Fast Syndrome-Based Chase Decoding of Binary BCH Codes Through Wu List Decoding

Yaron Shany and Amit Berman, Senior Member, IEEE

Abstract—We present a new fast Chase decoding algorithm for binary BCH codes. The new algorithm reduces the complexity in comparison to a recent fast Chase decoding algorithm for Reed–Solomon (RS) codes from [16] and [38]. In the BCH case, the algorithms updated throughout the same minimum distance, every algorithm for decoding RS codes is also automatically an algorithm for decoding BCH codes. Hence, the list decoding algorithms for RS codes from [16] and [38] can be used for decoding BCH codes up to the \(q\)-ary Johnson radius. However, if the BCH code is defined over a proper subfield \(\mathbb{F}_q\), then one may expect a larger decoding radius, since the \(q^\prime\)-ary Johnson bound is in general larger than the \(q\)-ary Johnson bound for \(q^\prime < q\). For \(q^\prime = 2\), [38] includes also a list decoding algorithm for binary BCH codes that takes advantage of the binary alphabet, and reaches the binary Johnson bound.

It is well known that using channel reliability information and moving from HD decoding to soft-decision (SD) decoding may significantly improve the decoding performance. Köetter and Vardy [22] presented a polynomial time SD list decoding algorithm for RS codes that converts the channel reliability information into a multiplicity matrix, with one dimension corresponding to the coordinates of the transmitted codeword, and the other to the \(q\) possible symbols of the alphabet.

In the conclusion of [38], Wu mentions in passing that his list decoding algorithms for RS and BCH codes can be converted to SD list decoding algorithms, by using different multiplicities for different coordinates, depending on the channel reliability information. While Wu’s “one-dimensional” multiplicity assignment is not as general as the two-dimensional multiplicity assignment of [22] for \(q > 2\), this is not the situation in the binary case.

Before [22] and [38], the main algebraic SD decoding algorithms for binary BCH codes were the generalized minimum distance (GMD) decoding [14], and the Chase decoding algorithms [8]. GMD decoding consists of successively erasing an even number of the least reliable \(\eta\) coordinates and applying errors-and-erasures decoding. In Chase decoding, there is a predefined list of test error patterns on the \(\eta\) least reliable coordinates for some small \(\eta\). For example, this list may consist of a random list of vectors, all possible non-zero vectors, all vectors of a low enough weight, etc.. The decoder runs on error patterns from the list and subtracts them from the received word, feeding the result to an HD decoder. If the HD decoder succeeds, then its output is saved into the output list of the decoder.

While the Chase algorithms typically have exponential complexity, they are known to have better performance than the algorithms of [22] and [38] for high-rate short to medium length codes [39]. Therefore, there is still a great interest in finding low-complexity Chase decoding algorithms.

I. INTRODUCTION

A. Motivation and Known Results

Binary BCH codes are widely used in storage and communication systems. Traditionally, these codes are decoded by hard-decision (HD) decoding algorithms that perform unique decoding up to half the minimum distance, such as the Berlekamp–Massey (BM) algorithm [23].

The revolutionary work [16] of Guruswami and Sudan (following Sudan’s original work [33]) introduced a polynomial time list decoding algorithm for Reed–Solomon (RS) codes over \(\mathbb{F}_q\) (where \(q\) is a prime power and \(\mathbb{F}_q\) is the finite field of \(q\) elements) up to the so-called \(q\)-ary Johnson bound [29, Sec. 9.6]. Wu [38] introduced an even more efficient algorithm for HD list decoding of RS codes up to the Johnson bound.

Since BCH codes are subfield subcodes of RS codes with the same minimum distance, every algorithm for decoding RS codes is also automatically an algorithm for decoding BCH codes. Hence, the list decoding algorithms for RS codes from [16] and [38] can be used for decoding BCH codes up to the \(q\)-ary Johnson radius. However, if the BCH code is defined over a proper subfield \(\mathbb{F}_q\), then one may expect a larger decoding radius, since the \(q^\prime\)-ary Johnson bound is in general larger than the \(q\)-ary Johnson bound for \(q^\prime < q\). For \(q^\prime = 2\), [38] includes also a list decoding algorithm for binary BCH codes that takes advantage of the binary alphabet, and reaches the binary Johnson bound.
In fast Chase decoding algorithms, the decoder shares computations between HD decodings of different test error patterns (see, e.g., [4], [20], [30], [39], [41], [42], and [46]). Several papers used special properties of binary codes to further reduce the complexity of their Chase decoding algorithms. Kamiya [20] presented a fast Chase decoding algorithm for binary BCH codes building on the Welch–Berlekamp algorithm. Wu [39, Appendix C] presented a simplified version of his BM-based polynomial-update algorithm for binary BCH codes. Zhang et al. [45] extend [39] by introducing a “backward” step, enabling to order the test vectors according to a Gray map instead of using Wu’s tree. Finally, Zhang [42] suggested an optimized version of [4], using both optimizations from [43] and [44] (for RS codes in general), and the binary alphabet of the BCH codes. It should be noted that [4], [42] are “time domain” algorithms, that is, they work directly on the received vector, and not on the syndrome vector. For practical applications using high-rate BCH codes, it is typically beneficial to replace the long received vector by the short syndrome vector before the decoding begins.

Because our fast Chase algorithm is based on a Gröbner bases formulation of the Wu list decoding algorithm, let us recall the main works in this line of research. For RS codes, the Gröbner bases formulation of the Wu list decoding algorithm was introduced by Trifonov [35], using a time domain approach. Beelen et al. [3] presented a Gröbner bases formulation working with the syndrome vector. Irreducible binary Goppa codes were also considered in [3], where an algorithm for HD list decoding up to the binary Johnson radius was developed.

B. Results and Method

We present a syndrome-based fast Chase decoding algorithm for (primitive, narrow-sense) binary BCH codes that considerably reduces the complexity in comparison to [30, Alg. C] (reduction by a factor of about 5 for polynomial updates; see Subsection V-E for a precise statement). To do that, we use a different approach than that of [30].

First, we establish an isomorphism between two solution modules for binary BCH codes, one for the key equation and one for a modified key equation. This isomorphism gives a general framework for reducing the decoding complexity in the binary case, both in HD bounded-distance decoding, and in fast Chase decoding. For example, for HD bounded-distance decoding, this isomorphism enables to benefit from the binary alphabet with practically any existing algorithm (such as the Euclidean algorithm), without being tied only to Berlekamp’s well-known simplification of his algorithm [5, pp. 24–32]. We remark that a similar idea exists in the literature for specific algorithms [7], [17]. Also, [17] introduces a general method with a different approach (using Newton’s identities) and with a distinction between even and odd correction radius.

Building on the above isomorphism, we then use an SD version of the Wu list decoding algorithm to derive a syndrome-based fast Chase decoding algorithm for binary BCH codes. In the new algorithm there is just one Kötter iteration per edge of the decoding tree (see ahead for details), as opposed to two iterations in [30] for RS codes. Halving the number of Kötter iterations has a double effect in reducing the complexity: once in reducing the degrees of the maintained polynomials, and once in performing less substitutions and scalar multiplications.

We note that it is not possible to simply omit an iteration from the algorithm of [30] in the binary case: omitting the derivative iteration from the algorithm of [30] makes it into a fast GMD algorithm, where coordinates are dynamically erased, which is different from dynamically flipping bits as in Chase decoding.

Similarly to [30, Alg. C], instead of tracking the error-locator polynomial (ELP) itself, we track low-degree “coefficient polynomials” in the decomposition of the ELP according to a suitable basis for the solution module of the modified key equation. Consequently, if the unique error-correction radius is $t$, the total number of errors is $t + r$, and the total number of errors in the non-reliable coordinates is at least $r$, then the maximum sum of degrees of all maintained polynomials is about $2r$ (for a precise statement, see Proposition 35 ahead), as opposed to about $2(t + r)$ in the algorithm of Wu [39].

To benefit from the low degree of the updated polynomials, we avoid working with evaluation vectors. While working with evaluation vectors is a theoretical means for achieving a complexity of $O(n)$ per modified coordinate, it is inefficient for practical code parameters. For this reason, Wu [39, Sec. 5] suggests a criterion for entering the evaluation step, based on a length variable of his algorithm. As in [30], we use such a criterion which is based on a discrepancy of the algorithm: ELP evaluation is performed when the discrepancy is zero. This criterion never misses a required evaluation, but might invoke an unnecessary evaluation with a low probability, and also slightly degrades the probability of decoding success: When the total number of errors is $t + r$, we require $r + 1$ (instead of $r$) errors on the unreliable coordinates, similarly to [39].

Because our algorithm tracks low-degree coefficient polynomials, evaluating an ELP hypothesis amounts to evaluating two polynomials of degree about $r$. Hence the number of finite-field multiplications of the evaluation step (when the above criterion is satisfied) is $O(n \cdot \min\{r, \log(n)\})$, where the minimum is between naïve point-by-point evaluation, and the fast additive FFT algorithm of Gao and Mateer [15].

We use the language of Gröbner-bases for $\mathbb{F}_q[X]$-modules, following the works of Fitzpatrick [12], Trifonov [35], Trifonov and Lee [36], and Beelen et al. [3]. Along the way, we briefly derive the Gröbner-bases formulation of the (SD) Wu list decoding algorithm for binary BCH codes, which is missing in the literature. While quite similar to the case of binary irreducible Goppa codes considered in [3], there are some differences that must be considered. For example, the method of [3] builds on the fact that for an irreducible $g(X) \in \mathbb{F}_{2^s}[X]$ (for $s \in \mathbb{N}^+$), $\mathbb{F}_{2^s}[X]/\langle g(X) \rangle$ is a field, so that any non-zero element has a multiplicative inverse. However, this is not the case for the ring $\mathbb{F}_{2^s}[X]/\langle X^{d-1} \rangle$ involved in the key equation for binary BCH codes.

In summary, the main contributions of the paper are as follows:

- We introduce a new fast Chase algorithm for binary BCH codes, namely, Algorithm A, that requires a single Kötter
iteration per edge of the decoding tree, and maintains polynomials of a lower degree than those of [19], [39].

- We use a degenerate version of the SD Wu list decoding algorithm for devising the new fast Chase algorithm, thus establishing a connection between two prominent algebraic SD decoding algorithms for binary BCH codes.

- We establish an isomorphism between two solution modules for decoding binary BCH codes.

- We derive the Gröbner-bases formulation of the SD Wu list decoding algorithm for binary BCH codes, which is missing in the literature.

### C. Organization

Section II contains some definitions and results used throughout the paper. In Section III, we prove the isomorphism between two solution modules for decoding binary BCH codes, and show how this isomorphism can be used for benefitting from the binary alphabet in a systematic way. Section IV contains some definitions and results used for decoding binary BCH codes. The new fast Chase decoding algorithm is derived in Section V, in which we describe the update rule on an edge of the decoding tree, the stopping criterion for entering exhaustive evaluation, and efficient methods for preforming the evaluation. The same section also includes a detailed complexity analysis. Finally, Section VI includes some conclusions and open questions for further research.

### II. Preliminaries

#### A. Binary BCH Codes and the Key Equation

For $s, t \in \mathbb{N}^*$, let $BCH(t)$ be the primitive binary BCH code of length $2^s - 1$ and designed distance $d := 2t + 1$, that is, let $BCH(t)$ be the cyclic binary code of length $n = 2^s - 1$ with zeroes $\gamma, \gamma^3, \ldots, \gamma^{2t-1}$ and their conjugates, where $\gamma$ is a primitive element in $\mathbb{F}_2^n$.

Suppose that a codeword $x \in BCH(t)$ is transmitted, and the HD received word is $y := x + e$ for some error vector $e \in \mathbb{F}_2^t$. For $j \in \mathbb{N}$, let $S_j := y(\gamma^j)$, where for a field $K$ and a vector $f = (f_0, f_1, \ldots, f_{n-1}) \in K^n$, we let $f(X) := f_0 + f_1X + \ldots + f_{n-1}X^{n-1} \in K[X]$. The syndrome polynomial associated with $y$ is $S(X) := S_1 + S_2X + \ldots + S_{d-1}X^{d-2}$.

Suppose that the error locations are some distinct elements $\alpha_1, \ldots, \alpha_e \in \mathbb{F}_2^t$ (where $e \in \{1, \ldots, n\}$ is the number of errors), and let the error locator polynomial (ELP), $\sigma(X) \in \mathbb{F}_2[X]$, be defined by

$$\sigma(X) := \sum_{i=1}^e (1 + \alpha_iX),$$

and the error evaluator polynomial (EEP), $\omega(X) \in \mathbb{F}_2[X]$, be defined by

$$\omega(X) := \sum_{i=1}^e \alpha_i \prod_{j \neq i} (1 + \alpha_jX) = \sigma'(X),$$

where $\sigma'(X)$ is the formal derivative of $\sigma(X)$. Then $\omega$ and $\sigma$ satisfy the following well-known key equation:

$$\omega \equiv S\sigma \mod (X^{2t})$$

(see, e.g., [29, Sect. 6.3] for a proof).

#### B. A Useful Monomial Ordering

For monomial orderings and Gröbner bases for modules over polynomial rings, we refer to [9, Sec. 5.2]. In this section, we recall a monomial ordering used throughout the paper.

**Definition 1:** Let $K$ be an arbitrary field.

1) For $w = (w_1, w_2, w_3) \in \mathbb{Q}^3$ and for $0 \neq f(X, Y, Z) \in K[X, Y, Z]$, we let $\text{wdeg}_w(f)$ be the maximum value of $w_1\text{deg}_x + w_2\text{deg}_y + w_3\text{deg}_z$ over all monomials $X^mY^nZ^p$ appearing in $f$. For $f = 0$, we put $\text{wdeg}_w(f) := -\infty$. We call $\text{wdeg}_w(f)$ the $w$-weighted degree of $f$. For $f(X, Y) \in K[X, Y]$ and $w \in \mathbb{Q}^2$, $\text{wdeg}_w(f)$ is defined similarly.

2) For $w \in \mathbb{Q}$, and for $f_0(X), f_1(X) \in K[X]$, we define $\text{wdeg}(w)(f_0, f_1) := \max\{\text{deg}(f_0), \text{deg}(f_1) + w\}$.

3) Using the notation of [29], put $\text{ord}(f_0, f_1) := \max\{\text{deg}(f_0) + 1, \text{deg}(f_1)\}$, and note that $\text{ord}(f_0, f_1) = \text{wdeg}(1, -1)(f_0, f_1) + 1$. As in [29], we refer to $\text{ord}(f_0, f_1)$ as the order of the pair $(f_0, f_1)$.

4) Following [12], define a monomial ordering $\prec_w$ on $K[X]$, as follows. For monomials $m_1, m_2 \in K[X]^2$, define $m_1 \prec_w m_2$ if $\text{wdeg}(1, w)(m_1) < \text{wdeg}(1, w)(m_2)$, or $\text{wdeg}(1, w)(m_1) = \text{wdeg}(1, w)(m_2)$, and, when a pair $(f(X), g(X))$ is considered as the bivariate polynomial $f(X) + Yg(X)$, $m_1$ is smaller than $m_2$ with respect to the lex ordering with $Y > X$. Explicitly, we define $(X^{2t}, 0) \prec_w (X^{2s}, 0)$ if $j_1 < j_2$, $(X^{2s}, 0) \prec w (0, X^{2s})$ if $j_1 < j_2$, and $(X^{2s}, 0) \prec (0, X^{2s})$ if $j_1 \leq j_2 + w$. Note that this is a monomial ordering even when $w$ is not positive or not an integer (as in [3]).

In what follows, for a monomial ordering $\prec$ on $K[X]$, for $\ell \in \mathbb{N}$, and for $f \in K[X]$, we will write $LM_{\prec}(f)$ for the leading monomial of $f$ with respect to $\prec$, that is, for the leftmost monomial appearing in $f$. When $\prec$ is clear from the context, we will sometimes write $LM(f)$ for $LM_{\prec}(f)$. Finally, for monomials in $K[X]^2$, we say that the monomial $(X^t, 0)$ is on the left, while the monomial $(0, X^t)$ is on the right. More generally, for $\ell \in \mathbb{N}$ and for $j \in \{0, \ldots, \ell\}$, we call $e_j := (0, \ldots, 0, 1, 0, \ldots, 0) \in K[X]^{\ell+1}$ (1 in $j$-th coordinate, counting from 0) the $j$-th unit vector. A monomial of the form $X^e e_j$ (for some $i, j \in \mathbb{N}$) is said to contain the $j$-th unit vector.

#### C. Kötter’s Iteration

In this subsection we recall the general form of Kötter’s iteration [21], [27], as presented by McEliece [24, Sect. VII.C].

