands, June 2017. Springer.

[Arı09] Erdal Arıkan. Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans. Inform. Theory*, 55(7):3051–3073, 2009.

[Bar97] Alexander Barg. Complexity issues in coding theory. *Electronic Colloquium on Computational Complexity*, October 1997.

[BBC$^+$13] Marco Baldi, Marco Bianchi, Franco Chiaraluce, Joachim Rosenthal, and Davide Schipani. Using LDGM codes and sparse syndromes to achieve digital signatures. In *Post-Quantum Cryptography 2013*, volume 7932 of *LNCS*, pages 1–15. Springer, 2013.

[BCD$^+$16] Magali Bardet, Julia Chaulet, Vlad Dragoi, Ayoub Otmani, and Jean-Pierre Tillich. Cryptanalysis of the McEliece public key cryptosystem based on polar codes. In *Post-Quantum Cryptography2016*, LNCS, pages 118–143, Fukuoka, Japan, February 2016.

[BDK$^+$11] Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, François-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 1–20, 2011.

[Ber10] Daniel J. Bernstein. Grover vs. McEliece. In Nicolas Sendrier, editor, *Post-Quantum Cryptography 2010*, volume 6061 of *LNCS*, pages 73–80. Springer, 2010.

[BGK17] Thierry P. Berger, Cheikh Thiécoumba Gueye, and Jean Belo Klamti. A np-complete problem in coding theory with application to code based cryptography. In *Codes, Cryptology and Information Security - Second International Conference, C2SI 2017, Rabat, Morocco, April 10-12, 2017, Proceedings - In Honor of Claude Carlet*, pages 230–237, 2017.

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012*, LNCS. Springer, 2012.

[BMS11] Paulo S.L.M Barreto, Rafael Misoczki, and Marcos A. Jr. Simplicio. One-time signature scheme from syndrome decoding over generic error-correcting codes. *Journal of Systems and Software*, 84(2):198–204, 2011.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73, 1993.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures-how to sign with rsa and rabin. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *LNCS*, pages 399–416. Springer, 1996.

[CD17] André Chailloux and Thomas Debris-Alazard. Tight security reduction in the quantum random oracle model for code-based signature schemes. preprint, September 2017. arXiv:1709.06870.

[CFS01] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 157–174, Gold Coast, Australia, 2001. Springer.

[CJ04] Jean-Sebastien Coron and Antoine Joux. Cryptanalysis of a provably secure cryptographic hash function. IACR Cryptology ePrint Archive, Report 2004/013, 2004. `http://eprint.iacr.org/`.

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, pages 272–287, 2002.

[COV07] Pierre-Louis Cayrel, Ayoub Otmani, and Damien Vergnaud. On Kabatianskii-Krouk-Smeets signatures. In *Arithmetic of Finite Fields - WAIFI 2007*, volume 4547 of *LNCS*, pages 237–251, Madrid, Spain, June 21–22 2007.

[CTS16] Rodolfo Canto-Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In *Post-Quantum Cryptography 2016*, LNCS, pages 144–161, Fukuoka, Japan, February 2016.

[DS06] Ilya Dumer and Kirill Shabunov. Soft-decision decoding of Reed-Muller codes: recursive lists. *IEEE Trans. Inform. Theory*, 52(3):1260–1266, 2006.

[DST17] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. A new signature scheme based on $(U|U + V)$ codes. preprint, June 2017. arXiv:1706.08065v1.

[DT17] Thomas Debris-Alazard and Jean-Pierre Tillich. Statistical decoding. preprint, January 2017. arXiv:1701.07416.

[Dum91] Ilya Dumer. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, pages 50–52, Moscow, 1991.

[FGO$^+$11] Jean-Charles Faugère, Valérie Gauthier, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A distinguisher for high rate McEliece cryptosystems. In *Proc. IEEE Inf. Theory Workshop- ITW 2011*, pages 282–286, Paraty, Brasil, October 2011.

[FGO$^+$13] Jean-Charles Faugère, Valérie Gauthier, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A distinguisher for high rate McEliece cryptosystems. *IEEE Trans. Inform. Theory*, 59(10):6830–6844, October 2013.

[Fin10] Matthieu Finiasz. Parallel-CFS - strengthening the CFS McEliece-based signature scheme. In *Selected Areas in Cryptography 17th International Workshop, 2010, Waterloo, Ontario, Canada, August 12-13, 2010, revised selected papers*, volume 6544 of *LNCS*, pages 159–170. Springer, 2010.

[FS09] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 88–105. Springer, 2009.

[GHPT17] Philippe Gaborit, Adrien Hauteville, Duong Hieu Phan, and Jean-Pierre Tillich. Identity-based encryption from rank metric. In *Advances in Cryptology - CRYPTO2017*, volume 10403 of *LNCS*, pages 194–226. Springer, August 2017.

[GRSZ14] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. New results for rank-based cryptography. In *Progress in Cryptology - AFRICACRYPT 2014*, volume 8469 of *LNCS*, pages 1–12, 2014.

[GS12] Philippe Gaborit and Julien Schrek. Efficient code-based one-time signature from automorphism groups with syndrome compatibility. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2012*, pages 1982–1986, Cambridge, MA, USA, July 2012.

[GSJB14] Danilo Gligoroski, Simona Samardjiska, Håkon Jacobsen, and Sergey Bezzateev. McEliece in the world of Escher. IACR Cryptology ePrint Archive, Report2014/360, 2014. `http://eprint.iacr.org/`.

[JJ02] Thomas Johansson and Fredrik Jönsson. On the complexity of some cryptographic problems based on the general decoding problem. *IEEE Trans. Inform. Theory*, 48(10):2669–2678, October 2002.

[KKS97] Gregory Kabatianskii, Evgenii Krouk, and Ben. J. M. Smeets. A digital signature scheme based on random error-correcting codes. In *IMA Int. Conf.*, volume 1355 of *LNCS*, pages 161–167. Springer, 1997.

[KKS05] Gregory Kabatianskii, Evgenii Krouk, and Sergei Semenov. *Error Correcting Coding and Security for Data Networks: Analysis of the Superchannel Concept.* John Wiley & Sons, 2005.

[Kor09] Satish Babu Korada. *Polar Codes for Channel and Source Coding.* PhD thesis, 'Ecole Polytechnique Fédérale de Lausanne (EPFL), July 2009.

[KT17] Ghazal Kachigar and Jean-Pierre Tillich. Quantum information set decoding algorithms. preprint, arXiv:1703.00263 [cs.CR], February 2017.

[LJ12] Carl Löndahl and Thomas Johansson. A new version of McEliece PKC based on convolutional codes. In *Information and Communications Security, ICICS*, volume 7168 of *LNCS*, pages 461–470. Springer, 2012.

[LT13] Grégory Landais and Jean-Pierre Tillich. An efficient attack of a McEliece cryptosystem variant based on convolutional codes. In P. Gaborit, editor, *Post-Quantum Cryptography'13*, volume 7932 of *LNCS*, pages 102–117. Springer, June 2013.

[MCT16a] Irene Márquez-Corbella and Jean-Pierre Tillich. Using Reed-Solomon codes in the $(u|u+v)$ construction and an application to cryptography. preprint, 2016. arXiv:1601:08227.

[MCT16b] Irene Márquez-Corbella and Jean-Pierre Tillich. Using Reed-Solomon codes in the $(u|u+v)$ construction and an application to cryptography. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 930–934, 2016. for a full version see, arXiv:1601:08227.

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $O(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, 2011.

[MO15] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 203–228. Springer, 2015.

[MS09] L. Minder and A. Sinclair. The extended k-tree algorithm. In C. Mathieu, editor, *Proceedings of SODA 2009*, pages 586–595. SIAM, 2009.

[Nie86] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.

[OT11] Ayoub Otmani and Jean-Pierre Tillich. An efficient attack on all concrete KKS proposals. In *Post-Quantum Cryptography 2011*, volume 7071 of *LNCS*, pages 98–116, 2011.

[PMIB17] Sven Puchinger, Sven Müelich, Karim Ishak, and Martin Bossert. Code-based cryptosystems using generalized concatenated codes. In Ilias S. Kotsireas and Edgar Martínez-Moro, editors, *Applications of Computer Algebra, ACA 2015*, volume 198 of *Proceedings in Mathematics & Statistics*, pages 397–423, Kalamata, Greece, 2017. Springer.

[Pra62] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[PT16] Aurélie Phesso and Jean-Pierre Tillich. An efficient attack on a code-based signature scheme. In *Post-Quantum Cryptography 2016*, volume 9606 of *LNCS*, pages 86–103, Fukuoka, Japan, February 2016. Springer.

