Reliable Architectures for Composite-Field-Oriented Constructions of McEliece Post-Quantum Cryptography on FPGA

Alvaro Cintas Canto, *Student Member, IEEE*, Mehran Mozaffari Kermani, *Senior Member, IEEE*, and Reza Azarderakhsh, *Member, IEEE*

***Abstract*—Code-based cryptography based on binary Goppa codes is a promising solution for thwarting attacks based on quantum computers. The McEliece cryptosystem is a code-based public-key cryptosystem which is believed to be resistant against quantum attacks. In fact, it is successfully advanced to the second round of the post-quantum cryptography standardization competition early 2019. Due to its very large key size, different variants of binary Goppa codes have been proposed. Nevertheless, research has shown that such codes can be thwarted through the injection of faults, causing erroneous outputs. In this work, we present countermeasures for the implementation of different composite field arithmetic units used in the McEliece cryptosystem. The proposed architectures use overhead-aware and tailored signatures. We apply these error detection signatures to the McEliece cryptosystem and perform field-programmable gate array (FPGA) implementations to show the feasibility of adopting the proposed schemes. We benchmark the overhead and performance degradation of the proposed approaches and show their suitability for constrained embedded systems.**

***Index Terms*—Composite fields, fault detection, McEliece.**

I. Introduction

Quantum computers are presumed to be able to break nearly all public-key encryption algorithms used today. There are five popular PQC algorithm classes: 1) code-based; 2) hash-based; 3) isogeny-based; 4) lattice-based; and 5) multivariate-quadratic-equation-based cryptosystems. Code-based cryptography, such as the McEliece and Niederreiter cryptosystems, differs from others by that its security relies on the hardness of decoding in a linear error-correcting code. Hash-based cryptography creates signature algorithms based on the security of a selected cryptographic hash function. The security of isogeny-based cryptography is based on the hard problem to find an isogeny between two given supersingular elliptic curves. Lattice-based cryptography is capable of creating a public-key cryptosystem based on lattices. Finally, multivariate-quadratic-equation-based cryptography is founded on asymmetric cryptography primitives using multivariate polynomials over a finite field. The McEliece cryptosystem is successfully advanced to the second round of the post-quantum cryptography standardization competition early 2019. Since the McEliece cryptosystem uses public-key encryption, it can serve in a wide variety of applications, such as digital signatures, authentication protocols, exchange of a secret key over an insecure channel, or even digital cash systems such as Bitcoin. Even though there are different alternatives to Goppa codes, such as LDPC and MDPC codes, Reed-Solomon codes, or convolutional codes, they are not as secure as the binary Goppa codes [1]. Since the key size of the McEliece scheme is very large, other slight variations of binary Goppa codes, such as quasi-cyclic and quasi-dyadic codes had also been explored [2], [3]. These alternant codes allow a smaller key size, but they are more sensitive to fault injection attacks [4]. Having the architectures of [5] as example, the McEliece cryptosystem uses Goppa codes in its different arithmetic units. The McEliece cryptosystem is based on linear error-correcting codes; however, it is vulnerable to fault injection attacks as it has been previously indicated [6].

Side-channel attacks, such as differential power analysis (DPA) for the McEliece cryptosystems have been studied in a number of previous works, e.g., [7] and [8]. Chen *et al.* [7] presented a successful DPA of a state-of-the-art McEliece implementation based on quasi-cyclic MDPC codes. Von Maurich and Güneysu [8] successfully demonstrate side-channel attacks of the McEliece cryptosystem implemented on constrained devices. Approaches on countering fault attacks, such as [9]–[12] have been the center of research attention for cryptography. In this work, for the first time, based on the underlying composite fields, error detection schemes are derived for different Goppa arithmetic units that the McEliece cryptosystem uses, e.g., evaluation, multiplication, squaring, division, square root, inversion, and greatest common division. These error detection schemes are based on signatures providing high error coverage. We note that the term signature here refers to appended bits used for error detection through error-detecting codes and not the typical signatures commonly used for proof of authenticity in cryptography. We also benchmark the overhead of the proposed schemes by implementing the arithmetic units used in the McEliece cryptosystem on field-programmable gate array (FPGA).