For a $K[X]$-submodule $M \subseteq K[X]^{\ell+1}$ (for $\ell \in \mathbb{N}$) of rank $\ell + 1$, let $G = \{g_0, \ldots, g_{\ell}\}$ be a Gröbner basis for $M$ with respect to some monomial ordering $\prec$ on $K[X]^{\ell+1}$. We will assume that the leading monomial of $g_i$ contains the $j$-th unit vector, for all $j \in \{0, \ldots, \ell\}$, where coordinates of vectors are indexed by $0, \ldots, \ell$ (it is easily verified that there is no loss of generality in this assumption).

Recall that a monomial in $K[X]^2$ is a pair of the form $(X^t, 0)$ or $(0, X^t)$ for some $i \in \mathbb{N}$.
Let $D: K[X]^{\ell+1} \to K$ be a non-zero linear functional for which $M^+ := M \cap \ker(D)$ is a $K[X]$-module, Köter’s iteration converts the $(\ell+1)$-element Gröbner basis $G$ of $M$ to an $(\ell+1)$-element Gröbner basis $G^+=\{g_0^+,\ldots,g_{\ell}^+\}$ of $M^+$, while maintaining the property that $LM(g_j^+)$ contains the $j$-th unit vector for all $j \in \{0,\ldots,\ell\}$. It is presented in the following pseudo-code.

**Köter’s iteration**

**Input**
A Gröbner basis $G = \{g_0,\ldots,g_{\ell}\}$ for the submodule $M \subseteq K[X]^{\ell+1}$, with $LM(g_j)$ containing the $j$-th unit vector for all $j$

**Output**
A Gröbner basis $G^+ = \{g_0^+,\ldots,g_{\ell}^+\}$ for $M^+$ with $LM(g_j^+)$ containing the $j$-th unit vector for all $j$

**Algorithm**
- For $j = 0,\ldots,\ell$, calculate $\Delta_j := D(g_j)$
- Set $J := \{j \in \{0,\ldots,\ell\} | \Delta_j \neq 0\}$
- For $j \in \{0,\ldots,\ell\} \setminus J$
  - Set $g_j^+ := g_j$
  - Let $g_j^+ \in J$ be such that $LM(g_j^+) = \min_{j \in J}\{LM(g_j)\}$
- For $j \in J$
  - If $j \neq J^*$
    - Set $g_j^+ := g_j - \frac{\Delta_j}{X}\cdot g_{J^*}$
  - Else if $j = J^*$
    - Set $g_j^+ := X\cdot g_{J^*} - \frac{D(X)g_{J^*}}{\Delta_j}\cdot g_j$
    - Set $i^* = (\frac{X - \frac{D(X)g_{J^*}}{\Delta_j}}{\Delta_j}\cdot g_j, J^*)$

Note that the introduction of the additional set of variables $\{g_j^+\}$ is only for clarity of presentation. For a proof that at the end of Köter’s iteration it indeed holds that $G^+ = \{g_0^+,\ldots,g_{\ell}^+\}$ is a Gröbner basis for $M^+$ and for all $j$, $LM(g_j^+)$ contains the $j$-th unit vector, see [24, Sec. VII.C].

**Remark:** Algorithms similar to Köter’s iteration were already presented in [1], [2], which came before [21] (see [18, Sec. 2.6]). For some problems over univariate polynomial rings, algorithms for computing the (weak or canonical) shifted Popov form of $K[X]$-matrices have the lowest asymptotic complexity; see, e.g., [25] and [28] and the references therein. We currently do not know if such methods can be used to further reduce the complexity in the context of the present paper.

**D. Hasse Derivatives and Multiplicities**

In this subsection, we recall some definitions and results regarding Hasse derivatives and multiplicities of polynomials in $K[X_1,\ldots,X_n]$.

In what follows, we let $X := (X_1,\ldots,X_n)$ and $Y := (Y_1,\ldots,Y_n)$, where $X_1,\ldots,X_n,Y_1,\ldots,Y_n$ are algebraically independent over $K$. For $g(X) \in K[X]$ and $i = (i_1,\ldots,i_n) \in \mathbb{N}^n$, the $i$-th Hasse derivative of $g$ is defined as the unique coefficient $g^{(i)}(X)$ of $Y^i := Y_1^{i_1}\ldots Y_n^{i_n}$ in the decomposition $g(X+Y) = \sum g^{(j)}(X)Y^j$.

For binomial coefficients, we will use the convention that $\binom{\ell}{k} = 0$ if $\ell > k$. We omit the straightforward proof of the following well-known proposition.

**Proposition 2:** If $g(X) = \sum a_i X^i \in K[X]$, then for all $j$,

$$g^{(j)}(X) = \sum_i a_i \binom{i}{j_1} \ldots \binom{i}{j_n} X^{i-j}$$

(note that all terms with a non-zero coefficient involve only non-negative exponents).

For $i = (i_1,\ldots,i_n)$, let $\omega(i) := \sum i_j$. For non-zero $g \in K[X]$ and for $a \in K^n$, we define the multiplicity $\text{mult}(g, a)$ of $g$ at $a$ to be the unique integer $m$ for which for all $i = (i_1,\ldots,i_n)$ with $\omega(i) < m$, $g^{(i)}(a) = 0$, while there exists an $i$ with $\omega(i) = m$ for which $g^{(i)}(a) \neq 0$. If $g$ is the zero polynomial, then we define $\text{mult}(g, a) := \infty$ for all $a$.

Note that by the definition of the Hasse derivative, $g^{(i)}(a)$ is just the coefficient of $X^i$ in the decomposition $g(a+X) = \sum g^{(i)}(a)X^i$ for all $X \in K$ for all $i$. Hence, $\text{mult}(g, a)$ is simply the smallest value of $\omega(i)$ for a monomial $X_1^{i_1}\ldots X_n^{i_n}$ appearing in $g(X+a)$.

The following proposition, involving polynomials with a (possibly) different number of variables, is a special case of [10, Prop. 6].

**Proposition 3 ([10]):** Let $g \in K[X_1,\ldots,X_n]$, and let $h_1,\ldots,h_n \in K[Y_1,\ldots,Y_n]$. Then for all $a \in K^n$,

$$\text{mult}(g(h_1(Y),\ldots,h_n(Y)), a) \geq \text{mult}(g(h_1(a),\ldots,h_n(a)))$$

(3)

**E. Partially Homogenized Polynomials**

Trifonov [35] introduced partially homogenized trivariate polynomials as a means for avoiding points at infinity in Wu’s list decoding algorithm [38]. While in this paper we eventually do (implicitly) use points at infinity, it is still convenient to work with partially homogenized polynomials.

For a field $K$, a $(Y,Z)$-homogeneous polynomial is a polynomial of the form

$$f(X,Y,Z) := \sum_{\rho=0}^{\rho} a_{i,j} X^i Y^{i-j} Z^{\rho-j} \in K[X,Y,Z]$$

for some $\rho \in \mathbb{N}$, where $a_{i,j} \in K$ for all $i,j$. We will say that $f$ has a $(Y,Z)$-homogeneous degree $\rho$, and write $\text{Homog}_K^{Y,Z}(\rho)$ for the $K[X]$-module of polynomials in $K[X,Y,Z]$ which are $(Y,Z)$-homogeneous degree $\rho$. We also write $\text{Homog}_K^{Y,Z} := \bigcup_{\rho \in \mathbb{N}} \text{Homog}_K^{Y,Z}(\rho)$. When the underlying field $K$ is clear from the context, we will write simply $\text{Homog}_K^{Y,Z}(\rho)$ and $\text{Homog}_K^{Y,Z}$ for $\text{Homog}_K^{Y,Z}(\rho)$ and $\text{Homog}_K^{Y,Z}$, respectively.

**Definition 4:** For $f(X,Y,Z) := \sum_{\rho=0}^{\rho} a_{i,j} X^i Y^{i-j} Z^{\rho-j}$, where $\rho \in \mathbb{N}$ and the $a_{i,j}$ are in $K$, define the associated (de-homogenized) bivariate polynomial of $f$, $f_1(X,Y)$, by setting

$$f_1(X,Y) := f(X,Y,1) = \sum_{\rho=0}^{\rho} a_{i,j} X^i Y^j.$$
Also, let \( f^1(X, Y) \) by the \( Y \)-reversed version of \( f_1(X, Y) \), that is,
\[
f^1(X, Y) := f(X, 1, Y) = Y^\rho f_1(X, Y^{-1}).
\]

Note that \( f \) is determined by \( f_1 \) by homogenization, as \( f(X, Y, Z) = Z^\rho f_1(X, Y/Z) \). Hence \( f \mapsto f_1 \) is a bijection between \((Y, Z)\)-homogeneous trivariate polynomials of \((Y, Z)\)-homogeneous degree \( \rho \) and bivariate polynomials of \( Y \)-degree at most \( \rho \). In fact, this is an isomorphism of \( K[X] \)-modules, as \( f \mapsto f_1 \) is clearly \( K[X] \)-linear.

The following proposition from [35] assures that multiplicities are preserved when moving from \( f \) to \( f_1 \) (or \( f^i \)), where below we take special care of points of the form \((x_0, 0, 0)\).

**Proposition 5 ([35, Lemma 1]):** Let
\[
f(X, Y, Z) := \sum_{j=0}^\rho \sum_{i} a_{ij} X^i Y^j Z^{\rho-j} \in \text{Homog}_{Y,Z}(\rho),
\]
and let \((x_0, y_0, z_0) \in K^3\). Then

1. For all \( \alpha \in K^* \), mult\((f, (x_0, y_0, z_0)) = mult(f_1, (x_0, y_0, z_0))\).
2. If \( z_0 \neq 0 \), then \( mult(f, (x_0, y_0, z_0)) = mult(f_1, (x_0, y_0/z_0))\).
3. If \( y_0 \neq 0 \), then \( mult(f, (x_0, y_0, z_0)) = mult(f_1, (x_0, y_0, z_0))\).
4. \( mult(f, (x_0, 0, 0)) \geq mult(f_1, (x_0, 0, 0))\).

Parts 1–3 were proved in [35]. For a proof of part 4, see Appendix A.

III. AN ISOMORPHISM BETWEEN TWO SOLUTION MODULES

The solution set of the key equation \( u' \equiv Su \mod (X^{2t}) \) is a \( F_{2^t} \)-vector subspace of \( F_{2^t}[X] \) that is not an \( F_{2^t}[X] \)-submodule (i.e., an ideal). Also, to work with Gröbner bases, it will be convenient to work with a submodule of \( F_{2^t}[X]^2 \). In this section, we will see that with an appropriate scalar polynomial multiplication, the key equation does define a module, and this module is isomorphic to a sub-module of \( F_{2^t}[X]^2 \) which will be useful for a Gröbner-bases formulation.

**Definition 6:** For a field \( K \), an even \(^4\) integer \( s \in \mathbb{N} \), and a polynomial \( f(X) = f_0 + f_1 X + \cdots + f_s X^s \in K[X] \), the odd part of \( f(X) \), \( f(X) \), is
\[
odd \quad f(X) := f_1 + f_3 X + \cdots + f_{s-1} X^{s-1},
\]
while the even part of \( f(X) \), \( f(X) \), is
\[
even \quad f(X) := f_0 + f_2 X + \cdots + f_s X^s/2.
\]

In the above definition, note that \( f(X) = even \ f(X) + X \ odd \ f(X) \).

**Definition 7:** 1) Let us define an \( F_{2^t}[X] \)-module, \( L \), as follows. As an abelian group, \( L = F_{2^t}[X] \), but the scalar multiplication \( \cdot_1 : F_{2^t}[X] \times L \to L \) is defined by \( f(X) \cdot_1 g(X) := f(X^2) g(X) \).

\(^4\)There is no loss in generality in the requirement that \( s \) is even.

2) Let \( \mu : F_{2^t}[X] \times F_{2^t}[X] \to L \)
\[
(u(X), v(X)) \mapsto v(X^2) + Xu(X^2),
\]
and note that \( \mu \) is an isomorphism of \( F_{2^t}[X] \)-modules, with inverse \( f \mapsto (f \cdot f)^{5} \).

Let us now define two solution sets, one for the key equation, and the other for a modified key equation.

**Definition 8:** Let
\[
M := \{ u \in F_{2^t}[X] \mid u' \equiv Su \mod (X^{2t}) \},
\]
and
\[
N := \left\{ (u, v) \in F_{2^t}[X] \times F_{2^t}[X] \mid \begin{array}{c}
\text{even} \quad S(X) u \\ \text{odd} \quad S(X) v \\
1 + X S(X) \end{array} \right\}. \]

The following proposition relates \( M \) and \( N \). We note that the proof is inspired by the proof of the lemma on p. 25 of [5].

**Proposition 9:** \( M \) is an \( F_{2^t}[X] \)-submodule of \( L \) and \( \mu(N) = M \). Hence \( \mu \) restricts to an isomorphism \( \mu' : N \to M \), as in the following commutative diagram:

\[
\begin{array}{ccc}
F_{2^t}[X] \times F_{2^t}[X] & \xrightarrow{\sim} & L \\
\cup & & \cup \\
N & \xrightarrow{\sim} & M
\end{array}
\]

**Proof:** While the first assertion follows from the second, it is worthwhile (and straightforward) to prove it directly: First, \( M \) is clearly an abelian subgroup of \( L \), and so it remains to verify that for all \( f \in F_{2^t}[X] \) and all \( u \in M \), \( f \cdot_1 u \in M \). But
\[
\begin{align*}
(f(X) \cdot_1 u(X))^t &= (f(X^2) u(X))^t \\
&= f(X^2) u'(X) \equiv f(X^2) Su = S \cdot (f(X) \cdot_1 u(X)),
\end{align*}
\]
where characteristic 2 was used in the second equation, and where \( \equiv \) stands for congruence modulo \( X^{2t} \).

Let us now turn to the second assertion. For a polynomial \( u \in F_{2^t}[X] \) we have \( u' = odd \ u(X^2) \), and so \( u \in M \) iff
\[
odd \ u(X^2) \equiv \left( even \ u(X^2) X + odd \ u(X^2) \right) \cdot
\]
\[
\equiv \left( even \ u(X^2) + odd \ u(X^2) \right) \cdot
\]
\[
\equiv even \ \left( S(X^2) \ u(X^2) + X^2 \odd \ u(X^2) \right) \cdot
\]
\[
+ \ X \odd \ \left( S(X^2) \ u(X^2) + even \ S(X^2) \ u(X^2) \right) \cdot
\]
\[
\equiv odd \ \left( S(X^2) \ u(X^2) + even \ S(X^2) \ u(X^2) \right) \cdot
\]
\[
\equiv even \ \left( S(X^2) \ u(X^2) + odd \ u(X^2) \right) \cdot
\]
\[
(4)
\]
\(^5\)Characteristic 2 is not required for making \( L \) a module and for showing that \( \mu \) is a homomorphism.
Now, \((4)\) is equivalent to the following pair of equations:
\[
\begin{align*}
\text{odd } u(X^2) & \equiv \text{even } S(X^2) \text{ even } u(X^2) + \text{odd } X^2 S(X^2) \text{ odd } u(X^2) \mod (X^{2t}), \\
\text{odd } u(X^2) & \equiv \text{even } S(X^2) \text{ odd } u(X^2) \mod (X^{2t}).
\end{align*}
\]
and
\[
0 \equiv \text{odd } S(X^2) \cdot u(X^2) + \text{even } S(X^2) \cdot \text{odd } u(X^2) \mod (X^{2t}).
\]

Hence, \(u \in M\) iff \((5)\) and \((6)\) hold. To prove the assertion, we will show that for all \(u \in F_{2r}[X]\), \((i)\) \((6)\) follows from \((5)\) (so that \(u \in M\) iff \((5)\) holds), and \((ii)\) \((5)\) is equivalent to \(\mu^{-1}(u) \in N\) (so that \(u \in M\) iff \(\mu^{-1}(u) \in N\)).

Let us start with \((ii)\). Observe that for all \(u\), \((5)\) is equivalent to
\[
\text{odd } u(X) \equiv \text{even } S(X) \cdot u(X) + X \cdot S(X) \cdot u(X) \mod (X^t),
\]
(7)
Collecting terms in the last equation, and noting that
\[
1 + X \cdot S(X)
\]
indeed has an inverse in \(F_{2r}[X]/(X^t)\), we see that for all \(u \in F_{2r}[X]\), \((5)\) is equivalent to \(\mu^{-1}(u) \in N\), as required.

