The **AIMer** Signature Scheme*

Seongkwang Kim[1]** , Jincheol Ha[2]** , Mincheol Son[2], Byeonghak Lee[2], Dukjae Moon[1], Joohee Lee[3], Sangyub Lee[1], Jihoon Kwon[1], Jihoon Cho[1], Hyojin Yoon[1], Jooyoung Lee[2]

[1]Samsung SDS, Seoul, Korea
[2]KAIST, Daejeon, Korea
[3]Sungshin Women's University, Seoul, Korea

[1]{sk39.kim, dukjae.moon, sangyub0.lee, jihoon.kwon, jihoon1.cho, hj1230.yoon}@samsung.com
[2]{smilecjf, encrypted.def, lbh0307, hicalf}@kaist.ac.kr
[3]jooheelee@sungshin.ac.kr

Abstract. Post-quantum signature schemes based on the MPC-in-the-Head (MPCitH) paradigm are recently attracting significant attention as their security solely depends on the one-wayness of the underlying primitive, providing diversity for the hardness assumption in post-quantum cryptography. Recent MPCitH-friendly ciphers have been designed using simple algebraic S-boxes operating on a large field in order to improve the performance of the resulting signature schemes. Due to their simple algebraic structures, their algebraic immunity should be comprehensively studied.
In this paper, we refine algebraic cryptanalysis of power mapping based S-boxes over binary extension fields, and cryptographic primitives based on such S-boxes. In particular, for the Gröbner basis attack over $\mathbb{F}_2$, we experimentally show that the exact number of Boolean quadratic equations obtained from the underlying S-boxes is critical to correctly estimate the theoretic complexity based on the degree of regularity. Similarly, it turns out that the XL attack might be faster when all possible quadratic equations are found and used from the S-boxes. This refined cryptanalysis leads to more precise estimation on the algebraic immunity of cryptographic primitives based on algebraic S-boxes.
Considering the refined algebraic cryptanalysis, we propose a new one-way function, dubbed AIM, as an MPCitH-friendly symmetric primitive with high resistance to algebraic attacks. The security of AIM is comprehensively analyzed with respect to algebraic, statistical, quantum, and generic attacks. AIM is combined with the BN++ proof system, yielding a new signature scheme, dubbed AIMer. Our implementation shows that AIMer significantly outperforms existing signature schemes based on symmetric primitives in terms of signature size and signing time.

Keywords: Signature Scheme, MPC-in-the-Head, One-way Function, Algebraic Cryptanaysis

1 Introduction

With a substantial amount of research on quantum computers in recent years, the security threats posed by quantum computers are rapidly becoming a reality. Cryptography is considered particularly risky in the quantum computing environment since the security of most widely used public key schemes relies on the hardness of factoring or discrete logarithm, which is solved in polynomial time with a quantum computer [67]. This encourages the cryptographic community to investigate post-quantum cryptographic schemes which are resilient to quantum attacks. NIST initiated a competition for post-quantum cryptography (PQC) standardization, and recently announced its selected algorithms: CRYSTALS-Kyber [65] as a public key encryption scheme, and CRYSTALS-Dilithium [60], Falcon [62], and SPHINCS$^+$ [45] as digital signature schemes.

MPC-in-the-Head based Signature. MPC-in-the-Head (MPCitH), proposed by Ishai et al. [46], is a paradigm to construct a zero-knowledge proof (ZKP) system from a multiparty computation (MPC) protocol. Its practicality is demonstrated by the ZKBoo scheme, the first efficient MPCitH-based proof scheme

* This work has been submitted to Korean Post-Quantum Cryptography Competition (`https://kpqc.or.kr`). The ePrint version of this paper can be retrieved in `https://ia.cr/2022/1387`

** The first two authors have contributed equally to this work.

proposed by Giacomelli et al. [37]. One of the main applications of the MPCitH paradigm is to construct a post-quantum signature as follows. Given a one-way function f and an input-output pair (x, y) such that $f(x) = y$, one can construct a digital signature scheme with secret key x, public key y, and non-interactive zero-knowledge proof of the knowledge (NIZKPoK) of the secret x as a signature.

The main advantage of MPCitH-based signature schemes is that their security solely depends on the security of the one-way function used in key generation, which makes them more reliable compared to the schemes whose security is based on the hardness assumption of certain mathematical problems with a potential gap in the security reduction. For example, a multivariate signature scheme Rainbow [27] has been recently broken by exploiting the gap between its hardness assumption and the actual security [13]. Also, an isogeny-based key exchange algorithm SIKE [47] reveals its weakness as its security assumption does not hold for a certain class of curves [16]. In this context, MPCitH-based signature schemes are attracting significant attention as they provide diversity for the underlying hardness assumption. The recent call of NIST for additional digital signature schemes[1] also expressed primary interest in signature schemes that are not based on structured lattices. The internal function of an MPCitH-based scheme can be easily updated when any weakness is found in it, which can be seen as an advantage in terms of cryptographic agility.

Picnic [17] is the first and the most famous signature scheme based on the MPCitH paradigm; it combines an MPC-friendly block cipher LowMC [2] and an MPCitH proof system called ZKB++, which is an optimized variant of ZKBoo. Katz et al. [50] proposed a new proof system KKW by further improving the efficiency of ZKB++ with pre-processing, and updated Picnic accordingly. The updated version of Picnic was the only ZKP-based scheme that advanced to the third round of the NIST PQC competition.

LowMC is relatively a new design which can be computed efficiently in the MPC environment, where the AND operation is significantly expensive compared to XOR. There have been various attacks on LowMC, partially motivated by the LowMC challenge[2], some of which have worked effectively [6, 7, 29, 31, 56–58, 63], and the LowMC parameters have been modified accordingly. Due to the security concern on LowMC, there have been attempts to construct MPCitH-based signature schemes from the one-wayness of the standard AES block cipher. In this way, the hardness of key recovery from a single evaluation of AES is reduced to the security of the basing signature scheme. BBQ [24] and Banquet [11] are AES-based signature schemes, where BBQ employs the KKW proof system and Banquet improves BBQ by using an MPCitH proof system optimized for an arithmetic circuit over a large field $\mathbb{F}_{2^{32}}$.

To fully exploit efficient multiplication over a large field in the Banquet proof system, Dobraunig et al. proposed MPCitH-friendly ciphers LS-AES and Rain. They are substitution-permutation ciphers based on the inverse S-box over a large field [32]. This design strategy increases the efficiency of the resulting MPCitH-based signature scheme, while the number of rounds should be carefully determined by comprehensive analysis on any possible aglebraic attack due to their simple algebraic structures. Kales and Zaverucha [48] proposed a number of optimization techniques to further improve the efficiency of the Baum and Nof's proof system [10], and their variant is called BN++. When Rain is combined with BN++, the resulting signature scheme enjoys the shortest signature size for the same level of signing/verification time (compared to existing MPCitH-based signatures) to the best of our knowledge.

1.1 Our Contribution

In this work, we refine algebraic cryptanalysis of power mapping based S-boxes over binary extension fields, and cryptographic primitives based on such S-boxes. In particular, we focus on the Gröbner basis and the XL (eXtended Linearization) attacks since they allow one to solve a system of equations from only a single evaluation of a one-way function, which is the case when it is used in an MPCitH-based signature scheme. Most of previous works on symmetric primitives over large fields analyzed their security against the Gröbner basis attack only over the large fields [1, 3, 32, 39]. Dobraunig et al. consider the analysis over $\mathbb{F}_2$ [32], but only deal with the equations of high degrees. We apply the Gröbner basis attack to the system of quadratic

[1] `https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf`

[2] `https://lowmcchallenge.github.io/`

equations over $\mathbb{F}_2$ using intermediate variables. When it comes to the Gröbner basis attack over $\mathbb{F}_2$, we experimentally show that the exact number of Boolean quadratic equations obtained from the underlying S-boxes is critical to correctly estimate the theoretic complexity based on the degree of regularity. Similarly, it turns out that the XL attack might be faster when all possible quadratic equations are found and used from the S-boxes. These results lead to more precise estimation on the algebraic immunity of cryptographic primitives based on algebraic S-boxes. Our refined algebraic cryptanalysis will be given in Appendix A as the main focus of this paper is put on the design of a one-way function and a signature scheme based on it.

With a design rationale based on the refined algebraic cryptanalysis, we propose a new one-way function, dubbed AIM[3], as an MPCitH-friendly symmetric primitive with high resistance to algebraic attacks. AIM uses Mersenne S-boxes based on power mappings with exponents of the form $2^e - 1$. Compared to the typical inverse S-box, Mersenne S-boxes turn out to provide higher resistance to algebraic attacks. The security of AIM is comprehensively analyzed with respect to algebraic, statistical, quantum and generic attacks. AIM is combined with the BN++ proof system, one of the state-of-the-art MPCitH proof systems working on large fields, yielding a new signature scheme, dubbed AIMer. The AIM function has been designed to fully exploit various optimization techniques of the BN++ proof system to reduce the overall signature size without significantly sacrificing the signing and the verification time.

We implement the AIMer signature scheme and compare its benchmark to existing post-quantum signatures on the same machine. We present a brief list of our C standalone implementation results in Table 1. The detailed values can be found in Section 6. We also implement AIMer with AVX2 instructions and compare the performance to existing post-quantum digital signature schemes in Appendix E. Compared to the signature schemes based on the BN++ proof system combined with the 3-round (resp. 4-round) Rain, which is the state-of-the-art MPCitH-based signature scheme, AIMer enjoys not only 8.21% (resp. 21.15%) shorter signature size but also 1.22% (resp. 13.41%) improved signing performance at 128-bit security level with the number of parties N being set to 16.

Scheme	N	τ	Keygen (ms)	Sign (ms)	Verify (ms)	*pk* Size (B)	*sk* Size (B)	*sig* Size (B)
AIMer-I	16	33	0.06	2.26	2.17	32	16	5 904
	1615	13	0.06	86.62	85.99	32	16	3 840
AIMer-III	16	49	0.13	4.75	4.58	48	24	13 080
	1621	19	0.13	178.72	174.64	48	24	8 352
AIMer-V	16	65	0.31	10.88	10.50	64	32	25 152
	1623	25	0.31	395.65	391.87	64	32	15 392

Table 1: Brief list of performance of AIMer reference implementation.

2 Preliminaries

2.1 Notation

For two vectors a and b over a finite field, their concatenation is denoted by $a\|b$. For a positive integer n, $\mathsf{hw}(n)$ denotes the Hamming weight of n in its binary representation, and we write $[n] = \{1, \cdots, n\}$.

In the multiparty computation setting, $x^{(i)}$ denotes the i-th party's additive share of x, which implies that $\sum_i x^{(i)} = x$.

[3] This name is an abbreviation of Affine-Interleaved Mersenne S-boxes.

For a set S, we will write $a \leftarrow S$ to denote that a is chosen uniformly at random from S. For a probability distribution $\mathcal{D}$, $a \leftarrow \mathcal{D}$ denotes that a is sampled according to the distribution $\mathcal{D}$. The binomial distribution with the number of trials n and the success probability p is denoted by $\mathrm{Bin}(n, p)$.

Unless stated otherwise, all logarithms are to the base 2. The complexity of matrix multiplication of two $n \times n$ matrices is $O(n^\omega)$ for some ω such that $2 \le \omega \le 3$. The constant ω is called the matrix multiplication exponent, and it will be conservatively set to 2 in this paper.

2.2 Algebraic Attacks

An algebraic attack on a symmetric primitive is to model it as a system of multivariate polynomial equations and to solve it using algebraic techniques. A straightforward way of establishing a system of equations is to represent the output of the primitive as a polynomial of the input including the secret key. In order to reduce the degree of the system of equations, intermediate variables might be introduced. For example, all the inputs and outputs of the underlying S-boxes can be regarded as independent variables.

One of the well-known methods of solving a system of equations is to define a system of linear equations by replacing every monomial of degree greater than one by a new variable and solve it, which is called *trivial linearization*. In the linearizaton, a large number of new variables might be introduced, and that many equations are also needed to determine a solution to the system of (linear) equations. However, in most ZKP-based digital signature schemes, one is given only a single evaluation of the underlying primitive, which limits the total number of equations thereof. For this reason, our focus will be put on algebraic attacks possibly using a small number of equations such as the Gröbner basis attack and the XL attack.

Gröbner Basis Attack. The Gröbner basis attack is to solve a system of equations by computing its Gröbner basis. The attack consists of the following steps.

1. Compute a Gröbner basis in the *grevlex* (graded reverse lexicographic) order.
2. Change the order of terms to obtain a Gröbner basis in the *lex* (lexicographic) order.
3. Find a univariate polynomial in this basis and solve it.
4. Substitute this solution into the Gröbner basis and repeat Step 3.

When a system of equations has only finitely many solutions in its algebraic closure, its Gröbner basis in the *lex* order always contains a univariate polynomial. When a single variable of the polynomial is replaced by a concrete solution, the Gröbner basis still remains a Gröbner basis of the "reduced" system, allowing one to obtain a univariate polynomial again for the next variable. We refer to [64] for more details on Gröbner basis computation.

The complexity of Gröbner basis computation can be estimated using the *degree of regularity* of the system of equations [8]. Consider a system of m equations in n variables $\{f_i\}_{i=1}^m$. Let d_i denote the degree of f_i for $i = 1, 2, \ldots, m$. If the system of equations is over-defined, i.e., $m > n$, then the degree of regularity can be estimated by the smallest of the degrees of the terms with non-positive coefficients for the following Hilbert series under the semi-regular assumption [36].

$$\mathrm{HS}(z) = \frac{1}{(1-z)^n} \prod_{i=1}^{m} (1 - z^{d_i}).$$

Given the degree of regularity d_{reg}, the complexity of computing a Gröbner basis of the system is known to be

$$O\left(\binom{n + d_{reg}}{d_{reg}}^\omega\right).$$

In the Gröbner basis attack, one always obtains an over-defined system of equations since each variable x should be contained in a finite field $\mathbb{F}_{p^e}$ for some characteristic p, and hence x satisfies $x^{p^e} - x = 0$ called a *field equation*. By including field equations in the system of equations, one can remove any possible solution outside $\mathbb{F}_{p^e}$ (in the algebraic closure). For some symmetric primitives, the field equations have not been taken into account in their analysis of the Gröbner basis attack [2,3,32,39]. It does not mean that they are broken under the modified analysis, while considering the field equations would lead to more precise analysis of the Gröbner basis attack.

XL Attack. The XL algorithm, proposed by Courtois et al. [20], can be viewed as a generalization of the relinearization attack [52]. For a system of m quadratic equations in n variables over $\mathbb{F}_2$, the trivial linearization does not work if m is smaller than the number of monomials appearing in the system.

The XL algorithm extends the system of equations by multiplying all the monomials of degree at most $D-2$ for some $D > 2$ to obtain a larger number of linearly independent equations. Since the number of monomials of degree at most $D-2$ is $\sum_{i=1}^{D-2}\binom{n}{i}$, the resulting system consists of $\left(\sum_{i=1}^{D-2}\binom{n}{i}\right)m$ equations of degree at most D with at most $\sum_{i=1}^{D}\binom{n}{i}$ monomials of degree at most D. When the number of equations equals the number of monomials as D grows, one can solve the extended system of equations by linearization.

In contrast to the Gröbner basis attack, it is not easy to precisely estimate the complexity of the XL attack since there is no theoretic estimation for the number of linearly independent equations obtained from the XL algorithm. Instead, we can loosely upper bound the number of linearly independent equations by $(\sum_{i=1}^{D-2}\binom{n}{i})m$. Under the assumption that all the equations obtained from the XL algorithm are linearly independent, which is in favor of the attacker, we can search for the (smallest) degree D such that

$$\left(\sum_{i=1}^{D-2}\binom{n}{i}\right)m \geq T \tag{1}$$

where T denotes the exact number of monomials appearing in the extended system of equations, which is upper bounded by $\sum_{i=1}^{D}\binom{n}{i}$. Once D is fixed, the extended system of equations can be solved by trivial linearization whose time complexity is given as

$$O\left(T^{\omega}\right).$$

2.3 BN++ Zero-knowledge Protocol

In this section, we briefly review the BN++ proof system [48], one of the state-of-the-art MPCitH zero-knowledge protocols. The BN++ protocol will be combined with our symmetric primitive AIM to construct the AIMer signature scheme which is fully described in Section 5. At a high level, BN++ is a variant of the BN protocol [10] with several optimization techniques applied to reduce the signature size.

Protocol Overview. The BN++ protocol follows the MPCitH paradigm [46]. In order to check C multiplication triples $(x_j, y_j, z_j = x_j \cdot y_j)_{j=1}^{C}$ over a finite field $\mathbb{F}$ in the multiparty computation setting with N parties, *helping values* $((a_j, b_j)_{j=1}^{C}, c)$ are required, where $a_j \leftarrow \mathbb{F}, b_j = y_j$, and $c = \sum_{j=1}^{C} a_j \cdot b_j$. Each party holds secret shares of the multiplication triples $(x_j, y_j, z_j)_{j=1}^{C}$ and helping values $((a_j, b_j)_{j=1}^{C}, c)$. Then the protocol proceeds as follows.

- A prover is given random challenges
$$\epsilon_1, \cdots, \epsilon_C \in \mathbb{F}.$$
- For $i \in [N]$, the i-th party locally sets
$$\alpha_1^{(i)}, \cdots, \alpha_C^{(i)}$$
where $\alpha_j^{(i)} = \epsilon_j \cdot x_j^{(i)} + a_j^{(i)}$.
- The parties open $\alpha_1, \cdots, \alpha_C$ by broadcasting their shares.
- For $i \in [N]$, the i-th party locally sets
$$v^{(i)} = \sum_{j=1}^{C} \epsilon_j \cdot z_j^{(i)} - \sum_{j=1}^{C} \alpha_j \cdot b_j^{(i)} + c^{(i)}.$$
- The parties open v by broadcasting their shares and output `Accept` if $v = 0$.

The probability that there exist incorrect triples and the parties output `Accept` in a single run of the above steps is upper bounded by $1/|\mathbb{F}|$.