II. Preliminaries

The McEliece cryptosystem includes three operations: 1) key generation, which generates a pair of keys (public key and private key) needed to keep the message secret; 2) encryption, which creates the ciphertext using the public key; and 3) decryption, which allows to obtain the original message using the private key. The key generation in the McEliece cryptosystem uses the dimension of the code subspace m, the maximum number of errors that can be corrected t, the code length n, and code rank k. In this work, the security parameters used are $m = 13$, $t = 128$, and $n = 8192$ (k can be calculated by performing $k = n - mt$), which are one of the possible security parameters submitted to NIST in late 2017 [13]. However, the proposed approaches are oblivious of the sizes of these three parameters. The McEliece cryptosystem produces the pair of keys by first constructing a basic finite field $GF(2^m)$. Since $m = 13$, this field contains 8192 elements, i.e., $\alpha_0, \alpha_1, \ldots, \alpha_{8191}$, where each element is a vector of 13 bits. Next, a random monic irreducible polynomial $g(x) = x^t + g_{t-1}x^{t-1} + \cdots + g_1x + g_0$, with degree t is also generated, called Goppa polynomial. This monic irreducible polynomial

Manuscript received April 26, 2020; revised June 27, 2020; accepted August 24, 2020. Date of publication August 28, 2020; date of current version April 21, 2021. This work was supported by the U.S. National Science Foundation (NSF) under Award SaTC-1801488. This article was recommended by Associate Editor W. Zhang. *(Corresponding author: Mehran Mozaffari Kermani.)*

Alvaro Cintas Canto and Mehran Mozaffari Kermani are with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620 USA (e-mail: alvarocintas@usf.edu; mehran2@usf.edu).

Reza Azarderakhsh is with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: razarderakhsh@fau.edu).

Digital Object Identifier 10.1109/TCAD.2020.3019987

TABLE I
McEliece Operations and Corresponding Processes

Operation	Process
Goppa Division	Key Generation, Decryption
Goppa Multiplication/Addition	Key Generation, Encryption, Decryption
Goppa Squaring	Key Generation
Goppa Square Root	Decryption
Goppa GDC, Goppa Inversion, and Goppa Polynomial Decomposition	Key Generation, Decryption
Goppa Polynomial Evaluation	Key Generation, Decryption

conforms part of the private key, and all its coefficients are elements of the basic finite field. After producing the Goppa polynomial, a control matrix H is constructed by multiplying three auxiliary matrices based on the private key, denoted as X, Y, and Z. The control matrix is permutated by using a random permutation matrix called P. Then, it is expanded into a binary form H_2 over $GF(2)$, converted into the systematic form $\tilde{G}$, and transposed into G to obtain its public key.

To get the ciphertext z, a random n-bit error vector e is created. The error vector e has to contain a total of t bits having the value 1. Then, z is calculated by performing $z = pG \oplus e$, where p refers to plaintext. The decryption process is fairly more complex than the encryption process. First, an error locator polynomial $\sigma(x)$ is created, which will reveal all errors of the elements α_i. Then, the error vector is reconstructed to obtain the original plaintext. Since this article focuses mainly on the key generation, readers interested in other aspects of the McEliece cryptosystem can refer to [5].

III. Proposed Fault Detection Schemes

The McEliece cryptosystem is based on three different Galois fields: 1) the Goppa field GF($(2^m)^t$) used by the Goppa polynomial [i.e., GF($(2^{13})^{128}$) in this article]; 2) GF(2^{13}), which is the field polynomial (we use $p(x) = x^{13} + x^4 + x^3 + x + 1$ as described by [13]); and 3) the binary field GF(2). The McEliece cryptosystem uses (based on the underlying composite fields) different Goppa field arithmetic units to perform a number of its operations, e.g., evaluation, multiplication, squaring, division, square root, inversion, and greatest common division. These Goppa field operations work on polynomials of degree $t-1$ with coefficients from GF(2^m). In Table I, the McEliece operations and corresponding processes are shown.

A. Goppa Division

Polynomial division is required to find the greatest common divisor. It follows the long division method by first, inverting the highest coefficient of the polynomial divisor. This inverted coefficient is then multiplied by the highest-degree coefficient of the dividend to obtain the quotient. Next, the quotient coefficient is multiplied by all the coefficients of the polynomial divisor and the product is finally subtracted from the dividend polynomial using modulo-2 addition. Since each coefficient is in GF(2^{13}), signatures for inversion, multiplication, and XORing in the field GF(2^{13}) are needed. Reyhani and Hasan [14] performed error detection over binary extension fields by adding parities which can only detect an odd number of faults and therefore, to obtain higher error coverage, two other signatures are derived in this article. The first alternative, called two-part signature, divides the 13 bits into two blocks (from bit 0 to bit 6, and from bit 7 to bit 12), and the other alternative, called three-part signature, divides the bits into three blocks (from bit 0 to bit 4, from bit 5 to bit 9, and from bit 10 to bit 12).

