AIM: Symmetric Primitive for Shorter Signatures with Stronger Security

Seongkwang Kim\(^{1\dagger}\), Jinchul Ha\(^{2\dagger}\), Mincheol Son\(^{2}\), Byeonghak Lee\(^{2}\), Dukjae Moon\(^{1}\), Joohee Lee\(^{3}\)
Sangyup Lee\(^{1}\), Jihoon Kwon\(^{1}\), Jihoon Cho\(^{1}\), Hyojin Yoon\(^{1}\), Jooyoung Lee\(^{2}\)

\(^{1}\)Samsung SDS \(^{2}\)KAIST \(^{3}\)Sungshin Women’s University

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.

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 [66]. 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 [64] as a public key encryption scheme, and CRYSTALS-Dilithium [59], Falcon [61], and SPHINCS\(^{+}\) [44] as digital signature schemes.

MPC-IN-THE-HEAD BASED SIGNATURE. MPC-in-the-Head (MPCitH), proposed by Ishai et al. [45], 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 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 [46] 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. [49] 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, 55–57, 62], 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 \(F_{2^{32}}\).

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 \(F_{2^{32}}\). 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 algebraic attack due to their simple algebraic structures. Kales and Zaverucha [47] 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.

---

\(^2\)https://lowmcchallenge.github.io/

---

1.1 Our Contribution

The main contribution of this paper is two-fold. First, 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 \([1, 3, 32, 38]\). Dobraunig et al. consider the analysis over \(F_2\) [32], but only deal with the equations of high degrees. We apply the Gröbner basis attack to the system of quadratic equations over \(F_2\) using intermediate variables. When it comes to the Gröbner basis attack over \(F_2\), 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.

Second, 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. Our implementation result is summarized in Section 6. Compared to the signature schemes based on the BN++ proof system combined with the 3-round (resp. 4-round) Rain, 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.

---

\(^3\)This name is an abbreviation of Affine-Interleaved Mersenne S-boxes.
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 $n$, $\text{hw}(n)$ 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 $\sum x^{(i)} = x$.

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 $\text{Bin}(n, p)$.

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 $\omega$ such that $2 \leq \omega \leq 3$. The constant $\omega$ 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 linearization, 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 [63] 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].

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

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

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

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 $\mathbb{F}_p$, for some characteristic $p$, and hence $x$ satisfies $x^p - x = 0$ called a field equation. By including field equations in the system of equations, one can remove any possible solution outside $\mathbb{F}_p$ (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, 38]. It does not mean that they are broken under the modified analysis, while considering the field equations would lead to a 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 [51]. 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 $(\sum_{i=1}^{D-2} \binom{n}{i}) m$ 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
Then the protocol proceeds as follows.

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

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

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

\[
O(T^w).
\]

### 2.3 BN++ Zero-knowledge Protocol

In this section, we briefly review the BN++ proof system [47], 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 Appendix D. 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 [45]. In order to check \(C\) multiplication triples \((x_j, y_j, z_j = x_j \cdot y_j)\) over a finite field \(\mathbb{F}\) in the multiparty computation setting with \(N\) parties, helping values \((a_j, b_j)\) 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 \((x_j, y_j, z_j)\) and helping values \((a_j, b_j)\). Then the protocol proceeds as follows.

- A prover is given random challenges
  
  \[
  \varepsilon_1, \cdots, \varepsilon_C \in \mathbb{F}.
  \]

- For \(i \in [N]\), the \(i\)-th party locally sets
  
  \[
  \alpha^{(i)}_1, \cdots, \alpha^{(i)}_C
  \]

  where \(\alpha^{(i)}_j = \varepsilon_j \cdot x_j^{(i)} + a_j^{(i)}\).

- The parties open \(\alpha_1, \cdots, \alpha_C\) by broadcasting their shares.

- For \(i \in [N]\), the \(i\)-th party locally sets
  
  \[
  v^{(i)} = \sum_{j=1}^{C} \varepsilon_j \cdot z_j^{(i)} - \sum_{j=1}^{C} \alpha_j \cdot b_j^{(i)} + c^{(i)}.
  \]

- The parties open \(v\) by broadcasting their shares and output \(\text{Accept}\) if \(v = 0\).