SIGNATURE SIZE. By applying the Fiat-Shamir transform [33], one can obtain a signature scheme from the BN++ proof system. In this signature scheme, the signature size is given as

$$6\lambda + \tau \cdot (3\lambda + \lambda \cdot \lceil \log_2(N) \rceil + \mathcal{M}(C)),$$

where λ is the security parameter, C is the number of multiplication gates in the underlying symmetric primitive, and $\mathcal{M}(C) = (2C+1) \cdot \log_2(|\mathbb{F}|)$. In particular, $\mathcal{M}(C)$ has been defined so from the observation that sharing the secret share offsets for $(z_j)_{j=1}^{C}$ and c, and opening shares for $(\alpha_j)_{j=1}^{C}$ occurs for each repetition, using C, 1, and C elements of $\mathbb{F}$, respectively. For more details, we refer to [48].

OPTIMIZATION TECHNIQUES. If multiplication triples use an identical multiplier in common, for example, given (x_1, y, z_1) and (x_2, y, z_2), then the corresponding α values can be batched to reduce the signature size. Instead of computing $\alpha_1 = \epsilon_1 \cdot x_1 + a_1$ and $\alpha_2 = \epsilon_2 \cdot x_2 + a_2$, $\alpha = \epsilon_1 \cdot x_1 + \epsilon_2 \cdot x_2 + a$ is computed, and v is defined as

$$v = \epsilon_1 \cdot z_1 + \epsilon_2 \cdot z_2 - \alpha \cdot y + c,$$

where $c = a \cdot y$. This technique is called *repeated multiplier* technique. Our symmetric primitive design allows us to take full advantage of this technique to reduce the number of α values in each repetition of the protocol.

If the output of the multiplication z_i can be locally generated from each share, then the secret share offset is not necessarily included in the signature.

3 AIM: Our New Symmetric Primitive

3.1 Specification

AIM is designed to be a "tweakable" one-way function so that it offers multi-target one-wayness. Given input/output size n and an $(\ell+1)$-tuple of exponents $(e_1, \ldots, e_\ell, e_*) \in \mathbb{Z}^{\ell+1}$, $\mathsf{AIM} : \mathbb{F}_{2^n} \times \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is defined by

$$\mathsf{AIM}(\mathsf{iv}, \mathsf{pt}) = \mathsf{Mer}[e_*] \circ \mathsf{Lin}[\mathsf{iv}] \circ \mathsf{Mer}[e_1, \ldots, e_\ell](\mathsf{pt}) \oplus \mathsf{pt}$$

where each function will be described below. See Figure 1 for the pictorial description of AIM with $\ell = 3$.

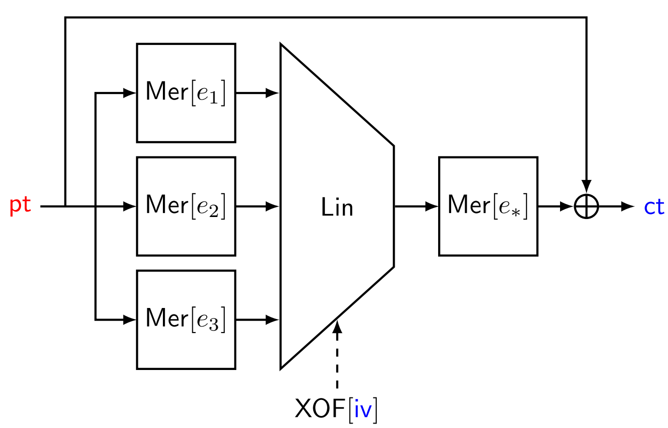

Fig. 1: The AIM-V one-way function with $\ell = 3$. The input pt (in red) is the secret key of the signature scheme, and (iv, ct) (in blue) is the corresponding public key.

S-boxes. In AIM, S-boxes are exponentiation by Mersenne numbers over a large field. More precisely, for $x \in \mathbb{F}_{2^n}$,

$$\mathsf{Mer}[e](x) = x^{2^e-1}$$

for some e. Note that this map is a permutation if $\gcd(e, n) = 1$. As an extension, $\mathsf{Mer}[e_1, \ldots, e_\ell] : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}^\ell$ is defined by

$$\mathsf{Mer}[e_1, \ldots, e_\ell](x) = \mathsf{Mer}[e_1](x) \| \ldots \| \mathsf{Mer}[e_\ell](x).$$

Linear Components. AIM includes two types of linear components: an affine layer and feed-forward. The affine layer is multiplication by an $n \times \ell n$ random binary matrix A_{iv} and addition by a random constant $b_{\mathsf{iv}} \in \mathbb{F}_2^n$. The matrix

$$A_{\mathsf{iv}} = \left[A_{\mathsf{iv},1} \middle| \ldots \middle| A_{\mathsf{iv},\ell}\right] \in (\mathbb{F}_2^{n \times n})^\ell$$

is composed of ℓ random invertible matrices $A_{\mathsf{iv},i}$. The matrix A_{iv} and the vector b_{iv} are generated by an extendable output function (XOF) with the initial vector iv. Each matrix $A_{\mathsf{iv},i}$ can be equivalently represented by a linearized polynomial $L_{\mathsf{iv},i}$ on $\mathbb{F}_{2^n}$. For $x = (x_1, \ldots, x_\ell) \in (\mathbb{F}_{2^n})^\ell$,

$$\mathsf{Lin}[\mathsf{iv}](x) = \sum_{1 \le i \le \ell} L_{\mathsf{iv},i}(x_i) \oplus b_{\mathsf{iv}}.$$

By abuse of notation, we will denote Ax as the same meaning as $\sum_{1 \le i \le \ell} L_{\mathsf{iv},i}(x_i)$. Feed-forward operation, which is addition by the input itself, makes the entire function non-invertible.

Recommended Parameters. Recommended sets of parameters are given in Table 2. The number of S-boxes is determined by taking into account the complexity of the XL attack, which is described in Section 4.1. Exponents e_1 and e_* are chosen as small numbers to provide smaller differential probability, and exponent e_2 is chosen so that $e_2 \cdot e_* \ge \lambda$, while all the exponents are distinct in the same set of parameters. The irreducible polynomials for extension fields $\mathbb{F}_{2^{128}}$, $\mathbb{F}_{2^{192}}$, and $\mathbb{F}_{2^{256}}$ are the same as those used in Rain [32].

Scheme	λ	n	ℓ	e_1	e_2	e_3	e_*
AIM-I	128	128	2	3	27	-	5
AIM-III	192	192	2	5	29	-	7
AIM-V	256	256	3	3	53	7	5

Table 2: Recommended sets of parameters of AIM.

3.2 Design Rationale

Choice of Field. When a symmetric primitive is built upon field operations, the field is typically binary since bitwise operations are cheap in most of modern architectures. However, when the multiplicative complexity of the primitive becomes a more important metric for efficiency, it is hard to generally specify which type of field has merits with respect to security and efficiency.

Focusing on a primitive for MPCitH-style zero-knowledge protocols, a primitive over a large field generally requires a small number of multiplications, leading to shorter signatures. However, any primitive operating on a large field of large prime characteristic might permit algebraic attacks since the number of variables and the algebraic degree will be significantly limited for efficiency reasons. On the other hand, binary extension fields enjoy both advantages from small and large fields. In particular, matrix multiplication is represented by a polynomial of high algebraic degree without increasing the proof size.

Algebraically Sound S-boxes. In an MPCitH-style zero-knowledge protocol, the proof size of a circuit is usually proportional to the number of nonlinear operations in the circuit. In order to minimize the number of multiplications, one might introduce intermediate variables for some wires of the circuit. For example, the inverse S-box ($S(x) = x^{-1}$) has high (bitwise) algebraic degree $n-1$, while it can be simply represented by a quadratic equation $xy = 1$ by letting the output from the S-box be a new variable y. When an S-box is represented by a quadratic equation of its input and output, we will say it is *implicitly quadratic*. In particular, we consider implicitly quadratic S-boxes which are represented by a single multiplication over $\mathbb{F}_{2^n}$. This feature makes the proof size short and mitigates algebraic attacks at the same time.

The inverse S-box is one of the well-studied implicitly quadratic S-boxes. The inverse S-box has been widely adopted to symmetric ciphers [4, 23, 66] due to its nice cryptographic properties. It is invertible, is of high-degree, has good enough differential uniformity and nonlinearity. Recently, it is used in symmetric primitives for advanced cryptographic protocols such as multi-party computation and zero-knowledge proof [32, 39, 40].

Meanwhile, the inverse S-box has one minor weakness; a single evaluation of the n-bit inverse S-box as a form of $xy = 1$ produces $5n-1$ linearly independent quadratic equations over $\mathbb{F}_2$ [21]. The complexity of an algebraic attack is typically bounded (with heuristics) by the degree and the number of equations, and the number of variables. In particular, an algebraic attack is more efficient with a larger number of equations, while this aspect has not been fully considered in the design of recent symmetric ciphers based on algebraic S-boxes. When the number of rounds is small, this issue might be critical to the overall security of the cipher. For more details, see Appendix A and Section 4.1.

With the above observation, we tried to find an invertible S-box of high-degree which is moderately resistant to differential/linear cryptanalysis as well as implicitly quadratic, and *producing only a small number of quadratic equations*. Since our attack model does not allow multiple queries to a single instance of AIM, we allow a relaxed condition on the DC/LC resistance, not being necessarily maximal. As a family of S-boxes that beautifully fit all the conditions, we choose a family of Mersenne S-boxes; they are exponentiation by Mersenne numbers. A Mersenne S-box whose exponent is of the form $2^e - 1$ such that $\gcd(n, e) = 1$, is invertible, is of high-degree, needs only one multiplication for its proof, produces only $3n$ Boolean quadratic equations with its input and output, and provides moderate DC/LC resistance. Furthermore, when the implicit equation $xy = x^{2^e}$ of a Mersenne S-box is computed in the BN++ proof system, it is not required to broadcast the output share since the output of multiplication x^{2^e} can be locally computed from the share of x.

Repetitive Structure. The efficiency of the BN++ proof system partially comes from the optimization technique using *repeated multipliers*. When a multiplier is repeated in multiple equations to prove, the proof can be done in a batched way, reducing the overall signature size. In order to maximize the advantage of repeated multipliers, we put S-boxes in parallel at the first round in order to make the S-box inputs the same. Then, we put only one S-box at the second round with feed-forward. In this way, all the implicit equations have the same multiplier.

Affine Layer Generation. The main advantage of using binary affine layers in large S-box-based constructions is to increase the algebraic degree of equations over the large field. Multiplication by a random $n \times n$ binary matrix can be represented as (2). Similarly, our design uses a random affine map from $\mathbb{F}_2^{\ell n}$ to $\mathbb{F}_2^n$. In order to mitigate multi-target attacks (in the multi-user setting), the affine map is uniquely generated for each user; each user's iv is fed to an XOF, generating the corresponding linear layer.

4 Security Analysis of AIM

In order for the basing signature scheme to be secure, AIM with fixed iv should be preimage-resistant. An adversary is given a randomly chosen iv and an output ct that is defined by iv and a randomly chosen input pt^*. Given such a pair $(\mathsf{iv}, \mathsf{ct})$, the adversarial goal is to find any pt (not necessarily the same as pt^*) such that $\mathsf{AIM}(\mathsf{iv}, \mathsf{pt}) = \mathsf{ct}$. In the multi-user setting, the adversary is given multiple IV-output pairs $\{(\mathsf{iv}_i, \mathsf{ct}_i)\}_i$, and tries to find any pt such that $\mathsf{AIM}(\mathsf{iv}_i, \mathsf{pt}) = \mathsf{ct}_i$ for some i.

4.1 Algebraic Attacks

Since our attack model does not allow multiple evaluations for a single instance of AIM, we do not consider interpolation, higher-order differential, and cube attacks. Instead, as mentioned in Section 2.2, we mainly consider the Gröbner basis attack and the XL attack using a single evaluation of AIM. In particular, we exploit the implicit Boolean representations of power mapping based S-boxes over $\mathbb{F}_{2^n}$, obtaining a refined algebraic analysis for these attacks.

Representation in $\mathbb{F}_2$ and its Extension Field. When a symmetric primitive is defined with arithmetic in a large field, it is straightforward to establish a system of equations from a single evaluation of the primitive using the same field arithmetic. If the underlying field is a binary extension field $\mathbb{F}_{2^n}$ for some n, then it is also possible to establish a system of equations over $\mathbb{F}_2$. Suppose that $\{1, \beta, \ldots, \beta^{n-1}\}$ is a basis of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$. Then each variable $x \in \mathbb{F}_{2^n}$ can be represented as n variables $x_0, x_1, \ldots, x_{n-1} \in \mathbb{F}_2$ by setting $x = \sum_{i=0}^{n-1} x_i\beta^i$. Using the representation of β^n with respect to this basis, every polynomial equation over $\mathbb{F}_{2^n}$ can be transformed into n equations over $\mathbb{F}_2$.

On the other hand, a linear equation over $\mathbb{F}_2$ is represented by a *linearized polynomial* over $\mathbb{F}_{2^n}$:

$$\sum_{i=0}^{n-1} a_i x^{2^i} = a_0x + a_1x^{2^1} + a_2x^{2^2} + \cdots + a_{n-1}x^{2^{n-1}} \tag{2}$$

where $a_0, a_1, \ldots, a_{n-1} \in \mathbb{F}_{2^n}$.

Suppose that variables x and y in $\mathbb{F}_{2^n}$ are represented by $\{x_i\}_{i=0}^{n-1}$ and $\{y_i\}_{i=0}^{n-1}$, respectively, in $\mathbb{F}_2$. If $y = x^a$ for some a, then each y_i is represented as a polynomial of x_i's of degree $\mathsf{hw}(a)$. For instance, the inverse S-box $y = x^{2^n-2}$ can be represented as a system of n equations of degree $n-1$.

Most of previous works on symmetric primitives over a large field, their security against the Gröbner basis attack have been analyzed only over the large field [1, 3, 32, 39]. However, when the primitives are defined over the binary extension fields, it is also possible to represent them by systems of equations over $\mathbb{F}_2$. For example, Dobraunig et. al. consider the representation of Rain over $\mathbb{F}_2$ using the above description of the inverse S-box [32], obtaining equations of the highest degree that make the algebraic analysis infeasible. We apply the Gröbner basis attack to the system of *quadratic* equations over $\mathbb{F}_2$ using intermediate variables as described below.

Number of Quadratic Equations. The efficiency of algebraic cryptanalysis heavily depends on the number of variables, the number of equations, and their degrees for the system of equations. As discussed above, a powering function $y = x^a$ over $\mathbb{F}_{2^n}$ can be represented as a system of n equations of degree $\mathsf{hw}(a)$ over $\mathbb{F}_2$. The resulting equations are explicit ones in a sense that each output variable is represented by an equation only in the input variables. However, their *implicit* representation might consist of equations of degree smaller than the explicit ones. For example, $y = x^{2^n-2}$ obtained from the inverse S-box is equivalent to the quadratic equation $xy = 1$ over $\mathbb{F}_{2^n}$, assuming the input x is nonzero, or a certain set of n quadratic equations in n variables over $\mathbb{F}_2$.

Implicit representation over $\mathbb{F}_2$ might also increase the number of (linearly independent) equations. There has been a significant amount of research on the number of linearly independent quadratic equations obtained from power functions over $\mathbb{F}_{2^n}$ [19, 21, 42, 61]. For example, it is known that one has $5n$ quadratic equations over $\mathbb{F}_2$ from $xy = 1$ over $\mathbb{F}_{2^n}$ [19]. However, the relation $xy = 1$ holds for the inverse S-box only when x and y are nonzero. Courtois et al. [21] shows that $5n-1$ linearly independent quadratic equations are obtained from the exact representation of the inverse S-box. See Appendix A for the details of how the number of quadratic equations affects the complexity of the Gröbner basis attack and the XL attack.

The Gröbner Basis Attack on AIM. A single equation of an input pt to AIM over $\mathbb{F}_{2^n}$ is of high degree, so it is infeasible to solve this type of system using the Gröbner basis attack. Alternatively, one can construct a system of equations over $\mathbb{F}_{2^n}$ using certain intermediate variables. Let u_i denote the output of the S-box $\mathsf{Mer}[e_i]$ and let v_i denote the output of the linear component $L_{\mathsf{iv},i}$ for $i = 1, 2, \ldots, \ell$. Then, we obtain the following system of equations

$$\begin{cases} u_i = \mathsf{pt}^{2^{e_i}-1} & \text{for } i = 1, 2, \ldots, \ell, \\ v_i = L_{\mathsf{iv},i}(u_i) & \text{for } i = 1, 2, \ldots, \ell, \\ \mathsf{pt} \oplus \mathsf{ct} = (v_1 \oplus \ldots \oplus v_\ell \oplus b_{\mathsf{iv}})^{2^{e_*}-1} \end{cases}$$

where $L_{\mathsf{iv},i}(\cdot)$ denotes the linearized polynomial of degree 2^{n-1} (with high probability), induced from the random matrix $A_{\mathsf{iv},i}$. Together with $2\ell + 1$ field equations, we obtain the following Hilbert series.

$$\prod_{i=1}^{\ell} \left(\frac{1 - z^{2^{e_i}-1}}{1 - z} \right) \left(\frac{1 - z^{2^{e_*}-1}}{1 - z} \right) \left(\frac{1 - z^{2^{n-1}}}{1 - z} \right)^{\ell} (1 - z^{2^n})^{2\ell+1}.$$

So the degree of regularity is estimated to be greater than 2^n, obtaining the complexity

$$\binom{(2\ell+1) + 2^n}{2^n}^{\omega} > 2^n$$

for $\ell \geq 2$.

We can also construct a system of implicit Boolean quadratic equations over $\mathbb{F}_2$ as described above. The corresponding Hilbert series is as follows.

$$\mathrm{HS}(z) = \frac{1}{(1-z)^{(\ell+1)n}} (1 - z^2)^{(1+\nu)(\ell+1)n} = (1 + z)^{(\ell+1)n} (1 - z^2)^{(\ell+1)\nu n}$$

where $\nu = 3$ in case of AIM. We perform Gröbner basis computation on AIM with $\ell = 2, 3$ for toy parameters, summarizing the result in Figure 2. Being the same as the single-round Even-Mansour cipher, the solving degrees for the both basic and full systems of AIM are also close to the estimated values for the full system. The estimated degrees of regularity and corresponding time complexities to compute a Gröbner basis for the full system of AIM are summarized in Table 3.