Moving to \((i)\), recall that in the binary case in question, for all \(s_i = S_i\) (see, e.g., [5]), and hence \(S(X)^2 = S_0 + S_1 X + \cdots + S_{2a-2} X^{2a-4}\)
\[
\equiv \text{odd } S(X)^2 \mod (X^{2t}).
\]
Hence
\[
\left(\text{even } S(X^2) + X \cdot S(X^2)\right)^2 \equiv \text{odd } S(X)^2 \mod (X^{2t}),
\]
that is,
\[
\left(\text{even } S(X^2)\right)^2 \equiv X^2 \cdot \text{odd } S(X^2)^2 + \text{odd } S(X)^2 \mod (X^{2t}),
\]
which is equivalent to
\[
\left(\text{even } S(X)\right)^2 \equiv X \cdot S(X)^2 \cdot \text{odd } S(X)^2 + S(X) \mod (X^t).
\]
(9)
Now, to prove \((i)\), assume that \((5)\) holds, so that \((7)\) also holds.

Multiplying \((7)\) by \(S(X)\) and writing \(\equiv\) for congruence modulo \((X^t)\), we get
\[
\text{odd } u(X) \cdot S(X) \equiv \left(\text{even } S(X)^2 \cdot \text{even } u(X) + X \cdot S(X) \cdot \text{odd } S(X)^2 \cdot \text{odd } u(X)\right) \equiv \left(1 + X \cdot S(X)\right) \cdot \text{odd } S(X)^2 \cdot \text{odd } u(X) + X \cdot S(X) \cdot \text{even } S(X)^2 \cdot \text{even } u(X) \mod (X^{2t}),
\]
(9)
Recall that \(S(X) = S_1 + S_2 X + \cdots + S_{a-1} X^{d-2}\), so that \(\text{odd } S(X) = S_0 + S_1 X + \cdots + S_{a-1} X^{d-3}/2\) (and \(d-3) = t-1\), recalling that \(d = 2t + 1\).

\((\ast)\) follows from \((9)\). Collecting terms and canceling \(1 + X \cdot S(X)\) (which is invertible in \(F_{2r}[X]/(X^t)\)), we get
\[
0 \equiv \text{odd } S(X)^2 \cdot \text{even } u(X) + \text{even } S(X) \cdot \text{odd } u(X) \mod (X^t),
\]
which is equivalent to \((6)\), as required.

The following proposition shows that the isomorphism \(\mu\) behaves properly with respect to the order (equivalently, with respect to \(w_{\deg(f)}\)) of a pair of polynomials.

**Proposition 10:** For \(f \in F_{2r}[X]\), it holds that
\[
\text{ord}(\mu^{-1}(f)) = \left\lfloor \frac{\deg(f)}{2} \right\rfloor = \left\lfloor \frac{\text{ord}(f^* f)}{2} \right\rfloor.
\]

**Proof:** We have
\[
\text{ord}(\mu^{-1}(f)) = \text{ord}(f, f^*)
\]
\[
= \begin{cases} 
\deg(f) & \text{if } 2 \mid \deg(f) \\
\deg(f) + 1 & \text{if } 2 \nmid \deg(f)
\end{cases}
\]
\[
= \left\lfloor \frac{\deg(f)}{2} \right\rfloor,
\]
where for \((a)\), note that if \(2 \mid \deg(f)\), then \(\deg(f^*) > \deg(f)\), while if \(2 \nmid \deg(f)\), then \(\deg(f^*) \geq \deg(f)\).

Let us now describe an application of Proposition 10 for HD unique decoding when the number \(\varepsilon\) of errors satisfies \(\varepsilon \leq t\). Recall that in this case, up to a non-zero multiplicative constant, \(\sigma\) is the unique element of \(M\) with minimum degree [29, Prop. 6.6].

**Proposition 11:** If \(\varepsilon \leq t\), then \(\mu^{-1}(\sigma)\) has the minimum leading monomial in \(N \setminus \{(0, 0)\}\) with respect to \(<_{-1}\) (this identifies \(\mu^{-1}(\sigma)\) uniquely up to a non-zero multiplicative constant).

**Proof:** First, it follows from Proposition 10 that there is no element of \(N \setminus \{(0, 0)\}\) with a strictly lower order than \(\text{ord}(\mu^{-1}(\sigma))\).

If \(\varepsilon = \deg(\sigma)\) is even, then it follows from Proposition 10 that elements of \(N\) with the same order as \(\mu^{-1}(\sigma)\) must be of the form \(\mu^{-1}(f)\) for \(f \in M\) with \(\deg(f) \in \{(\deg(\sigma), \deg(\sigma)-1)\}\). By minimality, such an \(f\) must be \(c \cdot \sigma\) for some \(c \neq 0\). We conclude that when \(\varepsilon\) is even, \(\mu^{-1}(\sigma)\) is the unique (up to a multiplicative constant) order-minimizing element of \(N\).

On the other hand, if \(\varepsilon\) is odd, then it follows from Proposition 10 that the only elements of \(N\) with the same order as \(\mu^{-1}(\sigma)\) (other than \(\mu^{-1}(c \cdot \sigma)\)) must be of the form \(\mu^{-1}(f)\) for \(f \in M\) with \(\deg(f) = \deg(\sigma) + 1\). Such an \(f\) has an even degree, which implies that \(\text{LM}_\text{deg}(\mu^{-1}(f))\) is on the right. However, \(\text{LM}_\text{deg}(\mu^{-1}(\sigma))\) is on the left (as \(\deg(\sigma)\) is odd), and also \(\text{LM}_\text{deg}(\mu^{-1}(\sigma)) = (X^{(\deg(\sigma)-1)/2}, 0) <_{-1} (0, X^{(\deg(\sigma)-1)/2}) = \text{LM}_\text{deg}(\mu^{-1}(f))\), by the definition of \(<_{-1}\).

It follows from Proposition 11 that HD decoding for up to \(t\) errors can be performed by finding a Gröbner basis for the
module $N$ with respect to $<_{-1}$, and taking $\sigma$ as the $\mu$-image of the minimal element in the Gröbner basis. Finding a Gröbner basis for $N$ can be done, e.g., with any one of the algorithms described in [12]. Note that these algorithms include a BM-like algorithm (Alg. 4.7, described also in Appendix D), as well as an Euclidean algorithm (Alg. 3.7).

As the problem dimensions are halved in $N$, this part of HD decoding is substantially faster than decoding without considering the binary alphabet: for a typical $O(t^2)$ algorithm, the complexity is reduced by a factor of about 4. However, for performing the decoding in $N$, we need to calculate the modified syndrome $\tilde{S}(X)$, defined as the unique polynomial of degree $< t$ in the image of

$$R(X) := \frac{S(X)}{1 + X S(X)} \in \mathbb{F}_{2^t}[X]/(X^t)$$

in $\mathbb{F}_{2^t}[X]/(X^t)$ (so that $\tilde{S}(X)$ consists of the first $t$ terms of $R$).

Writing

$$R(X) = a_0 + a_1 X + a_2 X^2 + \cdots$$

$$S(X) = b_0 + b_1 X + \cdots + b_{t-1} X^{t-1}$$

$$\tilde{S}(X) = c_0 + c_1 X + \cdots + c_{t-1} X^{t-1}$$

(with $b_i = S_{2i+1}$ and $c_i = S_{2(i+1)}$ for all $i$), it follows from

$$(a_0 + a_1 X + a_2 X^2 + \cdots)(1 + c_0 X + c_1 X^2 + \cdots + c_{t-1} X^{t-1}) = b_0 + b_1 X + \cdots + b_{t-1} X^{t-1}$$

that $a_0 = b_0$, and for all $i \in \{1, \ldots, t-1\}$, $a_i$ can be recovered recursively as

$$a_i = b_i + a_{i-1} c_0 + a_{i-2} c_1 + \cdots + a_0 c_{i-1}. \quad (11)$$

The overall number of multiplications required for calculation $a_1, \ldots, a_{t-1}$ is therefore $t(t-1)/2$. Using the complexity estimations from [13], this is again about 1/4 of the complexity of decoding with [12, Alg. 4.7] without using the binary alphabet. So, calculating the modified syndrome and a Gröbner-basis for $N$ requires about half the operations for finding the ELP without taking advantage of the binary alphabet.

Similarly to [17], additional calculations are required if only one coordinate is maintained in the Gröbner-basis algorithm, and its output includes only the even part of the ELP; this is the case for [12, Alg. 4.7], but not for the Euclidean algorithm [12, Alg. 3.7]. In such a case, the modified key equation should be used to calculate the odd part of the ELP, at a cost of $t(t+1)/2$ additional multiplications. Hence in such a case, the gain over decoding without using the binary alphabet is by factor of about 3/4.

We note that the idea of calculating a modified syndrome and working with half the dimensions appears also in [7] (with the Euclidean algorithm) and in [17] (with the BM algorithm), where the modified key equation appears implicitly in the form of a system of linear equations relating the even and odd parts, including the recursion (11). However, [7] and [17] do not show that this is a consequence of an isomorphism between two solution modules, and that any algorithm for finding a Gröbner basis can be used. Also, the derivation in [17] uses an entirely different method, building on Newton’s identities, while the current proof is simpler, using only elementary properties of polynomials. Also, in [17] there are two different systems of equations for even/odd $t$. Finally, we emphasize that the modified key equation in the form $u \equiv S \text{ mod } (X^t)$ appearing in the definition of $N$ in Definition 8 does not appear in [17].

We also note that Berlekamp’s well-known method of using his algorithm only for half of the iterations in the binary case [5, pp. 24ff] is not applicable to other decoding algorithms, such as the Euclidean algorithm. Moving to the modified syndrome enables to benefit from the binary alphabet with any algorithm for computing a Gröbner basis to the solution module of a key equation.

IV. A GRÖBNER-BASES FORMULATION OF THE SD WU LIST DECODING ALGORITHM FOR BINARY BCH CODES

The Gröbner-bases formulation of the (HD and SD) Wu list decoding algorithm for binary BCH codes is missing in the literature. While quite similar to the case of irreducible Goppa codes [3], there are some differences that must be considered, even for the HD case. For example, while any non-zero polynomial has a multiplicative inverse modulo an irreducible polynomial, this is not the case when working modulo $X^{d-1}$. As another example, for BCH codes it seems more natural to use a different monomial ordering than that of [3]. Also, considering SD decoding, where different coordinates may have different multiplicities, a refinement of the interpolation theorem is required. Hence, for completeness, we will briefly sketch the Gröbner bases formulation of the (SD) Wu list decoding algorithm for binary BCH codes.

The following proposition is a small adjustment of [3, Prop. 2] to the monomial ordering of the current paper. The proof is similar to that of [3, Prop. 2], and is omitted.

**Proposition 12** (f 3, Prop. 2): Let $(h_1 = (h_{10}, h_{11}), h_2 = (h_{20}, h_{21}))$ be a Gröbner basis for the module $N$ of Definition 8 with respect to the monomial ordering $<_{-1}$, and assume w.l.o.g. that $LM(h_1)$ contains $(1, 0)$ and $LM(h_2)$ contains $(0, 1)$. Then

1) $\deg(h_{10}) + \deg(h_{21}) = t$.

2) For $u = (u_0, u_1) \in N$, let $f_1, f_2 \in \mathbb{F}_{2^t}[X]$ be the unique polynomials such that $u = f_1 h_1 + f_2 h_2$. Put $\varepsilon' := \deg(\mu(u)) = \deg(u_1 X^2 + X u_0 X^2)$.

   a) If $\varepsilon'$ is even, then $\varepsilon' = 2 \deg(u_1)$, then

   $$\deg(f_1(X)) \leq \deg(u_1(X)) - t + \deg(h_{21}(X)) =: w_1^{\text{even}}$$

   $$\deg(f_2(X)) = \deg(u_1(X)) - \deg(h_{21}(X)) =: w_2^{\text{even}}$$

   (the right-most equations define $w_1^{\text{even}}$, $w_2^{\text{even}}$).

   b) Otherwise, if $\varepsilon'$ is odd, so that $\varepsilon' = 2 \deg(u_0) + 1$, then

   $$\deg(f_1(X)) = \deg(u_0(X)) - t + \deg(h_{21}(X)) =: w_1^{\text{odd}}$$

   $$\deg(f_2(X)) \leq \deg(u_0(X)) - \deg(h_{21}(X)) =: w_2^{\text{odd}}$$
Definition 13: 1) Fixing $h_i$ ($i = 1, 2$) as in Proposition 12, let $\hat{h}_i := \mu(h_i)$, so that $\hat{h}_i$ is the polynomial obtained by “gluing” the odd and even parts specified in $h_i$.

2) Take $u := \mu^{-1}(\sigma)$ in part 2 of Proposition 12, so that $\varepsilon' = \deg(\sigma) = \varepsilon$ is the number of errors in the received word $y$. For $i = 1, 2$, let $f_i(X) \in \mathbb{F}_{2^r}[X]$ be the unique polynomials by the proposition for this choice of $u$, and let

$$w_i := \begin{cases} w_i^{\text{even}} & \text{if } \varepsilon \text{ is even} \\ w_i^{\text{odd}} & \text{if } \varepsilon \text{ is odd}. \end{cases}$$

Remark 14: Note that $w_1 + w_2 = \varepsilon - t - 1$, regardless of the parity of $\varepsilon$.

To continue, we note that since $\mu$ restricts to an isomorphism $N \cong M$, $M$ is free of rank 2, and $\mu$ induces a bijection between the set of bases of $N$ and the set of bases of $M$. The following proposition is a counterpart to the last part of the proof of [26, Prop. 5.27].

Proposition 15: If $\{g_1(X), g_2(X)\}$ is any free-module basis for $M$ (as a submodule of $L$), then $\gcd(g_1, g_2) = 1$.

Proof: Note that the $L$-submodule generated by a set of polynomials is contained in the ideal generated by the same polynomials. Hence if $r(X)$ is a common factor of $g_1, g_2$, then $r(X)$ is a factor of every element of $M$. To prove the assertion, it is therefore sufficient to show that there are some coprime polynomials in $M$. Clearly, $X^{2i+1} \in M$. Also, using (8), it can be verified that $1 + XS(X) \in M$.

Recall from Definition 13 that $f_1, f_2 \in \mathbb{F}_2[X]$ are the unique polynomials such that $\mu^{-1}(\sigma) = f_1 h_1 + f_2 h_2$, and that $\hat{h}_i := \mu(h_i)$, $i = 1, 2$. Since $\mu$ is a homomorphism, we have

$$\sigma = \mu(f_1 h_1 + f_2 h_2) = f_1(X) \cdot \mu(h_1) + f_2(X) \cdot \mu(h_2) = f_1(X^2) \hat{h}_1 + f_2(X^2) \hat{h}_2,$$

that is,

$$\sigma(X) = f_1(X^2) \hat{h}_1(X) + f_2(X^2) \hat{h}_2(X). \quad (12)$$

Hence,

$$f_1(x^2) \cdot \hat{h}_1(x) + f_2(x^2) \cdot \hat{h}_2(x) = 0, \forall x \in \text{Roots}(\sigma, \mathbb{F}_{2^r}), \quad (13)$$

and also, by Proposition 15,

$$\hat{h}_1(x) = 0 \implies \hat{h}_2(x) \neq 0, \forall x \in \mathbb{F}_{2^r}. \quad (14)$$

Remark 16: Since every element of $\mathbb{F}_{2^r}$ has a square root, we can write $f_i(X^2) = g_i(X)^2$ for some $g_i$, $i = 1, 2$. As noted in [3], since $\sigma$ is square-free, we must therefore have $\gcd(f_1, f_2) = 1$.

Remark 17: It follows from (14) that for all $g \in N$, written uniquely as $g = g_0 h_1 + g_1 h_2$, it holds that for all $x \in \mathbb{F}_{2^r}$,

$$\mu(g)(x) = 0 \iff \begin{cases} g_0(x^2) + \hat{h}_2(x) g_1(x^2) = 0 & \text{if } \hat{h}_1(x) \neq 0 \\
\hat{h}_1(x) = 0 & \text{if } \hat{h}_1(x) = 0. \end{cases} \quad (15)$$

We use subscripts starting from 0 to comply with the notation of the following sections, where the subscript stands for the power of the intermediate $Y$.

Note that $h_1$ and $h_2$ depend only on the received word (through the syndrome), and the decoder’s task is to find $f_1, f_2$, as $\sigma$ can be restored from $f_1, f_2$. The interpolation theorem indicates how (13) and (14) can be used for finding $f_1$ and $f_2$.

The following interpolation theorem is a refinement of a theorem of Trifonov [35]. We note that although for decoding we are only interested in the case where $u_1, u_2$ the theorem are coprime, this assumption is not required for the theorem.

For completeness, we include here the simple proof.