The probability that there exist incorrect triples and the parties output \(\text{Accept}\) 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

\[
6\lambda + \tau \cdot (3\lambda + \lambda \cdot \lceil \log_2(N) \rceil + M(C)),
\]

where \(\lambda\) is the security parameter, \(C\) is the number of multiplication gates in the underlying symmetric primitive, and \(M(C) = (2C + 1) \cdot \log_2(|\mathbb{F}|)\). In particular, \(M(C)\) 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 [47].

**Optimization Techniques.** If multiplication triples use an identical multiplier in common, for example, given \((x_1, y_1, z_1)\) and \((x_2, y_2, z_2)\), then the corresponding \(\alpha\) values can be batched to reduce the signature size. Instead of computing \(\alpha_1 = \varepsilon_1 \cdot x_1 + a_1 \) and \(\alpha_2 = \varepsilon_2 \cdot x_2 + a_2\), \(\alpha = \varepsilon_1 \cdot x_1 + \varepsilon_2 \cdot x_2 + a\) is computed, and \(v\) is defined as

\[
v = \varepsilon_1 \cdot z_1 + \varepsilon_2 \cdot z_2 - \alpha \cdot y + c,
\]

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 \(\alpha\) 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 Refining Algebraic Cryptanalysis of Power Functions over Binary Fields

**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 \(\beta^i\) 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 $\mathbb{F}_{2^n}$:

$$
\sum_{i=0}^{n-1} a_i x_i^n = a_0 x + a_1 x^2 + a_2 x^3 + \cdots + a_{n-1} x^{n-1} \quad (2)
$$

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 \((x_i)_{i=0}^{n-1} \text{ and } (y_i)_{i=0}^{n-1}\), 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 $\log_2(hw(a))$. 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 $[1, 3, 32, 38]$. 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, Dobranyi 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 $\log_2(hw(a))$ 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, 41, 60]. 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. In the following, 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.

### 3.1 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 $vn$ linearly independent quadratic equations for some $v \geq 1$. By introducing intermediate variables between rounds, we can establish a system of $vn$ quadratic equations in $rn$ variables. Adding $rn$ field equations to this system, we obtain the Hilbert series as follows.

$$
HS(z) = \frac{1}{(1-z)^n} (1 - z^2)^{(1+v)rn} = (1+z)^n (1-z^2)^{vn} \quad (3)
$$

We consider four types of S-boxes with different values for the constants $v$: the inverse S-box $y = x^{2^n-2}$, a Mersenne S-box $y = x^{2^n-1}$ for some $e$, an S-box $y = x^{2^{e+1}+2^{e+1}-1}$ 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 $y = x^{2^{e+1}+2^{e+1}-1}$ with $n = 2s$ will be called an NGG S-box (after the authors of [60] that studied its properties). Each S-box is a powering function of the form $y = x^k$ where $hw(k + 1) \in \{1, 2\}$. 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 [60], we can find $e$ such that the Mersenne S-box $y = x^{2^n-1}$ 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 [60]. 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].

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 $xy = x^2$, 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 [41].

---

4More 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.
Figure 1: Degree of regularity $d_{reg}$ estimated by (3) 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.

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 1 compares the degree of regularity estimated by (3) 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 2. We observe that Gröbner basis computation becomes faster given a larger number of quadratic equations.

Table 1 compares the degree of regularity estimated by (3) for an evaluation of a single-round Even-Mansour cipher, and the corresponding time complexity for Gröbner basis computation for various values of $v$ and $n \in \{128, 192, 256\}$. We observe that the time complexity significantly decreases as $v$ grows. We conclude that the exact number of quadratic equations from an S-box, represented by the constant $v$, is critical to algebraic cryptanalysis of a primitive built on the S-box.

3.2 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 3 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
Table 1: Degree of regularity estimated by (3) for a single-round Even-Mansour cipher and the corresponding time complexity for computing a Gröbner basis according to the value of \( \nu \) and the block size \( n \in \{128, 192, 256\} \).

<table>
<thead>
<tr>
<th>( n )</th>
<th>Degree of Regularity</th>
<th>Complexity (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \nu = 1 )</td>
<td>( \nu = 2 )</td>
</tr>
<tr>
<td>128</td>
<td>17</td>
<td>11</td>
</tr>
<tr>
<td>192</td>
<td>23</td>
<td>15</td>
</tr>
<tr>
<td>256</td>
<td>29</td>
<td>19</td>
</tr>
</tbody>
</table>

Figure 2: 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.

degrees up to \( D \) appear in the experiment.