[Sen11] Nicolas Sendrier. Decoding one out of many. In *Post-Quantum Cryptography 2011*, volume 7071 of *LNCS*, pages 51–67, 2011.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

[Ste88] Jacques Stern. A method for finding codewords of small weight. In G. D. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *LNCS*, pages 106–113. Springer, 1988.

[Ste93] Jacques Stern. A new identification scheme based on syndrome decoding. In D.R. Stinson, editor, *Advances in Cryptology - CRYPTO'93*, volume 773 of *LNCS*, pages 13–21. Springer, 1993.

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, 2002.

A Proofs for §4

A.1 List Emulation

In the security proof, we need to build lists of indices (salts) in $\mathbb{F}_2^{\lambda_0}$. Those lists have size q_{sign}, the maximum number of signature queries allowed to the adversary, a number which is possibly very large. For each message $\mathbf{m}$ which is either hashed or signed in the game we need to be able to

- create a list $L_{\mathbf{m}}$ of q_{sign} random elements of $\mathbb{F}_2^{\lambda_0}$, when calling the constructor `new list()`;
- pick an element in $L_{\mathbf{m}}$, using the method $L_{\mathbf{m}}$.`next`(), this element can be picked only once;
- decide whether or not a given salt $\mathbf{r}$ is in $L_{\mathbf{m}}$, when calling $L_{\mathbf{m}}$.`contains`($\mathbf{r}$).

The straightforward manner to achieve this is to draw q_{sign} random numbers when the list is constructed, this has to be done once for each different message $\mathbf{m}$ used in the game. This may result in a quadratic cost $q_{\text{hash}}q_{\text{sign}}$ just to build the lists. Once the lists are constructed, and assuming they are stored in a proper data structure (a heap for instance) picking an element or testing membership has a cost at most $O(\log q_{\text{sign}})$, that is at most linear in the security parameter λ.

```
class list
  elt, index
  list()
    index ← 0
    for $i = 1, \dots, q_{\text{sign}}$
      elt[$i$] ← randint($2^{\lambda_0}$)

method list.contains($\mathbf{r}$)
  return $\mathbf{r} \in \{\texttt{elt}[i], 1 \leq i \leq q_{\text{sign}}\}$

method list.next()
  index ← index + 1
  return elt[index]
```

Fig. 6. Standard implementation of the list operations.

Note that in our game we condition on the event that *all elements of* $L_{\mathbf{m}}$ *are different*. This implies that now $L_{\mathbf{m}}$ is obtained by choosing among the subsets of size q_{sign} of $\mathbb{F}_2^{\lambda_0}$ uniformly at random. We wish to emulate the list operations and never construct them explicitly such that the probabilistic model for $L_{\mathbf{m}}$.`next`() and $L_{\mathbf{m}}$.`contains`($\mathbf{r}$) stays the same as above (but again conditioned on the event that all elements of $L_{\mathbf{m}}$ are different). For this purpose, we want to ensure that at any time we call either $L_{\mathbf{m}}$.`contains`($\mathbf{r}$) or $L_{\mathbf{m}}$.`next`() we have

$$\mathbb{P}(L_{\mathbf{m}}.\texttt{contains}(\mathbf{r}) = \texttt{true}) = \mathbb{P}(\mathbf{r} \in L_{\mathbf{m}} | \mathcal{Q}) \tag{22}$$

$$\mathbb{P}(\mathbf{r} = L_{\mathbf{m}}.\texttt{next}()) = p(\mathbf{r}|\mathcal{Q}) \tag{23}$$

for every $\mathbf{r} \in \mathbb{F}_2^{\lambda_0}$. Here $\mathcal{Q}$ represents the queries to $\mathbf{r}$ made so far and whether or not these $\mathbf{r}$'s belong to $L_{\mathbf{m}}$. Queries to $\mathbf{r}$ can be made through two different calls. The first one is a call of the form `Sign`($\mathbf{m}$) when it chooses $\mathbf{r}$ during the random assignment $\mathbf{r} \hookleftarrow \{0,1\}^{\lambda_0}$. This results in a call to `Hash`($\mathbf{m}, \mathbf{r}$) which queries itself whether $\mathbf{r}$ belongs to $L_{\mathbf{m}}$ or not through the call $L_{\mathbf{m}}$.`contains`($\mathbf{r}$). The answer is necessarily positive in this case. The second way to query $\mathbf{r}$ is by calling `Hash`($\mathbf{m}, \mathbf{r}$) directly. In this case, both answers `true` and `false` are possible. $p(\mathbf{r}|\mathcal{Q})$ represents the probability distribution of $L_{\mathbf{m}}$.`next`() that we have in the above implementation of the list operations given the previous queries $\mathcal{Q}$.

A convenient way to represent $\mathcal{Q}$ is through three lists S, H_{true} and H_{false}. S is the list of $\mathbf{r}$'s that have been queried through a call `Sign`($\mathbf{m}$). They belong necessarily to $L_{\mathbf{m}}$. H_{true} is the set of $\mathbf{r}$'s that have not been queried so far through a call to `Sign`($\mathbf{m}$) but have been queried through a direct call `Hash`($\mathbf{m}, \mathbf{r}$) and for which $L_{\mathbf{m}}$.`contains`($\mathbf{r}$) returned `true`. H_{false} is the list of $\mathbf{r}$'s that have been queried by a call of the form `Hash`($\mathbf{m}, \mathbf{r}$) and $L_{\mathbf{m}}$.`contains`($\mathbf{r}$) returned `false`.

We clearly have

$$\mathbb{P}(\mathbf{r} \in L_{\mathbf{m}} | \mathcal{Q}) = 0 \text{ if } \mathbf{r} \in H_{\text{false}} \tag{24}$$

$$\mathbb{P}(\mathbf{r} \in L_{\mathbf{m}} | \mathcal{Q}) = 1 \text{ if } \mathbf{r} \in S \cup H_{\text{true}} \tag{25}$$

$$\mathbb{P}(\mathbf{r} \in L_{\mathbf{m}} | \mathcal{Q}) = \frac{q_{\text{sign}} - |H_{\text{true}}| - |S|}{2^{\lambda_0} - |H_{\text{true}}| - |S| - |H_{\text{false}}|} \text{ else.} \tag{26}$$

To compute the probability distribution $p(\mathbf{r}|\mathcal{Q})$ it is helpful to notice that

$$\mathbb{P}(L_{\mathbf{m}}.\texttt{next}() \text{ outputs an element of } H_{\text{true}}\ ) = \frac{|H_{\text{true}}|}{q_{\text{sign}} - |S|}. \tag{27}$$

This can be used to derive $p(\mathbf{r}|\mathcal{Q})$ as follows

$$p(\mathbf{r}|\mathcal{Q}) = 0 \text{ if } \mathbf{r} \in H_{\text{false}} \cup S \tag{28}$$

$$p(\mathbf{r}|\mathcal{Q}) = \frac{1}{q_{\text{sign}} - S} \text{ if } \mathbf{r} \in H_{\text{true}} \tag{29}$$

$$p(\mathbf{r}|\mathcal{Q}) = \frac{q_{\text{sign}} - |S| - |H_{\text{true}}|}{(q_{\text{sign}} - S)(2^{\lambda_0} - |H_{\text{true}}| - |S| - |H_{\text{false}}|)} \text{ else.} \tag{30}$$

(28) is obvious. (29) follows from that all elements of H_{true} have the same probability to be chosen as return value for $L_{\mathbf{m}}.\texttt{next}()$ and (27). (30) follows by a similar reasoning by arguing (i) that all the elements of $\mathbb{F}_2^{\lambda_0} \setminus (S \cup H_{\text{true}} \cup H_{\text{false}})$ have the same probability to be chosen as return value for $L_{\mathbf{m}}.\texttt{next}()$, (ii) the probability that $L_{\mathbf{m}}.\texttt{next}()$ outputs an element of $\mathbb{F}_2^{\lambda_0} \setminus (S \cup H_{\text{true}} \cup H_{\text{false}})$ is the probability that it does not output an element of H_{true} which is $1 - \frac{|H_{\text{true}}|}{q_{\text{sign}} - |S|} = \frac{q_{\text{sign}} - |S| - |H_{\text{true}}|}{q_{\text{sign}} - |S|}$.