Theorem 18 ([35]): For any field $K$, let $Q(X, Y, Z) \in \text{Homog}_Y^K$, and let $u_1(X), u_2(X) \in K[X]$ be polynomials with $\deg(u_1(X)) \leq w_1'$ and $\deg(u_2(X)) \leq w_2'$ for some $w_1', w_2' \in \mathbb{Q}$. For $\varepsilon \in \mathbb{N}^*$, let $\{(x_i, y_i, z_i)\}_{i=1}^\varepsilon \subseteq K^3$ be a set of triples satisfying the following conditions:

1) The $x_i$ are distinct.
2) The $y_i$ and $z_i$ are not simultaneously zero, that is, for all $i$, $y_i = 0 \implies z_i \neq 0$.
3) For all $i$, $z_i u_1(x_i) + y_i u_2(x_i) = 0$.

Then, if

$$\sum_{i=1}^\varepsilon \text{mult}\left(Q(X, Y, Z), (x_i, y_i, z_i)\right) > \sum_{i=1}^\varepsilon \text{wdeg}_{w_1', w_2'}\left(Q(X, Y, Z)\right),$$

then $Q(X, u_1(X), u_2(X)) = 0$ in $K[X]$.

Proof: Write $I := \{1, \ldots, \varepsilon\}$, $I_1 := \{i \in I, y_i \neq 0\}$ and $I_2 := I \setminus I_1 = \{i \in I, z_i = 0\}$. Note that by assumption, for all $i \in I_2$, $y_i \neq 0$. Then

$$u_1(x_i) = -y_i u_2(x_i), \forall i \in I_1, \quad \text{and} \quad u_2(x_i) = -z_i u_1(x_i) / y_i = 0, \forall i \in I_2. \quad (16)$$

Now, it follows from Proposition 3 that

$$\sum_{i=1}^\varepsilon \text{mult}\left(Q(X, u_1(X), u_2(X)), x_i\right) \geq \sum_{i=1}^\varepsilon \text{mult}\left(Q(X, Y, Z), (x_i, u_1(x_i), u_2(x_i))\right). \quad (17)$$

Writing $m$ for the right-hand side of (17), we have

$$m = \sum_{i \in I_1} \text{mult}\left(Q, (x_i, -y_i u_2(x_i) / z_i, u_2(x_i))\right) + \sum_{i \in I_2} \text{mult}\left(Q, (x_i, u_1(x_i), -z_i u_1(x_i) / y_i)\right).$$

\[= \sum_{i \in I_1} \text{mult}\left(Q, (x_i, -y_i u_2(x_i), z_i u_2(x_i))\right) + \sum_{i \in I_2} \text{mult}\left(Q, (x_i, y_i u_1(x_i), -z_i u_1(x_i))\right) \geq \sum_{i \in I_1} \text{mult}\left(Q, (x_i, y_i, -z_i)\right) + \sum_{i \in I_2} \text{mult}\left(Q, (x_i, y_i, -z_i)\right) \geq \text{wdeg}_{w_1', w_2'}\left(Q(X, Y, Z)\right) \geq \text{deg}\left(Q(X, u_1(X), u_2(X))\right).$$
where (a) follows from (16), (b) follows from part 1 of Proposition 5, (c) follows from parts 1 and 4 of Proposition 5 (part 4 is required if \( u_2(x_i) = 0 \) or \( u_1(x_i) = 0 \) for some \( i \)), (d) is by assumption, and (e) holds since \( \deg u_i \leq w_i', i = 1, 2 \).

It follows that \( Q(X, u_1(X), v_2(X)) \) is the zero polynomial. \( \square \)

**Corollary 19:** For \( Q(X, Y, Z) \in \text{Homog}_{\mathbb{F}_2^{2\times 2}} \) and for the \( f_1, w_i, \hat{h}_i \) \( (i = 1, 2) \) from Definition 13, suppose that

\[
\sum_{x \in \text{Roots}(\sigma)} \text{mult} \left( Q(X, Y, Z), (x^2, \hat{h}_2(x), \hat{h}_1(x)) \right) > \text{wdeg}_{1, w_1, w_2} (Q(X, Y, Z)).
\]

Then \( Q(X, f_1(X), f_2(X)) = 0 \). Consequently, if \( f_2(X) \neq 0 \), then \( \{ Y f_2(X) + f_1(X) \} Q_1(X, Y) \) in \( \mathbb{F}_2[X, Y] \).

**Proof:** The first assertion is a straightforward consequence of (13), (14), and Theorem 18. For the second assertion, let \( \rho \) be the \( (Y, Z) \)-homogeneous degree of \( Q \). Then \( Q(X, Y, Z) = Z^\rho Q_1(X, Y/Z) \), and hence if \( f_2(X) \neq 0 \), \( Q_1(X, f_1(X)/f_2(X)) = 0 \). It follows that \( Y + f_1(X)/f_2(X) \) divides \( Q_1(X, Y) \) in \( \mathbb{F}_2[X, Y] \). Hence, we can write

\[
Q_1(X, Y) = \frac{u(X, Y)}{v(X)} \left( \frac{Y + f_1(X)}{f_2(X)} \right) = \frac{u_1(X, Y)}{v_1(X)} (Y f_2(X) + f_1(X)),
\]

for some \( u, u_1 \in \mathbb{F}_2[X, Y], v, v_1 \in \mathbb{F}_2[X] \), and where \( u_1(X, Y)/v_1(X) \) is a reduced fraction. As \( f_1, f_2 \) are coprime and \( Q_1(X, Y) \) in \( \mathbb{F}_2[X, Y], v_1(X) \) must be a constant. \( \square \)

**Definition 20:** For \( m = \{m_x \}_{x \in \mathbb{F}_2^{2\times 2}} \in \mathbb{N}^{2^{2\times 2}} \), let

\[
a(m) := \left\{ Q \in \mathbb{F}_2^{2\times 2}, (x, y, z) \right\} \exists x \in \mathbb{F}_2^{2\times 2}, \text{mult} \left( Q, (x^2, \hat{h}_2(x), \hat{h}_1(x)) \right) \geq m_x \}
\]

be the ideal of all polynomials having the multiplicities specified in \( m \). Also, For \( \rho \in \mathbb{N} \), let \( M(m, \rho) \) be the \( \mathbb{F}_2[X]- \)submodule of \( \mathbb{F}_2^{2\times 2} \)-submodule of \( \mathbb{F}_2^{2\times 2} \), defined by

\[
M(m, \rho) := a(m) \cap \text{Homog}_{\mathbb{F}_2^{2\times 2}}(\rho).
\]

To continue, it will be convenient to introduce some notation similar to that of [22]. Regarding the following definition, we note that it is important to take special care when considering the number of free variables with non-integer weights. In this context, we will restrict attention to weights (and weighted degrees) in \( \frac{1}{2} \mathbb{Z} \).

**Definition 21:** 1) For \( \rho \in \mathbb{N}, w_1', w_2' \in \frac{1}{2} \mathbb{Z}, \) and \( d \in \frac{1}{2} \mathbb{N} \), let

\[
N^\rho_{w_1, w_2}(d) := \left\{ (i, j) \in \mathbb{N}^{2} \mid i + w_1' j + w_2' (\rho - j) \leq d \text{ and } j \leq \rho \right\}
\]

be the number of free coefficients in a polynomial \( Q \in \text{Homog}_{\mathbb{Z}}(\rho) \) with \( \text{wdeg}_{1, w_1, w_2}(Q) \leq d \). Note that \( N^\rho_{w_1, w_2}(d) \) is always an integer.

2) For \( n_{eq} \in \mathbb{N} \), let \( \Delta^\rho_{w_1, w_2}(n_{eq}) := \min \{ d \in \frac{1}{2} \mathbb{N} \mid N^\rho_{w_1, w_2}(d) > n_{eq} \} \) be the minimum \( (1, w_1', w_2') \)-weighted degree required for getting at least \( n_{eq} \) \( + 1 \) free coefficients in a polynomial \( Q \in \text{Homog}_{\mathbb{Z}}(\rho) \).

3) For \( m = \{m_x \}_{x \in \mathbb{F}_2^{2\times 2}} \), let \( \text{cost}(m) := \frac{1}{2} \sum_{x \in \mathbb{F}_2^{2\times 2}} m_x (m_x + 1) \).

The following simple bound on \( \Delta^\rho_{w_1', w_2'}(n_{eq}) \) appears implicitly in [35] for integer weights. It also appears implicitly in [3] and [26] for non-integer weights.

**Proposition 22:** With the notation of Definition 21, suppose that \( w_1' + w_2' \geq 0 \) and that exactly one of \( w_1', w_2' \) is an integer (and hence equals an odd multiple of \( 1/2 \)). Then

\[
\Delta^\rho_{w_1', w_2'}(n_{eq}) \leq \begin{cases} 
\frac{n_{eq} + 1/4}{\rho + 1} + \frac{\rho}{2} (w_1' + w_2') & \text{if } \rho \text{ is odd} \\
\frac{n_{eq} + 1/4}{\rho + 1} + \frac{\rho}{2} (w_1' + w_2') & \text{if } \rho \text{ is even}.
\end{cases}
\]

If both \( w_1', w_2' \) are integers, then the first case of (18) holds, with an integer \( \Delta^\rho_{w_1', w_2'}(n_{eq}) \).

Since the case of non-integer weights is subtle, we include the proof of Proposition 22 in Appendix B.

**Corollary 23:** Let \( m \in \mathbb{N}^{2\times 2} \) be a vector of multiplicities, let \( \rho \in \mathbb{N} \), and let \( f_1, w_i, \hat{h}_i \) \( (i = 1, 2) \) be as in Definition 13. Then,

1) There exists a non-zero polynomial \( P \in M(m, \rho) \) with \( \text{wdeg}_{1, w_1, w_2}(P) \leq \Delta^\rho_{w_1, w_2}(\text{cost}(m)) \).
2) Let \( P \in M(m, \rho) \) be a polynomial satisfying the condition in part 1. Then, if

\[
\sum_{x \in \text{Roots}(\sigma)} m_x \geq \Delta^\rho_{w_1, w_2}(\text{cost}(m)),
\]

then \( P(X, f_1(X), f_2(X)) = 0 \). In particular, if (19) holds, then a non-zero polynomial \( Q \in M(m, \rho) \) of minimum \( (1, w_1, w_2) \)-weighted degree satisfies \( Q(X, f_1(X), f_2(X)) = 0 \).

**Proof:** 1. Follows from the usual “more variables than equations” argument (recalling Proposition 5).

2. Writing \( p_x := (x^2, \hat{h}_2(x), \hat{h}_1(x)) \), we have

\[
\sum_{x \in \text{Roots}(\sigma)} \text{mult}(P, p_x) \geq \sum_{x \in \text{Roots}(\sigma)} m_x \geq \Delta^\rho_{w_1, w_2}(\text{cost}(m)) \geq \text{wdeg}_{1, w_1, w_2}(P).
\]

The proposition now follows from Corollary 19. \( \square \)

Corollary 23 indicates that the decoder should search for a non-zero \( Q \in M(m, \rho) \) of minimum \( (1, w_1, w_2) \)-weighted degree. Up until now, we considered partially homogeneous trivariate polynomials (following Trifonov) for avoiding points at infinity. While the most efficient algorithms for list decoding require this setup, for the fast Chase decoding algorithm of the following section it is sufficient to consider Köttter’s iteration. For this purpose, it will be convenient to work with bivariate polynomials, by de-homogenizing.

**Proposition 24:** Let \( M_1(m, \rho) \) be the image of \( M(m, \rho) \) under the isomorphism \((\cdot)_{1}\) of Definition 4. Then \( M_1(m, \rho) \)

\[\text{Recall that } \text{cost}(m) \text{ is the number of homogeneous linear equations on the coefficients of a bivariate polynomial } P \in \mathbb{F}_2[X, Y] \text{ describing the condition } \forall x \in \mathbb{F}_2^2, \text{mult}(P, (x, y_x)) \geq m_x \text{ (for some family } \{y_x\}_{x}).\]
is the $\mathbb{F}_{2^r}[X]$-module of all polynomials $P \in \mathbb{F}_{2^r}[X,Y]$ satisfying
\begin{enumerate}
\item $\text{wdeg}(0,1)(P) \leq \rho$,
\item for all $x$ with $h_1(x) \neq 0$,
\[
\text{mult}\left(P, \left(x^2, \frac{h_2(x)}{h_1(x)}\right)\right) \geq m_{x-1},
\]
\item for all $x$ with $h_1(x) = 0$ (and hence with $\hat{h}_2(x) \neq 0$),
\[
\text{mult}\left(Y^\rho P(X,Y^{-1}), (x^2, 0)\right) \geq m_{x-1}.
\]
\end{enumerate}
Moreover, for a polynomial $R \in M(m, \rho)$, we have
\[
\text{wdeg}_{1,w_1,w_2}(R) = \text{wdeg}_{1,w_1-w_2}(R_1) + w_2\rho.
\]
Hence, if $P \in M_1(m, \rho) \setminus \{0\}$ has the minimal $(1, w_1-w_2)$-weighted degree in $M_1(m, \rho) \setminus \{0\}$, then the $R \in M(m, \rho)$ with $R_1 = P$ has the minimal $(1, w_1-w_2)$-weighted degree in $M(m, \rho) \setminus \{0\}$.

**Proof:** The characterization of $M_1(m, \rho)$ follows from Proposition 5, and the second assertion is easily verified. \□

Note that
\[
\begin{align*}
\hat{w}_1 - \hat{w}_2 &= \begin{cases}
2\text{deg}(h_1) - t - 1 & \text{if } \varepsilon \text{ is even} \\
2\text{deg}(h_1) - t & \text{if } \varepsilon \text{ is odd}.
\end{cases}
\end{align*}
\]

Hence $w_1 - w_2$ depends only on the parity of $\varepsilon$, and the decoder may know this parity if the even subcode of the BCH code considered up to this point is used instead of the BCH code itself (at the cost of loosing a single information bit). Alternatively, one may use a non-integer weight, as suggested in [3], in order to avoid losing an information bit. We note that while for list decoding there is a small gain in knowing the parity of $\varepsilon$, this is not the case for the fast Chase algorithm of the following section. Hence, we do not require working with the even subcode.

Let us now consider multiplicities vectors that take only a single non-zero value, say, $m$. Typically and informally, this non-zero multiplicity is assigned to all coordinates whose reliability is below some threshold.

**Corollary 25:** Suppose that $I \subseteq \mathbb{F}_{2^r}$ is the set of erroneous coordinates (so that $|I| = \varepsilon$). Let $J \subseteq \mathbb{F}_{2^r}$ be some subset, and for $m \in \mathbb{N}^*$, define $m$ by setting, for all $x \in \mathbb{F}_{2^r}$,
\[
m_x := \begin{cases}
m & \text{if } x \in J, \\
0 & \text{otherwise}.
\end{cases}
\]

Let also $\rho \in \mathbb{N}^*$, and put $w := 2\text{deg}(h_1) - t - 1$. Then if
\[
m \cdot |I \cap J| > \begin{cases}
\frac{|J| \cdot m(m+1)}{2(\rho+1)} + \frac{\rho}{2}(\varepsilon - t - 1/2) & \text{if } \rho \text{ is odd} \\
\frac{|J| \cdot m(m+1)/2}{2(\rho+1)} + \frac{\rho}{2}(\varepsilon - t - 1/2) & \text{if } \rho \text{ is even}
\end{cases}
\]
then a non-zero polynomial $Q(X,Y)$ minimizing the $(1, w)$-weighted degree in $M_j(m, \rho)$ satisfies $(Y f_2(X) + f_1(X))Q(X,Y)$ when $f_2(X) \neq 0$.

**Proof:** First note that when $\varepsilon$ is even, we may take $w_1 + 1/2$ instead of $w_1$ as an upper bound on $\text{deg}(f_1)$, while when $\varepsilon$ is odd, we may take $w_2 + 1/2$ instead of $w_2$ as an upper bound on $\text{deg}(f_2)$. In both cases, the result is adding $1/2$ to the sum $w_1 + w_2 = \varepsilon - t - 1$ (Remark 14), while replacing $w_1 - w_2$ by $2\text{deg}(h_2) - t - 1/2 = w$, regardless of the error parity. The assertion now follows from Corollary 23, Proposition 24, and Proposition 22. \□

For the fast Chase decoding algorithm of the following section, we take $\rho = 1$, and hence only the case of odd $\rho$ is relevant.