1) Goppa Addition and Goppa Multiplication: Goppa addition and Goppa multiplication need a total of t GF(2^m) additions and t GF(2^m) multiplications, respectively. As it is shown in [14], the multiplication of any two elements A and B of GF(2^m) can be represented as $A \cdot B \bmod p(x) = \sum_{i=0}^{m-1} b_i \cdot ((A\alpha^i) \bmod p(x)) = \sum_{i=0}^{m-1} b_i \cdot X^{(i)}$, where the set of α^i's is the polynomial basis of element A, the set of b_i's is the B coefficients, $p(x)$ is the field polynomial, $X^{(i)} = \alpha \cdot X^{(i-1)} \bmod p(x)$, and $X^{(0)} = A$. To perform polynomial basis multiplications over binary extension fields, α, *sum*, and *pass-thru* modules are used. The *pass-thru* module multiplies a GF(2^m) element by a GF(2) element, the α module multiplies an element of GF(2^m) by α and it reduces the result modulo $p(x)$, and the *sum* module adds two elements in GF(2^m) using m two-input XOR gates. The latter one is used for addition of polynomials in GF(2^{13}).

In the *sum* module, the parity bits of the inputs A and B, and the predicted parities of the output P are divided into two or three blocks, depending on if two-part signature or three-part signature is used. If two-part signature is used, the parity bits of A and B are divided into p_{A1} and p_{A2}, and p_{B1} and p_{B2}, respectively. The addition of elements A and B in GF(2^m) produce the predicted parities $\hat{p}_{P1} = p_{A1} + p_{B1}$ and $\hat{p}_{P2} = p_{A2} + p_{B2}$. On the other hand, if three-part signature is used, the parity bits of A are divided into p_{A1}, p_{A2}, and p_{A3}, and p_{B1}, p_{B2}, and p_{B3}, obtaining the predicted parities $\hat{p}_{P1} = p_{A1} + p_{B1}$, $\hat{p}_{P2} = p_{A2} + p_{B2}$, and $\hat{p}_{P3} = p_{A3} + p_{B3}$.

In the *pass-thru* module, the parity bits of the input A and the predicted parities of the output P are also divided into two or three blocks. If two-part signature is used, the parity bits of A are multiplied by an element b of GF(2) to obtain the predicted parities $\hat{p}_{P1} = b \cdot p_{A1}$ and $\hat{p}_{P2} = b \cdot p_{A2}$. On the other hand, if three-part signature is used, the predicted parities of output P are $\hat{p}_{P1} = b \cdot p_{A1}$, $\hat{p}_{P2} = b \cdot p_{A2}$, and $\hat{p}_{P3} = b \cdot p_{A3}$.

Finally, for the α module, Theorems 1 and 2 are derived for two-part signature and three-part signature, respectively.

Theorem 1: Let $p_{A1} = \sum_{i=0}^{[(m-1)/2]} a_i$ be the first parity for bits of A, and $p_{A2} = \sum_{i=[(m-1)/2]+1}^{m-1} a_i$ be the last parity for bits of A, where m is assumed to be odd, used for the dimension of the code subspace, and $f_i \in GF(2)$ for $i = 0, 1, \ldots, m-1$. Then, the predicted parities $\hat{p}_{X1}$ and $\hat{p}_{X2}$, are $\hat{p}_{X1} = a_{m-1} + \sum_{i=1}^{[(m-1)/2]} (a_{i-1} + a_{m-1} \cdot f_i)$, $\hat{p}_{X2} = \sum_{i=[(m-1)/2]+1}^{m-1} (a_{i-1} + a_{m-1} \cdot f_i)$, and for $m = 13$, we get to the following concise formulations: $\hat{p}_{X1} = p_{A1} + a_6$ and $\hat{p}_{X2} = p_{A2} + a_6 + a_{12}$.

Proof: Using the below formulations to calculate the X coordinates [14]

$$x_i = \begin{cases} a_{i-1} + a_{m-1} \cdot f_i, & 1 \le i \le m-1 \\ a_{m-1}, & i = 0 \end{cases}$$

the predicted parity $\hat{p}_X$ can be split as $\hat{p}_X = a_{m-1} + \sum_{i=1}^{[(m-1)/2]} (a_{i-1} + a_{m-1} \cdot f_i) + \sum_{i=[(m-1)/2]+1}^{m-1} (a_{i-1} + a_{m-1} \cdot f_i) = \hat{p}_{X1} + \hat{p}_{X2}$. Since $p(x) = x^{13} + x^4 + x^3 + x + 1$, one obtains $f_{13} = f_4 = f_3 = f_1 = f_0 = 1$, achieving the concise formulations. This completes the proof. ■