Figure 7 explains how we perform the emulation of the list operations so that they perform similarly to genuine list operations as specified above. The idea is to create and to operate explicitly on the lists S, H_{true} and H_{false} described earlier. We have chosen there

$$\beta = \frac{q_{\text{sign}} - |H_{\text{true}}| - |S|}{2^{\lambda_0} - |H_{\text{true}}| - |S| - |H_{\text{false}}|} \text{ and } \gamma = \frac{|H_{\text{true}}|}{q_{\text{sign}} - |S|}.$$

we also assume that when we call `randomPop()` on a list it outputs an element of the list uniformly at random and removes this element from it. The method `push` adds an element in a list. The procedure `rand`() picks a real number between 0 and 1 uniformly at random.

class list	method list.contains($\mathbf{r}$)	method list.next()
H_{true}, H_{false}, S	if $\mathbf{r} \notin H_{\text{true}} \cup H_{\text{false}} \cup S$	if `rand`() $\leq \gamma$
list()	if rand() $\leq \beta$	$\mathbf{r} \leftarrow H_{\text{true}}$.`randomPop`()
$H_{\text{true}} \leftarrow \emptyset$	H_{true}.`push`($\mathbf{r}$)	else
$H_{\text{false}} \leftarrow \emptyset$	else	$\mathbf{r} \hookleftarrow \mathbb{F}_2^{\lambda_0} \setminus (H_{\text{true}} \cup S \cup H_{\text{false}})$
$S \leftarrow \emptyset$	H_{false}.`push`($\mathbf{r}$)	S.`push`($\mathbf{r}$)
	return $\mathbf{r} \in H_{\text{true}} \cup S$	return $\mathbf{r}$

Fig. 7. Emulation of the list operations.

The correctness of this emulation follows directly from the calculations given above. For instance the correctness of the call $L_{\mathbf{m}}.\texttt{next}()$ follows from the fact that with probability $\frac{|H_{\text{true}}|}{q_{\text{sign}} - |S|} = \gamma$ it outputs an element of H_{true} chosen uniformly at random (see (27)). In such a case the corresponding element has to be moved from H_{true} to S (since it has been queried now through a call to $\texttt{Sign}(\mathbf{m})$). The correctness of $L_{\mathbf{m}}.\texttt{contains}(\mathbf{r})$ is a direct consequence of the formulas for $\mathbb{P}(\mathbf{r} \in L_{\mathbf{m}} | \mathcal{Q})$ given in (24), (25) and (26). All `push`, `pop`, membership testing above can be implemented in time proportional to λ_0.

A.2 Proof of Lemma 1

The goal of this subsection is to estimate the probability of a collision in a signature query for a message $\mathbf{m}$ when we allow at most q_{sign} queries (the event F in the security proof) and to deduce Lemma 1 of §4.3. We recall that in $\mathcal{S}_{\text{code}}$ for each signature query, we pick $\mathbf{r}$ uniformly at random in $\{0,1\}^{\lambda_0}$. Then the probability we are looking for is bounded by the probability to pick the same $\mathbf{r}$ at least twice after q_{sign} draws. The following lemma will be useful.

Lemma 5. *The probability to have at least one collision after drawing uniformly and independently t elements in a set of size n is upper bounded by t^2/n for sufficiently large n and $t^2 < n$.*

Proof. The probability of no collisions after drawing independently t elements among n is:

$$p_{n,t} \stackrel{\triangle}{=} \prod_{i=0}^{t-1}\left(1-\frac{i}{n}\right) \geq 1-\sum_{i=0}^{t-1}\frac{i}{n} = 1-\frac{t(t-1)}{2n}$$

from which we easily get $1-p_{n,t} \leq t^2/n$, concluding the proof.

In our case, the probability of the event F is bounded by the previous probability for $t = q_{\text{sign}}$ and $n = 2^{\lambda_0}$, so, with $\lambda_0 = \lambda + 2\log_2 q_{\text{sign}}$, we can conclude that

$$\mathbb{P}(F) \leq \frac{q_{\text{sign}}^2}{2^{\lambda_0}} = \frac{1}{2^{\lambda_0 - 2\log_2(q_{\text{sign}})}} = \frac{1}{2^{\lambda}}$$

which concludes the proof of Lemma 1.

A.3 Proof of Proposition 5 and Lemma 2

Our goal in this subsection is to prove Lemma 2 of §4.3 and to achieve this we will first prove Proposition 5 of §4.2 which asserts that syndromes by $\mathbf{H}_{\text{pub}}$ of errors of weight w are indistinguishable from random elements in $\mathbb{F}_2^{n-k}$:

Proposition 5. *Let $\mathcal{D}_w^{\mathbf{H}}$ be the distribution of the syndromes $\mathbf{H}\mathbf{e}^T$ when $\mathbf{e}$ is drawn uniformly at random among the binary vectors of weight w and $\mathcal{U}$ be the uniform distribution over the syndrome space $\mathbb{F}_2^{n-k}$. We have*

$$\mathbb{E}_{\mathbf{H}_{\text{pub}}}\left(\rho(\mathcal{D}_w^{\mathbf{H}_{\text{pub}}},\mathcal{U})\right) \leq \frac{1}{2}\sqrt{\varepsilon}$$

with

$$\varepsilon = \frac{2^{n-k}}{\binom{n}{w}} + \frac{2^{n/2-k_U}\binom{n/2}{w/2}}{\binom{n}{w}} + \sum_{\substack{j\in\{0,\dots,w\}\\ j\equiv w \pmod 2}} \frac{2^{2j+n/2-k_V}\binom{n/2}{(w-j)/2}^2\binom{n/2-(w-j)/2}{j}^2}{\binom{n/2}{j}\binom{n}{w}^2}.$$

Proposition 5 is based on two lemmas. The first one is a general lemma given in §4.3.

Lemma 3. *Consider a finite family $\mathcal{H} = (h_i)_{i\in I}$ of functions from a finite set E to a finite set F. Denote by ε the bias of the collision probability, i.e. the quantity such that*

$$\mathbb{P}_{h,e,e'}(h(e) = h(e')) = \frac{1}{|F|}(1+\varepsilon)$$

where h is drawn uniformly at random in $\mathcal{H}$, e and e' are drawn uniformly at random in E. Let $\mathcal{U}$ be the uniform distribution over F and $\mathcal{D}(h)$ be the distribution of the outputs $h(e)$ when e is chosen uniformly at random in E. We have

$$\mathbb{E}_h\left\{\rho(\mathcal{D}(h),\mathcal{U})\right\} \leq \frac{1}{2}\sqrt{\varepsilon}.$$

Proof. Let $q_{h,f}$ be the probability distribution of the discrete random variable $(h_0, h_0(e))$ where h_0 is drawn uniformly at random in $\mathcal{H}$ and e drawn uniformly at random in E (i.e. $q_{h,f} = \mathbb{P}_{h_0,e}(h_0 = h, h_0(e) = f)$). By definition of the statistical distance we have

$$\begin{aligned}
\mathbb{E}_h \left\{\rho(\mathcal{D}(h),\mathcal{U})\right\} &= \sum_{h\in\mathcal{H}} \frac{1}{|\mathcal{H}|}\rho(\mathcal{D}(h),\mathcal{U}) \\
&= \sum_{h\in\mathcal{H}} \frac{1}{2|\mathcal{H}|} \sum_{f\in F} \left|\mathbb{P}_e(h(e)=f) - \frac{1}{|F|}\right| \\
&= \frac{1}{2} \sum_{(h,f)\in\mathcal{H}\times F} \left|\mathbb{P}_{h_0,e}(h_0 = h, h_0(e) = f) - \frac{1}{|\mathcal{H}|\cdot|F|}\right| \\
&= \frac{1}{2} \sum_{(h,f)\in\mathcal{H}\times F} \left|q_{h,f} - \frac{1}{|\mathcal{H}|\cdot|F|}\right|. \qquad (31)
\end{aligned}$$

Using the Cauchy-Schwarz inequality, we obtain

$$\sum_{(h,f)\in\mathcal{H}\times F} \left|q_{h,f} - \frac{1}{|\mathcal{H}|\cdot|F|}\right| \leq \sqrt{\sum_{(h,f)\in\mathcal{H}\times F} \left(q_{h,f} - \frac{1}{|\mathcal{H}|\cdot|F|}\right)^2} \cdot \sqrt{|\mathcal{H}|\cdot|F|}. \qquad (32)$$

Let us observe now that

$$\begin{aligned}
\sum_{(h,f)\in\mathcal{H}\times F} \left(q_{h,f} - \frac{1}{|\mathcal{H}|\cdot|F|}\right)^2 &= \sum_{h,f} \left(q_{h,f}^2 - 2\frac{q_{h,f}}{|\mathcal{H}|\cdot|F|} + \frac{1}{|\mathcal{H}|^2\cdot|F|^2}\right) \\
&= \sum_{h,f} q_{h,f}^2 - 2\frac{\sum_{h,f} q_{h,f}}{|\mathcal{H}|\cdot|F|} + \frac{1}{|\mathcal{H}|\cdot|F|} \\
&= \sum_{h,f} q_{h,f}^2 - \frac{1}{|\mathcal{H}|\cdot|F|}. \qquad (33)
\end{aligned}$$