**Remark 26:** Corollary 25 is valid also when $\varepsilon \leq t$, in which case it holds also for $f_2 = 0$, as we shall now explain. When $\varepsilon \leq t$, (21) holds for $J = \emptyset$, for which $M_1(m, \rho)$ is just the set of polynomials of $Y$-degree at most $\rho$ in $\mathbb{F}_{2^r}[X,Y]$. There are two cases to consider, depending on the sign of $w$ (note that by definition, $w$ cannot be 0). If $w > 0$, then the minimum $(1, w)$-weighted degree is achieved exactly by any non-zero constant polynomial. Factoring, we find that $Y f_2 + f_1$ must also be a non-zero constant. In vector form, $(f_1, f_2) = (c, 0)$ for some $c \in \mathbb{F}_{2^r}$. Applying $\mu$, we obtain that in this case, $\sigma = c \cdot h_1$. On the other hand, if $w < 0$, then the minimum $(1, w)$-weighted degree is achieved exactly by the polynomials of the form $c Y^\rho$ ($c \in \mathbb{F}_{2^r}$), and factoring gives $Y f_2 + f_1 = c' Y$ for some $c'$. In vector form, $(f_1, f_2) = (0, c')$, so that $\sigma = c' \cdot h_2$. Note that as $w = \text{deg}(h_1) - \text{deg}(h_1) - 1/2$, we have $w > 0 \iff \text{LM}_{<1}h_1 < -1 \text{LM}_{<1}h_2$, and the above agrees with Proposition 11.

To complete the description of the SD Wu list decoding algorithm, one has to specify methods for translating the channel reliability information into the multiplicity vector $m$ in Corollary 23, to find an appropriate value of the list size $\rho$, to consider efficient interpolation (i.e., finding the minimizing $Q$), efficient factorization, etc. However, this is outside the main scope of the current paper. Also, most of the above questions can be answered using small modifications of existing works, such as [22], [3], and [38] and others. We therefore omit further details.

V. THE FAST CHASE ALGORITHM

A. Fast Chase Decoding on a Tree

In the original Chase-II decoding algorithm [8] for decoding a binary code of minimum distance $d$, all possible $2^{(d/2)}$ error patterns on the $\lfloor d/2 \rfloor$ least reliable coordinates are tested (i.e., subtracted from the received word). Bounded distance decoding is performed for each tested error pattern, resulting in a list of $\left\lceil 2^{(d/2)} \right\rceil$ candidate codewords. If the list is non-empty, then the most likely codeword in the list is chosen as the decoder output.

The type of Chase algorithm considered in the current paper is the following variant of the Chase-II algorithm. As in [8], it is assumed that the decoder has probabilistic reliability information on the received bits. Using this information, the decoder identifies the $\eta$ least reliable coordinates for some fixed $\eta \in \mathbb{N}^*$. Labeling coordinates by elements of $\mathbb{F}_{2^r}$, and writing $\alpha_1, \ldots, \alpha_\eta$ for the least reliable coordinates, let $U := \{\alpha_1, \ldots, \alpha_\eta\}$.

The Chase decoding considered in the current paper runs over all test error patterns on $U$ that have a weight of up
to some $r_{\text{max}} \in \{1, \ldots, \eta\}$, and performs (the equivalent of) bounded distance decoding for each such pattern. When $r_{\text{max}} = \eta$, the examined error patterns are all the error patterns on $U$.

As in [30] and [39], a directed tree $T$ of depth $r_{\text{max}}$ is constructed as follows. The root is the all-zero vector in $\mathbb{F}_q^\eta$, and for all $r \in \{1, \ldots, r_{\text{max}}\}$, the vertices at depth $r$ are the vectors of weight $r$ in $\mathbb{F}_q^\eta$. To define the edges of $T$, for each $r \geq 1$ and for each vertex $\beta = (\beta_1, \ldots, \beta_\eta) \in \mathbb{F}_q^\eta$ at depth $r$ with non-zero entries at coordinates $i_1, \ldots, i_r$, we pick a single vertex $\beta' = (\beta'_1, \ldots, \beta'_r)$ at depth $r - 1$ that is equal to $\beta$ on all coordinates, except for one $\ell \in \{1, \ldots, r\}$, for which $\beta'_\ell = 0$. Note that there are $r$ distinct ways to choose $\beta'$ given $\beta$, and we simply fix one such choice. Now the edges of $T$ are exactly all such pairs $(\beta', \beta)$. For examples of the tree $T$, we refer to Figure 1 (see also [39], Fig. 1, [30], Fig. 1).

The edge $(\beta', \beta)$ defined above corresponds to adjoining exactly one additional flipped coordinate (i.e., $\alpha_{i_\ell}$). Hence, the edge $(\beta', \beta)$ can be alternately marked by $(\beta', \alpha_{i_\ell})$. Similarly, we can identify a path from the root to a vertex at depth $r \geq 1$ (and hence the vertex itself) with a vector $(\alpha_{i_1}, \ldots, \alpha_{i_r})$ with distinct $\alpha_{i_\ell}$’s.

The main ingredient of [30] and [39], as well as of the current paper, is an efficient algorithm for updating polynomials for adding a single modified coordinate $\alpha_{i_\ell}$. The tree $T$ is then traversed depth first, saving intermediate results on vertices whose out degree is larger than 1, and applying the polynomial-update algorithm on the edges. Because the tree is traversed depth first and has depth $r_{\text{max}}$, there is a need to save at most $r_{\text{max}}$ vertex calculations at each time (one for each depth). See, e.g., [30, Sec. 4.3] for details.

Observing the top tree in Figure 1, as well as the tree of [30, Fig. 1], we observe that in these examples, for $r \geq \lceil(\eta - 1)/2\rceil$, the out degree of each vertex is at most 1. This is a desirable property, since it means that a single common memory can be used for all depths $r \geq \lceil(\eta - 1)/2\rceil$. The following proposition shows that the tree can always be constructed in this way.

**Proposition 27:** Using the above notation, suppose that $r_{\text{max}} \geq 1 + \lceil(\eta - 1)/2\rceil$. Then the tree $T$ can be constructed such that for all $\lceil(\eta - 1)/2\rceil \leq r < r_{\text{max}}$, all vertices at depth $r$ have an out degree of at most 1.

**Proof:** Suppose that $r$ satisfies the condition in the theorem. Let $U, V \subseteq \mathbb{F}_q^\eta$ be the sets of vectors of weight $r + 1$ and $r$, respectively. Let $G = (U, V, E)$ be the bipartite graph with bipartition $(U, V)$ and with edge set $E \subseteq U \times V$ defined by $\{(u, v) \in E \mid \text{the support of } v \text{ is contained in that of } u \}$ (equivalently, $v$ can be obtained from $u$ by replacing a single “1” by a “0”). Note that $G$ is $(r + 1, \eta - r)$-regular. When $r + 1 \geq \eta - r$, a standard application of Hall’s Theorem (see, e.g., [6, Thm. 16.4]) shows that $G$ has a matching that covers $U$: Let $S \subseteq U$, and let $N(S) \subseteq V$ be the neighborhood of $S$. Let $n_e$ be the number of edges between $S$ and $N(S)$. Then on one hand, $n_e = (r + 1)|S|$, while on the other hand, $n_e \leq (\eta - r)|N(S)|$. Hence $|N(S)| \geq \frac{r + 1}{\eta - r}|S|$, and by assumption $r + 1 \geq \eta - r$. We conclude that the condition of Hall’s theorem is satisfied, and $G$ has a matching that covers $U$. By the definition of $G$, this proves the assertion.

We note that finding a maximum matching in a bipartite graph of $N_v$ vertices and $N_e$ edges can be done in complexity $O(\sqrt{N_v} \cdot N_e)$ [11, Sec. 6.4], where in Proposition 27, $N_v = \binom{\eta}{r+1} + \binom{\eta}{r}$, and $N_e = \binom{\eta}{r+1} \cdot (r + 1)$.

**B. The Update Rule on an Edge of the Decoding Tree**

In this subsection we will use the results on SD Wu list decoding from the previous section in order to define appropriate polynomials and an update rule for a tree-based fast Chase decoding of binary BCH codes.

**Definition 28:** For a subset $J \subseteq \mathbb{F}_q^\eta$, let $L(J) := M_1(1_J, 1)$, where $1_J$ is the vector $\{m_x\}_{x \in \mathbb{F}_q^\eta}$ with

$$m_x := \begin{cases} 1 & \text{if } x \in J, \\ 0 & \text{otherwise}. \end{cases}$$

For simplicity, we will sometimes write $L(x_1, \ldots, x_\ell)$ for $L(\{x_1, \ldots, x_\ell\})$, where $\ell \in \mathbb{N}$ and $x_1, \ldots, x_\ell \in \mathbb{F}_q^\eta$ are distinct.

Writing $J_1^{-1} := \{x| x^{-1} \in J \text{ and } \hat{h}_1(x) \neq 0\}$ and $J_2^{-1} := \{x| x^{-1} \in J \text{ and } \hat{h}_1(x) = 0\}$, it follows from Proposition 24 that $L(J)$ consists of all polynomials $g_0(X) + Yg_1(X)$ that satisfy the following two conditions:

$$g_0(x^2) + \frac{\hat{h}_2(x)}{\hat{h}_1(x)}g_1(x^2) = 0, \quad \forall x \in J_1^{-1}, \quad (22)$$

and

$$g_1(x^2) = 0, \quad \forall x \in J_2^{-1}. \quad (23)$$

**Remark 29:** Recalling Remark 17, it follows from (22) and (23) that $L(J)$ consists of all polynomials $g_0(X) + Yg_1(X)$
such that \( \mu(g_0 h_1 + g_1 h_2) \) vanishes on \( J^{-1} := \{ x | x^{-1} \in J \} = J_1^{-1} \cup J_2^{-1} \).

For the fast Chase decoding algorithm of the current paper, \( J \) will be the support of a error pattern. Recall that \( f_1, f_2 \in \mathbb{F}_2^*[X] \) are such that \( \sigma(X) = f_1(X^2) h_1(X) + f_2(X^2) h_2(X) \).

Recall also from Corollary 25 that \( w := 2 \deg(h_{21}) - t - 1/2 \). The following theorem shows that when the maximum list size is 1, the SD Wu list decoding algorithm becomes a means for flipping coordinates: assigning a multiplicity of 1 to a coordinate has the same effect as flipping it in a Chase-decoding trial.

**Theorem 30:** Suppose that \( f_2(X) \neq 0 \). For \( J \subseteq \mathbb{F}_2^* \), let \( n_1 \) be the number of erroneous coordinates on \( J \), and let \( n_2 \) be the number of correct coordinates on \( J \) (so that \( |J| = n_1 + n_2 \)). Then if \( n_1 - n_2 \geq \varepsilon - t \) and \( \varepsilon \geq t + 1 \), then for a non-zero polynomial \( g_0(X) + Y g_1(X) \) that minimizes the \((1, w)\)-weighted degree in \( L(J) \), it holds that
\[
(f_1(X) + Y f_2(X)) (g_0(X) + Y g_1(X)).
\]
Hence there exists some \( t(X) \in \mathbb{F}_2^*[X] \) such that
\[
g_0(X) + Y g_1(X) = t(X) (f_1(X) + Y f_2(X)),
\]
so that
\[
\mu(g_0 h_1 + g_1 h_2) = t(X^2) \sigma(X),
\]
Moreover, \( \gcd(t(X^2), \sigma(X)) = 1 \) and \( t(X^2) \) has no roots outside \( J^{-1} \).

**Proof:** Suppose that \( I \subseteq \mathbb{F}_2^* \) is the set of erroneous coordinates (so that \( |I| = \varepsilon \)). Then by definition, \( |I \cap J| = n_1 \), and (21) reads
\[
n_1 > \frac{n_1 + n_2}{2} + \frac{1}{2} (\varepsilon - t - 1/2),
\]
which is equivalent to \( n_1 - n_2 > \varepsilon - t - 1/2 \). Hence, this condition holds by assumption, and the first assertion follows from Corollary 25.

Next, we would like to prove that \( \gcd(t(X^2), \sigma(X)) = 1 \). Suppose not. Then for some error location \( \beta \), it holds that \( (X+\beta^{-1}) t(X^2) \). As before, we may write \( t(X^2) = t(X)^2 \) for some \( t(X) \), so that \( (X+\beta^{-1})^2 = X^2 + \beta^{-2} \) divides \( t(X^2) \). Hence, \( X + \beta^{-2} \) divides \( t(X) \). Moreover,
\[
\mu\left( \frac{g_0(X)}{X + \beta^{-2}} h_1 + \frac{g_1(X)}{X + \beta^{-2}} h_2 \right) = \frac{t(X^2)}{(X+\beta^{-1})^2} \sigma(X)
\]
has the same roots as \( \mu(g_0 h_1 + g_1 h_2) \). It therefore follows from Remark 29 that
\[
\frac{g_0(X)}{X + \beta^{-2}} + Y \frac{g_1(X)}{X + \beta^{-2}} \in L(J),
\]
contradicting the minimality of \( g_0(X) + Y g_1(X) \).

Finally, suppose that there exists some \( \beta \neq J \) such that \( \beta^{-1} \) is a root of \( t(X^2) \). As above, it follows that \( (X+\beta^{-2}) t(X) \), and (26) shows that the left-hand side of (27) is in \( L(J) \), as \( \mu(\cdots) \) in (26) has the same roots as \( \mu(g_0 h_1 + g_1 h_2) \) on \( J^{-1} \). Again, this contradicts the minimality of \( g_0(X) + Y g_1(X) \).

In Theorem 30, we have considered only the case \( f_2 \neq 0 \). If \( f_2 = 0 \), then \( f_1 \) must be some non-zero constant from \( \mathbb{F}_2^* \) (as \( \sigma(X) \) is square-free), so that \( \sigma(X) = c \cdot h_1(X) \) for some \( c \not \in \mathbb{F}_2^g \). In general SD Wu list decoding, one has to check directly whether \( \hat{h}_1(X) \) is a valid ELP by performing exhaustive substitution, etc. (even when the degree is beyond \( t \)), similarly to Step 3 of [3, Alg. 2]. However, for the fast Chase decoding algorithm of the current paper, this need not be checked separately, since if \( \sigma(X) = c \cdot h_1(X) \), the criterion for polynomial evaluation from the following section will assure in particular that the validity of \( h_1(X) \) as an ELP will be checked. For further details, see Subsection V-C ahead.

**Remark 31:** 1) Note that in Theorem 30, if \( n_2 = 0 \), that is, if \( J \) is a “direct hit” including only error locations, then it follows from (13), (14), (22), (23) that \( f_1(X) + Y f_2(X) \in L(J) \). Hence, the minimality of \( g_0(X) + Y g_1(X) \) and (24) imply that \( (f_1, f_2) = c \cdot (g_0, g_1) \) for some \( c \in \mathbb{F}^*_g \), when \( f_2 \neq 0 \). For speeding up the decoding, we would also like to consider the case of an “indirect hit”, that is, the case where \( n_1 - n_2 \geq \varepsilon - t \), while \( n_2 > 0 \) in the proposition. By (25), in this case we may restore \( t(X^2) \sigma(X) \) from a minimizing element in \( L(J) \), rather than \( \sigma(X) \) itself.

2) To restore \( \sigma(X) \) itself, one possible method is as follows. As \( \gcd(f_1, f_2) = 1 \), in Theorem 30 we have \( t(X) = \gcd(g_0, g_1) \). Hence, we may calculate \( t(X) \), and consequently \( f_1, f_2 \) from the output \( g_0, g_1 \). There is also an additional method, that has a somewhat higher complexity, but avoids the calculation of \( t(X) = \gcd(g_0, g_1) \) and the division by \( t(X) \) – see Subsection V-C for more details.

For simplicity, from this point on we will sometimes identify a polynomial \( g_0(X) + Y g_1(X) \) with the vector \( (g_0(X), g_1(X)) \) without further mention; it should be clear from the context which representation is used.

Theorem 30 indicates that the decoder should look for a minimizing element in \( L(J) \), and such an element always appears in a Gröbner basis with respect to any monomial ordering that resolves ties for the \((1, w)\)-weighted degree, e.g., for the ordering \( <_w \) when working with the vector representation \( (g_0(X), g_1(X)) \). The core idea of the fast Chase decoding algorithm of the current paper is that a single application of Kötter’s iteration is required for moving from \( L(\alpha_{i_1}, \ldots, \alpha_{i_{r-1}}) \) to \( L(\alpha_{i_1}, \ldots, \alpha_{i_r}) \), where \( r \in \{1, \ldots, n\} \).

Recalling that the edges of the decoding tree \( T \) defined in the previous subsection correspond exactly to moving from a subset of \( \mathbb{F}_2^* \) to a subset containing one additional element, we have the following adaptation of Kötter’s iteration for the edges of \( T \). Note that the root of \( T \) contains the Gröbner basis \( \{g_1 = (1,0), g_2 = (0,1)\} \) for \( \mathbb{F}_2^*[X]^2 \).