Theorem 2: Let $p_{A1} = \sum_{i=0}^{\lfloor[(m-1)/3]\rfloor} a_i$ be the first parity for bits of A, $p_{A2} = \sum_{i=\lfloor[(m-1)/3]\rfloor+1}^{2(\lfloor[(m-1)/3]\rfloor)+1} a_i$ be the second parity for bits of A, and $p_{A3} = \sum_{i=2(\lfloor[(m-1)/3]\rfloor+1)}^{m-1} a_i$ be the last parity for bits of A. Then, the predicted parities of output X, referred to as $\hat{p}_{X1}$, $\hat{p}_{X2}$, and $\hat{p}_{X3}$, are $\hat{p}_{X1} = a_{m-1} + \sum_{i=1}^{\lfloor[(m-1)/3]\rfloor} (a_{i-1} + a_{m-1} \cdot f_i)$, $\hat{p}_{X2} = \sum_{i=\lfloor[(m-1)/3]\rfloor+1}^{2(\lfloor[(m-1)/3]\rfloor)+1} (a_{i-1} + a_{m-1} \cdot f_i)$, $\hat{p}_{X3} = \sum_{i=2(\lfloor[(m-1)/3]\rfloor+1)}^{m-1} (a_{i-1} + a_{m-1} \cdot f_i)$, and for the case study of $m = 13$, we get to the following concise formulations: $\hat{p}_{X1} = p_{A1} + a_4$, $\hat{p}_{X2} = p_{A2} + a_4 + a_9$, and $\hat{p}_{X3} = p_{A3} + a_9 + a_{12}$.

Proof: The predicted parity $\hat{p}_X$ can be split as $\hat{p}_X = a_{m-1} + \sum_{i=1}^{\lfloor[(m-1)/3]\rfloor} (a_{i-1} + a_{m-1} \cdot f_i) + \sum_{i=\lfloor[(m-1)/3]\rfloor+1}^{2(\lfloor[(m-1)/3]\rfloor)+1} (a_{i-1} + a_{m-1} \cdot$

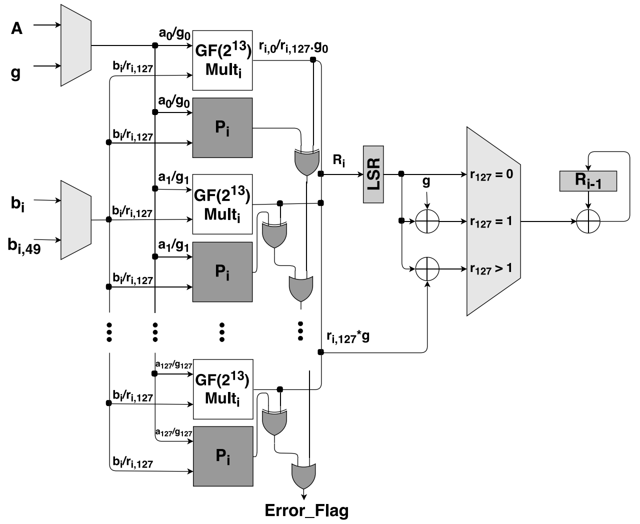

Fig. 1. Goppa multiplication unit with the proposed scheme.

$f_i) + \sum_{i=2(\lfloor[(m-1)/3]\rfloor+1)}^{m-1}(a_{i-1} + a_{m-1}\cdot f_i) = \hat{p}_{X1} + \hat{p}_{X2} + \hat{p}_{X3}$. Similar to the proof of previous theorem, the concise formulations are derived. This completes the proof. ■

Fig. 1 illustrates the architecture of the Goppa multiplication unit with our proposed fault detection schemes. It follows a shift-and-add approach with a chain of 127 GF(2^{13}) multiplications, and therefore, a total of 127 signatures are needed. These signatures are shown as P_i for $i = 0, 1, \ldots, 127$, and they can be implemented as normal, two-part, or three-part signatures (depicted by dark-grey, shaded blocks). a_i, b_i, and g_i are coefficients of the field GF(2^{13}). R_i is the intermediate result of the 127 multiplications, expressed as $R_i = \sum_{j=0}^{127} r_{i,j}x^j$, where $r_{i,j} = b_i a_j \bmod p(x)$. The multiplication in Fig. 1 is essentially a schoolbook multiplication. First, a_i is multiplied with b_i, obtaining R_i. If $i \neq 0$, R_i is shifted 13 bits using the left shift register (shown as LSR in Fig. 1). Then, R_i is reduced depending on the value of $r_{i,127}$: If $r_{i,127} = 0$, there is no reduction; if $r_{i,127} = 1$, R_i is XOR-ed with g; and if $r_{i,127} > 1$, R_i is XOR-ed with $r_{i,127}\cdot g$. Finally, the reduced result is XOR-ed with the previous result R_{i-1}. If a faulty output is detected in any of the 127 GF(2^{13}) multiplication modules, *Error_Flag* is asserted.

2) Goppa Inversion: To perform Goppa inversion, t GF(2^m) inversions are needed. The polynomial variant of Fermat's little theorem (FLT) is used. FLT achieves higher performance, allowing to calculate the inverse GF(2^{13}) using 12 squarings and 11 multiplications. Theorems 3 and 4 are derived for signatures of squaring.