Consider for $i \in \{0,1\}$ independent random variables h_i and e_i that are drawn uniformly at random in $\mathcal{H}$ and E respectively. We continue this computation by noticing now that

$$\begin{aligned}
\sum_{h,f} q_{h,f}^2 &= \sum_{h,f} \mathbb{P}_{h_0,e_0}(h_0 = h, h_0(e_0) = f)\mathbb{P}_{h_1,e_1}(h_1 = h, h_1(e_1) = f) \\
&= \mathbb{P}_{h_0,h_1,e_0,e_1}\left(h_0 = h_1, h_0(e_0) = h_1(e_1)\right) \\
&= \frac{\mathbb{P}_{h_0,e_0,e_1}\left(h_0(e_0) = h_0(e_1)\right)}{|\mathcal{H}|} \\
&= \frac{1+\varepsilon}{|\mathcal{H}|\cdot|F|}. \qquad (34)
\end{aligned}$$

By substituting for $\sum_{h,f} q_{h,f}^2$ the expression obtained in (34) into (33) and then back into (32) we finally obtain

$$\sum_{(h,f)\in\mathcal{H}\times F} \left|q_{h,f} - \frac{1}{|\mathcal{H}|\cdot|F|}\right| \leq \sqrt{\frac{1+\varepsilon}{|\mathcal{H}|\cdot|F|} - \frac{1}{|\mathcal{H}|\cdot|F|}}\sqrt{|\mathcal{H}|\cdot|F|} = \sqrt{\frac{\varepsilon}{|\mathcal{H}|\cdot|F|}}\sqrt{|\mathcal{H}|\cdot|F|} = \sqrt{\varepsilon}.$$

This finishes the proof of our lemma. □

In order to use this lemma to bound the statistical distance we are interested in, we perform now the following computation

Lemma 6. *Assume that* $\mathbf{x}$ *and* $\mathbf{y}$ *are random vectors of* S_w *that are drawn uniformly at random in this set. We have*

$$\mathbb{P}_{\mathbf{H}_{\text{pub}},\mathbf{x},\mathbf{y}}\left(\mathbf{H}_{\text{pub}}\mathbf{x}^T = \mathbf{H}_{\text{pub}}\mathbf{y}^T\right) \leq \frac{1}{2^{n-k}}(1+\varepsilon) \text{ with } \varepsilon \text{ given in Proposition 5.}$$

Proof. Recall that $\mathbf{H}_{\text{pub}}$ is obtained as

$$\mathbf{H}_{\text{pub}} = \mathbf{S}\mathbf{H}_{\text{sec}}\mathbf{P} \quad \text{with} \quad \mathbf{H}_{\text{sec}} \stackrel{\triangle}{=} \begin{pmatrix} \mathbf{H}_U & \mathbf{0} \\ \mathbf{H}_V & \mathbf{H}_V \end{pmatrix}$$

where $\mathbf{H}_U$ has been chosen uniformly at random in $\mathbb{F}_2^{(n/2-k_U)\times n/2}$, $\mathbf{H}_V$ has been chosen uniformly in $\mathbb{F}_2^{(n/2-k_V)\times n/2}$, $\mathbf{S}$ has been chosen uniformly at random among the invertible matrices in $\mathbb{F}_2^{(n-k)\times(n-k)}$ and $\mathbf{P}$ among the $n \times n$ permutation matrices. As $\mathbf{S}$ is non-singular and $\mathbf{P}$ is a permutation, the probability of the event $\mathbf{H}_{\text{pub}}\mathbf{x}^T = \mathbf{H}_{\text{pub}}\mathbf{y}^T$ is the same as the probability of the event

$$\begin{pmatrix} \mathbf{H}_U & \mathbf{0} \\ \mathbf{H}_V & \mathbf{H}_V \end{pmatrix}\mathbf{x}^T = \begin{pmatrix} \mathbf{H}_U & \mathbf{0} \\ \mathbf{H}_V & \mathbf{H}_V \end{pmatrix}\mathbf{y}^T.$$

Let $\mathbf{x}$ be a vector of $\mathbb{F}_2^n$, we will denote in the following by $\mathbf{x}_1$ (resp. $\mathbf{x}_2$) the vector formed by its first (resp. last) $n/2$ coordinates. In other words, the probability we are looking for is

$$\mathbb{P}_{\mathbf{H}_U,\mathbf{H}_V,\mathbf{x},\mathbf{y}}\big(\mathbf{H}_U(\mathbf{x}_1+\mathbf{y}_1)^T = \mathbf{0} \wedge \mathbf{H}_V(\mathbf{x}_1+\mathbf{x}_2+\mathbf{y}_1+\mathbf{y}_2)^T = \mathbf{0}\big).$$

To compute this probability we use Lemma 7 which says that:

$$\mathbb{P}_{\mathbf{H}}\left(\mathbf{H}\mathbf{e}^T = \mathbf{0}\right) = \frac{1}{2^{n-k}} \text{ if } \mathbf{e} \neq 0 \text{ and } 1 \text{ otherwise} \tag{35}$$

when $\mathbf{H}$ is chosen uniformly at random in $\mathbb{F}_2^{(n-k)\times n}$. This lemma motivates to distinguish between four disjoint events

Event 1:
$$\mathcal{E}_1 \stackrel{\triangle}{=} \{\mathbf{x}_1+\mathbf{y}_1 = \mathbf{0} \wedge \mathbf{x}_1+\mathbf{x}_2+\mathbf{y}_1+\mathbf{y}_2 \neq \mathbf{0}\}$$

Event 2:
$$\mathcal{E}_2 \stackrel{\triangle}{=} \{\mathbf{x}_1+\mathbf{y}_1 \neq \mathbf{0} \wedge \mathbf{x}_1+\mathbf{x}_2+\mathbf{y}_1+\mathbf{y}_2 = \mathbf{0}\}$$

Event 3:
$$\mathcal{E}_3 \stackrel{\triangle}{=} \{\mathbf{x}_1+\mathbf{y}_1 \neq \mathbf{0} \wedge \mathbf{x}_1+\mathbf{x}_2+\mathbf{y}_1+\mathbf{y}_2 \neq \mathbf{0}\}$$

Event 4:
$$\mathcal{E}_4 \stackrel{\triangle}{=} \{\mathbf{x}_1+\mathbf{y}_1 = \mathbf{0} \wedge \mathbf{x}_1+\mathbf{x}_2+\mathbf{y}_1+\mathbf{y}_2 = \mathbf{0}\}$$

Under these events we get thanks to (35):

$$\begin{aligned}
&\mathbb{P}_{\mathbf{H}_{\text{sec}},\mathbf{x},\mathbf{y}}\left(\mathbf{H}_{\text{sec}}\mathbf{x}^T = \mathbf{H}_{\text{sec}}\mathbf{y}^T\right) \\
&= \sum_{i=1}^{4} \mathbb{P}_{\mathbf{H}_{\text{sec}}}\left(\mathbf{H}_{\text{sec}}\mathbf{x}^T = \mathbf{H}_{\text{sec}}\mathbf{y}^T | \mathcal{E}_i\right)\mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_i\right) \\
&= \frac{\mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_1\right)}{2^{n/2-k_V}} + \frac{\mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_2\right)}{2^{n/2-k_U}} + \frac{\mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_3\right)}{2^{n-k}} + \mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_4\right) \\
&= \frac{1}{2^{n-k}}\left(\frac{\mathbb{P}\left(\mathcal{E}_1\right)}{2^{n/2-k_V-n+k}} + \frac{\mathbb{P}\left(\mathcal{E}_2\right)}{2^{n/2-k_U-n+k}} + \mathbb{P}\left(\mathcal{E}_3\right) + 2^{n-k}\mathbb{P}\left(\mathcal{E}_4\right)\right) \\
&\leq \frac{1}{2^{n-k}}\left(1 + 2^{n/2-k_U}\mathbb{P}\left(\mathcal{E}_1\right) + 2^{n/2-k_V}\mathbb{P}\left(\mathcal{E}_2\right) + 2^{n-k}\mathbb{P}(\mathcal{E}_4)\right),
\end{aligned} \tag{36}$$

where we used for the last inequality the trivial upper-bound $\mathbb{P}\left(\mathcal{E}_3\right) \leq 1$. Let us now upper-bound (or compute) the probabilities of the events $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_4$. For $\mathcal{E}_4$ we clearly have

$$\mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_4\right) = \mathbb{P}(\mathbf{x}=\mathbf{y}) = \frac{1}{\binom{n}{w}}.$$