**Algorithm A: Kötter’s iteration on an edge of \( T \)**

**Input**

- The weight \( w := 2 \deg(h_{21}) - t - 1/2 \)
- A Gröbner basis \( G = \{g_1 = (g_{10}, g_{11}), g_2 = (g_{20}, g_{21})\} \) for \( L(\alpha_{i_1}, \ldots, \alpha_{i_{r-1}}) \) with respect to \( <_w \), with \( L(\alpha_{i_1}, \ldots, \alpha_{i_{r-1}}) \) containing the \((j - 1)\)-th unit vector for \( j \in \{1, 2\} \)
- \( h_1(X), \hat{h}_2(X) \)
- The next error location, \( \alpha_{i_r} \), with \( i_r \notin \{i_1, \ldots, i_{r-1}\} \)

**Output**

- By convention, for \( r = 1 \), \( L(\alpha_{i_1}, \ldots, \alpha_{i_{r-1}}) = \mathbb{F}_2^*[X]^2 \).
A Gröbner basis $G^+ = \{g_i^+ = (g_{i0}^+, g_{i1}^+), g_2^+ = (g_{20}^+, g_{21}^+)\}$ for
$L(\alpha_1, \ldots, \alpha_{r-1}, \alpha_r)$ with respect to $<_w$, with $LM_{<_w}(g_j^+)$ containing the $(j-1)$-th unit vector for $j \in \{1, 2\}$

**Algorithm**

- For $j = 1, 2$, calculate $/\ast$ justification: (22), (23)

\[
\Delta_j := \begin{cases} 
   g_{j0}(\alpha_r^{-2}) + \tilde{h}_2(\alpha_r^{-1})g_{j1}(\alpha_r^{-2}) & \text{if } \tilde{h}_1(\alpha_r^{-1}) \neq 0 \\
   g_{j1}(\alpha_r^{-2}) & \text{if } \tilde{h}_1(\alpha_r^{-1}) = 0 
\end{cases}
\]

- Set $A := \{j \in \{1, 2\} | \Delta_j \neq 0\}$
- For $j \in \{1, 2\} \setminus A$, set $g_j^+ := g_j$
- Let $j^* \in A$ be such that $LM_{<_w}(g_{j^*}) = \min_{j \in A} \{LM_{<_w}(g_j)\}$
- For $j \in A$
  - If $j \neq j^*$
    - Set $g_j^+ := \Delta_j^{-1}g_j + g_j^*$
    - Else $\ast j = j^*$ 
  - Set $g_{j^*} := (X + \alpha_r^{-2})g_{j^*}$

Note that the update for $j = j^*$ is justified by the fact that replacing $g_j$ by $Xg_j$ has the effect of multiplying $\Delta_j$ by $\alpha_r^{-2}$. Note also that the update $g_j^+ := \Delta_j^{-1}g_j + g_j^*$ (which is equivalent to the usual update of the form $g_j^+ := g_j + \sum_{j \neq j^*} g_j^*$) appears in this form only for the purpose of the complexity analysis ahead. This update rule assures that when $|A| = 2$, both $g_j$ and $g_{j^*}$ are multiplied once by a constant, where $j = \{1, 2\} \setminus j^*$.

**Remark 32:** Some remarks regarding Algorithm A are in order.

1) By Remark 29, it is clear that there is a strict inclusion

\[ L(\alpha_1, \ldots, \alpha_{r-1}) \supseteq L(\alpha_1, \ldots, \alpha_{r-1}, \alpha_r). \]

In some detail, $v(X) := X^{2t} - \prod_{i=0}^{r-1}X^2 + \alpha_r^{-2}$ is in the solution module $M$, and for the unique $g_0(X), g_1(X)$ such that $\mu^{-1}(v) = g_0h_1 + g_1h_2$, it holds that $g_0 + Yg_1 \in L(\alpha_1, \ldots, \alpha_{r-1})$, but $g_0 + Yg_1 \notin L(\alpha_1, \ldots, \alpha_r)$. Hence, in Algorithm A, $\Delta_j$ can be zero for at most one value of $j$, otherwise the above two modules would share a Gröbner basis.

2) The evaluations $\tilde{h}_1(\alpha_r^{-1}), \tilde{h}_2(\alpha_r^{-1})$ on all $\eta$ unreliable coordinates can be calculated in advance before starting the fast Chase algorithm, and so can the quotients $\tilde{h}_i(\alpha_r^{-1})$ (when $\tilde{h}_i(\alpha_r^{-1}) \neq 0$), and the values $\alpha_r^{-2}$ to be substituted in various polynomials.

3) Actually, it is not essential to work with a non-integer weight, and we could have chosen $w = 2 \deg(h_2) + 1$, i.e., the lower of the two values in (20). The reason that this value of $w$ works also for the odd case (where we should use $w + 1$ instead) is the following fact: For integer $w$, the orders $\prec_w$ and $\prec_{w+1}$ differ only in the way they compare monomials that have the same $(1, w+1)$-weighted degree (see Appendix C for details). In other words, if $w_{\deg(1,w+1)}(a, b) > w_{\deg(1,w+1)}(c, d)$, then $LM_{<w}(a, b) > LM_{<w}(c, d)$. Hence, if we only wish to minimize the $(1, w+1)$-weighted degree, we can use $<_w$ instead of $<_{w+1}$.\(^{12}\) Note that this observation is relevant only for $\rho = 1$; in the more general context of list decoding, we still have to use a non-integer $w$.

**C. A Criterion for Polynomial Evaluation, and Efficient Evaluation**

1) The Stopping Criterion: In [39], Wu suggested a stopping criterion for avoiding unnecessary exhaustive substitution in polynomials. Wu’s criterion is based on the idea that a length variable in his algorithm does not increase if the true ELP was found, and an additional flip hits a correct error location. Here, we have a similar stopping criterion based on the discrepancy of Algorithm A, in the lines of the criterion in [30].

In detail, using the terminology of Theorem 30, suppose that for $J = \{\alpha_1, \ldots, \alpha_{r-1}\}$ it holds that $n_1 - n_2 \geq \varepsilon - t + 1$, and that $\alpha_r$ is an error location. Then for $J' = \{\alpha_1, \ldots, \alpha_{r-1}\}$ we have $n_1 - n_2 \geq \varepsilon - t$, and the condition of Theorem 30 holds.

By the theorem, if $\tilde{h}_1$ is not the ELP (up to a multiplicative constant), then after performing Algorithm A on an edge reaching the vertex $(\alpha_1, \ldots, \alpha_{r-1})$, we have $\mu(g_0^+h_1 + g_1^+h_2) = t(\mu(\sigma(X))$ for the unique $j \in \{1, 2\}$ for which $w_{\deg(1,w)}(g_j^+)$ is minimal.\(^{13}\) Hence, on the edge connecting $(\alpha_1, \ldots, \alpha_{r-1})$ to $(\alpha_1, \ldots, \alpha_{r-2})$, it holds that $\Delta_j = 0$, by Remark 17 and the assumption that $\alpha_r$ is an error location.

Moreover, if $\tilde{h}_1 = c \cdot \sigma(X)$ for some $c \neq 0$, suppose that at least one of the $\eta$ unreliable coordinates, say, $\alpha_i$, is an error location. As the root of the tree $T$ contains the Gröbner basis $\{(1, 0), (0, 1)\}$ of $\mathbb{F}_q[X]^2$, and $\mu(1-h_1 + 0 \cdot h_2) = \tilde{h}_1$, it holds that $\Delta_1 = 0$ on the edge connecting the root to the vertex $(\alpha_i)$. We conclude that when moving from the root of the tree to depth 1, the criterion for starting an exhaustive substitution is having $\Delta_1 = 0$, while for $r > 1$, the criterion for exhaustive substitution on an edge connecting depth $r - 1$ to depth $r$ is $\Delta_j = 0$ for the unique $j \in \{1, 2\}$ for which $w_{\deg(1,w)}(g_j)$ is minimal.

If $\varepsilon \leq t + r$ for $r + 1 \leq r_{\text{max}}$, and $r + 1$ of the errors are in the $\eta$ unreliable coordinates, then the required exhaustive substitution will never be missed. Note that this slightly degrades the decoding performance, as we need $r + 1$ errors on the unreliable coordinates, instead of the $r$ required to restore the ELP on some vertex.

\(^{12}\)It is not true that the order $\prec_w$ can be used to minimize the $(1, w')$-weighted degree for all $w' \geq w$. A vector $h$ in a submodule $P \subseteq K[X]^2$ for a field $K$ with a minimal $\prec_w$-leading monomial minimizes both the $(1, w)$-weighted degree and the $(1, w + 1)$-weighted degree, but not necessarily the $(1, w + 2)$-weighted degree, as it need not be a minimal $\prec_{w+1}$-minimal. For example, considering $P = \langle K[X]^2 \rangle$, $(0, X)$ is the unique monomial minimizing the $(1, 1)$-weighted degree, and both $(X, 0) < c_0$ $(0, X)$ have the minimal $(1, 0)$-weighted degree, while $(X, 0)$ has the minimal $(1, 1)$-weighted degree.

\(^{13}\)Since $w$ is non-integer, we must have $w_{\deg(1,w)}(g_0^+) \neq w_{\deg(1,w)}(g_1^+)$. In detail, since the leading monomial of $g_0^+$ is on the right, $w_{\deg(1,w)}(g_0^+)$ is non-integer. On the other hand, the leading monomial of $g_1^+$ is on the left, and therefore $w_{\deg(1,w)}(g_1^+)$ is an integer.
It is possible that the stopping criterion holds falsely. Heuristically, it is reasonable to assume that the probability that one discrepancy is 0 by accident is about $1/q$.\footnote{We remark that the event of meeting the stopping criterion on the edge connecting $(\alpha_{i-1}, \ldots, \alpha_{i-2}, \alpha_i)$ to $(\alpha_{i-1}, \ldots, \alpha_{i-2}, \alpha_{i+1})$ is exactly the event that the minimal element in $L(\alpha_{i-1}, \ldots, \alpha_{i-2}, \alpha_i)$ (with respect to $\leq_w$) happens to fall in $L(\alpha_{i-1}, \ldots, \alpha_{i-2}, \alpha_{i+1})$.} When this happens, an unnecessary exhaustive substitution is performed. This somewhat increases the decoding complexity, but has no effect on the probability of decoding successfully.

Example: For $q = 256$ ($n = 255$) and $t = 8$, a random error of weight $\varepsilon = 14$ was drawn $10^4$ times. Algorithm A was run on a path corresponding to $\varepsilon - t = 6$ totally random distinct positions, and the number of times for which the stopping criterion was falsely met was counted. The resulting empirical probability was calculated as the total number of false positives on an edge, divided by $6 \cdot 10^4$, and was equal to 1/251.05.

2) Efficient Evaluation by Finding $t(X)$ and Division:

Let $(g_0, g_1)$ be the vector with the minimum $(1, w)$-weighted degree in the current Gröbner basis. As shown in Remark 31, in case of success, we have $(g_0, g_1) = t(X)/(f_1, f_2)$, and $t(X) = \gcd(g_0, g_1)$. This leads to the following evaluation strategy:

- Before starting the fast Chase decoding algorithm, calculate and store $\{\hat{h}_1(x)\}_{x \in \mathbb{F}_q^2}, \{\hat{h}_2(x)\}_{x \in \mathbb{F}_q^2}$.
- When traversing the decoding tree $T$, if the stopping criterion holds,
  1. Calculate $t(X) = \gcd(g_0, g_1)$ using the Euclidean algorithm. This requires $O(r^2)$ multiplications (by Proposition 35 ahead).
  2. Calculate $f_1(X) := g_0(X)/t(X), f_2(X) := g_1(X)/t(X)$. This also requires $O(r^2)$ multiplications.
  3. For all $x \in \mathbb{F}_q^2$,
     - Calculate $a := f_1(x^{-2}), b := f_2(x^{-2})$.
     - Read the stored values $c := \hat{h}_1(x^{-1}), d := \hat{h}_2(x^{-1})$.
     - Calculate $ac + bd$. If the result is 0, then adjoin $x$ to a set $E$ of error locations.
  4. Calculate $\delta := \deg(f_1(x^2)\hat{h}_1(x) + f_2(x^2)\hat{h}_2(x))$ as
     \[
     \delta = \max\left\{2\deg(f_1) + 2\deg(h_{10}) + 1, 2\deg(f_2) + 2\deg(h_{21})\right\}
     \]
     (by Proposition 33 ahead) and check if $\delta = |E|$. If equality holds, adjoin the error vector with support $E$ to the list of output error vectors (see justification ahead).

Note that the comparison of the degree of the estimated ELP to the number of its roots is equivalent to checking that the syndrome of the error vector defined by the estimated ELP is equal to the syndrome of $\mathbf{y}$: First, the estimated ELP $\hat{\sigma} = \mu(f_1h_1 + f_2h_2)$ is separable, by the degree test. Also, since $\hat{\sigma} \in \mu(N) = M$, we may use [30, Prop. 4.3] with $\omega := \hat{\sigma}'$ to show that the error vector with locations defined by $\hat{\sigma}$ and values obtained by $\hat{\sigma}$ and $\omega := \hat{\sigma}'$ through Forney’s formula (see e.g., [30, Eq. (3)]) has the same syndrome as $\mathbf{y}$. Finally, observing Forney’s formula for the case where $\omega = \sigma'$ and $\alpha_i = \alpha_i$ for all $i$ (using the terminology of [30]), as is the case for BCH codes, we see that all the error values obtained with this formula must be 1.

It remains to justify the calculation of the degree $\delta$ in the above steps. For this, we have the following proposition.

**Proposition 33.** For any polynomials $u_1(X), u_2(X) \in \mathbb{F}_2[X]$, it holds that

\[
\deg(u_1(X^2)\hat{h}_1(X) + u_2(X^2)\hat{h}_2(X)) = \max\left\{2\deg(u_1) + 2\deg(h_{10}) + 1, 2\deg(u_2) + 2\deg(h_{21})\right\}.
\]

**Proof:** Recall that $\hat{h}_j(X) = h_{j1}(X^2) + Xh_{j0}(X^2)$ ($j = 1, 2$). Consider the degree of $h_{j1}(X^2) + Xh_{j0}(X^2)$. As the leading monomial of $h_{j1}(X)$ with respect to $<_{-1}$ is on the left, we have $\deg(h_{j0}) > \deg(h_{j1}) - 1$, that is, $\deg(h_{j0}) \geq \deg(h_{j1})$. Hence

\[
\deg(\hat{h}_1) = \deg(h_{11}(X^2) + Xh_{10}(X^2)) = 1 + 2\deg(h_{10}).
\]

Similarly, as the leading monomial of $h_2$ is on the right, we have

\[
\deg(\hat{h}_2) = \deg(h_{21}(X^2) + Xh_{20}(X^2)) = 2\deg(h_{21}).
\]

Now the assertion follows from the fact that the degrees of $u_1(X^2)\hat{h}_1(X)$ and $u_2(X^2)\hat{h}_2(X)$ are distinct, as the first is odd, while the second is even. \qed

3) Efficient Evaluation Without Polynomial Division:

Let us now consider another efficient method to perform the exhaustive evaluation. The current method is slightly less efficient than that of the previous subsection, but it avoids polynomial division and gcd calculations (Euclidean algorithm), which may be desirable in some applications. Letting again $(g_0, g_1)$ be the vector with the minimum $(1, w)$-weighted degree in the current Gröbner basis and writing

\[
\hat{\sigma}(X) := \mu(g_0h_1 + g_1h_2) = g_0(X^2)\hat{h}_1(X) + g_1(X^2)\hat{h}_2(X),
\]

it follows from Theorem 30 that in case of success, $\hat{\sigma}(X) = t(X^2)\sigma(X)$ for some non-zero $t(X)$ such that $\gcd(t(X^2), \sigma(X)) = 1$ and $t(X^2)$ has no roots outside $J^{-1} := \{\alpha_{i_1}, \ldots, \alpha_{i_r}\}$ (the terminology of Subsection V-C.1). Noting that $\hat{\sigma}'(X) = t(X^2)\sigma'(X)$ and recalling that $\sigma, \sigma'$ are coprime, it follows that the roots of $\hat{\sigma}'(X)$ are exactly those roots of $\hat{\sigma}(X)$ that are not roots of $\hat{\sigma}'(X)$. Also, since $t(X^2)$ has no roots outside $J^{-1}$, all the roots of $\hat{\sigma}$ outside $J^{-1}$ are also roots of $\sigma$. As

\[
\hat{\sigma}'(X) = g_0(X^2)\hat{h}_1'(X) + g_1(X^2)\hat{h}_2'(X),
\]

this leads to the following evaluation strategy:

- Before starting the fast Chase decoding algorithm, calculate and store

\[
\{\hat{h}_1(x)\}_{x \in \mathbb{F}_q^2}, \{\hat{h}_2(x)\}_{x \in \mathbb{F}_q^2},
\]

\[
\{\hat{h}_1'(x)\}_{x \in J^{-1}}, \{\hat{h}_2'(x)\}_{x \in J^{-1}}.
\]

- When traversing the decoding tree $T$, if the stopping criterion of Subsection V-C.1 holds:

1. Find the estimated error locations by performing the following actions for all $x \in \mathbb{F}_q^2$.

   - Calculate $a := g_0(x^{-2}), b := g_1(x^{-2})$.
   - Read the stored values $c := \hat{h}_1(x^{-1})$ and $d := \hat{h}_2(x^{-1})$ and calculate $\hat{\sigma}(x^{-1}) = ac + bd$.
– If $x^{-1} \in J^{-1}$,
  * Read the stored values $c' := \hat{h}_1(x^{-1})$ and $d' := \hat{h}_2(x^{-1})$ and calculate $\sigma'(x^{-1}) = ac' + bd'$.  
  * If $\sigma_x(x^{-1}) = 0$ and $\sigma'(x^{-1}) \neq 0$, adjoin $x$ to a set $E$ of error locations.
– Otherwise ($x^{-1} \notin J^{-1}$)
  * If $\sigma_x(x^{-1}) = 0$ adjoin $x$ to a set $E$ of error locations.