Theorem 3: Let $p_{A1} = \sum_{i=0}^{[(m-1)/2]} a_i$ be the first parity for bits of A, and $p_{A2} = \sum_{i=[(m-1)/2]+1}^{m-1} a_i$ be the last parity for bits of A. Then, the predicted parities of output ν (instead of X to avoid confusion), referred to as $\hat{p}_{\nu1}$ and $\hat{p}_{\nu2}$, are $\hat{p}_{\nu1} = a_{m-1}\cdot f_{m-1} + a_{m-2} + a_{m-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_1 + \sum_{i=2}^{[(m-1)/2]}(a_{m-1}\cdot f_{i-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_i + a_{i-2})$, $\hat{p}_{\nu2} = \sum_{i=[(m-1)/2]+1}^{m-1}(a_{m-1}\cdot f_{i-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_i + a_{i-2})$, and for the case study of $m = 13$, we get to the following concise formulations: $\hat{p}_{\nu1} = p_{A1} + a_5 + a_6$ and $\hat{p}_{\nu2} = p_{A2} + a_5 + a_6 + a_{11} + a_{12}$.

Proof: The predicted parity $\hat{p}_\nu$ can be split as $\hat{p}_\nu = a_{m-1}\cdot f_{m-1} + a_{m-2} + a_{m-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_1 + \sum_{i=2}^{[(m-1)/2]}(a_{m-1}\cdot f_{i-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_i + a_{i-2}) + \sum_{i=[(m-1)/2]+1}^{m-1}(a_{m-1}\cdot f_{i-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_i + a_{i-2}) = \hat{p}_{\nu1} + \hat{p}_{\nu2}$. The concise formulations are derived similar to previous proofs. This completes the proof. ■

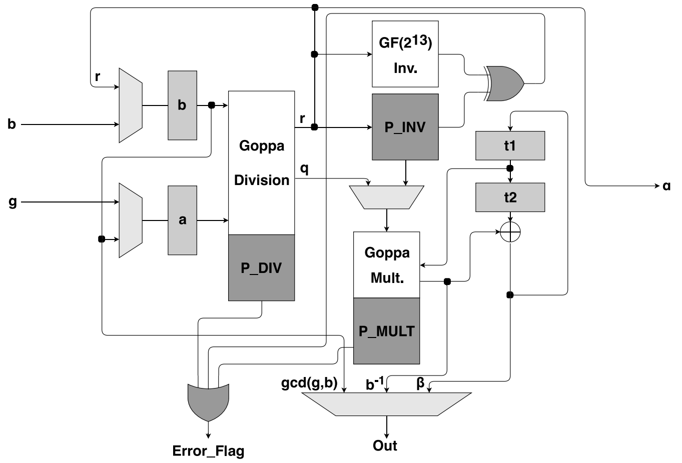

Fig. 2. GCDIPD with the proposed error detection scheme.

Theorem 4: Let $p_{A1} = \sum_{i=0}^{\lfloor[(m-1)/3]\rfloor} a_i$ be the first parity for bits of A, $p_{A2} = \sum_{i=\lfloor[(m-1)/3]\rfloor+1}^{2(\lfloor[(m-1)/3]\rfloor)+1} a_i$ be the second parity for bits of A, and $p_{A3} = \sum_{i=2(\lfloor[(m-1)/3]\rfloor+1)}^{m-1} a_i$ be the last parity for bits of A. Then, the predicted parities of output ν, referred to as $\hat{p}_{\nu1}$, $\hat{p}_{\nu2}$, and $\hat{p}_{\nu3}$, are $\hat{p}_{\nu1} = a_{m-1}\cdot f_{m-1} + a_{m-2} + a_{m-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_1 + \sum_{i=2}^{\lfloor[(m-1)/3]\rfloor}(a_{m-1}\cdot f_{i-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_i + a_{i-2})$, $\hat{p}_{\nu2} = \sum_{i=\lfloor[(m-1)/3]\rfloor+1}^{2(\lfloor[(m-1)/3]\rfloor)+1}(a_{m-1}\cdot f_{i-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_i + a_{i-2})$, $\hat{p}_{\nu3} = \sum_{i=2(\lfloor[(m-1)/3]\rfloor+1)}^{m-1}(a_{m-1}\cdot f_{i-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_i + a_{i-2})$, and for the case study of $m = 13$, we get to the following concise formulations: $\hat{p}_{\nu1} = p_{A1} + a_3 + a_4 + a_{12}$, $\hat{p}_{\nu2} = p_{A2} + a_8 + a_9 + a_{12}$, and $\hat{p}_{\nu3} = p_{A3} + a_{11} + a_{12}$.