For $\mathcal{E}_1$ we derive the following upper-bound

$$\begin{aligned}
\mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_1\right) &\leq \mathbb{P}\left(\mathbf{x}_1=\mathbf{y}_1\right) \\
&= \sum_{w_1=0}^{w} \frac{\binom{n/2}{w_1}\binom{n/2}{w-w_1}^2}{\binom{n}{w}^2} \\
&\leq \sum_{w_1=0}^{w} \frac{\binom{n/2}{w_1}\binom{n/2}{w-w_1}}{\binom{n}{w}^2}\binom{n/2}{w/2} && (37)\\
&= \frac{\binom{n/2}{w/2}}{\binom{n}{w}} && (38)
\end{aligned}$$

where (37) follows from $\binom{n/2}{w-w_1}^2 \leq \binom{n/2}{w-w_1}\binom{n/2}{w/2}$ for all w_1 in $\{0,\dots,w\}$ and (38) from $\sum_{w_1=0}^{w}\binom{n/2}{w_1}\binom{n/2}{w-w_1} = \binom{n}{w}$. To upper-bound $\mathbb{P}\left(\mathcal{E}_2\right)$, let us first derive the distribution of $\mathbf{x}_1+\mathbf{x}_2$. We first observe that

$$\begin{aligned}
\mathbb{P}(\mathbf{x}_1+\mathbf{x}_2=\mathbf{e}) &= \mathbb{P}\Big(\mathbf{x}_1+\mathbf{x}_2=\mathbf{e}\Big|\ |\mathbf{x}_1+\mathbf{x}_2|=w_e\Big)\mathbb{P}(|\mathbf{x}_1+\mathbf{x}_2|=w_e) \\
&= \frac{1}{\binom{n/2}{w_e}} 2^{w_e} \frac{\binom{n/2}{(w-w_e)/2}\binom{n/2-(w-w_e)/2}{w_e}}{\binom{n}{w}} \text{ (by Prop. 10)} && (39)
\end{aligned}$$

if $w_e \equiv w \pmod 2$, where w_e is the Hamming weight of $\mathbf{e}$. If w_e does not have the same parity as w, then this probability is equal to 0. From this we deduce that

$$\begin{aligned}
\mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_2\right) &\leq \mathbb{P}\left(\mathbf{x}_1+\mathbf{x}_2=\mathbf{y}_1+\mathbf{y}_2\right) \\
&= \sum_{\substack{j\in\{0,\dots,w\}\\ j\equiv w \pmod 2}} \sum_{\mathbf{e}\in\mathbb{F}_2^{n/2}:|\mathbf{e}|=j} \mathbb{P}_{\mathbf{x}}\left(\mathbf{x}_1+\mathbf{x}_2=\mathbf{e}\right)^2 \\
&= \sum_{\substack{j\in\{0,\dots,w\}\\ j\equiv w \pmod 2}} \frac{1}{\binom{n/2}{j}} 2^{2j} \frac{\binom{n/2}{(w-j)/2}^2\binom{n/2-(w-j)/2}{j}^2}{\binom{n}{w}^2} \quad \text{(by Eq. (39))}
\end{aligned}$$

By plugging these upper-bounds in (36), we finally obtain:

$$\begin{aligned}
&\mathbb{P}_{\mathbf{H}_{\text{pub}},\mathbf{x},\mathbf{y}}\left(\mathbf{H}_{\text{pub}}\mathbf{x}^T=\mathbf{H}_{\text{pub}}\mathbf{y}^T\right) \\
&\leq \frac{1}{2^{n-k}}\left(1+\frac{2^{n-k}}{\binom{n}{w}}+\frac{2^{n/2-k_U}\binom{n/2}{w/2}}{\binom{n}{w}}+\sum_{\substack{j\in\{0,\dots,w\}\\ j\equiv w \pmod 2}}^{w}\frac{2^{2j+n/2-k_V}\binom{n/2}{(w-j)/2}^2\binom{n/2-(w-j)/2}{j}^2}{\binom{n/2}{j}\binom{n}{w}^2}\right)
\end{aligned}$$

which concludes the proof. □

These two lemmas imply directly Proposition 5.

Proof (Proposition 5). Indeed we let in Lemma 3, $E \stackrel{\triangle}{=} \mathbb{F}_2^n$, $F \stackrel{\triangle}{=} \mathbb{F}_2^{n-k}$ and $\mathcal{H}$ be the set of functions associated to the 4-tuples $(\mathbf{H}_U, \mathbf{H}_V, \mathbf{S}, \mathbf{P})$ used to generate a public parity-check matrix $\mathbf{H}_{\text{pub}}$ through (9). These functions h are given by $h(\mathbf{e}) = \mathbf{H}_{\text{pub}}\mathbf{e}^T$. Lemma 6 gives an upper-bound for the ε term in Lemma 3 and this finishes the proof of Proposition 5. □

We are now able to prove Lemma 2 (we use here notations of the security proof in §4.3).

Lemma 2.

$$\mathbb{P}(S_1) \leq \mathbb{P}(S_2) + \frac{q_{\text{hash}}}{2}\sqrt{\varepsilon} \textit{ where } \varepsilon \textit{ is given in Proposition 5.}$$

Proof (Lemma 2). To simplify notation we let $q \overset{\triangle}{=} q_{\text{hash}}$. Then we notice that

$$\mathbb{P}(S_1) \leq \mathbb{P}(S_2) + \rho(\mathcal{D}^{\text{pub}}_{w,q}, \mathcal{D}_{\text{pub}} \otimes \mathcal{U}^{\otimes q}), \tag{40}$$

where

- $\mathcal{U}$ is the uniform distribution over $\mathbb{F}_2^{n-k}$;
- $\mathcal{D}^{\text{pub}}_{w,q}$ is the distribution of the $(q+1)$-tuples $(\mathbf{H}_{\text{pub}}, \mathbf{H}_{\text{pub}}\mathbf{e}_1^T, \cdots, \mathbf{H}_{\text{pub}}\mathbf{e}_q^T)$ where the $\mathbf{e}_i$'s are independent and uniformly distributed in S_w;
- $\mathcal{D}_{\text{pub}} \otimes \mathcal{U}^{\otimes q}$ is the distribution of the $(q+1)$-tuples $(\mathbf{H}_{\text{pub}}, \mathbf{s}_1^T, \cdots, \mathbf{s}_q^T)$ where the $\mathbf{s}_i$'s are independent and uniformly distributed in $\mathbb{F}_2^{n-k}$.

We now observe that

$$\begin{aligned}
\rho(\mathcal{D}^{\text{pub}}_{w,q}, \mathcal{D}_{\text{pub}} \otimes \mathcal{U}^{\otimes q}) &= \sum_{\mathbf{H}\in\mathbb{F}_2^{(n-k)\times n}} \mathbb{P}(\mathbf{H}_{\text{pub}} = \mathbf{H}) \rho((\mathcal{D}^{\mathbf{H}}_w)^{\otimes q}, \mathcal{U}^{\otimes q}) \\
&\leq q \sum_{\mathbf{H}\in\mathbb{F}_2^{(n-k)\times n}} \mathbb{P}(\mathbf{H}_{\text{pub}} = \mathbf{H}) \rho(\mathcal{D}^{\mathbf{H}}_w, \mathcal{U}) \text{ (by Prop. 4)} \\
&= q\mathbb{E}_{\mathbf{H}_{\text{pub}}}\left\{\rho(\mathcal{D}^{\text{pub}}_w, \mathcal{U})\right\} \\
&\leq q\frac{\sqrt{\varepsilon}}{2} \text{ (by Prop. 5).}
\end{aligned}$$

□

B Proofs for §5

B.1 Proof of Proposition 11 and Theorem 2

First of all it is straightforward to check that the distributions p_i^u are given by

Proposition 10 (Distribution of w_1 and w_2). *For all i in $\{0, \dots, w\}$ such that $w \equiv i \pmod 2$*

$$p_2^u\left(\frac{w-i}{2}\right) = p_1^u(i) = 2^i \frac{\binom{n/2}{(w-i)/2}\binom{n/2-(w-i)/2}{i}}{\binom{n}{w}}$$

and for other choices of i, $p_1(i)$ and $p_2(i)$ are equal to 0.

On the other hand the distributions p_i^{sdd} of the source distortion decoder are given by

Proposition 11. *Let θ denote the internal coin used in the probabilistic algorithm D and*

$$p(i) \overset{\triangle}{=} \mathbb{P}_{\mathbf{s},\theta}(|D(\mathbf{H}, \mathbf{s})| = i)$$

If two executions of D are independent, then for all i in $\{0, \dots, w\}$ such that $w - i \equiv 0 \pmod 2$ we have

$$p_2^{sdd}\left(\frac{w-i}{2}\right) = p_1^{sdd}(i) = \frac{x_i\, p(i)}{p_w^1} \tag{41}$$

where

$$p_w^1 \overset{\triangle}{=} \sum_{\substack{0\leq j\leq w \\ j\equiv w \pmod 2}} x_j\, p(j)$$

and $p_1^{sdd}(i) = 0$ for other choices of i.