2) Check if the potential error vector with support $E$ has the same syndrome as $y$. If it does, adjoin it to the output error list of the decoder. This requires $t|E|$ additions in $\mathbb{F}_2^v$ (no multiplications in $\mathbb{F}_2^v$ are required).

With this method, we do not know how to avoid calculating the syndrome of the error vector for validation. Also, the evaluated polynomials $g_0, g_1$ here typically have a higher degree than the polynomials evaluated in the previous subsection. The degree is higher by $d_g := \deg(gcd(g_0, g_1))$, and the overall complexity increment of evaluation is therefore $O(d_g \cdot n)$ (if the FFT of $[15]$ is not used). If $d_g > 0$, this is typically higher than the complexity of $O(r^2)$ required in the previous subsection for finding $t(X)$ and division.\(^{15}\)

D. High-Level Description of the Decoding Process

Let us now wrap-up the entire decoding process with the fast Chase algorithm of the current paper.

1) Perform HD decoding in the following way:
   - Calculate the syndrome polynomial $S(X)$.
   - Calculate the modified syndrome $\tilde{S}(X) = \frac{1 + X^{2d}(X)}{X} \mod (X^l)$ using the recursion (11).
   - Find a Gröbner basis $\{h_1 = (h_{10}, h_{11}), h_2 = (h_{20}, h_{21})\}$ with respect to $<_{-1}$ for the module $N$ of Definition 8, where the leading monomial of $h_1$ is in the first coordinate and the leading monomial of $h_2$ in the second coordinate. As explained in [12], this can be practically done with any of the syndrome-based algorithms for HD decoding of RS codes. For completeness, we include in Appendix D the complete listing of one particular algorithm of Fitzpatrick for finding a Gröbner basis for $N$.  
   - Let $j$ be such
     \[ \mathrm{LM}_{<_{-1}}(h_j) = \min \{\mathrm{LM}_{<_{-1}}(h_1), \mathrm{LM}_{<_{-1}}(h_2)\}. \]
   - Take $\sigma := \mu(h_j) = h_{j1}(X^2) + X h_{j0}(X^2)$.
   - If $\deg(\sigma) \leq t$, find the roots of $\sigma$ in $\mathbb{F}_2^v$ by exhaustive substitution. If the number of roots equals $\deg(\sigma)$, declare success and output the corresponding error vector.

2) If HD decoding fails, use reliability information to identify the set $U$ of $\eta$ least reliable coordinates. Calculate $w := 2\deg(h_{21}) - t - 1/2$, store the basis $\{(1, 0), (0, 1)\}$ at the memory for depth 0, and perform fast Chase decoding by traversing the tree $T$, depth first:
   - When visiting the edge $(\beta', \beta)$ at depth $r-1$ and a vertex $\beta$ at depth $r$ with an additional 1 in coordinate $\alpha_i$:

$^{15}$Note that when [15] is not used for evaluation, it is reasonable to assume that $r = O(\log(n))$, for otherwise evaluation with [15] is more efficient.

– Perform Algorithm A, taking the inputs $g_1, g_2$ from the memory for depth $r - 1$. If the stopping criterion holds, perform efficient exhaustive substitution, in one of the methods described in Subsection V-C.2 and Subsection V-C.3. If the resulting error passes the verification criterion described in the respective subsections, adjoin the error to the list of output errors.
– Store the outputs $g_1^+, g_2^+$ in the memory for depth $r$.

E. Complexity Analysis

Next, we would like to evaluate the complexity of Algorithm A. To do this, we first have to bound the degrees of the updated polynomials.

\[ \text{Definition 34: For } r \in \{1, \ldots, r_{\text{max}}\} \text{ and } j \in \{1, 2\}, \text{ write } \begin{cases} g_j^{(r)} = (g_{j0}^{(r)}, g_{j1}^{(r)}) \text{ and } g_j^{+} = (g_{j0}^{+}, g_{j1}^{+}) \text{ for the respective polynomials in Algorithm A when adjoining error location } \alpha_i. \text{ By definition, } g_j^{(1)} = (1, 0) \text{ and } g_j^{(0)} = (0, 1) \text{ are the initial values used in the first call to the algorithm. We will use the convention } g_1^{(0)} := (1, 0) \text{ and } g_2^{(0)} := (0, 1). \end{cases} \]

\[ \text{Proposition 35: For all } r, \]

\[ \deg(g_{j0}^{(r)}) + \deg(g_{j1}^{(r)}) + \deg(g_{j0}^{+}) + \deg(g_{j1}^{+}) \leq 2r - 1, \quad (31) \]

where the sum on the left-hand side is taken only over non-zero monomials.

Proof: For simplicity, let us define the degree of monomials in $\mathbb{F}_2[X]^2$ in the obvious way, by setting $\deg(X^k, 0) := k$ and $\deg(0, X^k) := k$. In each application of Kötter’s iteration, there is at most one $j \in \{1, 2\}$ for which $\text{LM}(g_j^{(r)}) > \text{LM}(g_j^{+})$, namely $j = j^*$. Moreover, $\text{LM}(g_j^{+}) = X \text{LM}(g_j^{*})$, so that $\deg \text{LM}(g_j^{+}) = \deg \text{LM}(g_j^{*}) + 1$. Also, in the initialization we have $\deg \text{LM}(g_1^{(1)}) = \deg \text{LM}(g_2^{(1)}) = 0$. Hence, for all $r$,

\[ \deg \text{LM}(g_1^{+}) + \deg \text{LM}(g_2^{+}) \leq r. \quad (32) \]

Recall that our monomial ordering is $<_w$, and that in Kötter’s iteration, the leading monomial of $g_j^{+}$ contains the $(j - 1)$-th unit vector. Hence

\[ \deg \text{LM}(g_1^{+}) = \deg(g_{j0}^{(r)}) = \deg(g_{j0}^{+}) + w \quad (33) \]

and

\[ \deg \text{LM}(g_2^{+}) = \deg(g_{j1}^{(r)}) = \deg(g_{j1}^{+}) - w. \quad (34) \]

From (32) and the equality part of (33) and (34), we get

\[ \deg(g_{j0}^{(r)}) + \deg(g_{j1}^{(r)}) \leq r. \quad (35) \]

Suppose that both $g_{j0}^{(r)}$ and $g_{j1}^{(r)}$ are non-zero (note that $g_{j0}^{(r)}$ and $g_{j1}^{(r)}$ containing the respective leading monomials,
are never zero). Then from the inequality part of (33) and (34), we see that
\[ \deg(g_{11}^{(r)}) + \deg(g_{20}^{(r)}) < \deg(g_{t0}^{(r)}) + \deg(g_{21}^{(r)}) \leq r. \]
Summing two inequalities, we get (31).

Now, if \( g_{11}^{(r)} = g_{20}^{(r)} = 0 \), then the assertion readily follows from (35), and so it remains to consider only the case where exactly one of \( g_{11}^{(r)}, g_{20}^{(r)} \) is zero, say, \( g_{11}^{(r)} = 0 \) and \( g_{20}^{(r)} \neq 0 \).

We will use induction on \( r \) to prove that \( \deg(g_{20}^{(r)}) \leq r - 1 \).

The basis of the induction, on the root of the tree \( T \), is clear. Suppose that the assumption holds for \( r - 1 \), and recall that \( g_j^{(r-1)} = \deg(f_i) \) for \( j = 1, 2 \). If \( \Delta_2 = 0 \) in iteration \( r \), then there is nothing to prove. Suppose, therefore, that \( \Delta_2 \neq 0 \). If \( j^* = 2 \), then the assertion follows immediately from the update rule for \( j^* \) and the induction hypothesis. Otherwise, since \( \deg(g_{20}^{(r)}) \leq r - 1 \) by (35), the assertion follows by the update rule for \( j \neq j^* \) and the induction hypothesis.

Now, on an edge connecting depth \( r - 1 \) to depth \( r \) in the decoding tree \( T \), we have four polynomials \( g_{ij}^{(r)} \), whose sum of degrees is at most \( 2(2r - 1) - 1 = 2r - 3 \), by Proposition 35. For each one of these polynomials, we have to perform evaluation once, and multiply by a scalar once. We also have to calculate \( \Delta_j^{(r)}/\Delta_j^{(r-1)} \), at the cost of a single multiplication, assuming the inverse is calculated by a table. Finally, there are 2 multiplications by the pre-computed \( h_2(\alpha_i^{-1})/h_1(\alpha_i^{-1}) \). In general, for a polynomial \( f \), evaluation takes \( \deg(f) \) multiplications (using Horner’s method), and multiplying by a scalar takes \( \deg(f) + 1 \) multiplications. Hence, the overall number of multiplications for performing Algorithm \( A \) on an edge between depth \( r - 1 \) and \( r \) is \( 3 + 2(2r - 3) + 4 = 4r + 1 \). In comparison, the corresponding complexity for [30, Alg. C] is \( 20r + 3 \) (this is \( M_C^{\text{max}} \)’s from [30]).

Hence, the gain from moving to the binary alphabet is by a factor of about 5. We note that having a single Kötter iteration per edge affects the complexity twice, both in halving the degrees of the maintained polynomials, and in halving the number of evaluations and multiplications in each stage. The reason that the gain in comparison to the \( q \)-ary case is by a factor of 5 (instead of 4), is that the so-called derivate step of [30, Alg. C], which does not have an equivalent in Algorithm \( A \), is slightly more complicated than the root step.

When \( r_{\text{max}} = \eta \), if the entire tree \( T \) is traversed, the total number of multiplications for all edge updates is at most
\[
M_{\text{Alg. } A}^{\text{tree}} := \sum_{r=1}^{\eta} (4r + 1) \binom{\eta}{r}
= 4 \sum_{r=1}^{\eta} \eta \binom{\eta - 1}{r - 1} + 2\eta - 1 = \eta 2^{\eta + 1} + 2\eta - 1
\]
(\text{using } r \binom{n}{r} = \eta \binom{\eta - 1}{r - 1}).

A complexity analysis of Wu’s algorithm for binary BCH codes [39, Alg. 5] is missing from [39]. While a precise estimation of the complexity of Wu’s algorithm is outside the scope of the current paper, we note that it is initialized by two polynomials whose sum of degrees is \( 2t + 1 \), and on each edge, typically the sum of degrees is increased by 2. Hence, it seems reasonable to replace the total degree bound of Proposition 35 by about \( 2(t + 1) + 1 \) for Wu’s algorithm.

The polynomials participating on an edge connecting a vertex at depth \( r - 1 \) with an edge at depth \( r \) therefore have a sum of degrees of \( 2(t + r - 1) + 1 = 2(t + r) - 1 \). Hence, the evaluation part on such an edge requires \( 2(t + r) - 1 \) multiplications, while the multiplication of polynomials by constants takes \( 2(t + r) + 1 \) multiplications (adding 2 to account, as before, for 2 free coefficients). An additional single multiplication is required to calculate a ratio of two scalars, as above. The overall number of multiplication for such an edge is therefore \( 4(t + r) + 1 \).

When \( r_{\text{max}} = \eta \), if the entire tree \( T \) is traversed, the total number of multiplications for all edge updates is at most
\[
M_{\text{Wu}} := \sum_{t=1}^{\eta} \binom{\eta}{t} + 2t + 1 = 2^\eta + 1(\eta + 2t + 1/2) - 4t - 1.
\]
It follows that the algorithm of the current paper reduces the complexity of traversing the decoding tree by a factor of about \( \eta + 2t/\eta = 1 + 2t/\eta \) when \( r_{\text{max}} = \eta \) and the entire tree is traversed. The complexity reduction is even higher when \( r_{\text{max}} < \eta \).

Regarding [20], we note that from the discussion on p. 2004 of the precursor [19], it follows that the complexity of polynomial updates for the fast Chase decoding of [20] is in \( O(2^{n+1}t) \). While it is not entirely clear what is the precise constant involved, it seems that the overall complexity is higher than that of [39] (see [45]).

Let us now consider the cost of unnecessary polynomial evaluations, focusing only the method of Subsection V-C.2. First, it can be verified from (35) that, using the terminology of Subsection V-C.2, \( \deg(g_0) + \deg(g_1) \leq r - 1 \) on an edge connecting depth \( r \) to depth \( r + 1 \). Hence, on an edge connecting depth \( r - 1 \) to depth \( r \), we have \( \deg(g_0) + \deg(g_1) \leq r - 2 \).

The number of multiplications for the gcd calculation and the divisions from Subsection V-C.2 is therefore \( O(r^2) \) (see, e.g., [32, Thm. 17.3]). Evaluation of two polynomials whose sum of degrees is at most \( r - 2 \) (for \( r \geq 2 \)) can be done using \( n \cdot \min\{4 \log_2(n), r - 2\} \) multiplications, where the minimum is between point-by-point evaluation using Horner’s method and the FFT algorithm of [15]. In addition, there are \( 2n \) multiplications in calculations of the form \( ac + bd \).

As already mentioned, it seems reasonable to assume that the probability of falsely meeting the stopping condition of Subsection V-C.1 is about \( 1/q \). With this assumption, the mean contribution per-edge of unnecessary exhaustive evaluations is
\[
\frac{1}{q} O(r^2) + 2n + n \cdot \min\{4 \log_2(n), r - 2\},
\]
which is dominated by \( \hat{N}_{\text{eval}} := 2 + \min\{4 \log_2(n), r - 2\} \).

When the minimum is \( r - 2 \), this increases the overall mean number of multiplications by a factor of about \( 5/4 \) over the number \( 4r + 1 \) solely for polynomial updates.

Comparing to Wu’s algorithm, recall first that the approximated probability of 1/q applies also to the stopping

\[ \text{To see this, we note that if } \begin{array}{ll} L & < \log_2(g_1), \end{array} \begin{array}{ll} L & < \log_2(g_2), \end{array} \text{ then } \begin{array}{ll} \deg(g_1) + w \neq \deg(g_10) \leq \deg(g_2) + w \end{array} \text{ (using an obvious notation), so that, } \begin{array}{ll} \deg(g_20) + \deg(g_1) < \deg(g_10) + \deg(g_21) \leq r, \end{array} \text{ where the last inequality is from (35). Similarly, if } \begin{array}{ll} \log_2(g_2) < \log_2(g_1), \end{array} \begin{array}{ll} \log_2(g_20) \leq \deg(g_21) + w \neq \deg(g_10), \end{array} \text{ so that } \begin{array}{ll} \deg(g_20) + \deg(g_21) < r. \end{array} \]
condition of [39]. Hence, arguing as above, the mean contribution per-edge of unnecessary exhaustive evaluations in [39] is about \( \min\{2 \log_2(n), t + r\} \) (note that for [39], only a single polynomial has to be evaluated). Again, when the minimum is \( t + r \), this increases the overall mean number of multiplications by a factor of about 5/4 over polynomial evaluation.

Finally, we note that the complexity of the pre-computations in part 2 of Remark 32 is in \( O(n d) \), and hence negligible. However, for the overall complexity comparison ahead, it will be useful to calculate this complexity. First, note that by (28), (29), we have \( \deg(h_1) + \deg(h_2) = 1 + 2(\deg(h_{10}) + \deg(h_{21})) = 1 + 2t \), where we have used Proposition 12 for the last equation. Hence, evaluation of \( \hat{h}_1, \hat{h}_2 \) on \( \eta \) elements requires a total of \((1 + 2t)\eta\) multiplications. Accounting for additional \( \eta \) multiplications in the calculation of the quotients \( \frac{h_2(x_{i_j}^{-1})}{h_1(x_{i_j})} \) (where again we assume a table for calculating inverses) requires \( \eta \) additional multiplications. Hence, the total number of multiplications in this stage is \( M_{\text{pre-sub.}} := (2 + 2t)\eta \). Note that squarings amount to cyclic shifts if a normal basis is used, and hence do not increase the complexity.