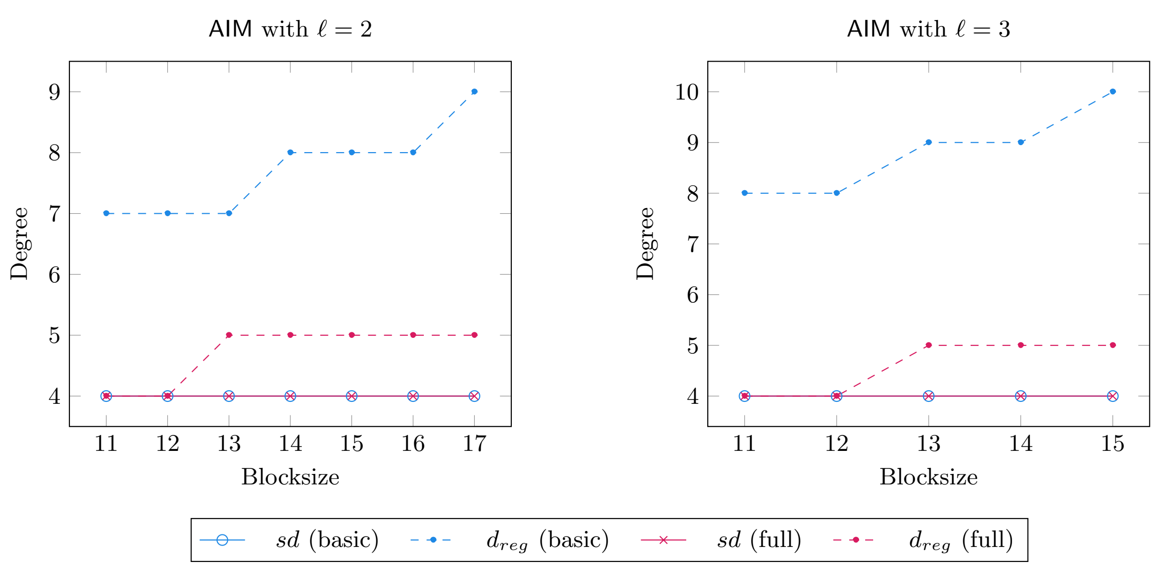

Fig. 2: Degree of regularity d_{reg} estimated by (7) and the solving degree sd for AIM with $\ell = 2, 3$ using Mersenne S-boxes.

Scheme	n	ν	Gröbner Basis		XL	
			d_{reg}	Time (bits)	D	Time (bits)
AIM-I	128		20	222.8	12	148.0
AIM-III	192	3	27	310.8	15	194.1
AIM-V	256		45	530.3	19	266.1
	128		14	168.5	10	$127.9^\dagger$
Rain_3	192	5	19	235.9	12	$162.1^\dagger$
	256		24	303.1	13	$183.9^\dagger$
	128		17	219.2	11	147.3
Rain_4	192	5	24	303.1	13	$183.9^\dagger$
	256		30	385.9	15	$219.2^\dagger$

Table 3: Analyses of the Gröbner basis attack and the XL attack for AIM and Rain. d_{reg} is the estimated value for the degree of regularity and D is the target degree of the XL attack to obtain equations more than the number of monomials (under the independence assumption) for the full systems of AIM and Rain. †We note that the time complexity for the XL attack is a lower bound that is smaller than the actual complexity due to the independence assumption and the use of $\omega = 2$, so that this values does not imply that the Rain parameters are broken.

The XL Attack on AIM. The XL attack complexity can be loosely bounded in terms of the number of monomials T appearing in the extended system of the target degree D. We observe that the systems of equations defined by the inverse and the Mersenne S-boxes are dense for toy parameters (see Appendix A.2). Letting $T = \sum_{i=1}^{D} \binom{n}{i}$, we can find the smallest degree D satisfying (1). We emphasize again that the time complexity computed from the smallest degree D is rather loose since the estimation is based on the assumption that all the equations obtained by the XL algorithm are linearly independent, which might not be the case in general. The degree D and the corresponding time complexity of the XL attack on the full system of AIM are summarized in Table 3. We observe that AIM is secure for all the parameter sets even with this (loose) lower bound on the complexity of the XL attack.

AIM vs. Rain. We perform experiments for the 3-round Rain (denoted Rain_3) with toy parameters. It can be viewed as a 3-round Even-Mansour-type cipher based on the inverse S-box, so the degree of regularity is estimated by (7) with $r = 3$ and $\nu = 5$. Figure 3 shows the estimated degree of regularity and the solving degree for Rain_3. The result suggests that the exact number of quadratic equations should be considered to estimate the degree of regularity.

We note that the number of variables, the number of equations and their degrees are the same for the basic systems of Rain_3 and AIM with $\ell = 2$, and for the basic systems of Rain_4 and AIM with $\ell = 3$. This implies that the difference of algebraic cryptanalysis between the full systems of AIM and Rain only comes from the values of ν, determined by the number of linearly independent quadratic equations of their S-boxes. Table 3 compares the complexities of the Gröbner basis and the XL attacks for the full systems of AIM and Rain. Compared to Rain, AIM provides stronger security against the Gröbner basis and the XL attacks.

4.2 Quantum Attacks

Quantum attacks are classified into two types according to the attack model. In the Q1 model, an adversary is allowed to use quantum computation without making any quantum query, while in the Q2 model, both quantum computation and quantum queries are allowed [70].

As a generic algorithm for exhaustive key search, Grover's algorithm has been known to give quadratic speedup compared to the classical brute-force attack [41]. In this section, we investigate if any specialized

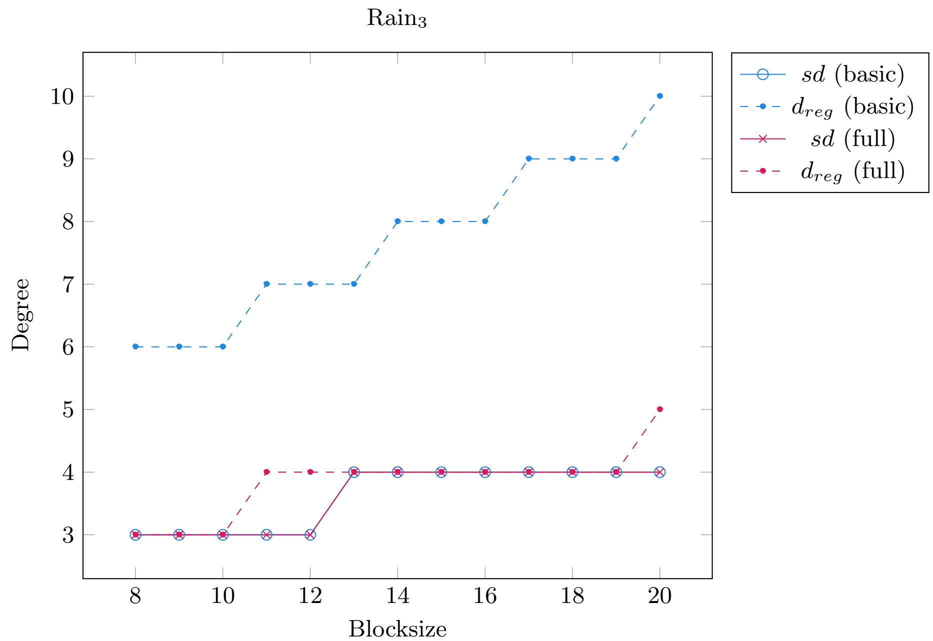

Fig. 3: Degree of regularity d_{reg} estimated by (7) and the solving degree sd for the Rain$_3$ cipher.

quantum algorithm targeted at AIM might possibly achieve better efficiency than Grover's algorithm in the Q1 model.

Quantum Algebraic Attack. When an algebraic root-finding algorithm works over a small field, the guess-and-determine strategy might be effectively combined with Grover's algorithm, reducing the overall time complexity.

The GroverXL algorithm [12] is a quantum version of the FXL algorithm [20], which solves a system of multivariate quadratic equations over a finite field. A single evaluation of AIM can be represented by Boolean quadratic equations using intermediate variables. Precisely, we have a system of $4(\ell+1)n$ equations (including field equations) in $(\ell+1)n$ variables. For this system of equations, the complexity of GroverXL is given as $O(2^{(0.3687+o(1))(\ell+1)n})$, which is worse than Grover's algorithm.

The `QuantumBooleanSolve` algorithm [35] is a quantum version of the `BooleanSolve` algorithm [9], which solves a system of Boolean quadratic equations. In [35], its time complexity has been analyzed only for a system of equations with the same number of variables and equations. A single evaluation of AIM can be represented by $(\ell+1)n$ equations in $(\ell+1)n$ variables. In this case, the complexity of `QuantumBooleanSolve` is given as $O(2^{0.462(\ell+1)n})$, which is worse than Grover's algorithm.

In contrast to the algorithms discussed above, Chen and Gao [18] proposed a quantum algorithm to solve a system of multivariate equations using the Harrow-Hassidim-Lloyd (HHL) algorithm [43] that solves a sparse system of linear equations with exponential speedup. In brief, Chen and Gao's algorithm solves a system of linear equations from the Macaulay matrix by the HHL algorithm. It has been claimed that this algorithm enjoys exponential speedup for a certain set of parameters. When applied to AIM, the hamming weight of the secret key should be smaller than $O(\log n)$ to achieve exponential speedup [26]. Otherwise, this algorithm is slower than Grover's algorithm [26].

Quantum Generic Attack. A generic attack does not use any particular property of the underlying components (e.g., S-boxes for AIM). The underlying smaller primitives are typically modeled as public random permutations or functions. The Even-Mansour cipher [34], the FX-construction [51] and a Feistel cipher [59] have been analyzed in the classic and generic attack model. As their quantum analogues, the Even-Mansour cipher [14, 54], the FX-construction [44, 55] and a Feistel cipher [53] have been analyzed in the Q1 or Q2 model. Most of these attacks can be seen as a combination of Simon's period finding algorithm [68] (in the Q2 model), and Grover's/offline Simon's algorithms [14] (in the Q1 model). Since Simon's period finding algorithm requires multiple queries to a *keyed* construction (which is not the case for AIM), we believe that the above attacks do not apply to AIM in a straightforward manner.

4.3 Statistical Attacks

The adversary is allowed to evaluate AIM with an arbitrary input pair $(\mathsf{pt}, \mathsf{iv})$ in an offline manner. However, such an evaluation is independent of the actual secret key pt^*, so the adversary is not able to collect a sufficient amount of statistical data which are related to pt^*. Furthermore, the linear layer of AIM is generated independently at random for every user. For this reason, we believe that our construction is secure against any type of statistical attacks including (impossible) differential, boomerang, and integral attacks.

That said, to prevent any unexpected variant of differential and linear cryptanalysis, we summarize differential and linear probabilities in this section. For more details, see Appendix B and C.

Differential Cryptanalysis. For the differential probability, we bound the maximum differential probability without expectation as AIM is a key-less primitive. We bound the probability

$$\mathrm{MDP}^{\mathsf{AIM}} = \max_{\Delta x \neq 0, \Delta y} \Pr_{x} \left[\mathsf{AIM}(x \oplus \Delta x) \oplus \mathsf{AIM}(x) = \Delta y\right].$$

As $\mathrm{MDP}^{\mathsf{AIM}}$ cannot be less than $2^{-\lambda}$ for security parameter λ, The values of $\log\gamma$ such that

$$\Pr_{A,b}\left[\mathrm{MDP}^{\mathsf{AIM}} > \gamma\right] < 2^{-\lambda}$$

is summarized in Table 4 according to the security level, where A (resp. b) are the random matrix (resp. vector) in the affine layer. We remark that $\gamma > 2^{-\lambda}$ does not imply the feasibility of differential cryptanalysis for each λ.

λ	128	192	256
$\log\gamma$	-118.4	-178.0	-245.9

Table 4: $\log\gamma$ such that $\Pr_{A,b}\left[\mathrm{MDP}^{\mathsf{AIM}} > \gamma\right] < 2^{-\lambda}$ for each security level λ.

Linear Cryptanalysis. In contrast to differential cryptanalysis, security against linear cryptanalysis has been rarely evaluated for key-less primitives. For this reason, we find the condition when the bias of a correlation trail are less than $2^{-\lambda}$ assuming the masked sums of inputs and outputs are independent. When

$$\min_{1\le i\le \ell}(2^{e_i}-2)^2(2^{e_*}-2)^2 < 2^n,$$

the bias of a correlation trail in AIM is smaller 2^{-n}, and the amount of data required for linear cryptanalysis becomes at least 2^n.

4.4 Security Proof

In this section, we prove the one-wayness of AIM when the underlying S-boxes are modeled as public random permutations and iv is (implicitly) fixed. For simplicity, we will assume that $\ell = 2$. The security of AIM with $\ell > 2$ is similarly proved.

In the public permutation model and in the single-user setting, AIM is defined as

$$\mathsf{AIM}(\mathsf{pt}) = S_3(A_1 \cdot S_1(\mathsf{pt}) \oplus A_2 \cdot S_2(\mathsf{pt}) \oplus b) \oplus \mathsf{pt}$$

for $\mathsf{pt} \in \{0,1\}^n$, where S_1, S_2, S_3 are independent public random permutations, and A_1 and A_2 are fixed $n \times n$ invertible matrices, and b is a fixed $n \times 1$ vector over $\mathbb{F}_2$.

An adversary $\mathcal{A}$ is allowed to choose any value $\mathsf{ct} \in \{0,1\}^n$ on its own, and then make a certain number of forward and backward queries to S_1, S_2 and S_3. Specifically, suppose that $\mathcal{A}$ makes q permutation queries in total. If $\mathcal{A}$ succeeds in finding all the S-box evaluations that make up an evaluation $\mathsf{AIM}(\mathsf{pt}) = \mathsf{ct}$ for some $\mathsf{pt} \in \{0,1\}^n$, then we say that $\mathcal{A}$ wins the preimage-finding game, breaking the one-wayness of AIM. The goal of our security proof is to show that $\mathcal{A}$'s winning probability, denoted $\mathbf{Adv}^{\mathsf{pre}}_{\mathsf{AIM}}(q)$, is small.

We will assume that $\mathcal{A}$ is information-theoretic, and hence deterministic. Furthermore, we assume that $\mathcal{A}$ does not make any redundant query. We will also slightly modify $\mathcal{A}$ so that whenever $\mathcal{A}$ makes a (forward or backward) query to S_1 (resp. S_2) obtaining $S_1(x) = y$ (resp. $S_2(x) = y$), $\mathcal{A}$ makes an additional *forward* query to S_2 (resp. S_1) with *x for free*. This additional query will not degrade $\mathcal{A}$'s preimage-finding advantage since $\mathcal{A}$ is free to ignore it.

An evaluation $\mathsf{AIM}(\mathsf{pt}) = \mathsf{ct}$ consists of three S-box queries. Among the three S-box queries, the lastly asked one is called the *preimage-finding query*. We distinguish two cases.

Case 1. The preimage-finding query is made to either S_1 or S_2. Since $\mathcal{A}$ consecutively obtains a pair of queries of the form $S_1(x) = y_1$ and $S_2(x) = y_2$, any preimage-finding query to either S_1 or S_2 should be forward. If it is $S_1(x)$ (without loss of generality), then there should be queries $S_2(x) = y$ for some y and $S_3(z) = x \oplus \mathsf{ct}$ for some z that have already been made by $\mathcal{A}$. In order for $S_1(x)$ to be the preimage-finding query, it should be the case that

$$S_3(A_1 \cdot S_1(x) \oplus A_2 \cdot S_2(x) \oplus b) = x \oplus \mathsf{ct}$$

or equivalently,

$$S_1(x) = A_1^{-1} \cdot (z \oplus b \oplus A_2 \cdot y)$$

which happens with probability at most $\frac{1}{2^n - q}$. Therefore, the probability of this case is upper bounded by $\frac{q}{2^n - q}$.

Case 2. The preimage-finding query is made to S_3. In order to address this case, we use the notion of a *wish list*, which was first introduced in [5]. Namely, whenever $\mathcal{A}$ makes a pair of queries $S_1(x) = y_1$ and $S_2(x) = y_2$, the evaluation

$$S_3 : A_1 \cdot y_1 \oplus A_2 \cdot y_2 \oplus b \mapsto x \oplus \mathsf{ct}$$

is included in the wish list $\mathcal{W}$. In order for an S_3-query to complete an evaluation $\mathsf{AIM}(\mathsf{pt}) = \mathsf{ct}$ for any pt, at least one "wish" in $\mathcal{W}$ should be made come true. Each evaluation in $\mathcal{W}$ is obtained with probability at most $\frac{1}{2^n - q}$, and $|\mathcal{W}| \leq q$. Therefore, the probability of this case is upper bounded by $\frac{q}{2^n - q}$.

Overall, we conclude that

$$\mathbf{Adv}^{\mathsf{pre}}_{\mathsf{AIM}}(q) \leq \frac{2q}{2^n - q}.$$

The lesson of this security proof is that one cannot break the one-wayness of AIM in $O(2^n)$ time without exploiting any particular properties of the underlying S-boxes.

In the multi-user setting with u users, $\mathcal{A}$ is given u different target images. The proof of the multi-user security follows the same line of argument as the single-user security proof. The difference is that the probability that each query to either S_1 or S_2 becomes the preimage-finding one is upper bounded by $\frac{uq}{2^n - q}$ and the size of the wish list (in the second case) is upper bounded by uq. Overall, the adversarial preimage finding advantage in the multi-user setting is upper bounded by

$$\frac{2uq}{2^n - q}.$$

It does not mean that AIM provides only the birthday-bound security in the multi-user setting. The straightforward birthday-bound attack is mitigated since AIM is based on a distinct linear layer for every user.

5 The AIMer Signature Scheme

5.1 Specification

The AIMer signature scheme consists of three algorithms: key generation, signing, and verification algorithms. The key generation takes as input a security parameter and outputs a public key $(\mathsf{iv}, \mathsf{ct})$ and a secret key pt such that $\mathsf{ct} = \mathsf{AIM}(\mathsf{iv}, \mathsf{pt})$. The signing algorithm takes as input the pair of secret and public keys $(\mathsf{pt}, (\mathsf{iv}, \mathsf{ct}))$ and a message m and outputs the corresponding signature σ. The verification algorithm takes as input the public key $(\mathsf{iv}, \mathsf{ct})$, a message m and a signature σ and outputs either Accept or Reject. We describe the AIMer signing and verification algorithms in Algorithm 1 and 2, respectively.