Proof. Let $\mathbf{e}$ be the output of Algorithm 2. Recall that

$$p_1^{sdd}(j) \stackrel{\triangle}{=} \mathbb{P}_{\mathbf{e}}\left(w_1(\mathbf{e}) = j\right) = \mathbb{P}_{\mathbf{s},\theta}(|D(\mathbf{H}_V, \mathbf{s})| = j).$$

As two executions of D are independent, by a disjunction of independent events the probability to get an error $\mathbf{e}$ such that $w_1(\mathbf{e}) = i$ is given by:

$$\sum_{l=0}^{+\infty} \alpha^l \beta_i = \frac{\beta_i}{1-\alpha} \tag{42}$$

where α denotes the probability that the output of D at Instruction 2 of Algorithm 2 is rejected and β_i the probability to have an error of weight i which is accepted. These probabilities are readily seen to be equal to:

$$\beta_i = p(i)x_i \quad ; \quad \alpha = 1 - \sum_{\substack{0 \leq j \leq w \\ j \equiv w \pmod 2}} x_j p(j).$$

Plugging this expression in (42) finishes the proof. □

Let us recall that $\mathcal{D}_w$ is the distribution $\{D_{\mathbf{H}_{\text{sec}},w}(\mathbf{s}) : \mathbf{s} \hookleftarrow \mathbb{F}_2^{n-k}\}$ where $D_{\mathbf{H}_{\text{sec}},w}$ is Algorithm 2. Recall now Theorem 2

Theorem 2. *If the source decoder D used in Algorithm 2 behaves uniformly for $\mathbf{H}_V$ and uniformly for $\mathbf{H}''_U$ which is obtained from $(\mathbf{H}_U, \mathbf{e}_V)$ in Proposition 2 (see §3.2) for all error patterns $\mathbf{e}_V$ obtained as $\mathbf{e}_V = D(\mathbf{H}_V, \mathbf{s}_2)$, we have:*

$$\rho\left(\mathcal{D}_w, \mathcal{U}_w\right) = \rho\left(p_1^{sdd}, p_1^u\right)$$

where $\mathcal{D}_w$ is the output distribution of Algorithm 2. Then, output of Algorithm 2 is the uniform distribution over S_w if in addition two executions of D are independent and the no-rejection probability vector $\mathbf{x}$ is chosen for any i in $\{0, \dots, w\}$ as

$$x_i = \frac{1}{M_{rs}} \frac{p_1^u(i)}{p(i)} \text{ if } w \equiv i \pmod 2 \;\; x_i = 0 \text{ otherwise}$$

with $p(i) \stackrel{\triangle}{=} \mathbb{P}_{\mathbf{s},\theta}(|D(\mathbf{H}_V, \mathbf{s})| = i)$ and $M_{rs} \stackrel{\triangle}{=} \sup_{\substack{0 \leq i \leq w \\ i \equiv w \pmod 2}} \frac{p_1^u(i)}{p(i)}$.

Proof. Let us first introduce some notation. Let $\mathbf{e}$ be a random variable whose distribution is $\mathcal{U}_w$, *i.e.* the uniform distribution over S_w, and let $\tilde{\mathbf{e}}$ be a random variable whose distribution is $\mathcal{D}_w$. The last random variable can be viewed in a natural way as the output of Algorithm 2 and is of the form $\tilde{\mathbf{e}} = (\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V)$. We view $\mathbf{e}_U$ and $\mathbf{e}_V$ as random variables. We have

$$\rho\left(\mathcal{D}_w, \mathcal{U}_w\right) = \sum_{\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{F}_2^{n/2} : |(\mathbf{e}_1, \mathbf{e}_2)| = w} \left|\mathbb{P}(\tilde{\mathbf{e}} = (\mathbf{e}_1, \mathbf{e}_2)) - \mathbb{P}(\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2))\right|. \tag{43}$$

We notice now that

$$\begin{aligned} \mathbb{P}(\tilde{\mathbf{e}} = (\mathbf{e}_1, \mathbf{e}_2)) &= \mathbb{P}(\mathbf{e}_U = \mathbf{e}_1 | \mathbf{e}_V = \mathbf{e}_1 + \mathbf{e}_2)\mathbb{P}(\mathbf{e}_V = \mathbf{e}_1 + \mathbf{e}_2) \\ &= \mathbb{P}(\mathbf{e}_U = \mathbf{e}_1 | \mathbf{e}_V = \mathbf{e}_1 + \mathbf{e}_2)\mathbb{P}_{\mathbf{s}_2,\theta}(D(\mathbf{H}_V, \mathbf{s}_2) = \mathbf{e}_1 + \mathbf{e}_2). \end{aligned} \tag{44}$$

From the assumption on the uniform behavior of D we deduce that $\mathbb{P}_{\mathbf{s}_2,\theta}(D(\mathbf{H}_V, \mathbf{s}_2) = \mathbf{e}_1 + \mathbf{e}_2)$ only depends on the Hamming weight $|\mathbf{e}_1 + \mathbf{e}_2|$ of $\mathbf{e}_1 + \mathbf{e}_2$. We recall now that in Algorithm 2 we have

$$\mathbf{e}_U = D(\mathbf{H}_U, \mathbf{s}_1, \mathbf{e}_V)$$

Let

$$n' \overset{\triangle}{=} n/2 - |\mathbf{e}_V| \quad ; \quad w' \overset{\triangle}{=} \frac{w - |\mathbf{e}_V|}{2}$$

and $\mathbf{H}''_U, \mathbf{s}''_1$ are elements given by $(\mathbf{H}_U, \mathbf{s}_1, \mathbf{e}_V)$ in Proposition 2 in §3.2. It will now be convenient to split $\mathbf{e}_U$ and $\mathbf{e}_1$ into two parts: the first one, denoted respectively by $\mathbf{e}'_U$, and $\mathbf{e}'_1$ is the restriction of these vectors to the complement of the support of $\mathbf{e}_V$, whereas the second one, denoted respectively by $\mathbf{e}''_U$ and $\mathbf{e}''_1$ is the restriction of these vectors to the support of $\mathbf{e}_V$. With this notation, we now notice that

$$\begin{aligned}
\mathbb{P}(\mathbf{e}_U = \mathbf{e}_1 | \mathbf{e}_V = \mathbf{e}_1 + \mathbf{e}_2) &= \mathbb{P}_{(\mathbf{s}_1,\mathbf{s}_2),\theta}(\mathbf{e}'_U = \mathbf{e}'_1, \mathbf{e}''_U = \mathbf{e}''_1 | \mathbf{e}_V = \mathbf{e}_1 + \mathbf{e}_2) \\
&= \mathbb{P}_{\mathbf{s}_1,\theta}(\mathbf{e}'_U = \mathbf{e}'_1)\mathbb{P}_{\mathbf{s}_1,\theta}(\mathbf{e}''_U = \mathbf{e}''_1) \\
&= \mathbb{P}_{\mathbf{s}_1,\theta}(D_{w'}(\mathbf{H}''_U, \mathbf{s}''_1) = \mathbf{e}'_1)\mathbb{P}_{\mathbf{s}_1,\theta}(\mathbf{e}''_U = \mathbf{e}''_1) \\
&= \frac{1}{\binom{n'}{w'}} \frac{1}{2^{n/2-n'}}. \qquad (45)
\end{aligned}$$