So far, we have only considered the complexity of traversing the decoding tree \( T \). For a complete complexity comparison between Algorithm A of the current paper and the algorithm of Wu [39, Alg. 5], the preceding operations should also be considered.

For Algorithm A of the current paper, the preceding operations are comprised of calculating the syndrome polynomial from the received word, calculating the modified syndrome from the syndrome polynomial, finding a Gröbner basis for the module \( N \), and computing the syndrome polynomial from the received word, and performing the BM algorithm.

The calculation of the syndrome polynomial from the received word (common to [39] and the current paper) does not involve any finite-field multiplications; only multiplications of bits by constants (i.e., selection of constants) and finite-field additions are involved. There are \( nt \) multiplications of bits by constants, and \((n - 1)t\) finite-field additions for calculating \( S_j \) for odd \( j \) (the values for even \( j \) can then be calculated by about \( t \) finite-field squarings). Although the current complexity analysis is focused on multiplications, it is not reasonable to neglect this part, as it involves a large number of additions. We will roughly approximate the equivalent number of finite-field multiplications for this stage as \( M_{\text{syn}} := nt/s \).

The number of multiplications for calculating the modified syndrome from the syndrome polynomial is \( M_{\text{mod. syn.}} := t(t - 1)/2 \) (see the discussion after (11)). For finding the Gröbner basis for \( N \), it is possible to first use Fitzpatrick’s algorithm [13], and then to use the modified key equation twice to move from two polynomials to two pairs of polynomials, as explained in Appendix D. The application of Fitzpatrick’s algorithm (as appearing in Appendix D) requires at most \( t^2/2 \) multiplications. Note that this complexity bound was obtained by substituting \( t/2 \) for \( t \) in the bound of [13, Coro. 3.7], as here “\( n \)” of [13] equals \( t \) instead of \( 2t \), where we assume that \( t \) is even for simplicity. The two following multiplications by the modified syndrome modulo \( X^t \) require a total of \( 2 \cdot t(t + 1)/2 = t^2 + t \), where we have used the fact that multiplying two polynomials modulo \( X^t \) requires at most \( t(t + 1)/2 \) multiplications. Hence, the overall complexity of obtaining the Gröbner basis is \( M_{\text{GB}}(N) := 3t^2/2 + t \). All-in-all, the number of multiplications in all steps of the algorithm of the current paper is typically not higher than

\[
M_{\text{total}} := M_{\text{synd}} + M_{\text{mod. syn.}} + M_{\text{GB}}(N) + M_{\text{pre-sub.}} + \frac{5}{4} M_{\text{tree}}
\]

\[
= \frac{nt}{s} + 2t^2 + \frac{t}{2} + (2 + 2t)\eta + \frac{5}{4}(\eta^2 + 1 + 2^n - 1).
\]

For making a similar calculation for [39, Alg. 5], we first note that the number of multiplications for the binary BM algorithm is at most \( M_{\text{BW}} := t^2 \). Hence the total number of multiplications in all steps of the algorithm of [39] is typically not higher than

\[
M_{\text{BW}} := M_{\text{synd}} + M_{\text{BM}} + \frac{5}{4} M_{\text{tree}}
\]

\[
= \frac{nt}{s} + t^2 + \frac{5}{4}(2^n + 1)(\eta + 2t + 1/2) - 4t - 1).
\]

In Table I, we compare the values of \( M_{\text{total}} \) and \( M_{\text{BW}} \) for several values of the parameters \( n, t, k, s, \eta \) (where \( k \) is the dimension). We note that for simplicity, the stated dimension \( k \) is given by the lower bound \( n - st \), and in some cases the actual dimension may be in fact higher. We also note that the parameters of the code in the last two examples are taken from the last row of Table IV of [40].

| \( n \) | \( t \) | \( k \) | \( s \) | \( \eta \) | \( M_{\text{total}} \) | \( M_{\text{BW}} \) |
|-------|-------|-------|-------|-------|----------------|----------------|
| 4200  | 8     | 4096  | 13    | 4     | 3.0 \cdot 10^8 | 3.4 \cdot 10^8  |
| 4200  | 8     | 4096  | 13    | 6     | 3.9 \cdot 10^8 | 6.2 \cdot 10^8  |
| 511   | 10    | 421   | 9     | 5     | 1.3 \cdot 10^9 | 2.6 \cdot 10^9  |
| 511   | 10    | 421   | 9     | 7     | 3.3 \cdot 10^9 | 9.4 \cdot 10^9  |
| 17344 | 64    | 16384 | 15    | 10    | 1.1 \cdot 10^9 | 4.3 \cdot 10^9  |
| 17344 | 64    | 16384 | 15    | 15    | 1.4 \cdot 10^9 | 11.8 \cdot 10^9 |

VI. Conclusion

Building on the SD Wu list decoding algorithm, we have presented a new syndrome-based fast Chase decoding algorithm for binary BCH codes. The new algorithm requires only a single Kötter iteration per edge of the decoding tree, as opposed to two iterations in the corresponding algorithm for RS codes [30]. Also, the algorithm has a

Recall that the factor 5/4 comes from the average contribution of unnecessary polynomial evaluations.

This number comes from summing only over the odd terms in the proof of [13, Lemma 2.1], where for part (ii), we substitute \( k - 1 \) instead of \( k + 1 \) in part (i).
lower complexity than that of [39], as it updates low-degree coefficient polynomials. As in [30], and as opposed to [39], the current fast Chase algorithm can work also if the total number of errors is beyond \( d - 1 \).

We have also established an isomorphism between two solution modules for decoding binary BCH codes. This isomorphism can be used to benefit from the binary alphabet for reducing the complexity of HD bounded-distance decoding in a systematic way, practically for all syndrome-based HD decoding algorithms.

The new fast Chase algorithm is based on the idea that for a maximum list size of 1, \( \{0,1\} \)-multiplicity assignment in the SD Wu list decoding algorithm is equivalent to flipping bits in locations with a non-zero multiplicity (Theorem 30). This allows to accumulate flipfings through Kötter’s iteration, and leads to the fast Chase decoding algorithm.

We note that the BM algorithm itself can be used to find the Gröbner basis \( \{h_1, h_2\} \) for the module \( V \). The idea is that two vectors are extracted from the output of the algorithm, and then at most a single leading monomial cancellation is required for achieving the required Gröbner basis. In fact, the calculation of \( h_1, h_2 \) can be done without calculating the modified syndrome at all, using a slightly augmented BM algorithm: In addition to tracking the connection polynomial before the last length change, one tracks one additional past version of the connection polynomial. Since the proofs are rather long and this is outside the main scope of the current paper, we omit further details. For a precise listing of the relevant algorithms, see [31].

It seems plausible that the fast Chase decoding algorithm of the current paper can also be derived through a suitable minimization problem over a module, as done in [30] for RS codes. It would be interesting to formalize and prove this. It would be even more interesting to find a way to obtain the results of [30] through the SD Wu list decoding algorithm for RS codes. Beyond the interest in finding new connections between decoding algorithms, there are also additional advantages in using the Wu list decoding as a means for deriving a fast Chase algorithm. For example, the case of indirect hits (Remark 31) is handled trivially, as opposed to for deriving a fast Chase algorithm. For example, the case of connections between decoding algorithms, there are also possibilities in \( \{0,1\} \)-multiplicity assignment in the SD Wu list decoding algorithm is equivalent to flipping bits in locations with a non-zero multiplicity (Theorem 30). This allows to accumulate flipfings through Kötter’s iteration, and leads to the fast Chase decoding algorithm.

Appendix A

Proof of Part 4 of Proposition 5

Proof: Writing \( f \) as a sum of the form \( \sum_{i_1,i_2,i_3} b_{i_1,i_2,i_3} X_{i_1} Y_{i_2} Z_{i_3} \) with \( b_{i_1,i_2,i_3} := a_{i_1,i_2} \) and with \( b_{i_1,i_2,i_3} := 0 \) for \( i_3 \neq p - i_2 \), it follows from Proposition 2 that for all \( (j_1,j_2,j_3) \) and for all \( \alpha \in K \)

\[
f^{(j_1,j_2,j_3)}(x_0,\alpha y_0,\alpha z_0) = \sum_{i_1 \geq j_1, i_2 \geq j_2, i_3 = p - i_2, i_3 \neq j_3} b_{i_1,i_2,i_3} (\alpha y_0)^{i_2-j_2} (\alpha z_0)^{i_3-j_3} \rho^{i_1-j_1} x_0^{i_1-j_1}.
\]

Note that as \( \alpha \) is always raised to a non-negative power, the above indeed holds also for \( \alpha = 0 \). For this choice of \( \alpha \), we obtain

\[
f^{(j_1,j_2,j_3)}(x_0,y_0,z_0) = 0 \implies f^{(j_1,j_2,j_3)}(x_0,0,0) = 0,
\]

which is sufficient for proving the assertion.\footnote{It also follows that \( f^{(j_1,j_2,j_3)}(x_0,0,0) \) is non-zero only if \( j_3 = \rho - j_2 \) (this can also be verified directly).}

Appendix B

Proof of Proposition 22

Proof: If \( w_1', w_2', d \) are integers, then

\[
N_{1,w_1',w_2'}^d(d) \geq \sum_{j=0}^{\rho} (d - w_1'j - w_2'(\rho - j) + 1) = (\rho + 1)(d + 1 - \frac{\rho}{2}(w_1' + w_2')) \quad (36)
\]

Suppose now that \( w_2' \) is an integer, while \( w_1' \) is not an integer, say, \( w_1' = w_1' + 1/2 \) for some \( w_1' + 1/2 \in Z \). In this case, if \( d \) is an integer and \( j \) is odd, then

\[
|d - jw_1' - (\rho - j)w_2' + 1| = \left| d - jw_1' - \frac{j - 1}{2} - (\rho - j)w_2' - \frac{1}{2} \right| = d - jw_1' - \frac{j - 1}{2} - (\rho - j)w_2' = d - jw_1' - (\rho - j)w_2' + \frac{1}{2}. \quad (37)
\]

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
while if \( j \) is even, the floor function can be simply removed. Similarly, if \( d \) is not an integer and \( j \) is even, then (37) holds, while if \( j \) is odd, then the floor function can be removed. This means that when \( w'_1 \) is not an integer, we have to subtract \( 1/2 \) either from all odd \( j \) or all even \( j \) of the sum in (36). If \( \rho \) is odd, we therefore subtract exactly \((\rho + 1)/4\) from (36) in any possible case, while if \( \rho \) is even, we subtract at most \((\rho + 2)/4\). Hence,

\[
N^\rho_{1,w_1,w_2}(d) \geq \begin{cases}
(\rho + 1)(d + \frac{3}{4} - \frac{\rho}{2}(w'_1 + w'_2)) & \text{if } \rho \text{ is odd} \\
(\rho + 1)(d + \frac{3}{4} - \frac{\rho}{2}(w'_1 + w'_2)) - \frac{1}{4} & \text{if } \rho \text{ is even}
\end{cases}
\]

The case where only \( w'_2 \) is non-integer is handled in a similar way.

It follows that when exactly one of \( w'_1, w'_2 \) is not an integer, for odd \( \rho \), \( N^\rho_{1,w_1,w_2}(n_{eq}) \) is not above the minimum possible \( d \in \frac{1}{2}\mathbb{N} \) for which

\[
n_{eq} < (\rho + 1)(d + \frac{3}{4} - \frac{\rho}{2}(w'_1 + w'_2)),
\]

while for even \( \rho \), we have to replace \( n_{eq} \) by \( n_{eq} + 1/4 \) in (38). As (38) clearly holds if

\[
d > \frac{n_{eq} + 1}{\rho + 1} + \frac{\rho}{2}(w'_1 + w'_2) - \frac{1}{2},
\]

(18) follows. The proof for the case where both \( w'_1 \) and \( w'_2 \) are integers (using (36) without modifications) is similar. \( \square \)

**Appendix C**

**Using the Monomial Order \( \prec_w \) to Minimize the \((1, 1+w)\)-Weighted Degree**

In the last part of Remark 32, it was stated that the lower of the two weights in (20) can be used for both the even and odd cases. The following proposition makes this statement precise.

**Proposition 36:** For \( w \in \mathbb{Z} \) and for monomials \( m_1, m_2 \in K[X]^2 \) (where \( K \) is a field), suppose that \( m_1 <_{\prec_w} m_2 \) but \( m_1 >_{\prec_w} m_2 \). Then \( w_{deg_{\prec_w}}(m_1) = w_{deg_{\prec_w}}(m_2) \).

**Proof:** The assumption implies that \( m_1, m_2 \) contain distinct unit vectors. Hence, there are two cases to consider: (1) \( m_1 = (X^i, 0) \) and \( m_2 = (0, X^j) \), and (2) \( m_1 = (0, X^i) \) and \( m_2 = (X^j, 0) \), \( i, j \in \mathbb{N} \). In case (1), we have \( i \leq j + w + 1 \) and \( i > j + w \), which contradicts \( i = j + w \). As required. In case (2), we have \( i + w + 1 < j \) and \( j \leq i + w \), which is a contradiction. \( \square \)

**Appendix D**

**A Concrete Algorithm for Finding a Gröbner Basis for the Module \( N \) – Fitzpatrick’s Algorithm**

For completeness, we include here the listing of one particular algorithm for finding a Gröbner basis for the module \( N \) of Definition 8 with respect to \( \prec_{-1} \). Algorithm 4.7 of [12] tailored to the case \( r = -1 \) in the ordering \( \prec_r \). We also use Fitzpatrick’s idea to finding only the second coordinates of the elements in the Gröbner basis in order to reduce the complexity (the first coordinates are then found by an application of the modified key equation; see ahead for details). The below algorithm also coincides with Algorithm F of [13], whose complexity is thoroughly analyzed in [13].

In the following algorithm, for \( i \in \{0, 1\} \), we write \( i \) for the complement of \( \bar{i} \) (\( \bar{0} := 1, \bar{1} := 0 \)). Also, for two polynomials \( u(X), v(X) \in \mathbb{F}_2[X] \) and for \( k \in \mathbb{Z} \), we write \( [uv]_k \) for the coefficient of \( X^k \) in \( uv \), where by convention, \( k < 0 \), \( [uv]_k := 0 \).

**Algorithm F:** Fitzpatrick’s algorithm for finding a Gröbner basis for \( N \)

**Input**

- The modified syndrome, \( \hat{S} \)
- \( t \)

**Output**

Two polynomials \( b_0(X), b_1(X) \in \mathbb{F}_2[X] \) such that \( \{(\hat{S}b_0 \mod X^t, \hat{S}b_1 \mod X^t, b_0, b_1)\} \) is a Gröbner basis for \( N \) with respect to \( \prec_{-1} \).

**Initialization**

\( b_0 := 0, b_1 := 1, \alpha_0 := 1, i := 1, j := 1, d := 1 \)

**Algorithm**

- For \( k = 0, \ldots, t - 1 \)
  - \( - \alpha_j := [\hat{S}b_j]_k \)
    - If \( \alpha_i \neq 0 \)
      * \( b_i := b_i \mod \alpha_i b_i \)
      * \( b_i := Xb_i \)
      * \( j := \bar{i} \)
      * \( d := d - 1 \)
    - Else
      * \( b_i := Xb_i \)
      * \( j := \bar{i} \)
      * \( d := d + 1 \)

After applying Algorithm F, the following post-processing steps are required to finally obtain the required Gröbner basis \{\( h_1, h_2 \)\} for \( N \):

1. For \( i = 0, 1 \), calculate \( a_i(X) := \hat{S}b_i(X) \mod X^t \)
2. Write \( G := \{(a_0, b_0), (a_1, b_1)\} \). By construction, this is a Gröbner basis for \( N \) with respect to \( \prec_{-1} \). This implies that one of the two elements of \( G \) has its \( \prec_{-1} \)-leading monomial on the left, while the other has its \( \prec_{-1} \)-leading monomial on the right.\( ^{20} \) Take \( h_1 \) to be the element with leading monomial on the left, and \( h_2 \) to be the element with leading monomial on the right. Note that there is practically no computational cost to this stage, which is only comprised of some degree comparisons.

**Acknowledgment**

The authors would like to thank Avner Dor and Izhak Tamo for some very helpful discussions, also would like to thank Johan Rosenkilde for pointing them to [1], [2], [25], and [28], and also would like to thank the two anonymous reviewers for their helpful and constructive comments which considerably improved the presentation of the article.

\( ^{20} \)Recall that \( (X^t, 0), (0, X^t) \in N \).