4 AIM: Our New Symmetric Primitive

4.1 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 \((e_1, \ldots, e_\ell, e_s) \in \mathbb{Z}^{\ell+1}_+,\)

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

where each function will be described below. See Figure 4 for the pictorial description of AIM with \( \ell = 3 \).

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

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

for some \( e \). Note that this map is a permutation if \( \gcd(e, n) = 1 \).

As an extension, \( \text{Mer}[e_1, \ldots, e_\ell] : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}^{\ell} \) is defined by

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

Figure 3: 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.

**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 \( b_{iv} \in \mathbb{F}_2^n \). The matrix

\[
A_{iv} = [A_{iv,1} \ldots A_{iv,\ell}] \in (\mathbb{F}_2^{n \times n})^\ell
\]

is composed of \( \ell \) random invertible matrices \( A_{iv,i} \). The matrix \( A_{iv} \) and the vector \( b_{iv} \) are generated by an extendable output function (XOF) with the initial vector \( iv \). Each matrix \( A_{iv,i} \) can be equivalently represented by a linearized polynomial \( L_{iv,i}(x_i) \) on \( \mathbb{F}_2^n \). For \( x = (x_1, \ldots, x_\ell) \in (\mathbb{F}_2^n)^\ell \),

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

By abuse of notation, we will denote \( Ax \) as the same meaning as \( \sum_{1 \leq i \leq \ell} L_{iv,i}(x_i) \). 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 5.1. Exponents $e_1$ and $e_s$ are chosen as small numbers to provide smaller differential probability, and exponent $e_2$ is chosen so that $e_2 \cdot e_s \geq \lambda$, 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 \[32\].

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$\lambda$</th>
<th>$n$</th>
<th>$\ell$</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM-I</td>
<td>128</td>
<td>128</td>
<td>2</td>
<td>3</td>
<td>27</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>AIM-III</td>
<td>192</td>
<td>192</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>29</td>
<td>-</td>
</tr>
<tr>
<td>AIM-V</td>
<td>256</td>
<td>256</td>
<td>3</td>
<td>3</td>
<td>53</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 2: Recommended sets of parameters of AIM.

### 4.2 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 $\mathbb{F}_2$. 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 \[4,23,65\] 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 \[32,38,39\].

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 Section 5.1.

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 $x$

**REPEETITIVE 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 $\mathbb{F}_2^{2n}$ 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.

## 5 Security Analysis

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 (iv, ct), the adversarial goal is to find any pt (not necessarily the same as pt*) such that AIM(iv, pt) = ct. In the multi-user setting, the adversary is given multiple IV-output pairs \{ (iv, ct) \}_i, and tries to find any pt such that AIM(iv, pt) = ct, for some $i$.

### 5.1 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. As mentioned in Section 3, we mainly consider the Gröbner basis attack and the XL attack using a single affine map.

**THE GRÖBNER BASIS ATTACK.** 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 $Mer[i]$ and let $v_i$ denote the output of the linear component $L_{iv, i}$ for $i = 1, 2, \ldots, \ell$. Then, we obtain the following system of equations

\[
\begin{align*}
u_i = pt^{2^{v_i} - 1} & \quad \text{for } i = 1, 2, \ldots, \ell, \\
v_i = L_{iv, i}(u_i) & \quad \text{for } i = 1, 2, \ldots, \ell, \\
pt \oplus ct = (v_1 \oplus \cdots \oplus v_\ell \oplus b_r)^{2^{v_r} - 1} & \quad \text{for } r = 1 \ldots \ell,
\end{align*}
\]

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

\[
\prod_{i=1}^{\ell} \left( \frac{1 - z^{2^{v_i} - 1}}{1 - z} \right) \left( \frac{1 - z^{2^{v_r} - 1}}{1 - z} \right)^{\ell} = \frac{1 - z^{2^n}}{1 - z^{2^{v_r} - 1}}.
\]

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

\[
\left( \frac{2\ell + 1 + 2^n}{2^n} \right)^{\omega} > 2^n
\]

for $\ell \geq 2$.

We can also construct a system of equations over $\mathbb{F}_2$ as discussed in Section 3.1. The corresponding Hilbert series is the same as obtained from an $(\ell + 1)$-round Even-Mansour cipher. We perform Gröbner basis computation on AIM with $\ell = 2, 3$ for toy parameters, summarizing the result in Figure 5. 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.