The last equality follows from the fact that D behaves uniformly on $\mathbf{H}''_U$ for all patterns $\mathbf{e}_V$ and therefore the output of $D_{w'}(\mathbf{H}''_U, \mathbf{s}''_1)$ is the uniform distribution over the set of words of weight w' in $\mathbb{F}_2^{n'}$. Equality (45) implies that $\mathbb{P}(\mathbf{e}_U = \mathbf{e}_1 | \mathbf{e}_V = \mathbf{e}_1 + \mathbf{e}_2)$ only depends on the weight of w' which itself only depends on the weight of $\mathbf{e}_1 + \mathbf{e}_2$. Since $\mathbb{P}_{\mathbf{s}_2,\theta}(D(\mathbf{H}_V, \mathbf{s}_2) = \mathbf{e}_1 + \mathbf{e}_2)$ has the same property, we deduce from (44), that $\mathbb{P}(\tilde{\mathbf{e}} = (\mathbf{e}_1, \mathbf{e}_2))$ only depends on the weight of $\mathbf{e}_1 + \mathbf{e}_2$. Obviously $\mathbb{P}(\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2))$ also has this property. We may therefore write

$$\begin{aligned}
\mathbb{P}(\tilde{\mathbf{e}} = (\mathbf{e}_1, \mathbf{e}_2)) = f(|\mathbf{e}_1 + \mathbf{e}_2)|) \\
\mathbb{P}(\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)) = g(|\mathbf{e}_1 + \mathbf{e}_2)|)
\end{aligned}$$

for some functions f and g. Plugging these expressions in (43) yields by bringing in the quantity m_i which is the number of $\mathbf{e}$ in S_w such that $w_1(\mathbf{e}) = i$:

$$\begin{aligned}
\rho\left(\mathcal{D}_w, \mathcal{U}_w\right) &= \sum_{\substack{0 \le i \le w \\ i \equiv w \pmod 2}} \sum_{\mathbf{m} \in S_w | w_1(\mathbf{m}) = i} |\mathbb{P}(\tilde{\mathbf{e}} = \mathbf{m}) - \mathbb{P}(\mathbf{e} = \mathbf{m})| \\
&= \sum_{\substack{0 \le i \le w \\ i \equiv w \pmod 2}} m_i \left| f(i) - g(i) \right| \\
&= \sum_{\substack{0 \le i \le w \\ i \equiv w \pmod 2}} |m_i(f(i) - g(i))| \\
&= \sum_{\substack{0 \le i \le w \\ i \equiv w \pmod 2}} |\mathbb{P}(w_1(\tilde{\mathbf{e}}) = i) - \mathbb{P}(w_1(\mathbf{e}) = i)| \\
&= \rho(p_1^{sdd}, p_1^u).
\end{aligned}$$

The last part of the proposition follows from the fact that the $p_1^{sdd}(i)$'s are functions of the non-rejection probability vector $\mathbf{x} = (x_i)$. Thanks to what we just proved, we can compute the x_i's to have $\rho(p_1^{sdd}, p_1^u) = 0$. This will imply that the output of Algorithm 2 is the uniform distribution. Indeed, we first notice that for all i:

$$0 \le x_i = \frac{1}{M_{rs}} \frac{p_1^u(i)}{p(i)} = \left(\inf_{\substack{0 \le j \le w \\ w \equiv j \pmod 2}} \frac{p(j)}{p_1^u(j)} \right) \frac{p_1^u(i)}{p(i)} \le \frac{p(i)}{p_u^1(i)} \frac{p_1^u(i)}{p(i)} = 1$$

which allows to assert that $\mathbf{x}$ is a probability vector. We now use the following equations for all i:

$$\begin{aligned}
p_1^{sdd}(i) &= \frac{x_i\, p(i)}{p_w^1} \\
&= \frac{p_1^u(i)}{M_{rs} \sum\limits_{\substack{0 \le j \le w \\ w \equiv j \pmod 2}} \frac{1}{M_{rs}} p_1^u(j)} \\
&= p_1^u(i)
\end{aligned}$$

where the last line relies on the equality $\sum\limits_{\substack{0 \le j \le w \\ w \equiv j \pmod 2}} p_1^u(j) = 1$. □

B.2 Proof of Proposition 6 and discussion related to it

Here the internal coins are over the choices of the $n-k$ positions I (columns of the parity-check matrix $\mathbf{H}$) we choose to invert in the Prange algorithm. We have here

$$\begin{aligned}
p(w) &= \sum_{\mathbf{e}:|\mathbf{e}|=w} \mathbb{P}_{\mathbf{s},I}\left(\mathbf{e} = D^{\text{Prange}}(\mathbf{H},\mathbf{s})\right) \\
&= \sum_{I \subset \{1,\dots,n\}:|I|=n-k} \mathbb{P}(I) \sum_{\mathbf{e}:|\mathbf{e}|=w} \mathbb{P}_{\mathbf{s}}(\mathbf{e} = D^{\text{Prange}}(\mathbf{H},\mathbf{s})|I) \\
&= \sum_{I \subset \{1,\dots,n\}:|I|=n-k} \mathbb{P}(I) \sum_{\mathbf{e}:|\mathbf{e}|=w,\text{Supp}(\mathbf{e})\subset I} \frac{1}{2^{n-k}} \\
&= \frac{\binom{n-k}{w}}{2^{n-k}}.
\end{aligned}$$

This ends the proof of Proposition 6. □

C Proofs of results of §7

C.1 Proof of Proposition 7 in §7

Let us recall Proposition 7

Proposition 7. *Assume that we choose a $(U, U+V)$ code by picking the parity-check matrices of U and V uniformly at random among the binary matrices of size $(n/2-k_U)\times n/2$ and $(n/2-k_V)\times n/2$ respectively. Let $a_{(U,U+V)}(w)$, $a_{(U,U)}(w)$ and $a_{(0,V)}(w)$ be the expected number of codewords of weight w that are respectively in the $(U, U+V)$ code, of the form $(\mathbf{u},\mathbf{u})$ where $\mathbf{u}$ belongs to U and of the form $(\mathbf{0},\mathbf{v})$ where $\mathbf{v}$ belongs to V. These numbers are given for even w in $\{0,\dots,n\}$ by*

$$a_{(U,U+V)}(w) = \frac{\binom{n/2}{w/2}}{2^{n/2-k_U}} + \frac{\binom{n/2}{w}}{2^{n/2-k_V}} + \frac{1}{2^{n-k_U-k_V}}\left(\binom{n}{w} - \binom{n/2}{w} - \binom{n/2}{w/2}\right)$$

$$a_{(U,U)}(w) = \frac{\binom{n/2}{w/2}}{2^{n/2-k_U}} \quad ; \quad a_{(0,V)}(w) = \frac{\binom{n/2}{w}}{2^{n/2-k_V}}$$

and for odd w in $\{0,\dots,n\}$ by

$$a_{(U,U+V)}(w) = \frac{\binom{n/2}{w}}{2^{n/2-k_V}} + \frac{1}{2^{n-k_U-k_V}}\left(\binom{n}{w} - \binom{n/2}{w}\right)$$

$$a_{(U,U)}(w) = 0 \quad ; \quad a_{(0,V)}(w) = \frac{\binom{n/2}{w}}{2^{n/2-k_V}}$$

On the other hand, when we choose a code of length n with a random parity-check matrix of size $(n-k_U-k_V)\times n$ chosen uniformly at random, then the expected number $a(w)$ of codewords of weight $w>0$ is given by

$$a(w) = \frac{\binom{n}{w}}{2^{n-k_U-k_V}}.$$

We will need the following lemma.

Lemma 7. *Let $\mathbf{y}$ be a non-zero vector of $\mathbb{F}_2^n$ and $\mathbf{s}$ an arbitrary element in $\mathbb{F}_2^r$. We choose a matrix $\mathbf{H}$ of size $r\times n$ uniformly at random among the set of $r\times n$ binary matrices. In this case*

$$\mathbb{P}\left(\mathbf{H}\mathbf{y}^T = \mathbf{s}^T\right) = \frac{1}{2^r}$$

Proof. The coefficient of $\mathbf{H}$ at row i and column j is denoted by h_{ij}, whereas the coefficients of $\mathbf{y}$ and $\mathbf{s}$ are denoted by y_i and s_i respectively. The probability we are looking for is the probability to have

$$\sum_j h_{ij}y_j = s_i \tag{46}$$

for all i in $\{1,\dots,r\}$. Since $\mathbf{y}$ is non zero, it has at least one non-zero coordinate. Without loss of generality, we may assume that $y_1 = 1$. We may rewrite (46) as $h_{i1} = \sum_{j>1} h_{ij}y_j$. This event happens with probability $\frac{1}{2}$ for a given i and with probability $\frac{1}{2^r}$ on all r events simultaneously due to the independence of the h_{ij}'s.

The last part of Proposition 7 is a direct application of this lemma. We namely have

Proposition 12. *Let $a(w)$ be the expected number of codewords of weight w in a binary linear code $\mathcal{C}$ of length n whose parity-check matrix is chosen $\mathbf{H}$ uniformly at random among all binary matrices of size $r\times n$. We have*

$$a(w) = \frac{\binom{n}{w}}{2^r}.$$

Proof. Let $Z \overset{\triangle}{=} \sum_{\mathbf{x}\in\mathbb{F}_2^n:|\mathbf{x}|=w} Z_{\mathbf{x}}$ where $Z_{\mathbf{x}}$ is the indicator function of the event "$\mathbf{x}$ is in $\mathcal{C}$". We have

$$\begin{aligned} a(w) &= \mathbb{E}(Z) \\ &= \sum_{\mathbf{x}\in\mathbb{F}_2^n:|\mathbf{x}|=w} \mathbb{E}(Z_{\mathbf{x}}) \\ &= \sum_{\mathbf{x}\in\mathbb{F}_2^n:|\mathbf{x}|=w} \mathbb{P}(\mathbf{x}\in\mathcal{C}) \\ &= \sum_{\mathbf{x}\in\mathbb{F}_2^n:|\mathbf{x}|=w} \mathbb{P}(\mathbf{H}\mathbf{x}^T = 0) \\ &= \sum_{\mathbf{x}\in\mathbb{F}_2^n:|\mathbf{x}|=w} \frac{1}{2^r} \\ &= \frac{\binom{n}{w}}{2^r}. \end{aligned}$$

This proves the part of Proposition 7 dealing with the expected weight distribution of a random linear code. We are ready now to prove Proposition 7 concerning the expected weight distribution of a random $(U,U+V)$ code.