Proof: The predicted parity $\hat{p}_\nu$ can be split as $\hat{p}_\nu = a_{m-1}\cdot f_{m-1} + a_{m-2} + a_{m-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_1 + \sum_{i=2}^{\lfloor[(m-1)/3]\rfloor}(a_{m-1}\cdot f_{i-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_i + a_{i-2})\sum_{i=\lfloor[(m-1)/3]\rfloor+1}^{2(\lfloor[(m-1)/3]\rfloor)+1}(a_{m-1}\cdot f_{i-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_i + a_{i-2})\sum_{i=2(\lfloor[(m-1)/3]\rfloor+1)}^{m-1}(a_{m-1}\cdot f_{i-1} + (a_{m-1}\cdot f_{m-1} + a_{m-2})\cdot f_i + a_{i-2}) = \hat{p}_{\nu1} + \hat{p}_{\nu2} + \hat{p}_{\nu3}$. The concise formulations can be also derived and this completes the proof. ■

B. Goppa Square Root

Square root is needed in the decryption process. It is a rather time-consuming task because first, it multiplies the input polynomial by a matrix Q^{-1} [a polynomial of order t and with coefficients of GF(2^m) where its rows are formed by the square of a monomial modulo the Goppa polynomial], and then, it does the square root of each coefficient GF(2^m) obtained. In total, 16 384 finite field multiplications and 128 square roots are needed for the case study of $m = 13$. Since all the elements are in the field GF(2^{13}), 12 GF(2^{13}) squarings are needed to perform the square root of each element.

C. Goppa Greatest Common Division, Goppa Inversion, and Goppa Polynomial Decomposition

Goppa greatest common division, Goppa inversion, and Goppa polynomial decomposition (GCDIPD) unit performs three different operations in the Goppa field GF($(2^m)^t$). It finds the greatest common divisor of two polynomials of order t, it inverts the syndrome, and it does polynomial decomposition. To perform such operations, Goppa division, Goppa multiplication, and GF(2^{13}) inversion are needed. We show the architecture of the proposed fault detection constructions for this unit in Fig. 2, where r stands for remainder, q stands for quotient, α is the intermediate remainder, and $\beta = (b(x) \bmod a(x))^{-1} = t(x)$. Goppa division and Goppa multiplication have their

TABLE II
OVERHEADS OF THE PROPOSED ERROR DETECTION SCHEMES FOR THE GPE UNIT

Architecture	Area (occupied slices)	Delay (ns)	Power (mW) @50 MHz	Throughput (Gbps)	Error Coverage Percentage	Xilinx FPGA family and device
GPE	1370	4.205	0.205	3.09	Not Applicable	Kintex-7 (xc7k70tfbv676-1)
GPE with Normal Sign.	1447 (5.62%)	4.494 (6.87%)	0.213 (3.90%)	3.12 (0.97%)	$100 \cdot (1-(\frac{1}{2})^{6 \cdot 10^3})\%$	
GPE with Two-Part Sign.	1484 (8.32%)	4.415 (4.99%)	0.213 (3.90%)	3.17 (2.59%)	$100 \cdot (1-(\frac{1}{2})^{1.2 \cdot 10^4})\%$	
GPE with Three-Part Sign.	1487 (8.54%)	4.402 (4.68%)	0.213 (3.90%)	3.18 (2.91%)	$100 \cdot (1-(\frac{1}{2})^{1.8 \cdot 10^4})\%$	
GPE	1339	5.386	0.219	2.41	Not Applicable	Spartan-7 (xc7s100fgga676-1)
GPE with Normal Sign.	1470 (9.78%)	5.461 (1.39%)	0.225 (2.74%)	2.56 (6.22%)	$100 \cdot (1-(\frac{1}{2})^{6 \cdot 10^3})\%$	
GPE with Two-Part Sign.	1491 (11.35%)	5.431 (0.84%)	0.225 (2.74%)	2.58 (7.05%)	$100 \cdot (1-(\frac{1}{2})^{1.2 \cdot 10^4})\%$	
GPE with Three-Part Sign.	1467 (9.57%)	5.440 (1.00%)	0.225 (2.74%)	2.57 (6.64%)	$100 \cdot (1-(\frac{1}{2})^{1.8 \cdot 10^4})\%$	

own signatures inside their blocks denoted as P_DIV and P_MULT, respectively, while the signatures for inversion block are separated in Fig. 2, since it performs GF(2^{13}) inversion, instead of Goppa inversion. First, the polynomials $t_1(x)$ and $t_2(x)$ are set to 1 and 0, respectively. While $b(x) \neq 0 : g(x)$ is divided over $b(x)$ to obtain $q(x)$; $g(x)$ is mod with $b(x)$ to obtain $r(x)$; $t(x)$ is obtained by performing $t_2(x)$-$q(x)t_1(x)$; and finally, $g(x)$, $b(x)$, $t_2(x)$, and $t_1(x)$ are set to $b(x)$, $r(x)$, $t_1(x)$, and $t(x)$, respectively. If the degree of $r(x)$ is less than 25 and the mode is *Patterson* (which reconstructs the error vector needed in the process of decryption), $\alpha(x)$ and $\beta(x)$ are set to $r(x)$ and $t(x)$, respectively. If the mode is *gcd*, $gcd(g(x), b(x))$ is returned. Moreover, if the mode is *inverse*, $r^{-1}(x)$ is multiplied with $t_1(x)$ to obtain $b^{-1}(x)$. The *Error_Flag* becomes high for faults in any of the polynomial operations.