In Algorithm 1 and 2, the SHAKE128 (resp. SHAKE256) XOF is used to instantiate hash functions Commit, H_1, H_2 and pseudorandom generators Expand and $\mathsf{ExpandTape}$ in the signature scheme for $\lambda = 128$ (resp. $\lambda \in \{192, 256\}$). $\mathsf{Sample}(\mathsf{tape})$ samples an element from a random tape tape, which is an output of $\mathsf{ExpandTape}$, tracking the current position of the tape.

5.2 Optimization Technique

The BN++ proof system is combined with AIM, yielding the AIMer signature scheme. The AIM function has been designed to fully exploit the optimization techniques of the BN++ proof system using *repeated multipliers* for checking multiplication triples and *locally computed output shares* to reduce the overall signature.

Repeated Multiplier. If multiplication triples share the same multiplier, then the α values in the multiplication checking protocol can be batched as mentioned in Section 2.3. The $\ell + 1$ S-box evaluations in AIM produce the $\ell + 1$ multiplication triples that needs to be verified, reformulated as follows.

$$\mathsf{pt} \cdot t_i = \mathsf{pt}^{2^{e_i}}$$

for $i = 1, \ldots, \ell$, and

$$\mathsf{pt} \cdot \mathsf{Lin}[\mathsf{iv}](t) = (\mathsf{Lin}[\mathsf{iv}](t))^{2^{e_*}} + \mathsf{ct} \cdot \mathsf{Lin}[\mathsf{iv}](t)$$

where t_i, $i = 1, 2, \ldots, \ell$, is the output of the i-th S-box and $t \overset{\text{def}}{=} [t_1 | \ldots | t_\ell]$. Since every multiplication triple shares the same multiplier pt, a single value of α can be included in the signature instead of $\ell + 1$ different values.

Locally Computed Output Shares. For the above multiplication triples, every multiplication output share on the right-hand side can be locally computed without communication between parties. Hence, it is possible to remove the share Δz in the signature. This technique is similar with *multiplications with public output*, suggested in BN++.

For the first ℓ multiplications, each party computes the output as $(\mathsf{pt}^{(i)})^{2^{e_i}}$ based on their input share $\mathsf{pt}^{(i)}$ using linear operations. For the last multiplication output, the output is determined as follows.

$$\begin{cases} (A_{\mathsf{iv}} \cdot t^{(i)} + b_{\mathsf{iv}})^{2^{e_*}} + \mathsf{ct} \cdot (A_{\mathsf{iv}} \cdot t^{(i)} + b_{\mathsf{iv}}) & \text{for } i = 1, \\ (A_{\mathsf{iv}} \cdot t^{(i)})^{2^{e_*}} + \mathsf{ct} \cdot (A_{\mathsf{iv}} \cdot t^{(i)}) & \text{for } i \geq 2, \end{cases}$$

where $t^{(i)} \in \mathbb{F}_2^{\ell n}$ is the output shares of the first ℓ S-boxes for the i-th party: $t^{(i)} = [t_1^{(i)} | \ldots | t_\ell^{(i)}]$.

With the above optimization techniques applied, the signature size is given as

$$6\lambda + \tau \cdot (\lambda \cdot \lceil \log_2(N) \rceil + (\ell + 5) \cdot \lambda).$$

5.3 Security Proof of AIMer

We prove that AIMer is $\mathsf{EUF\text{-}CMA}$-secure (existential unforgeable under adaptive chosen-message attacks [38]). We first prove that AIMer is secure against key-only attack ($\mathsf{EUF\text{-}KO}$) where the adversary is given the public key and no access to signing oracle in Theorem 1. Then, we show AIMer is also $\mathsf{EUF\text{-}CMA}$-secure by showing

Algorithm 1: $\mathsf{Sign}(\mathsf{pt}, (\mathsf{iv}, \mathsf{ct}), m)$ - AIMer signature scheme, signing algorithm.

`// Phase 1: Committing to the seeds and the execution views of the parties.`

Sample a random salt $\mathsf{salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$.

Compute the first ℓ S-boxes' outputs $t_1, \ldots, t_\ell$.

Derive the binary matrix $A_{\mathsf{iv}} \in (\mathbb{F}_2^{n\times n})^\ell$ and the vector $b_{\mathsf{iv}} \in \mathbb{F}_2^n$ from the initial vector iv.

for *each parallel execution* $k \in [\tau]$ **do**

- Sample a root seed : $\mathsf{seed}_k \xleftarrow{\$} \{0,1\}^\lambda$.
- Compute parties' seeds $\mathsf{seed}_k^{(1)}, \ldots, \mathsf{seed}_k^{(N)}$ as leaves of binary tree from seed_k.
- **for** *each party* $i \in [N]$ **do**
  - Commit to seed: $\mathsf{com}_k^{(i)} \leftarrow \mathsf{Commit}(\mathsf{salt}, k, i, \mathsf{seed}_k^{(i)})$.
  - Expand random tape: $\mathsf{tape}_k^{(i)} \leftarrow \mathsf{ExpandTape}(\mathsf{salt}, k, i, \mathsf{seed}_k^{(i)})$.
  - Sample witness share: $\mathsf{pt}_k^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$.
- Compute witness offset and adjust first witness: $\Delta\mathsf{pt}_k \leftarrow \mathsf{pt} - \sum_i \mathsf{pt}_k^{(i)}$, $\mathsf{pt}_k^{(1)} \leftarrow \mathsf{pt}_k^{(1)} + \Delta\mathsf{pt}_k$.
- **for** *each S-box with index* j **do**
  - **if** $j \leq \ell$ **then**
    - For each party i, sample a S-box output: $t_{k,j}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$.
    - Compute output offset and adjust first share: $\Delta t_{k,j} = t_j - \sum_i t_{k,j}^{(i)}$, $t_{k,j}^{(1)} \leftarrow t_{k,j}^{(1)} + \Delta t_{k,j}$.
    - For each party i, set $x_{k,j}^{(i)} = t_{k,j}^{(i)}$ and $z_{k,j}^{(i)} = (\mathsf{pt}_k^{(i)})^{2^{e_j}}$.
  - **if** $j = \ell + 1$ **then**
    - For $i = 1$, set $x_{k,j}^{(i)} = A_{\mathsf{iv}} \cdot t_{k,*}^{(i)} + b_{\mathsf{iv}}$ where $t_{k,*}^{(i)} = [t_{k,1}^{(i)} | \ldots | t_{k,\ell}^{(i)}]$ is the output shares of the first ℓ S-boxes.
    - For each party $i \in [N]\backslash\{1\}$, set $x_{k,j}^{(i)} = A_{\mathsf{iv}} \cdot t_{k,*}^{(i)}$
    - For each party i, set $z_{k,j}^{(i)} = (x_{k,j}^{(i)})^{2^{e_*}} + \mathsf{ct} \cdot x_{k,j}^{(i)}$.
- For each party i, set $a_k^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$.
- Compute $a_k = \sum_{i=1}^N a_k^{(i)}$.
- Set $c_k = a_k \cdot \mathsf{pt}$.
- For each party i, set $c_k^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$.
- Compute offset and adjust first share : $\Delta c_k = c_k - \sum_i c_k^{(i)}$, $c_k^{(1)} \leftarrow c_k^{(1)} + \Delta c_k$.

Set $\sigma_1 \leftarrow \Big(\mathsf{salt}, ((\mathsf{com}_k^{(i)})_{i\in[N]}, \Delta\mathsf{pt}_k, \Delta c_k, (\Delta t_{k,j})_{j\in[\ell]})_{k\in[\tau]}\Big)$.

`// Phase 2: Challenging the checking protocol.`

Compute challenge hash: $h_1 \leftarrow H_1(m, \mathsf{iv}, \mathsf{ct}, \sigma_1)$.

Expand hash: $((\epsilon_{k,j})_{j\in[\ell+1]})_{k\in[\tau]} \leftarrow \mathsf{Expand}(h_1)$ where $\epsilon_{k,j} \in \mathbb{F}_{2^n}$.

`// Phase 3. Commit to the simulation of the checking protocol.`

for *each repetition* k **do**

- Simulate the triple checking protocol as in Section 2.3 for all parties with challenge $\epsilon_{k,j}$. The inputs are $((x_{k,j}^{(i)}, \mathsf{pt}_k^{(i)}, z_{k,j}^{(i)})_{j\in[\ell+1]}, a_k^{(i)}, b_k^{(i)}, c_k^{(i)})$, where $b_k^{(i)} = \mathsf{pt}_k^{(i)}$, and let $\alpha_k^{(i)}$ and $v_k^{(i)}$ be the broadcast values.

Set $\sigma_2 \leftarrow \Big(\mathsf{salt}, ((\alpha_k^{(i)}, v_k^{(i)})_{i\in[N]})_{k\in[\tau]}\Big)$.

`// Phase 4. Challenging the views of the MPC protocol.`

Compute challenge hash: $h_2 \leftarrow H_2(h_1, \sigma_2)$.

Expand hash: $(\bar{i}_k)_{k\in[\tau]} \leftarrow \mathsf{Expand}(h_2)$ where $\bar{i}_k \in [N]$.

`// Phase 5. Opening the views of the MPC and checking protocols.`

for *each repetition* k **do**

- $\mathsf{seeds}_k \leftarrow \{\lceil \log_2(N) \rceil$ nodes to compute $\mathsf{seed}_k^{(i)}$ for $i \in [N]\backslash\{\bar{i}_k\}\}$.

Output $\sigma \leftarrow (\mathsf{salt}, h_1, h_2, (\mathsf{seeds}_k, \mathsf{com}_k^{(\bar{i}_k)}, \Delta\mathsf{pt}_k, \Delta c_k, (\Delta t_{k,j})_{j\in[\ell]}, \alpha_k^{(\bar{i}_k)})_{k\in[\tau]})$.

Algorithm 2: $\mathsf{Verify}((\mathsf{iv},\mathsf{ct}),m,\sigma)$ - AIMer signature scheme, verification algorithm.

Parse σ as $\left(\mathsf{salt}, h_1, h_2, \left(\mathsf{seeds}_k, \mathsf{com}_k^{(\bar{i}_k)}, \Delta\mathsf{pt}_k, \Delta c_k, (\Delta t_{k,j})_{j\in[\ell]}, \alpha_k^{(\bar{i}_k)}\right)_{k\in[\tau]}\right)$.

Derive the binary matrix $A_{\mathsf{iv}} \in (\mathbb{F}_2^{n\times n})^\ell$ and the vector $b_{\mathsf{iv}} \in \mathbb{F}_2^n$ from the initial vector iv.

Expand hashes: $((\epsilon_{k,j})_{j\in[\ell+1]})_{k\in[\tau]} \leftarrow \mathsf{Expand}(h_1)$ and $(\bar{i}_k)_{k\in[\tau]} \leftarrow \mathsf{Expand}(h_2)$.

for *each parallel repetition* $k \in [\tau]$ **do**

- Uses seeds_k to recompute $\mathsf{seed}_k^{(i)}$ for $i \in [N] \setminus \{\bar{i}_k\}$.
- **for** *each party* $i \in [N] \setminus \{\bar{i}_k\}$ **do**
  - Recompute $\mathsf{com}_k^{(i)} \leftarrow \mathsf{Commit}(\mathsf{salt}, k, i, \mathsf{seed}_k^{(i)})$,
  - $\mathsf{tape}_k^{(i)} \leftarrow \mathsf{ExpandTape}(\mathsf{salt}, k, i, \mathsf{seed}_k^{(i)})$ and
  - $\mathsf{pt}_k^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$.
  - **if** $i = 1$ **then**
    - Adjust first share: $\mathsf{pt}_k^{(i)} \leftarrow \mathsf{pt}_k^{(i)} + \Delta\mathsf{pt}_k$
  - **for** *each S-box with index* j **do**
    - **if** $j \le \ell$ **then**
      - Sample a S-box output: $t_{k,j}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$.
      - **if** $i = 1$ **then**
        - Adjust first share: $t_{k,j}^{(1)} \leftarrow t_{k,j}^{(1)} + \Delta t_{k,j}$.
      - Set $x_{k,j}^{(i)} = t_{k,j}^{(i)}$ and $z_{k,j}^{(i)} = (\mathsf{pt}_k^{(i)})^{2^{e_j}}$.
    - **if** $j = \ell + 1$ **then**
      - **if** $i = 1$ **then**
        - Set $x_{k,j}^{(i)} = A_{\mathsf{iv}} \cdot t_{k,*}^{(i)} + b_{\mathsf{iv}}$ where $t_{k,*}^{(i)} = [t_{k,1}^{(i)}|\dots|t_{k,\ell}^{(i)}]$ is the output shares of the first ℓ S-boxes.
      - **else**
        - Set $x_{k,j}^{(i)} = A_{\mathsf{iv}} \cdot t_{k,*}^{(i)}$.
      - Set $z_{k,j}^{(i)} = (x_{k,j}^{(i)})^{2^{e_*}} + \mathsf{ct} \cdot x_{k,j}^{(i)}$.
  - Set $a_k^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$ and $c_k^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$.
  - **if** $i = 1$ **then**
    - Adjust first share $c_k^{(i)} \leftarrow c_k^{(i)} + \Delta c_k$.

Set $\sigma_1 \leftarrow \left(\mathsf{salt}, \left((\mathsf{com}_k^{(i)})_{i\in[N]}, \Delta\mathsf{pt}_k, \Delta c_k, (\Delta t_{k,j})_{j\in[\ell]}\right)_{k\in[\tau]}\right)$.

Set $h_1' \leftarrow H_1(m, \mathsf{iv}, \mathsf{ct}, \sigma_1)$.

for *each parallel execution* $k \in [\tau]$ **do**

- **for** *each party* $i \in [N] \setminus \{\bar{i}_k\}$ **do**
  - Simulate the triple checking protocol as defined in Section 2.3 for all parties with challenge $\epsilon_{k,j}$. The inputs are $((x_{k,j}^{(i)}, \mathsf{pt}_k^{(i)}, z_{k,j}^{(i)})_{j\in[\ell+1]}, a_k^{(i)}, b_k^{(i)}, c_k^{(i)})$, where $b_k^{(i)} = \mathsf{pt}_k^{(i)}$, and let $\alpha_k^{(i)}$ and $v_k^{(i)}$ be the broadcast values.

Compute $v_k^{(\bar{i}_k)} = 0 - \sum_{i\neq\bar{i}_k} v_k^{(i)}$.

Set $\sigma_2 \leftarrow \left(\mathsf{salt}, ((\alpha_k^{(i)}, v_k^{(i)})_{i\in[N]})_{k\in[\tau]}\right)$

Set $h_2' = H_2(h_1, \sigma_2)$.

Output Accept if $h_1 = h_1'$ and $h_2 = h_2'$.

Otherwise, output Reject.

that the signature can be simulated without using the secret key in Theorem 2. We assume that all adversaries are probabilistic polynomial time(in λ) algorithms. Many parts of the proofs are identical to [48], and we give full credit to it.

Theorem 1 (EUF-KO Security of AIMer). *Let Commit, H_1 and H_2 be modeled as random oracles, Expand be modeled as a random function, and let (N, τ, λ) be parameters of the AIMer signature scheme. Let $\mathcal{A}$ be an adversary against the EUF-KO security of AIMer that makes a total of Q random oracle queries. Assuming that KeyGen is an ϵ_{OWF}-hard one-way function, then $\mathcal{A}$'s advantage in the EUF-KO game is*

$$\epsilon_{\mathsf{KO}} \leq \epsilon_{\mathsf{OWF}} + \frac{(\tau N + 1)Q^2}{2^{2\lambda}} + Pr[X + Y = \tau], \tag{3}$$

where $Pr[X + Y = \tau]$ is as described in the proof.

Proof. We build an algorithm $\mathcal{B}$ to retrieve a pre-image for the key generation one-way function AIM using the EUF-KO adversary $\mathcal{A}$. Let the random oracles (RO) H_{c} (shorthand for Commit), H_1 and H_2, and the respective RO query lists $\mathcal{Q}_{\mathsf{c}}$, $\mathcal{Q}_1$ and $\mathcal{Q}_2$. We expand the output lengths of random oracles H_1 and H_2 instead of making the calls to $\mathsf{Expand}()$ in our analysis, since Expand is a random function used to expand outputs from H_1 and H_2.

Algorithm $\mathcal{B}$ takes the one-way function value $(\mathsf{iv}, \mathsf{ct})$ as an input, and forwards it to $\mathcal{A}$ as a AIMer public key for the EUF-KO game. $\mathcal{B}$ manages a set Bad to keep track of all the outputs of three random oracles and two tables to maintain the values derived from $\mathcal{A}$'s RO queries as follows :

- $\mathcal{T}_{\mathsf{sh}}$ to store secret shares of the parties, and
- $\mathcal{T}_{\mathsf{in}}$ to store inputs to the MPC protocol.

We also program the random oracles for $\mathcal{A}$ as follows :

- H_{c} : When $\mathcal{A}$ queries the commitment random oracle, $\mathcal{B}$ records the query to find out which commitment corresponds to which seed. See Algorithm 3.
- H_1 : When $\mathcal{A}$ commits to seeds and sends the offsets for the secret key pt and the multiplication triples, $\mathcal{B}$ check the query list $\mathcal{Q}_{\mathsf{c}}$ to see if the commitments were output by its simulation of H_{c}. If $\mathcal{B}$ finds matching results for all i's in some repetition k, then it can recover pt. See Algorithm 4.
- H_2 : See Algorithm 5.

When $\mathcal{A}$ terminates, $\mathcal{B}$ checks whether there is $\mathsf{pt}_k \in \mathcal{T}_{\mathsf{in}}$ satisfying $\mathsf{AIM}(\mathsf{iv}, \mathsf{pt}_k) = \mathsf{ct}$. If $\mathcal{B}$ finds a match pt_k, $\mathcal{B}$ outputs it as a pre-image for the AIM, otherwise $\mathcal{B}$ outputs $\perp$.

Algorithm 3: $H_{\mathsf{c}}(q_{\mathsf{c}} = (\mathsf{salt}, k, i, \mathsf{seed})$:

```
r ←$ {0,1}^{2λ}.
if r ∈ Bad then
    abort.
r → Bad.
(q_c, r) → Q_c.
Return r.
```

Given the algorithm of $\mathcal{B}$ as above, the probability that $\mathcal{A}$ wins is bounded as below.