Weight distributions of $(U,U)\overset{\triangle}{=}\{(\mathbf{u},\mathbf{u}) : \mathbf{u}\in U\}$ **and** $(0,V)\overset{\triangle}{=}\{(\mathbf{0},\mathbf{v}) : \mathbf{v}\in V\}$. This follows directly from Proposition 12 since $a_{(U,U)}(w) = 0$ for odd and $a_{(U,U)}(w)$ is equal to the expected number of codewords of weight $w/2$ in a random linear code of length $n/2$ with a parity-check matrix of size $(n/2 - k_U) \times n/2$ when w is even. On the other hand $a_{(0,V)}$ is equal to the expected number of weight w in a random linear code of length $n/2$ and with a parity-check matrix of size $(n/2 - k_V) \times n/2$. In other words

$$\begin{aligned} a_{(U,U)}(w) &= 0 \text{ if } w \text{ is odd} \\ a_{(U,U)}(w) &= \frac{\binom{n/2}{w/2}}{2^{n/2-k_U}} \text{ if } w \text{ is even} \\ a_{(0,V)}(w) &= \frac{\binom{n/2}{w}}{2^{n/2-k_V}} \end{aligned}$$

Weight distributions of $(U, U+V)$. The code $(U, U+V)$ is chosen randomly by picking up a parity-check matrix $\mathbf{H}_U$ of U uniformly at random among the set of $(n/2 - k_U) \times n/2$ binary matrices and a parity-check matrix $\mathbf{H}_V$ of V uniformly at random among the set of $(n/2-k_V)\times n/2$ binary matrices. Let $Z \overset{\triangle}{=} \sum_{\mathbf{x}\in\mathbb{F}_2^n:|\mathbf{x}|=w} Z_{\mathbf{x}}$ where $Z_{\mathbf{x}}$ is the indicator function of the event "$\mathbf{x}$ is in $(U, U+V)$".

We have

$$\begin{aligned} a_{(U,U+V)}(w) &= \mathbb{E}(Z) \\ &= \sum_{\mathbf{x}\in\mathbb{F}_2^n:|\mathbf{x}|=w} \mathbb{E}(Z_{\mathbf{x}}) \\ &= \sum_{\mathbf{x}\in\mathbb{F}_2^n:|\mathbf{x}|=w} \mathbb{P}(Z_{\mathbf{x}} = 1) \\ &= \sum_{\mathbf{x}\in\mathbb{F}_2^n:|\mathbf{x}|=w} \mathbb{P}(\mathbf{x} \in (U, U+V)) \end{aligned} \qquad (47)$$

By writing $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ where $\mathbf{x}_i$ is in $\mathbb{F}_2^{n/2}$ we know that $\mathbf{x}$ is in $(U, U+V)$ if and only if at the same time $\mathbf{x}_1$ is in U and $\mathbf{x}_2 + \mathbf{x}_1$ is in V, that is

$$\mathbf{H}_U \mathbf{x}_1^T = 0, \;\; \mathbf{H}_V \mathbf{x}_1^T = \mathbf{H}_V \mathbf{x}_2^T.$$

There are three cases to consider

Case 1: $\mathbf{x}_1 = 0$ and $\mathbf{x}_2 \neq 0$. In this case

$$\mathbb{P}(\mathbf{x} \in (U, U+V)) = \mathbb{P}(\mathbf{H}_V \mathbf{x}_2^T = \mathbf{0}) = \frac{1}{2^{n/2-k_V}} \qquad (48)$$

Case 2: $\mathbf{x}_1 = \mathbf{x}_2$. In this case

$$\mathbb{P}(\mathbf{x} \in (U, U+V)) = \mathbb{P}(\mathbf{H}_U \mathbf{x}_1^T = \mathbf{0}) = \frac{1}{2^{n/2-k_U}} \qquad (49)$$

Case 3: $\mathbf{x}_1 \neq \mathbf{x}_2$ and $\mathbf{x}_1 \neq 0$. In this case

$$\mathbb{P}(\mathbf{x} \in (U, U+V)) = \mathbb{P}(\mathbf{H}_U \mathbf{x}_1^T = \mathbf{0} \wedge \mathbf{H}_V(\mathbf{x}_1^T + \mathbf{x}_2^T) = \mathbf{0}) = \frac{1}{2^{n/2-k_U}} \frac{1}{2^{n/2-k_V}} \qquad (50)$$

Note that we used in each case Lemma 7.

By substituting $\mathbb{P}(\mathbf{x} \in (U, U+V))$ in (47) we obtain for even $0 < w \leq n$

$$a_{(U,U+V)}(w) = \frac{\binom{n/2}{w/2}}{2^{n/2-k_U}} + \frac{\binom{n/2}{w}}{2^{n/2-k_V}} + \frac{1}{2^{n-k_U-k_V}} \left(\binom{n}{w} - \binom{n/2}{w} - \binom{n/2}{w/2} \right)$$

and for odd $w \leq n$

$$a(w) = \frac{\binom{n/2}{w}}{2^{n/2-k_V}} + \frac{1}{2^{n-k_U-k_V}} \left(\binom{n}{w} - \binom{n/2}{w} \right)$$

which concludes the proof. $\square$

C.2 Proof of Theorem 3

First it is clear that the weak $(U, U+V)$-distinguishing problem is in NP. The proof that the problem is NP-complete relies on the hardness of the subcode equivalence problem [BGK17]:

Problem 5 (subcode equivalence).
Instance: Two linear codes $\mathcal{C}$ and $\mathcal{D}$ of length n
Question: Is there a permutation σ of the support such that $\sigma(\mathcal{C}) \subseteq \mathcal{D}$

This problem was proved to be NP-complete in [BGK17].

We will show that any instance of the subcode-equivalence problem can be transformed into an instance of Problem 4 with the same answer. Let us consider an instance $(\mathcal{C}, \mathcal{D})$ of the subcode equivalence problem. We will adopt the generator matrix point of view here which is more convenient for our purpose. In other words, we have access to generator matrices $\mathbf{G}_{\mathcal{C}}$ and $\mathbf{G}_{\mathcal{D}}$ of codes $\mathcal{C}$ and $\mathcal{D}$. Let $\mathbf{G}$ be the following matrix

$$\begin{pmatrix} \mathbf{G}_{\mathcal{C}} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_{\mathcal{D}} \end{pmatrix}$$

Suppose that there exists a permutation σ such that $\sigma(\mathcal{C}) \subseteq \mathcal{D} \iff \mathcal{C} \subseteq \sigma^{-1}(\mathcal{D})$. Then if we apply σ^{-1} on the last $n/2$ columns of $\mathbf{G}$ we get:

$$\begin{pmatrix} \mathbf{G}_{\mathcal{C}} & \mathbf{0} \\ \mathbf{0} & \sigma^{-1}(\mathbf{G}_{\mathcal{D}}) \end{pmatrix}$$

which generates the code $(\mathcal{C}|\sigma^{-1}(\mathcal{D}))$ which is equal to $(\mathcal{C}|\mathcal{C}+\sigma^{-1}(\mathcal{D}))$. Then the code $(\mathcal{C}, \mathcal{D}) \stackrel{\triangle}{=} \{(\mathbf{c}, \mathbf{d}) : \mathbf{c} \in \mathcal{C},\ \mathbf{d} \in \mathcal{D}\}$ is a YES instance of Problem 4.

Conversely, suppose that $(\mathcal{C}, \mathcal{D})$ is a YES instance of Problem 4. This code has generator matrix $\mathbf{G} = \begin{pmatrix} \mathbf{G}_{\mathcal{C}} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_{\mathcal{D}} \end{pmatrix}$ and this means that $\mathbf{G}$ should generate a permuted $(U, U+V)$ code with a permutation which acts only on the second half of the code positions. $\mathbf{G}_{\mathcal{C}}$ is therefore necessarily a generator matrix of U. U and $\mathcal{C}$ are therefore equal. It also follows that $\mathbf{G}_{\mathcal{D}}$ (which generates $\mathcal{D}$) has to generate a permutation of $U+V$. Since U is a subcode of $U+V$, it follows that $\mathcal{C}$ is a subcode, up to a permutation, of $\mathcal{D}$. □