D. Goppa Polynomial Evaluation

Goppa polynomial evaluation (GPE) unit is utilized in the key generation and decryption processes. Adopting the Horner scheme, it only needs to use a GF(2^{13}) multiplier because it allows removing high-degree polynomial multiplications, e.g., the element $f_1 + f_2\alpha_2 + f_3\alpha_2^2$ is written as $f_1 + (f_2 + f_3\alpha_2)\alpha_2$.

IV. ERROR COVERAGE AND FPGA IMPLEMENTATIONS

Since the GPE unit uses less signatures than most of the other units, we perform an analysis to show the efficiency of our proposed error detection schemes even for the smallest units of the McEliece cryptosystem. The GPE unit uses the Horner scheme over the Goppa polynomial, which has order t and coefficients of GF(2^m). To perform the GPE, a total of 128 GF(2^{13}) finite field multiplications and 128 GF(2^{13}) finite field additions are required. Each finite field multiplication and addition requires α, *sum*, and *pass-thru* modules. More precisely, a total of 12 α, 12 *sum*, and 13 *pass-thru* modules are needed for each finite field multiplication, and a total of 12 *sum* modules are required for each finite field addition. Moreover, the total number of signatures needed in the GPE unit is $128_{\text{mult.}} \cdot (12_\alpha + 12_{\text{sum}} + 13_{\text{pass-thru}}) + 128_{\text{add.}} \cdot (12_{\text{sum}})$ or a total of close to $6 \cdot 10^3$ signatures for the normal signature scheme. The percentage of error detection is calculated by applying the formula $100 \cdot (1 - (1/2)^{\#\text{sign}})\%$, where #sign is as the total number of signatures. Therefore, the percentage error coverage by this unit is approximately $100 \cdot (1 - (1/2)^{6 \cdot 10^3})\%$, which is close to 100%. In the two-part signature scheme, for every module, there are two error flags instead of one (as it is with normal signatures). Moreover, two-part signature scheme uses $2_{\text{error-flags}} \cdot (128_{\text{mult.}} \cdot (12_\alpha + 12_{\text{sum}} + 13_{\text{pass-thru}}) + 128_{\text{add.}} \cdot (12_{\text{sum}}))$ or a total of close to $1.2 \cdot 10^4$ signatures, which makes an error percentage of approximately $100 \cdot (1 - (1/2)^{1.2 \cdot 10^4})\%$. Finally, in the three-part signature scheme, for every module, there are three error flags needing $3_{\text{error-flags}} \cdot (128_{\text{mult.}} \cdot (12_\alpha + 12_{\text{sum}} + 13_{\text{pass-thru}}) + 128_{\text{add.}} \cdot (12_{\text{sum}}))$ or a total of close to $1.8 \cdot 10^4$ signatures with error coverage of $100 \cdot (1 - (1/2)^{1.8 \cdot 10^4})\%$.

We have implemented our error detection schemes benchmarked on two different Xilinx FPGA families. We note that the target platform does not necessarily affect the results because the chosen FPGAs belong to the same series (Xilinx series 7), being very similar from a technological point of view. As we are not using specific FPGA-related sub-blocks, such as large multipliers or inner multiplexers, the choice for hardware platform does not directly affect our derived overhead. The design entry is Verilog. We have implemented the proposed error schemes in the GPE unit. In Table II, the overheads in terms of area (occupied slices), delay, power (at the frequency of 50 MHz), and throughput of the different signatures on the GPE unit are presented. The overheads obtained are very acceptable, especially since the error detection coverage is close to 100%. There are several hardware implementations of the McEliece cryptosystem, e.g., [15] and [16]. López-García and Cantó-Navarro [15] produced a hardware/software implementation which is able to decipher 8192 bit-length in 47.39 ms. Bu *et al.* [16] proposed a new variant of McEliece cryptosystem and its encryption-decryption co-processor based on the generalized nonbinary orthogonal Latin square code (OLSC). The error detection schemes proposed in this work are suitable for any cryptosystem that uses any of the mentioned Goppa modules, such as the works in [15] and [16]. There has not been any prior work done on this type of error detection methods for the McEliece cryptosystem to the best of our knowledge. For qualitative comparison to verify that the overheads incurred are acceptable, let us go over some case studies. The work in [17] presented signature-based fault diagnosis for cryptographic block ciphers LED and HIGHT, obtaining a combined area and delay overhead of 21.9% and 31.9% for LED and HIGHT, respectively. Additionally, Mozaffari Kermani *et al.* [9] proposed efficient error detection architectures of hash-counter-hash tweakable enciphering schemes, obtaining a combined area and throughput overhead of less than 13.5%. The proposed schemes in this article have combined area and delay overheads of less than 12% (worst-case scenario). Such prior works on classical cryptography verify that the proposed error detection architectures obtain acceptable overhead.