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins} \wedge \mathcal{B} \text{ aborts}] + \Pr[\mathcal{A} \text{ wins} \wedge \mathcal{B} \text{ outputs } \perp] + \Pr[\mathcal{A} \text{ wins} \wedge \mathcal{B} \text{ outputs } \mathsf{pt}] \\ &\leq \Pr[\mathcal{B} \text{ aborts}] + \Pr[\mathcal{A} \text{ wins} \mid \mathcal{B} \text{ outputs } \perp] + \Pr[\mathcal{B} \text{ outputs } \mathsf{pt}]\end{aligned} \tag{4}$$

Algorithm 4: $H_1(q_1 = \sigma_1)$:

Parse σ_1 as $\Big(\mathsf{salt}, ((\mathsf{com}_k^{(i)})_{i\in[N]}, \Delta\mathsf{pt}_k, \Delta c_k, (\Delta t_{k,j})_{j\in[\ell]})_{k\in[\tau]}\Big)$.

for $k \in [\tau], i \in [N]$ **do**
  $\mathsf{com}_k^{(i)} \to \mathsf{Bad}$.

`// If the committed seed is known for some` k, i`, then` $\mathcal{B}$ `records the shares of the secret key and the multiplication output values for that party, derived from that seed and the offsets in` σ_1

for $k \in [\tau], i \in [N] : \exists \mathsf{seed}_k^{(i)} : ((\mathsf{salt}, k, i, \mathsf{seed}_k^{(i)}), \mathsf{com}_k^{(i)}) \in \mathcal{Q}_\mathsf{c}$ **do**
  $\mathsf{pt}_k^{(i)}, a_k^{(i)}, c_k^{(i)}, (t_{k,j}^{(i)})_{j\in[\ell]} \leftarrow \mathsf{ExpandTape}(\mathsf{salt}, k, i, \mathsf{seed}_k^{(i)})$.
  if $i = 1$ **then**
    $\mathsf{pt}_k^{(i)} \leftarrow \mathsf{pt}_k^{(i)} + \Delta\mathsf{pt}_k, c_k^{(i)} \leftarrow c_k^{(i)} + \Delta c_k$ and $(t_{k,j}^{(i)} \leftarrow t_{k,j}^{(i)} + \Delta t_{k,j})_{j\in[\ell]}$
  $(\mathsf{pt}_k^{(i)}, c_k^{(i)}, (t_{k,j}^{(i)}))_{j\in[\ell]} \to \mathcal{T}_\mathsf{sh}[q_1, k, i]$

`// If the shares of the various elements are known for every party in that repetition,` $\mathcal{B}$ `records the resulting secret key, multiplication inputs and S-box outputs`

for *each* $k : \forall i, \mathcal{T}_\mathsf{sh}[q_1, k, i] \neq \emptyset$ **do**
  $\mathsf{pt}_k \leftarrow \sum_i \mathsf{pt}_k^{(i)}, c_k \leftarrow \sum_i c_k^{(i)}, a_k \leftarrow \sum_i a_k^{(i)}, (t_{k,j} \leftarrow \sum_i t_{k,j}^{(i)})_{j\in[\ell]}$ and $t_{k,\ell+1}^{(i)} = A_\mathsf{iv} \cdot t_{k,*}^{(i)} + b_\mathsf{iv}$ where $t_{k,*}^{(i)} = [t_{k,1}^{(i)}|\dots|t_{k,\ell}^{(i)}]$ is the output shares of the first ℓ S-boxes.
  Derive the binary matrix $A_\mathsf{iv} \in (\mathbb{F}_2^{n\times n})^\ell$ and the vector $b_\mathsf{iv} \in \mathbb{F}_2^n$ from the initial vector iv.
  for $j \in [\ell]$ **do**
    Set $x_{k,j} = t_{k,j}$ and $z_{k,j} = (\mathsf{pt}_k)^{2^{e_j}}$.
  for $j = \ell + 1$ **do**
    Set $x_{k,j} = A_\mathsf{iv} \cdot t_{k,*} + b_\mathsf{iv}$ where $t_{k,*} = [t_{k,1}|\dots|t_{k,\ell}]$ is the output shares of the first ℓ S-boxes and $z_{k,j} = (x_{k,j})^{2^{e_*}} + \mathsf{ct} \cdot x_{k,j}$.
  $(\mathsf{pt}_k) \to \mathcal{T}_\mathsf{in}[q_1, k]$.

$r \xleftarrow{\$} \{0,1\}^{2\lambda}$.
if $r \in \mathsf{Bad}$ **then**
  abort.
$r \to \mathsf{Bad}$.
$(q_1, r) \to \mathcal{Q}_1$.

`// Compute the multiplication check protocol values.`

$(\epsilon_{k,j})_{j\in[\ell+1]} \leftarrow \mathsf{Expand}(r)$.
for *each* $k : \mathcal{T}_\mathsf{in}[q_1, e] \neq \emptyset$ **do**
  $\alpha_k = \sum_{j\in[\ell+1]} \epsilon_j \cdot x_j + a_k$.
  $v_k = \sum_{j\in[\ell+1]} \epsilon_j \cdot z_{k,j} - \alpha_k \cdot \mathsf{pt} + c_k$.
Return r.

Algorithm 5: $H_2(q_2 = (h_1, \sigma_2))$:

$h_1 \to \mathsf{Bad}$.
$r \xleftarrow{\$} \{0,1\}^{2^\lambda}$.
if $r \in \mathsf{Bad}$ **then**
  abort.
$r \to \mathsf{Bad}$.
$(q_2, r) \to \mathcal{Q}_2$.
Return r.

Let Q_{c}, Q_1 and Q_2 denote the number of queries made by $\mathcal{A}$ to random oracles H_{c}, H_1, H_2, respectively. Then we can bound the probability that $\mathcal{B}$ aborts (The first term on the RHS of (4)) as follows.

$$\begin{aligned}\Pr[\mathcal{B}\text{ aborts}] &= (\#\text{times an } r \text{ is sampled}) \cdot \Pr[\mathcal{B} \text{ aborts at that sample}] \\ &\leq (Q_{\mathsf{c}} + Q_1 + Q_2) \cdot \frac{\max|\mathsf{Bad}|}{2^{2\lambda}} = (Q_{\mathsf{c}} + Q_1 + Q_2) \cdot \frac{Q_{\mathsf{c}} + (\tau N + 1)Q_1 + 2Q_2}{2^{2\lambda}} \\ &\leq \frac{(\tau N+1)(Q_{\mathsf{c}} + Q_1 + Q_2)^2}{2^{2\lambda}} = \frac{(\tau N+1)Q^2}{2^{2\lambda}}, \end{aligned} \tag{5}$$

where $Q = Q_{\mathsf{c}} + Q_1 + Q_2$.

We now analyze $\Pr[\mathcal{A} \text{ wins} \mid \mathcal{B} \text{ outputs } \perp]$ (The second term on the RHS of (4)), which means pt corresponding to $(\mathsf{iv}, \mathsf{ct})$ is not found. We parse it into two cases, which correspond to cheating in the first round and the second round.

Cheating in the first round. Let $q_1 \in \mathcal{Q}_1$ be the query to H_1, and $h_1 = ((\epsilon_{k,j})_{j\in[\ell+1]})_{k\in[\tau]}$ be its corresponding answer. We collect the set of indices $k \in [\tau]$ representing "good executions" such that $\mathcal{T}_{\mathsf{in}}[q_1, k]$ is nonempty and $v_k = 0$, say $G_1(q_1, h_1)$. For $k \in G_1(q_1, h_1)$, the challenges $(\epsilon_{k,j})_{j\in[\ell+1]}$ were sampled such that the multiplication check presented in the Section (2.3) is passed in that repetition. By Lemma (1), since h_1 is sampled uniformly at random, this happens with probability at most $1/2^{\lambda}$.

Lemma 1. *If the secret-shared input $(x_j, y, z_j)_{j\in[C]}$ contains an incorrect multiplication triple, or if the shares of $((a_j, y)_{j\in[C]}, c)$ form an incorrect dot product, then the parties output `Accept` in the sub-protocol with probability at most $1/2^{\lambda}$.*

Proof. Let $\Delta_{z_j} = z_j - x_j \cdot y$ and $\Delta_c = -\sum_{j\in[C]} a_j \cdot y + c$. Then

$$\begin{aligned} v &= \sum_{j\in[C]} \epsilon_j \cdot z_j - \alpha \cdot y + c \\ &= \sum_{j\in[C]} \epsilon_j \cdot z_j - \sum_{j\in[C]} \epsilon_j \cdot x_j \cdot y - \sum_{j\in[C]} a_j \cdot y + c \\ &= \sum_{j\in[C]} \epsilon_j \cdot (z_j - x_j \cdot y) - \sum_{j\in[C]} a_j \cdot y + c \\ &= \sum_{j\in[C]} \epsilon_j \cdot \Delta_{z_j} + \Delta_c \end{aligned}$$

Define a multivariate polynomial

$$Q(X_1, \ldots, X_C) = X_1 \cdot \Delta_{z_1} + \cdots + X_C \cdot \Delta_{z_C} + \Delta_c$$

over $\mathbb{F}_{2^\lambda}$ and note that $v = 0$ iff $Q(\epsilon_1, \ldots, \epsilon_C) = 0$. In the case of a cheating prover, Q is nonzero, and by the multivariate version of the Schwartz-Zippel lemma, the probability that $Q(\epsilon_1, \ldots, \epsilon_C) = 0$ is at most $1/2^{\lambda}$, since Q has total degree 1 and $(\epsilon_1, \ldots, \epsilon_C)$ is chosen uniformly at random. $\square$

Given $\mathcal{B}$ outputs $\perp$, the number of elements $\#G_1(q_1, h_1)|_{\perp} \sim X_{q_1}$ where $X_{q_1} = \mathcal{B}(\tau, p_1)$, where $\mathcal{B}(\tau, p_1)$ is the binomial distribution with τ events, each with success probability $p_1 = 1/2^{\lambda}$. We select the query-response pair $(q_{\mathsf{best}_1}, h_{\mathsf{best}_1})$ such that $\#G_1(q_1, h_1)$ is the maximum. Then, the following holds.

$$\#G_1(q_{\mathsf{best}_1}, h_{\mathsf{best}_1})|_{\perp} \sim X = \max_{q_1 \in \mathcal{Q}_1} \{X_{q_1}\}.$$

Cheating in the second round. Let $q_2 = (h_1, \sigma_2)$ be the query to H_2. Note that q_2 can only be used in the winning $\mathsf{EUF\text{-}KO}$ game when the corresponding $(q_1, h_1) \in \mathcal{Q}_1$ exists. For the bad repetition $k \in [\tau] \backslash G_1(q_1, h_1)$, either $\mathcal{T}_{\mathsf{in}}[q_1, k]$ is empty (which means verification fails so that $\mathcal{A}$ loses) or $v_k \neq 0$ but the verification passes. Hence, it should be the case that one of the N parties cheated. Since $h_2 = (\bar{i}_k)_{k\in[\tau]} \in [N]^\tau$ is distributed uniformly at random, the probability that one of the N parties have cheated for all bad executions k is

$$\left(\frac{1}{N}\right)^{\tau-\#G_1(q_1,h_1)} \leq \left(\frac{1}{N}\right)^{\tau-\#G_1(q_{\mathsf{best}_1},h_{\mathsf{best}_1})}.$$

To sum up, we can analyze the probability that $\mathcal{A}$ wins conditioning on $\mathcal{B}$ outputting $\perp$ is

$$\Pr[\mathcal{A} \text{ wins} \mid \mathcal{B} \text{ outputs } \perp] \leq \Pr[X + Y = \tau], \tag{6}$$

where X is as before, and $Y = \max_{q_2\in\mathcal{Q}_2}\{Y_{q_2}\}$ where Y_{q_2} variables are independently and identically distributed as $\mathcal{B}(\tau - X, 1/N)$.

Finally, combining (4), (5) and (6) all together, we obtain the following.

$$\Pr[\mathcal{A} \text{ wins}] \leq \frac{(\tau N + 1)\cdot Q^2}{2^{2\lambda}} + \Pr[X + Y = \tau] + \Pr[\mathcal{B} \text{ outputs } \mathsf{pt}],$$

where $Q = Q_{\mathsf{c}} + Q_1 + Q_2$, X and Y are defined as above. Setting KeyGen as an ϵ_{OWF}-secure OWF, we achieve (3) as desired. □

In our $\mathsf{EUF\text{-}CMA}$ theorem, we assume $\mathsf{ExpandTape}$ is a secure pseudorandom generator (PRG). This implies the unopened seed is hidden to a computationally bounded adversary, and it holds in the random oracle model as proven in [69].

Theorem 2 (EUF-CMA Security of AIMer). *The* AIMer *signature scheme is EUF-CMA-secure, assuming that* $\mathsf{Commit}, H_1, H_2$ *and* Expand *are modeled as random oracles,* $\mathsf{ExpandTape}$ *is a secure PRG, the seed tree construction is computationally hiding, the* (N, τ, λ) *parameters are appropriately chosen, and that the key generation is a secure one-way function.*

Proof. Let $\mathcal{A}$ be an $\mathsf{EUF\text{-}CMA}$ adversary for given $(\mathsf{iv}, \mathsf{ct})$. Let G_0 be the original $\mathsf{EUF\text{-}CMA}$ game and $\mathcal{B}$ be an $\mathsf{EUF\text{-}KO}$ adversary that simulates the $\mathsf{EUF\text{-}CMA}$ game to $\mathcal{A}$. When $\mathcal{A}$ queries random oracles, $\mathcal{B}$ checks if the query has been recorded so that it sends back the recorded answer if so, and otherwise, it records a pair of the query and the result it retrieves and forwards the answer to $\mathcal{A}$.

- G_0: $\mathcal{B}$ knows the secret key pt for the forwarded public key $(\mathsf{iv}, \mathsf{ct})$;
- G_1: $\mathcal{B}$ replaces real signatures with simulated ones no longer using pt. $\mathcal{B}$ uses the $\mathsf{EUF\text{-}KO}$ challenge $\mathsf{pt}^*(\neq \mathsf{pt})$ in its simulation with $\mathcal{A}$.

We define G_0(resp. G_1) be a probability that $\mathcal{A}$ succeeds in Game G_0(resp. G_1). The advantage of $\mathcal{A}$ is $\epsilon_{\mathrm{CMA}} = \mathsf{G}_0 = (\mathsf{G}_0 - \mathsf{G}_1) + \mathsf{G}_1$.

Hybrid Arguments. We bound $(\mathsf{G}_0 - \mathsf{G}_1)$ by defining a sequence of games to connect G_0 and G_1 and constructing hybrid arguments. Upon receiving a signing query from $\mathcal{A}$, $\mathcal{B}$ simulates a signature using randomly sampled pt^*, selects one of the party $\mathcal{P}_{i^*}$ for cheating in the verification and broadcast of the output shares $v_k^{(i)}$ so that it passes multiplication checking protocols. We show that the signature values are sampled from a distribution that is computationally indistinguishable from that of real signatures while it is sampled independently of pt^*. $\mathcal{B}$ sets the random oracle H_1 and H_2 to output uniform random challenges $((\epsilon_{k,j})_{j\in[\ell+1]})_{k\in[\tau]}$ and $(\bar{i}_k)_{k\in[\tau]}$, respectively. The definition of subgames and hybrid arguments are the same as in the $\mathsf{EUF\text{-}CMA}$ proof in [48] (Theorem 7 in Appendix) except that we do not have to cheat on the broadcast of party $P_{\bar{i}_k}$'s output share $\mathsf{ct}_k^{(\bar{i}_k)}$, since the output broadcast is implicit in our protocol.

1. $\mathcal{B}$ knows pt so that it computes signatures honestly. $\mathcal{B}$ aborts if the salt that it samples in Phase 1 has already been queried.

2. $\mathcal{B}$ randomly choose h_2 and programs the random oracle H_2 to output h_2 when queried in Phase 4. The unopened parties $(\bar{i}_k)_{k\in[\tau]}$ is derived by expanding h_2. Simulation is aborted if the queries to H_2 have been made previously.
3. $\mathcal{B}$ replaces the seed of the unopened parties $\mathsf{seed}_k^{(\bar{i}_k)}$ in the binary tree by a random element.
4. $\mathcal{B}$ replaces the outputs of $\mathsf{ExpandTape}(\mathsf{salt}, k, \bar{i}_k, \mathsf{seed}_k^{(\bar{i}_k)}))$ by random elements. Since we assume that $\mathsf{ExpandTape}$ is a secure PRG, this is indistinguishable from the previous subgame.
5. $\mathcal{B}$ replaces the commitments of the unopened parties $\mathsf{com}_k^{(\bar{i}_k)}$ with random values. $\mathcal{B}$ aborts if the replaced value is collide with $\mathsf{Commit}(x)$ where x is queried by $\mathcal{A}$.
6. $\mathcal{B}$ randomly choose h_1 and programs the random oracle H_1 to output h_1 in Phase 2. The checking values $((\epsilon_{k,j})_{j\in[\ell+1]})_{k\in[\tau]}$ is derived by expanding h_1. Simulation is aborted if the queries to H_1 have been made previously.
7. $\mathcal{B}$ replaces $\alpha_k^{(\bar{i}_k)}$ with a uniformly random value and sets $v_k^{(\bar{i}_k)} \leftarrow -\sum_{i\neq\bar{i}_k} v_k^{(i)}$. Note that if the multiplication triple is wrong, then $v_k^{(\bar{i}_k)} \leftarrow -\sum_{i\neq\bar{i}_k} v_k^{(i)}$ is different from an honest value derived from legitimate calculation. However $(\bar{i}_k)$ is an unopened and the multiplication check is still passed.
8. $\mathcal{B}$ sets $(\Delta t_{k,j})_{j\in[\ell]}$ and Δc_k to random values in Phase 1.
9. $\mathcal{B}$ replaces the real pt by a random key pt^* as $\mathsf{pt}_k^{(\bar{i}_k)}$ is independent from the seeds $\mathcal{A}$ observes. The distribution of $\Delta\mathsf{pt}_k$ is not changed and $\mathcal{A}$ has no information about pt^*.

If the algorithm is not aborted, above games are all indistinguishable to each other, which results the simulated signatures in G_1 and the real signatures in G_0 are indistinguishable. The abort happens when:

- A_1 : The salt it sampled has been used before.
- A_2 : The committed value it replaces is queried.
- A_3 : Queries to H_1 and H_2 have been made previously.