**THE XL ATTACK.** As mentioned in Section 3.2, we observe that the systems of equations defined by the inverse and the Mersenne S-boxes are dense for toy parameters. Letting $T = \sum_{i=1}^\ell (\omega)$, 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 Rain3) 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 (3) with $r = 3$ and $v = 5$. Figure 6 shows the estimated degree of regularity and the solving degree for Rain3. 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 Rain3 and AIM with $\ell = 2$, and for the basic systems of Rain 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 $v$, determined by the number of linearly independent quadratic equations of their
Figure 5: Degree of regularity $d_{\text{reg}}$ estimated by (3) and the solving degree $sd$ for AIM with $\ell = 2, 3$ using Mersenne S-boxes.

Figure 6: Degree of regularity $d_{\text{reg}}$ estimated by (3) and the solving degree $sd$ for the Rain$_3$ cipher.

$sd$ (basic) $d_{\text{reg}}$ (basic) $sd$ (full) $d_{\text{reg}}$ (full)

5.2 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 [69].

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 [40]. In this section, we investigate if any specialized 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 $O(2^{0.3687(\ell+1)n})$, 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 [42] 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 [50] and...
The adversary is allowed to evaluate AIM. Table 3: Analyses of the Gröbner basis attack and the XL attack for AIM. The probability $d_{reg}$ 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.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$n$</th>
<th>$\nu$</th>
<th>$d_{reg}$</th>
<th>Time (bits)</th>
<th>$D$</th>
<th>Time (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM-I</td>
<td>128</td>
<td>20</td>
<td>222.8</td>
<td>12</td>
<td>148.0</td>
<td></td>
</tr>
<tr>
<td>AIM-III</td>
<td>192</td>
<td>3</td>
<td>27</td>
<td>15</td>
<td>194.1</td>
<td></td>
</tr>
<tr>
<td>AIM-V</td>
<td>256</td>
<td>45</td>
<td>530.3</td>
<td>19</td>
<td>266.1</td>
<td></td>
</tr>
<tr>
<td>Rain 3</td>
<td>128</td>
<td>14</td>
<td>168.5</td>
<td>10</td>
<td>127.9</td>
<td></td>
</tr>
<tr>
<td>Rain 4</td>
<td>192</td>
<td>5</td>
<td>235.9</td>
<td>12</td>
<td>162.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>24</td>
<td>303.1</td>
<td>13</td>
<td>183.9</td>
<td></td>
</tr>
</tbody>
</table>

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.

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

$$\Pr_{A,b} [\text{MDP}^{\text{AIM}} > \gamma] < 2^{-\lambda}$$

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 $\lambda$.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>128</th>
<th>192</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\log \gamma$</td>
<td>-118.4</td>
<td>-178.0</td>
<td>-245.9</td>
</tr>
</tbody>
</table>

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

**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

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

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

**5.4 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 AIM with \( \ell > 2 \) is similarly proved.

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

\[
\text{AIM}(pt) = S_3(A_1 \cdot S_1(pt) \oplus A_2 \cdot S_2(pt) \oplus b) \oplus pt
\]

for \( pt \in \{0, 1\}^n \), 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 \( A \) is allowed to choose any value \( ct \in \{0, 1\}^n \) 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 \( A \) makes \( q \) permutation queries in total. If \( \mathcal{A} \) succeeds in finding all the S-box evaluations that make up an evaluation \( \text{AIM}(pt) = ct \) for some \( pt \in \{0, 1\}^n \), 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 \( \text{Adv}^\text{pre}_{\text{AIM}}(q) \), 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 \( \text{AIM}(pt) = ct \) 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), there should be queries \( S_2(x) = y \) for some \( y \) and \( S_3(z) = x \oplus ct \) 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

\[
S_3(A_1 \cdot S_1(x) \oplus A_2 \cdot S_2(x) \oplus B) = x \oplus ct
\]

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

\[
S_3 : A_1 \cdot y_1 \oplus A_2 \cdot y_2 \oplus b \mapsto x \oplus ct
\]

is included in the wish list \( \mathcal{W} \). In order for an \( S_3 \)-query to complete an evaluation \( \text{AIM}(pt) = ct \) 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

\[
\text{Adv}^\text{pre}_{\text{AIM}}(q) \leq \frac{2q}{2^n-q}.
\]

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.