V. CONCLUSION

In this work, we have derived error detection schemes for the Goppa arithmetic units that the McEliece cryptosystem uses (based on the underlying composite fields). We also discussed that the proposed error detection schemes are applicable to the main functions of public-key cryptosystems which use composite fields as underlying arithmetic constructions as well. To show the efficiency of the presented schemes we implemented the GPE unit on FPGA and showed the different overheads with respect to the GPE unit with no signatures and the GPE unit with normal, two-part, and three-part signatures. Results prove that a very high error coverage of close to 100% is obtained [it is approximately at least $100 \cdot (1 - (1/2)^{6 \cdot 10^3})\%$] at the cost of low and viable overheads.

References

[1] D. J. Bernstein, T. Lange, and C. Peters, "Attacking and defending the McEliece cryptosystem," in *Proc. Int. Workshop Post Quantum Cryptogr. (PQC)*, 2008, pp. 31–46.

[2] R. Misoczki and P. Barreto, "Compact McEliece keys from Goppa codes," in *Proc. Int. Workshop Sel. Areas Cryptogr.*, 2009, pp. 376–392.

[3] P. Barreto, R. Lindner, and R. Misoczki, "Monoidic codes in cryptography," in *Proc. Int. Workshop Post Quantum Cryptogr.*, 2011, pp. 179–199.

[4] V. G. Umaña and G. Leander, "Practical key recovery attacks on two McEliece variants," IACR Cryptol. ePrint Archive, Lyon, France, Rep. 2009/509, 2009.

[5] A. Shoufan, T. Wink, S. Huss, and E. Kohnert, "A novel cryptoprocessor architecture for the McEliece public-key cryptosystem," *IEEE Trans. Comput.*, vol. 59, no. 11, pp. 1533–1546, Nov. 2010.

[6] F. Strenzke, E. Tews, H. G. Molter, R. Overbeck, and A. Shoufan, "Side channels in the McEliece PKC," in *Proc. Int. Workshop Post Quantum Cryptogr. (PQC)*, 2008, pp. 216–229.

[7] C. Chen, T. Eisenbarth, I. Von Maurich, and R. Steinwandt, "Differential power analysis of a McEliece cryptosystem," in *Proc. Conf. Appl. Cryptogr. Netw. Security*, 2015, pp. 538–556.

[8] I. Von Maurich and T. Güneysu, "Towards side-channel resistant implementations of QC-MDPC McEliece encryption on constrained devices," in *Proc. Conf. Appl. Cryptogr. Netw. Security*, 2014, pp. 266–282.

[9] M. M. Kermani, R. Azarderakhsh, A. Sarker, and A. Jalali, "Efficient and reliable error detection architectures of hash-counter-hash tweakable enciphering schemes," *ACM Trans. Embedded Comput. Syst.*, vol. 17, no. 2, pp. 54:1–54:19, May 2018.

[10] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "Security analysis of concurrent error detection against differential fault analysis," *J. Cryptogr. Eng.*, vol. 5, no. 3, pp. 153–169, 2015.

[11] M. Mozaffari Kermani, R. Azarderakhsh, and A. Aghaie, "Fault detection architectures for post-quantum cryptographic stateless hash-based secure signatures benchmarked on ASIC," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 2, pp. 1–19, 2019.

[12] M. Mozaffari Kermani and A. Reyhani-Masoleh, "A high-performance fault diagnosis approach for the AES SubBytes utilizing mixed bases," in *Proc. IEEE Workshop Fault Diagnosis Tolerance Cryptogr. (FDTC)*, Nara, Japan, Sep. 2011, pp. 80–87.

[13] D. J. Bernstein *et al.* (Nov. 2017). *Classic McEliece: Conservative Code-Based Cryptography*. [Online]. Available: https://classic.mceliece.org/nist/mceliece-20171129.pdf

[14] A. Reyhani and M. Hasan, "Error detection in polynomial basis multipliers over binary extension fields," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, 2002, pp. 515–528.

[15] M. López-García and E. Cantó-Navarro, "Hardware-software implementation of a McEliece cryptosystem for post-quantum cryptography," in *Proc. Future Inf. Commun. Conf.*, 2020, pp. 814–825.

[16] L. Bu, R. Agrawal, H. Cheng, and M. A. Kinsy, "A lightweight McEliece cryptosystem co-processor design," 2019. [Online]. Available: arXiv preprint:1903.03733.

[17] S. Subramanian, M. Mozaffari-Kermani, R. Azarderakhsh, and M. Nojoumian, "Reliable hardware architectures for cryptographic block ciphers LED and HIGHT," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 10, pp. 1750–1758, Oct. 2017.