Let Q_{salt} be the number of different salts queried during the game (by both $\mathcal{A}$ and $\mathcal{B}$), Q_{c} be the number of queries made to Commit by $\mathcal{A}$ including those made during signature queries and Q_1(resp. Q_2) be the number of queries made to H_1(resp. H_2) during the game. Then the probability of each event occurring is bounded by $\Pr[A_1] \leq Q_{\mathsf{salt}}/2^{2\lambda}$, $\Pr[A_2] \leq Q_{\mathsf{c}}/2^{\lambda}$, and $\Pr[A_3] \leq Q_1/2^{2\lambda} + Q_2/2^{2\lambda}$.

Therefore

$$\begin{aligned}\Pr[\mathcal{B} \text{ aborts}] &\leq Q_{\mathsf{salt}}/2^{2\lambda} + Q_{\mathsf{c}}/2^{\lambda} + Q_1/2^{2\lambda} + Q_2/2^{2\lambda} \\ &= (Q_{\mathsf{salt}} + Q_1 + Q_2)/2^{2\lambda} + Q_{\mathsf{c}}/2^{\lambda} \\ &\leq (Q_1 + Q_2)/2^{2\lambda-1} + Q_{\mathsf{c}}/2^{\lambda} \quad (\because Q_{\mathsf{salt}} \leq Q_1 + Q_2) \\ &\leq Q/2^{\lambda} \quad (\text{where } Q = Q_1 + Q_2 + Q_{\mathsf{c}})\end{aligned}$$

and

$$\begin{aligned}\mathsf{G}_0 - \mathsf{G}_1 \leq & Q_s \cdot (\tau \cdot \epsilon_{\text{PRG}} + \epsilon_{\text{TREE}} + \Pr[\mathcal{B} \text{ aborts}]) \\ \leq & Q_s \cdot (\tau \cdot \epsilon_{\text{PRG}} + \epsilon_{\text{TREE}} + Q/2^{\lambda}),\end{aligned}$$

where Q_s be the total number of signature queries.

Bounding G_1. If $\mathcal{A}$ outputs a valid signature in G_1, then $\mathcal{B}$ outputs a valid signature in the $\mathsf{EUF\text{-}KO}$ game. Finally we have

$$\mathsf{G}_1 \leq \epsilon_{\text{KO}} \leq \epsilon_{\text{OWF}} + \frac{(\tau N + 1)Q^2}{2^{2\lambda}} + \Pr[X + Y = \tau],$$

where the bound on the advantage ϵ_{KO} of a $\mathsf{EUF\text{-}KO}$ attacker follows from Theorem 1. We conclude that

$$\epsilon_{\mathrm{CMA}} \le \epsilon_{\mathrm{OWF}} + \frac{(\tau N+1)Q^2}{2^{2\lambda}} + \Pr[X+Y=\tau] + Q_s \cdot (\tau \cdot \epsilon_{\mathrm{PRG}} + \epsilon_{\mathrm{TREE}} + Q/2^{\lambda}).$$

Assuming that ExpandTape is a secure PRG that is ϵ_{PRG}-close to uniform, that the seed tree construction is hiding (so that ϵ_{TREE} is negligible), that key generation is a one-way function and that parameters (N, τ, λ) are appropriately chosen implies that ϵ_{CMA} is negligible in λ. □

6 Performance Evaluation of Reference Implementations

6.1 Environment.

The benchmark is performed in Intel Xeon E5-1650 v3 @ 3.50GHz CPU with 128GB memory with GNU C 7.5.0 compiler on the Ubuntu 18.04.1 operating system. For the instantiation of the XOF, we use SHAKE in `XKCP` library[4]. We use SHAKE128 for AIMer-I, and SHAKE256 for AIMer-III and AIMer-V. All the implementations used in the experiments are compiled at the `-O3` optimization level. Our experiments are measured by the average clock cycles executed 10^3 times. For a fair comparison, we measure the execution time for signature scheme on the same CPU using the `taskset` command with Hyper-Threading and Turbo Boost features disabled.

6.2 Performance of AIMer.

In this section, we provide the performance of C standalone implementation of the AIMer signature scheme in Table 5. The performance of signature scheme is represented as milliseconds (ms) and CPU clock cycles (cc) and the size of public key, secret key, and signature is represented as bytes (B). For the implementation results of AVX2 version and comparison to other signature schemes, we refer to Appendix E.

Scheme	N	τ	Keygen (ms)	Keygen (cc)	Sign (ms)	Sign (cc)	Verify (ms)	Verify (cc)	*pk* Size (B)	*sk* Size (B)	*sig* Size (B)
AIMer-I	16	33	0.06	210 332	2.26	7 927 283	2.17	7 587 588	32	16	5 904
	57	23	0.06	216 316	5.45	19 073 690	5.36	18 774 645	32	16	4 880
	256	17	0.06	215 055	17.91	62 701 819	17.86	62 505 586	32	16	4 176
	1615	13	0.06	214 695	86.62	303 154 431	85.99	300 970 342	32	16	3 840
AIMer-III	16	49	0.13	438 963	4.75	16 618 659	4.58	16 015 373	48	24	13 080
	64	33	0.13	438 550	12.31	43 099 886	12.24	42 827 096	48	24	10 440
	256	25	0.13	439 356	37.24	130 342 111	36.76	128 664 878	48	24	9 144
	1621	19	0.13	439 751	178.72	625 520 404	174.64	611 255 371	48	24	8 352
AIMer-V	16	65	0.31	1 078 521	10.88	38 085 070	10.50	36 763 832	64	32	25 152
	62	44	0.31	1 076 379	27.16	95 067 716	26.56	92 970 531	64	32	19 904
	256	33	0.31	1 080 828	80.77	282 682 033	80.50	281 757 450	64	32	17 088
	1623	25	0.31	1 081 808	395.65	1 384 758 687	391.87	1 371 544 537	64	32	15 392

Table 5: Performance of AIMer reference implementation for various parameter sets.

[4] `https://github.com/XKCP/XKCP`

References

1. Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In *ASIACRYPT 2016*, pages 191–219. Springer, 2016.
2. Martin R Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In *EUROCRYPT 2015*, pages 430–454. Springer, 2015.
3. Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols. *IACR Transactions on Symmetric Cryptology*, 2020(3), Sep. 2020.
4. Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita. Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms — Design and Analysis. In *SAC 2001*, pages 39–56. Springer, 2001.
5. Frederik Armknecht, Ewan Fleischmann, Matthias Krause, Jooyoung Lee, Martijn Stam, and John Steinberger. The preimage security of double-block-length compression functions. In *ASIACRYPT 2011*, pages 233–251. Springer, 2011.
6. Subhadeep Banik, Khashayar Barooti, F. Betül Durak, and Serge Vaudenay. Cryptanalysis of lowmc instances using single plaintext/ciphertext pair. *IACR Transactions on Symmetric Cryptology*, 2020(4):130–146, Dec. 2020.
7. Subhadeep Banik, Khashayar Barooti, Serge Vaudenay, and Hailun Yan. New Attacks on LowMC Instances with a Single Plaintext/Ciphertext Pair. In *ASIACRYPT 2021*, pages 303–331. Springer, 2021.
8. Magali Bardet, Jean-Charles Faugere, and Bruno Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proceedings of the International Conference on Polynomial System Solving*, pages 71–74, 2004.
9. Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer. On the complexity of solving quadratic Boolean systems. *Journal of Complexity*, 29(1):53–75, 2013.
10. Carsten Baum and Ariel Nof. Concretely-Efficient Zero-Knowledge Arguments for Arithmetic Circuits and Their Application to Lattice-Based Cryptography. In *PKC 2020*, pages 495–526. Springer, 2020.
11. Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and fast signatures from AES. In *PKC 2021*, pages 266–297. Springer, 2021.
12. Daniel J. Bernstein and Bo-Yin Yang. Asymptotically Faster Quantum Algorithms to Solve Multivariate Quadratic Equations. In *PQCrypto 2018*, pages 487–506. Springer, 2018.
13. Ward Beullens. Breaking rainbow takes a weekend on a laptop. Cryptology ePrint Archive, Paper 2022/214, 2022. `https://eprint.iacr.org/2022/214`.
14. Xavier Bonnetain, Akinori Hosoyamada, María Naya-Plasencia, Yu Sasaki, and André Schrottenloher. Quantum Attacks Without Superposition Queries: The Offline Simon's Algorithm. In *ASIACRYPT 2019*, pages 552–583. Springer, 2019.
15. Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).
16. Wouter Castryck and Thomas Decru. An efficient key recovery attack on sidh (preliminary version). Cryptology ePrint Archive, Paper 2022/975, 2022. `https://eprint.iacr.org/2022/975`.
17. Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *ACM CCS 2017*, pages 1825–1842, 2017.
18. Yu-Ao Chen and Xiao-Shan Gao. Quantum Algorithm for Boolean Equation Solving and Quantum Algebraic Attack on Cryptosystems. *Journal of Systems Science and Complexity*, 35(1):373–412, Feb 2022.
19. Jung Hee Cheon and Dong Hoon Lee. Resistance of S-Boxes against Algebraic Attacks. In *FSE 2004*, pages 83–93. Springer, 2004.
20. Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *EUROCRYPT 2000*, pages 392–407. Springer, 2000.
21. Nicolas T. Courtois, Blandine Debraize, and Eric Garrido. On Exact Algebraic [Non-]Immunity of S-Boxes Based on Power Functions. In *ACISP 2006*, pages 76–86. Springer, 2006.
22. Joan Daemen. Cipher and hash function design strategies based on linear and differential cryptanalysis. *Doctoral Dissertation, KU Leuven*, 1995.
23. Joan Daemen and Vincent Rijmen. *The Design of Rijndael*, volume 2. Springer, 2002.
24. Cyprien Delpech de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P Smart. BBQ: Using AES in picnic signatures. In *SAC 2019*, pages 669–692. Springer, 2019.

25. Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient Zero-Knowledge MPCitH-Based Arguments. In *ACM CCS 2021*, page 3022–3036. Association for Computing Machinery, 2021.
26. Jintai Ding, Vlad Gheorghiu, András Gilyén, Sean Hallgren, and Jianqiang Li. Limitations of the Macaulay matrix approach for using the HHL algorithm to solve multivariate polynomial systems. arXiv 2111.00405, 2021. `https://arxiv.org/abs/2111.00405`.
27. Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *ACNS 2005*, pages 164–175. Springer, 2005.
28. Jintai Ding and Dieter Schmidt. *Solving Degree and Degree of Regularity for Polynomial Systems over a Finite Fields*, pages 34–49. Springer, 2013.
29. Itai Dinur, Yunwen Liu, Willi Meier, and Qingju Wang. Optimized Interpolation Attacks on LowMC. In *ASIACRYPT 2015*, volume 9453, pages 535–560. Springer, 2015.
30. Hans Dobbertin. Almost Perfect Nonlinear Power Functions on GF(2n): The Niho Case. *Information and Computation*, 151(1):57–72, 1999.
31. Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Higher-Order Cryptanalysis of LowMC. In *ICISC 2015*, volume 9558, pages 87–101. Springer, 2016.
32. Christoph Dobraunig, Daniel Kales, Christian Rechberger, Markus Schofnegger, and Greg Zaverucha. Shorter Signatures Based on Tailor-Made Minimalist Symmetric-Key Crypto. In *ACM CCS 2022*, November 2022.
33. Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In *CRYPTO 2020*, page 602–631. Springer, 2020.
34. Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *Journal of Cryptology*, 10(3):151–161, Jun 1997.
35. Jean-Charles Faugère, Kelsey Horan, Delaram Kahrobaei, Marc Kaplan, Elham Kashefi, and Ludovic Perret. Fast Quantum Algorithm for Solving Multivariate Quadratic Equations. Cryptology ePrint Archive, Paper 2017/1236, 2017. `https://eprint.iacr.org/2017/1236`.
36. Ralf Fröberg. An Inequality for Hilbert Series of Graded Algebras. *MATHEMATICA SCANDINAVICA*, 56, Dec. 1985.
37. Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster Zero-Knowledge for Boolean Circuits. In *USENIX Security 2016*, pages 1069–1083. USENIX Association, 2016.
38. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, apr 1988.
39. Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In *USENIX Security 2021*, pages 519–535. USENIX Association, 2021.
40. Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. In *EUROCRYPT 2020*, pages 674–704. Springer, 2020.
41. Lov K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *ACM STOC '96*, page 212–219. Association for Computing Machinery, 1996.
42. Kishan Chand Gupta and Indranil Ghosh Ray. Finding Biaffine and Quadratic Equations for S-Boxes Based on Power Mappings. *IEEE Transactions on Information Theory*, 61(4):2200–2209, 2015.
43. Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009.
44. Akinori Hosoyamada and Yu Sasaki. Cryptanalysis Against Symmetric-Key Schemes with Online Classical Queries and Offline Quantum Computations. In *CT-RSA 2018*, pages 198–218. Springer, 2018.
45. Andreas Hulsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kolbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS+. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`.
46. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *ACM STOC 2007*, pages 21–30, 2007.
47. David Jao, Reza Azarderakhsh, Matt Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalili, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, and David Urbanik. SIKE: Supersingular Isogeny Key Encapsulation. *HAL*, 2017(0), 2017.
48. Daniel Kales and Greg Zaverucha. Efficient Lifting for Shorter Zero-Knowledge Proofs and Post-Quantum Signatures. Cryptology ePrint Archive, Paper 2022/588, 2022. `https://eprint.iacr.org/2022/588`.

49. Daniel J Katz, KU Schmidt, and A Winterhof. Weil sums of binomials: Properties applications and open problems. In *Combinatorics and Finite Fields: Difference Sets, Polynomials, Pseudorandomness and Applications*, volume 23, pages 109–134. De Gruyter, 2019.
50. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures. In *ACM CCS 2018*, pages 525–537. ACM, 2018.
51. Joe Kilian and Phillip Rogaway. How to Protect DES Against Exhaustive Key Search (an Analysis of DESX). *Journal of Cryptology*, 14(1):17–35, Jan 2001.
52. Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In *CRYPTO '99*, pages 19–30. Springer, 1999.
53. Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In *2010 IEEE International Symposium on Information Theory*, pages 2682–2685, 2010.
54. Hidenori Kuwakado and Masakatu Morii. Security on the quantum-type Even-Mansour cipher. In *2012 International Symposium on Information Theory and its Applications*, pages 312–316, 2012.
55. Gregor Leander and Alexander May. Grover Meets Simon – Quantumly Attacking the FX-construction. In *ASIACRYPT 2017*, pages 161–178. Springer, 2017.
56. Fukang Liu, Takanori Isobe, and Willi Meier. Cryptanalysis of full LowMC and LowMC-M with algebraic techniques. In *Annual International Cryptology Conference*, pages 368–401. Springer, 2021.
57. Fukang Liu, Willi Meier, Santanu Sarkar, and Takanori Isobe. New Low-Memory Algebraic Attacks on LowMC in the Picnic Setting. *IACR Transactions on Symmetric Cryptology*, 2022(3):102–122, Sep. 2022.
58. Fukang Liu, Santanu Sarkar, Gaoli Wang, Willi Meier, and Takanori Isobe. Algebraic Meet-in-the-Middle Attack on LowMC. Cryptology ePrint Archive, Paper 2022/019, 2022. `https://eprint.iacr.org/2022/019`, *to appear in* Asiacrypt 2022.
59. Michael Luby and Charles Rackoff. How to Construct Pseudo-random Permutations from Pseudo-random Functions. In *CRYPTO '85*, pages 447–447. Springer, 1986.
60. Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`.
61. Yassir Nawaz, Kishan Chand Gupta, and Guang Gong. Algebraic Immunity of S-Boxes Based on Power Mappings: Analysis and Construction. *IEEE Transactions on Information Theory*, 55(9):4263–4273, 2009.
62. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`.
63. Christian Rechberger, Hadi Soleimany, and Tyge Tiessen. Cryptanalysis of Low-Data Instances of Full LowMCv2. *IACR Transactions on Symmetric Cryptology*, 2018(3):163–181, 2018.
64. Jan Ferdinand Sauer and Alan Szepieneic. SoK: Gröbner Basis Algorithms for Arithmetization Oriented Ciphers. Cryptology ePrint Archive, Paper 2021/870, 2021. `https://eprint.iacr.org/2021/870`.
65. Peter Schwabe, Roberto Avanzi, Joppe Bos, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehle, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`.
66. Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-Bit Blockcipher CLEFIA (Extended Abstract). In *FSE 2007*, pages 181–195. Springer, 2007.
67. Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. IEEE, 1994.
68. Daniel R. Simon. On the Power of Quantum Computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.
69. Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, Vladmir Kolesnikov, and Daniel Kales. Picnic. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions`.
70. Mark Zhandry. How to Construct Quantum Random Functions. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 679–687, 2012.

A Refining Algebraic Cryptanalysis of Power Functions over Binary Fields

In this section, we will study how the number of quadratic equations obtained from a power mapping based S-box affects the complexity of the Gröbner basis attack and the XL attack on a symmetric primitive based on the S-box.

A.1 Gröbner Basis Attack over $\mathbb{F}_2$

In order to see how the number of quadratic equations from a power mapping based S-box affects the time complexity of the Gröbner basis computation, we compare the theoretic estimation of the degree of regularity and the *solving degree* [28], which is the highest degree reached during the actual Gröbner basis computation, for toy parameters. The solving degrees are obtained with *grevlex* order.

Consider an r-round Even-Mansour cipher [34] based on S-boxes, each of which defines νn linearly independent quadratic equations for some $\nu \geq 1$. By introducing intermediate variables between rounds, we can establish a system of νrn quadratic equations in rn variables. Adding rn field equations to this system, we obtain the Hilbert series as follows.

$$\mathrm{HS}(z) = \frac{1}{(1-z)^{rn}}(1-z^2)^{(1+\nu)rn} = (1+z)^{rn}(1-z^2)^{\nu rn}. \tag{7}$$

We consider four types of S-boxes with different values for the constants ν: the inverse S-box $y = x^{2^n-2}$, a Mersenne S-box $y = x^{2^e-1}$ for some e, an S-box $y = x^{2^{s+1}+2^{s-1}-1}$ for $n = 2s$, and a Niho S-box $y = x^a$, where a, called a Niho exponent, is defined as follows [30].

$$a = \begin{cases} 2^s + 2^{\frac{s}{2}} - 1 & \text{if } n = 2s+1 \text{ for some even } s, \\ 2^s + 2^{\frac{3s+1}{2}} - 1 & \text{if } n = 2s+1 \text{ for some odd } s. \end{cases}$$

In this paper, an S-box of the form $y = x^{2^{s+1}+2^{s-1}-1}$ with $n = 2s$ will be called an *NGG* S-box (after the authors of [61] that studied its properties). Each S-box is a powering function of the form $y = x^k$ where $\mathsf{hw}(k+1) \in \{1, 2\}$. Since x^{k+1} is linear or quadratic over $\mathbb{F}_2$, each S-box defines n quadratic equations over $\mathbb{F}_2$ from an implicit equation $xy = x^{k+1}$.

Using the algorithm proposed in [61], we can find e such that the Mersenne S-box $y = x^{2^e-1}$ defines $3n$ quadratic equations over $\mathbb{F}_2$. It is known that the NGG and the Niho S-boxes define $2n$ and n quadratic equations over $\mathbb{F}_2$ if $n \geq 8$, respectively [61]. When it comes to the inverse S-box, we will assume that it defines $5n$ quadratic equations over $\mathbb{F}_2$ from the quadratic relation $xy = 1$ over $\mathbb{F}_{2^n}$ [19].[5]

For each S-box, we consider two different types of systems of equations: the *basic* system containing only n quadratic equations directly obtained from the implicit quadratic relation such as $xy = 1$ and $xy = x^{2^e}$, and the *full* system containing the exact number of quadratic equations induced from the S-box definition. For the Niho S-box, we do not distinguish the basic and the full systems since both systems contain the same number of quadratic equations. The exact quadratic equations describing the full system can be computed by the algorithm proposed in [42].

We computed a Gröbner basis for a system of equations defined by a single evaluation of a single-round Even-Mansour cipher based on each of the four S-boxes, using MAGMA [15]. Figure 4 compares the degree of regularity estimated by (7) and the solving degree *sd*. We observe that for both the basic and the full systems, their solving degrees are close to the theoretically estimated values for the full system.

The four S-boxes differ in the actual running time of Gröbner basis computation as shown in Figure 5. We observe that Gröbner basis computation becomes faster given a larger number of quadratic equations.

Table 6 compares the degree of regularity estimated by (7) for an evaluation of a single-round Even-Mansour cipher, and the corresponding time complexity for Gröbner basis computation for various values of ν and $n \in \{128, 192, 256\}$. We observe that the time complexity significantly decreases as ν grows. We conclude that the exact number of quadratic equations from an S-box, represented by the constant ν, is critical to algebraic cryptanalysis of a primitive built on the S-box.

[5] More precisely, the inverse S-box defines $5n - 1$ quadratic equations [21], while one can assume that the input to the S-box is nonzero for a large field, in which case $5n$ quadratic equations are obtained.

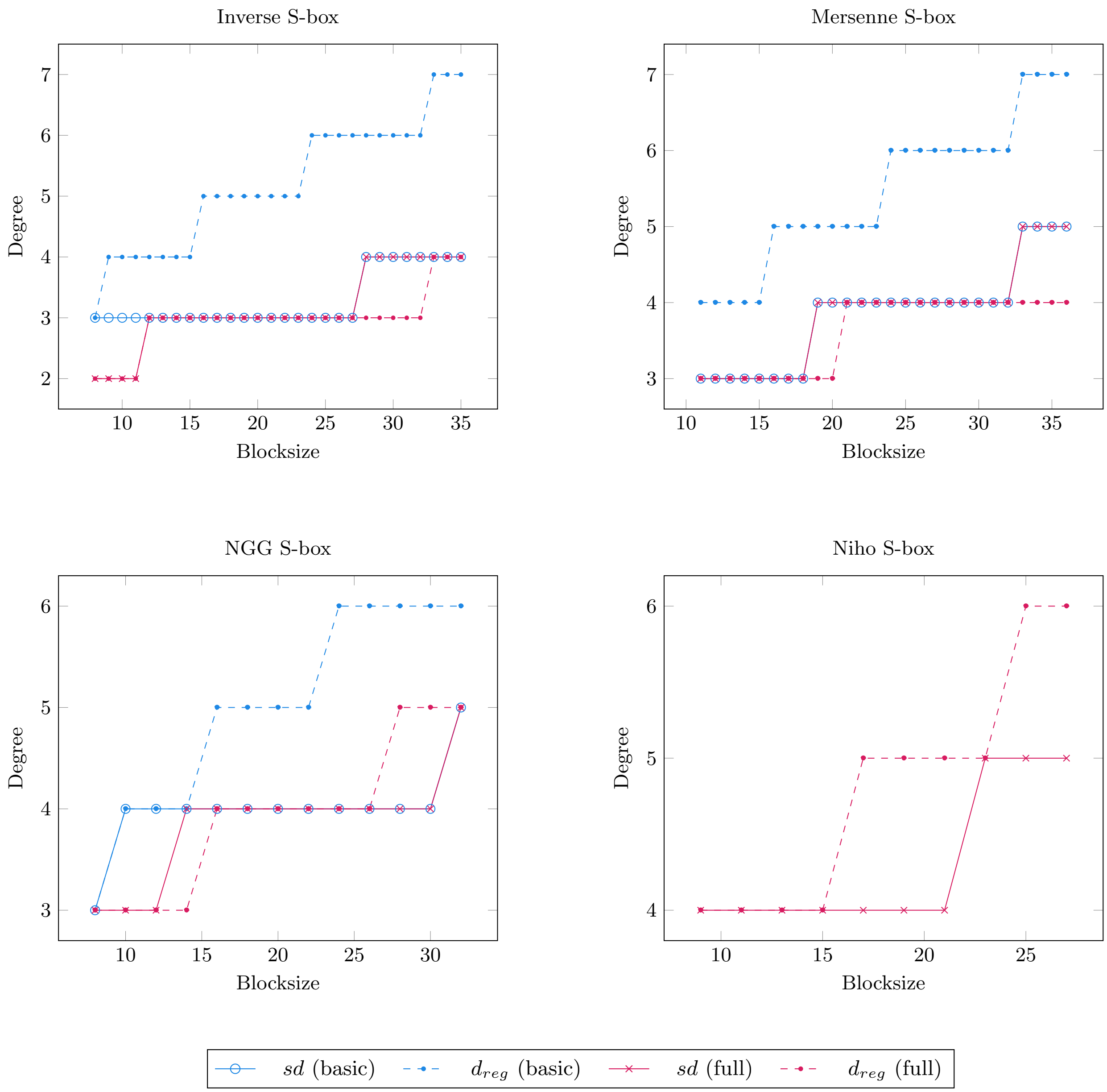

Fig. 4: Degree of regularity d_{reg} estimated by (7) and the solving degree sd for the basic and the full systems of equations constructed from a single evaluation of a single-round Even-Mansour cipher built on top of each of the inverse, Mersenne, NGG and Niho S-boxes.

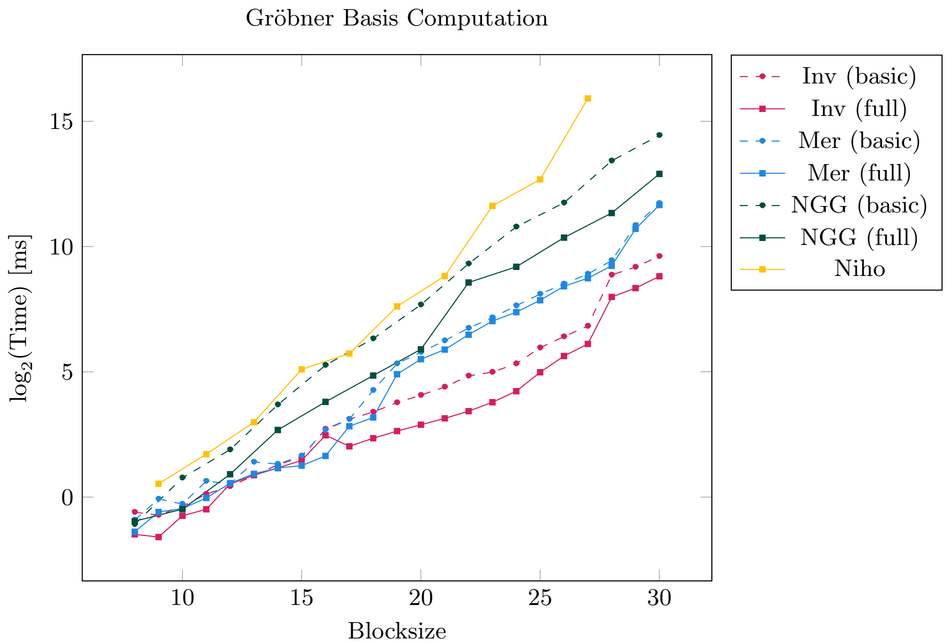

Fig. 5: Computation time of a Gröbner basis for a single-round Even-Mansour cipher. Inv, NGG and Niho represent the inverse, NGG and Niho S-boxes having $5n$, $3n$, $2n$, and n quadratic equations, respectively. This experiment is done in AMD Ryzen 7 2700X @ 3.70GHz with 128 GB memory.

n	Degree of Regularity					Complexity (bits)				
	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 4$	$\nu = 5$	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 4$	$\nu = 5$
128	17	11	9	8	7	144.6	104.9	90.1	82.2	74.0
192	23	15	12	10	9	204.0	148.8	125.5	108.9	100.3
256	29	19	14	12	10	263.1	192.6	152.5	135.2	117.0

Table 6: Degree of regularity estimated by (7) for a single-round Even-Mansour cipher and the corresponding time complexity for computing a Gröbner basis according to the value of ν and the block size $n \in \{128, 192, 256\}$.

A.2 XL Attack over $\mathbb{F}_2$

To see the impact of the number of quadratic equations of the S-box on the XL attack, we experiment with the XL algorithm for a single-round Even-Mansour cipher for toy parameters. Figure 6 shows the ratio of a rank to the number of monomials in the extended system according to the target degree D of the XL algorithm for the basic and the full systems of equations constructed from a single evaluation of a single-round Even-Mansour cipher of blocksize $n = 20$ using the inverse, Mersenne, and NGG S-boxes. The dashed lines show the results for the basic systems, and the solid lines show those for the full systems. We observe that, as expected, applying the XL algorithm for the full system results in a smaller target degree D achieving a rank equal to the number of monomials than the basic system. We also find that all the systems induced by those S-boxes are dense; all the monomials of degrees up to D appear in the experiment.

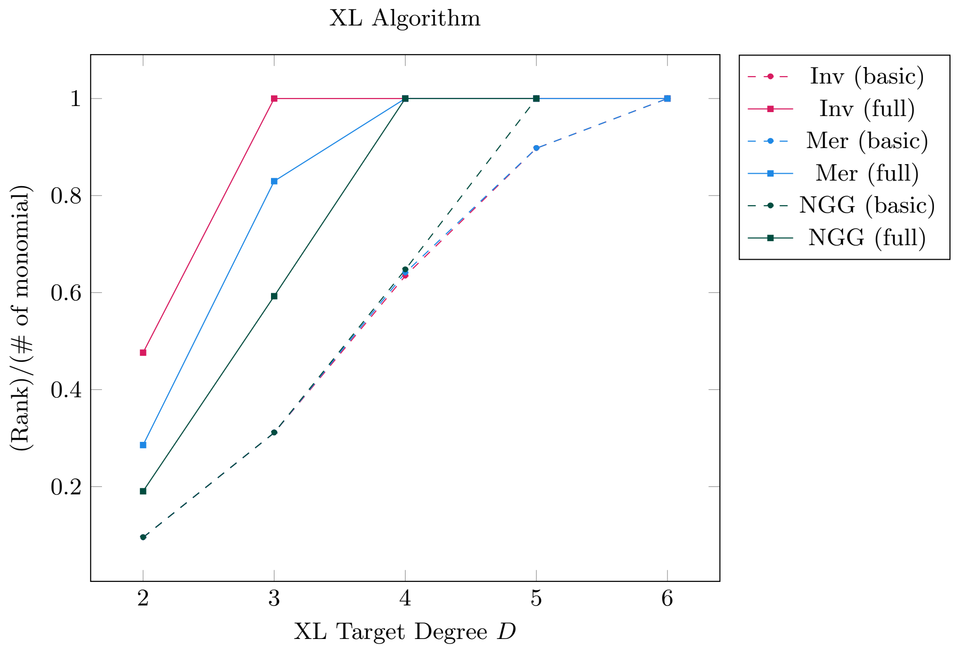

Fig. 6: The ratio of a rank to the number of appearing monomials in the extended system according to the target degree D of the XL algorithm for a single-round Even-Mansour cipher of the blocksize $n = 20$. Inv, Mer, and NGG represent the inverse, Mersenne, and NGG S-boxes having $5n$, $3n$, and $2n$ quadratic equations, respectively. The dashed lines show the results for the basic systems and the solid lines show those for the full systems.

B Differential Cryptanalysis

Resistance of a substitution-permutation cipher against differential cryptanalysis is typically estimated by the maximum expected probability of differential trails [22]. As AIM is a key-less primitive, we bound the maximum differential probability without expectation.

Given a pair $(\Delta x, \Delta y)$, the differential probability of $f : \{0,1\}^m \to \{0,1\}^n$ is defined by

$$\mathrm{DP}^f(\Delta x, \Delta y) \stackrel{\text{def}}{=} \Pr_x [f(x \oplus \Delta x) \oplus f(x) = \Delta y]\,.$$

The maximal differential probability is defined as follows.

$$\mathrm{MDP}^f \stackrel{\text{def}}{=} \max_{\Delta x \neq 0, \Delta y} \mathrm{DP}^f(\Delta x, \Delta y).$$

So $\mathrm{DP}^{\mathsf{Mer}[e]}(\Delta x, \Delta y)$ is determined by the number of solutions to $\mathsf{Mer}[e](X \oplus \Delta x) \oplus \mathsf{Mer}[e](X) = \Delta y$, which is an equation of degree $2^e - 2$. Therefore, there are at most $2^e - 2$ solutions to this equation, which implies

$$\mathrm{MDP}^{\mathsf{Mer}[e]} \leq \frac{2^e - 2}{2^n}.$$

Now, we can bound the differential probability of the entire function. See Figure 7 for the notations used in the following argument. We will write $\Delta y = (\Delta y_1, \ldots, \Delta y_\ell)$, and simply

$$\mathrm{DP}(\Delta x, \Delta z) = \mathrm{DP}^{\mathsf{Lin} \circ \mathsf{Mer}[e_1, \ldots, e_\ell]}(\Delta x, \Delta z)$$

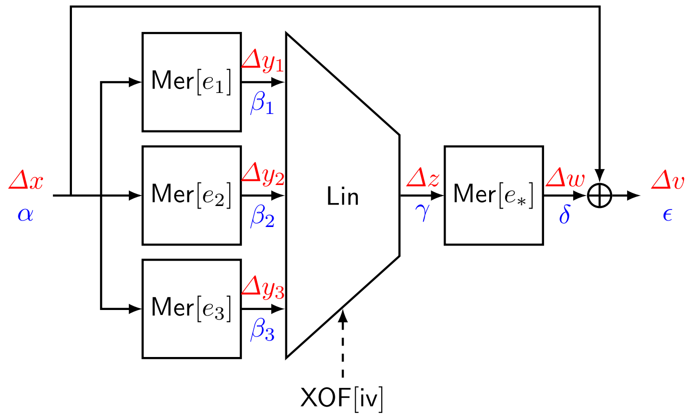

Fig. 7: Differential and linear cryptanalysis against the AIM-V one-way function. See values in red (resp. blue) for differential cryptanalysis (resp. linear cryptanalysis).

where the function in superscript is omitted if it is obvious (e.g., $\mathsf{Lin} \circ \mathsf{Mer}[e_1, \ldots, e_\ell]$ for $\Delta x \to \Delta z$). Then we want to upper bound

$$\mathrm{MDP}^{\mathsf{AIM}} = \max_{\Delta x^* \neq 0, \Delta v^*} \mathrm{DP}(\Delta x^*, \Delta v^*).$$

Let $\mathrm{Img}(\Delta x^*) = \{\Delta y : \Delta x = \Delta x^*\}_{x \in \mathbb{F}_{2^n}}$. Note that $|\mathrm{Img}(\Delta x^*)| \leq 2^n$ for any Δx^*. For a fixed $n \times \ell n$-matrix A, we have

$$\begin{aligned}\mathrm{DP}(\Delta x^*, \Delta z^*) &= \sum_{\Delta y \in \mathrm{Img}(\Delta x^*) \cap A^{-1}(\Delta z^*)} \mathrm{DP}(\Delta x^*, \Delta y) \\ &\leq \sum_{\Delta y \in \mathrm{Img}(\Delta x^*) \cap A^{-1}(\Delta z^*)} \min_{1 \leq i \leq \ell} \mathrm{MDP}^{\mathsf{Mer}[e_i]}.\end{aligned}$$

Let $\varepsilon = \min_{1 \leq i \leq \ell} \mathrm{MDP}^{\mathsf{Mer}[e_i]}$. Let $\delta > 0$ and let A be a block-wise invertible matrix as in AIM. Assuming an event $\Delta y \in A^{-1}(\Delta z^*)$ is independent for each $\Delta y \in \mathrm{Img}(\Delta x^*)$, we have

$$\Pr_A[\mathrm{DP}(\Delta x^*, \Delta z^*) > (1+\delta)\varepsilon] \leq \Pr_{X \sim \mathcal{B}}[X > 1 + \delta]$$

where $\mathcal{B} = \mathrm{Bin}\left(|\mathrm{Img}(\Delta x^*)|, \Pr_A[\Delta y \in A^{-1}(\Delta z^*)]\right)$ is a binomial distribution. The probabilities $\Pr_A[\Delta y \in A^{-1}(\Delta z^*)]$ for $\ell \in \{2, 3\}$ are summarized in Table 7, and the proof is given in Appendix D.

	$\Delta z^* = 0$	$\Delta z^* \neq 0$
$\ell = 2$	$\frac{1}{2^n - 1}$	$\frac{2^n - 2}{(2^n - 1)^2}$
$\ell = 3$	$\frac{2^n - 2}{(2^n - 1)^2}$	$\frac{(2^n - 2)^2}{(2^n - 1)^3} + \frac{1}{(2^n - 1)^2}$

Table 7: $\Pr_A[\Delta y \in A^{-1}(\Delta z^*)]$ for $\ell \in \{2, 3\}$.

For a binomial distribution $\mathcal{B}' = \text{Bin}(2^n, 1/2^n + 2/2^{2n})$, we have

$$\begin{aligned}\Pr_A[\mathrm{DP}(\Delta x^*, \Delta z^*) > (1+\delta)(1+2/2^n)\varepsilon] &\le \Pr_{X\sim\mathcal{B}}[X > (1+\delta)(1+2/2^n)] \\ &\le \Pr_{X'\sim\mathcal{B}'}[X' > (1+\delta)(1+2/2^n)] \\ &< \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{1+2/2^n}\end{aligned}$$

by the Chernoff bound. Now, $\mathrm{DP}(\Delta x^*, \Delta v^*)$ can be expressed in terms of $\mathrm{DP}(\Delta x^*, \Delta z^*)$ as follows.

– If $\Delta v^* = \Delta x^*$, then

$$\mathrm{DP}(\Delta x^*, \Delta v^*) = \mathrm{DP}^{\Delta x \to \Delta z}(\Delta x^*, 0).$$

– Otherwise, for a fixed $B \in \mathbb{F}_2^n$,

$$\mathrm{DP}(\Delta x^*, \Delta v^*) = \sum_{\Delta z}\left(\mathrm{DP}(\Delta x^*, \Delta z)\cdot \Pr_x\left[\begin{array}{c}H_b(X\oplus\Delta x^*)\oplus H_b(X)\\ =\Delta x^*\oplus\Delta v^*\end{array}\middle|\begin{array}{c}F(X\oplus\Delta x^*)\oplus F(X)\\ =\Delta z\end{array}\right]\right)$$

where $F(x) = A\cdot\mathsf{Mer}[e_1,\ldots,e_\ell](x)$, $G = \mathsf{Mer}[e_*]$, and $H_b(x) = G(F(x)\oplus b)$. The vector $b \in \mathbb{F}_2^n$ is from the constant addition in affine layers.

We remark that

$$\mathbb{E}_b[\mathrm{DP}(\Delta x^*, \Delta v^*)] = \sum_{\Delta z}\mathrm{DP}(\Delta x^*, \Delta z)\,\mathrm{DP}(\Delta z, \Delta v^*)$$

but we do not use this equation since b is public. For $\delta' > 0$, assuming the independence, we have

$$\begin{aligned}\Pr_b[\mathrm{DP}(\Delta x^*, \Delta v^*) > (1+\delta')(2^{e_*}-2)\max_{\Delta z}\mathrm{DP}(\Delta x^*, \Delta z)] &\le \Pr_{X''\sim\mathcal{B}''}[X'' > (1+\delta')(2^{e_*}-2)] \\ &\le \left(\frac{e^{\delta'}}{(1+\delta')^{1+\delta'}}\right)^{2^{e_*}-2}\end{aligned}$$

where $\mathcal{B}'' = \text{Bin}\left(2^n, \max_{\Delta z\neq 0}\mathrm{DP}(\Delta z, \Delta v^*)\right)$ is a binomial distribution.

For any $\Delta x^* \neq 0, \Delta v^*$, we have

$$\begin{aligned}&\Pr_{A,B}[\mathrm{DP}(\Delta x^*, \Delta v^*) > (1+\delta)(1+2/2^n)(1+\delta')(2^{e_*}-2)\varepsilon] \\ &\le \Pr_A[\max_{\Delta z}\mathrm{DP}(\Delta x^*, \Delta z) > (1+\delta)(1+2/2^n)\varepsilon]\cdot\left(\frac{e^{\delta'}}{(1+\delta')^{1+\delta'}}\right)^{2^{e_*}-2} \\ &< \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{1+2/2^n}\left(\frac{e^{\delta'}}{(1+\delta')^{1+\delta'}}\right)^{2^{e_*}-2}.\end{aligned}$$

We set the bound at

$$\left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{1+2/2^n}\left(\frac{e^{\delta'}}{(1+\delta')^{1+\delta'}}\right)^{2^{e_*}-2} = 2^{-\lambda}$$

for security parameter λ and summarize the values of $\log\gamma$ such that

$$\Pr_{A,B}\left[\mathrm{MDP}^{\mathsf{AIM}} > \gamma\right] < 2^{-\lambda}$$

according to its security level in Table 4. We remark that $\gamma > 2^{-\lambda}$ for each λ does not imply the feasibility of differential cryptanalysis.

C Linear Cryptanalysis

In contrast to differential cryptanalysis, security against linear cryptanalysis has been rarely evaluated for key-less primitives. The reason is that differential cryptanalysis helps finding a collision or a second preimage while linear cryptanalysis does not. That said, in order to prevent any possible variant of linear cryptanalysis, we briefly compute the bias of a correlation trail assuming the masked sums of inputs and outputs are independent.

Given a pair $(\alpha, \beta) \in \mathbb{F}_{2^n} \times \mathbb{F}_{2^n}^{\times}$, the linear probability of $\mathsf{Mer}[e]$ is defined by

$$\mathrm{LP}^{\mathsf{Mer}[e]}(\alpha, \beta) \stackrel{\text{def}}{=} \frac{2}{2^n} \cdot \left|\{x \in \mathbb{F}_{2^n} : \alpha^\top x = \beta^\top \mathsf{Mer}[e](x)\}\right| - 1.$$

The maximal linear probability is defined as follows.

$$\mathrm{MLP}^{\mathsf{Mer}[e]} \stackrel{\text{def}}{=} \max_{\alpha, \beta \neq 0} \mathrm{LP}^{\mathsf{Mer}[e]}(\alpha, \beta).$$

For a non-power-of-2 exponent d such that x^d is invertible, the maximum linear probability of $f(x) = x^d$ on $\mathbb{F}_2^n$ has a generic bound $\mathrm{MLP}^f \leq (d-1)/2^{n/2}$ [49]. Specifically, the maximum linear probability of a Mersenne S-box is bounded by

$$\mathrm{MLP}^{\mathsf{Mer}[e]} \leq \frac{2^e - 2}{2^{n/2}}.$$

Now, we can bound the linear probability of the entire function. See Figure 7 for the notations used in the following argument. We will simply write

$$\mathrm{LP}(\alpha, \gamma) = \mathrm{LP}^{\mathsf{Lin} \circ \mathsf{Mer}[e_1, \ldots, e_\ell]}(\alpha, \gamma)$$

where the function in superscript is omitted if it is obvious (e.g., $\mathsf{Lin} \circ \mathsf{Mer}[e_1, \ldots, e_\ell]$ for $\alpha \to \gamma$). Then the bias of a trail

$$\alpha^* \to \beta^* \to \gamma^* \to \delta^* \to \epsilon^*$$

is computed as

$$\begin{aligned}
(\mathrm{LP}(\alpha^*, \beta^*)\,\mathrm{LP}(\beta^*, \gamma^*)\,\mathrm{LP}(\gamma^*, \delta^*)\,\mathrm{LP}(\delta^*, \epsilon^*))^2 &\leq (\mathrm{LP}(\alpha^*, \beta^*)\,\mathrm{LP}(\gamma^*, \delta^*))^2 \\
&\leq \left(\min_{1 \leq i \leq \ell} \mathrm{MLP}^{\mathsf{Mer}[e_i]} \mathrm{MLP}^{\mathsf{Mer}[e_*]}\right)^2 \\
&\leq \min_{1 \leq i \leq \ell} \frac{(2^{e_i} - 2)^2 (2^{e_*} - 2)^2}{2^{2n}}
\end{aligned}$$

assuming independence of each edge. When

$$\min_{1 \leq i \leq \ell} (2^{e_i} - 2)^2 (2^{e_*} - 2)^2 < 2^n,$$

the bias of AIM is smaller 2^{-n}, and the amount of data required for linear cryptanalysis becomes at least 2^n.

D Computing $\mathbf{Pr}_{\boldsymbol{A}}[\boldsymbol{\Delta y} \in \boldsymbol{A}^{-1}(\boldsymbol{\Delta z}^*)]$

Let $\mathcal{L}_n$ denote a set of $n \times n$ invertible matrices over $\mathbb{F}_2$. Then we have

$$\Pr_{L \leftarrow \mathcal{L}_n}[Lx = y] = \frac{1}{2^n - 1}$$

for nonzero vectors $x, y \in \mathbb{F}_2^n$. Note that the zero vector is a fixed point of any linear transformation.

Case $\ell = 2$. The $n \times 2n$ matrix A is written as

$$A = [A_1 | A_2]$$

where $A_1, A_2 \in \mathcal{L}_n$. Then

$$\begin{aligned}\Pr_A[A\Delta y = \Delta z^*] &= \Pr_{A_1,A_2}[A_1\Delta y_1 \oplus A_2\Delta y_2 = \Delta z^*] \\ &= \Pr_{A_1,A_2}\begin{bmatrix} A_2\Delta y_2 \neq \Delta z^* \\ \wedge \\ A_1\Delta y_1 = A_2\Delta y_2 \oplus \Delta z^* \end{bmatrix}\end{aligned}$$

since $\Delta y_1 \neq 0$ for $\Delta x \neq 0$. If $\Delta z^* = 0$ then $A_2\Delta y_2 \neq \Delta z^*$ for every $\Delta y_2 \neq 0$, and hence

$$\Pr_A[A\Delta y = 0] = \Pr_A[A_1\Delta y_1 = A_2\Delta y_2] = \frac{1}{2^n - 1}.$$

On the other hand, if $\Delta z^* \neq 0$ then

$$\begin{aligned}\Pr_A[A\Delta y = \Delta z^*] &= \Pr_{A_2}[A_2\Delta y_2 \neq \Delta z^*] \Pr_A[A_1\Delta y_1 = A_2\Delta y_2 \oplus \Delta z^* | A_2\Delta y_2 \oplus \Delta z^* \neq 0] \\ &= \frac{2^n - 2}{(2^n - 1)^2}.\end{aligned}$$

Case $\ell = 3$. The $n \times 3n$ matrix A is written as

$$A = [A_1 | A_2 | A_3]$$

where $A_1, A_2, A_3 \in \mathcal{L}_n$. Then

$$\begin{aligned}\Pr_A[A\Delta y = \Delta z^*] &= \Pr_{A_1,A_2,A_3}[A_1\Delta y_1 \oplus A_2\Delta y_2 \oplus A_3\Delta y_3 = \Delta z^*] \\ &= \Pr_{A_1,A_2,A_3}\begin{bmatrix} A_2\Delta y_2 \oplus A_3\Delta y_3 \neq \Delta z^* \\ \wedge \\ A_1\Delta y_1 = A_2\Delta y_2 \oplus A_3\Delta y_3 \oplus \Delta z^* \end{bmatrix}\end{aligned}$$

since $\Delta y_1 \neq 0$ for $\Delta x \neq 0$.

If $\Delta z^* = 0$, then we have

$$\Pr_{A_2,A_3}[A_2\Delta y_2 \neq A_3\Delta y_3] = \frac{2^n - 2}{2^n - 1}. \tag{8}$$

For any nonzero a, we have

$$\Pr_A[A_1\Delta y_1 = a | A_2\Delta y_2 \oplus A_3\Delta y_3 = a] = \frac{1}{2^n - 1}. \tag{9}$$

Combining (8) and (9), we obtain

$$\begin{aligned}\Pr_A[A\Delta y = 0] &= \Pr_{A_2,A_3}[A_2\Delta y_2 \neq A_3\Delta y_3] \Pr_A[A_1\Delta y_1 = A_2\Delta y_2 \oplus A_3\Delta y_3 | A_2\Delta y_2 \neq A_3\Delta y_3] \\ &= \frac{2^n - 2}{(2^n - 1)^2}.\end{aligned}$$

Suppose that $\Delta z^* \neq 0$. Since Δy_2 is nonzero, we have

$$\begin{aligned}\Pr_{A_2,A_3}[A_2\Delta y_2 \oplus A_3\Delta y_3 \neq \Delta z^*] &= \Pr_{A_2,A_3}\begin{bmatrix} A_3\Delta y_3 \neq \Delta z^* \\ \wedge \\ A_2\Delta y_2 \neq A_3\Delta y_3 \oplus \Delta z^* \end{bmatrix} + \Pr_{A_3}[A_3\Delta y_3 = \Delta z] \\ &= \left(\frac{2^n - 2}{2^n - 1}\right)^2 + \frac{1}{2^n - 1}.\end{aligned}$$

Therefore, we obtain

$$\Pr_A[A\Delta y = \Delta z^*] = \frac{(2^n - 2)^2}{(2^n - 1)^3} + \frac{1}{(2^n - 1)^2}.$$

E Performance Evaluation of AVX2-Optimized Implementations

E.1 Environment.

The source codes are developed in C++17, using the GNU C++ 8.4.0 (GNU C 7.5.0 for running the algorithms in the third round submission packages for NIST PQC standardization) compiler with the AVX2 instructions on the Ubuntu 18.04 operating system. All the implementations used in the experiments are compiled at the `-O3` optimization level. For the instantiation of the XOF, we use SHAKE in `XKCP` library[6]. We use SHAKE128 for AIMer-I, and SHAKE256 for AIMer-III and AIMer-V. Our experiments are measured in Intel Xeon E5-1650 v3 @ 3.50GHz with 128 GB memory. For a fair comparison, we measure the execution time for each signature scheme on the same CPU using the `taskset` command with Hyper-Threading and Turbo Boost features disabled.

E.2 Performance of AIMer.

As mentioned in Section 2.3, AIM has been designed to take full advantage of optimization by repeated multipliers to reduce the number of α values. Due to this technique, the overall performance of the signature scheme is improved in terms of both the signature size and the signing time. The performance of AIMer is summarized in Table 8. Parameter sets (i.e., the number of parties N and the number of parallel repetitions τ) for various security levels are chosen in the same way of [48]. We observe that AIMer enjoys the best trade-off between the signature size and the signing/verification time.

Scheme	N	τ	Sign (ms)	Verify (ms)	Size (B)
AIMer-I	16	33	0.82	0.78	5 904
AIMer-I	57	23	1.82	1.77	4 880
AIMer-I	256	17	5.96	5.90	4 176
AIMer-I	1615	13	29.62	29.17	3 840
AIMer-III	16	49	1.57	1.48	13 080
AIMer-III	64	33	3.86	3.62	10 440
AIMer-III	256	25	10.57	10.42	9 144
AIMer-III	1621	19	58.70	58.10	8 352
AIMer-V	16	65	2.87	2.78	25 152
AIMer-V	62	44	6.60	6.54	19 904
AIMer-V	256	33	19.21	19.19	17 088
AIMer-V	1623	25	98.49	98.64	15 392

Table 8: Performance of AIMer for various parameter sets with AVX2 instruction set.

In Table 9, AIMer is compared to the state-of-the-art Rainier signature scheme combined with the BN++ proof system (denoted BN++Rain$_r$, where $r \in \{3, 4\}$) with all the optimizations from [48] applied at the 128-bit security level. AIMer-I enjoys 5.14 to 8.21% shorter signature size than BN++Rain$_3$ with similar signing and verification time. Compared to BN++Rain$_4$, AIMer achieves more significant improvement with 13.98 to 21.15% shorter signature size and 5.59 to 14.84% improved signing and verification performance for all the parameter sets.

[6] `https://github.com/XKCP/XKCP`

Scheme	N	τ	Sign (ms)	Verify (ms)	Size (B)
BN++Rain$_3$ [48]	16	33	0.83	0.77	6 432
BN++Rain$_3$ [48]	57	23	1.83	1.77	5 248
BN++Rain$_3$ [48]	256	17	5.92	5.94	4 448
BN++Rain$_3$ [48]	1615	13	28.95	28.33	4 048
BN++Rain$_4$ [48]	16	33	0.93	0.86	7 488
BN++Rain$_4$ [48]	57	23	2.09	2.01	5 984
BN++Rain$_4$ [48]	256	17	6.45	6.23	4 992
BN++Rain$_4$ [48]	1615	13	32.85	31.86	4 464
AIMer-I	16	33	0.82	0.78	5 904
AIMer-I	57	23	1.82	1.77	4 880
AIMer-I	256	17	5.96	5.90	4 176
AIMer-I	1615	13	29.62	29.17	3 840

Table 9: Performance of AIMer, BN++Rain$_3$, and BN++Rain$_4$ at 128-bit security level.

E.3 Comparison.

We compare the performance of AIMer to existing post-quantum signature schemes at the 128-bit security level in Table 10. In the first group, we provide the performance of three selected algorithms in the NIST competition for PQC standardization - CRYSTALS-Dilithium [60], Falcon [62], and SPHINCS$^+$ [45].

CRYSTALS-Dilithium and Falcon are lattice-based signature schemes with high efficiency in both bandwidth (signature size plus public key size) and signing/verification time. We implemented SPHINCS$^+$ using the SHAKE256 hash function for a fair comparison between symmetric primitives based signature schemes. Compared to any of the small and the fast variants of SPHINCS$^+$, AIMer obviously provides smaller bandwidth and faster signing time at the cost of slightly slower verification.

In the second group, we compare existing ZKP-based signature schemes based on symmetric primitives: Picnic [69], Limbo [25], Banquet[7], Rainier[8], and BN++Rain[9]. In particular, Picnic is one of the alternate candidates of the third round of the NIST competition. For Limbo-AES128, we cited the numbers from the paper as its public implementation is not available (to the best of our knowledge). When the number of parties N is set to 16, these schemes require bandwidth of 12,495 to 30,957 bytes, while AIMer requires 5,936 bytes with comparable performance in signing and verification time.

[7] `https://github.com/dkales/banquet`
[8] `https://github.com/IAIK/rainier-signatures`
[9] `https://github.com/IAIK/bnpp_helium_signatures`

Scheme	$\|pk\|$ (B)	$\|sig\|$ (B)	Sign (ms)	Verify (ms)
Dilithium2 [60]	1312	2 420	0.10	0.03
Falcon-512 [62]	897	690	0.27	0.04
SPHINCS$^+$-128s* [45]	32	7 856	315.74	0.35
SPHINCS$^+$-128f* [45]	32	17 088	16.32	0.97
Picnic1-L1-full [69]	32	30 925	1.16	0.91
Picnic3-L1 [69]	32	12 463	5.83	4.24
Banquet [11]	32	19 776	7.09	5.24
Limbo-AES128† [25]	32	21 520	2.70	2.00
Rainier$_3$ [32]	32	8 544	0.97	0.89
BN++Rain$_3$ [48]	32	6 432	0.83	0.77
AIMer-I	32	5 904	0.82	0.78

*: -SHAKE-simple

†: measurements are from this paper.

Table 10: Comparison of AIMer to existing (post-quantum) signature schemes at 128-bit security level. The number of parties N is set to 16 for ZKP-based signature schemes.