### 6 Performance Evaluation

**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 -03 optimization level. For the instantiation of the XOF, we use SHAKE in XKCP library\(^5\). We use SHAKE128 for AlMer-I, and SHAKE256 for AlMer-III and AlMer-V. Our experiments are measured in Intel Xeon E5-1650 v3 \( \times 3.50\)GHz 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-Thread and Turbo Boost features disabled.

**PERFORMANCE OF AlMer.** As mentioned in Section 2.3, AIM has been designed to take full advantage of optimization by repeated multipliers to reduce the number of \( \alpha \) values. Due to this technique, the overall performance of the signature scheme is improved in terms of both the signature size and

\(^5\)https://github.com/XKCP/XKCP
Table 5: Performance of AIMer for various parameter sets.

In Table 6, AIMer is compared to the state-of-the-art.

Table 6: Performance of AIMer for various parameter sets.

Table 7: Comparison of AIMer to existing (post-quantum) signature schemes at 128-bit security level.

References


https://github.com/IAIK/bnpp_helium_signatures

https://github.com/IAIK/rainier-signatures

https://github.com/dkales/banquet

†: measurements are from this paper.

Table 7: 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.

| Scheme       | $|\text{pk}|$ (B) | $|\text{sig}|$ (B) | Sign (ms) | Verify (ms) | Size (B) |
|--------------|----------------|----------------|------------|-------------|----------|
| Dilithium2   | 1312           | 2420           | 0.10       | 0.03        |
| Falcon-512   | 897            | 690            | 0.27       | 0.04        |
| SPHINCS+128f | 32             | 7856           | 315.74     | 0.35        |
| SPHINCS+128f | 32             | 17088          | 16.32      | 0.97        |
| Picnic1-L1-full | 32         | 50925          | 1.16       | 0.91        |
| Picnic3-L1   | 32             | 12463          | 5.83       | 4.24        |
| Banquet      | 32             | 19776          | 7.09       | 5.24        |
| Limbo-AES128† | 32             | 21520          | 2.70       | 2.00        |
| Rainier1     | 32             | 8544           | 0.97       | 0.89        |
| BN++Rain3    | 32             | 6432           | 0.83       | 0.77        |
| AIMer-I      | 32             | 5904           | 0.82       | 0.78        |

*: SHAKE-simple

The signing time. The performance of AIMer is summarized in Table 5. Parameter sets (i.e., the number of parties $N$ and the number of parallel repetitions $\tau$) for various security levels are chosen in the same way of [47]. We observe that AIMer enjoys the best trade-off between the signature size and the signing/verification time. In Table 6, AIMer is compared to the state-of-the-art.

In Table 6, AIMer is compared to the state-of-the-art.

Rainier signature scheme combined with the BN++ proof system (denoted BN++Rain, where $r \in \{3, 4\}$) with all the optimizations from [47] applied at the 128-bit security level. AIMer-I enjoys 5.14 to 8.21% shorter signature size than BN++Rain with similar signing and verification time. Compared to BN++Rain, 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.

Table 6: Performance of AIMer, BN++Rain3, and BN++Rain4 at 128-bit security level.

The number of parallel repetitions $\tau$ is set to 16 for ZKP-based signature schemes.


A 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

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

The maximal differential probability is defined as follows.

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

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

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

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

$$\text{DP}(\Delta x, \Delta z) = \text{DP}^{\text{Lin} \circ \text{Mer}[\varepsilon_1, \ldots, \varepsilon_\ell]}(\Delta x, \Delta z)$$

where the function in superscript is omitted if it is obvious (e.g., $\text{Lin} \circ \text{Mer}[\varepsilon_1, \ldots, \varepsilon_\ell]$ for $\Delta x \to \Delta z$). Then we want to upper bound

$$\text{MDP}^{\text{AIM}} = \max_{\Delta x \neq 0, \Delta y^*} \text{DP}(\Delta x^*, \Delta y^*).$$

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

Let $\operatorname{Im}(\Delta x^*) = \{\Delta y : \Delta x = \Delta x^*\}_{x \in \mathbb{F}_2^n}$. Note that $|\operatorname{Im}(\Delta x^*)| \leq 2^n$ for any $\Delta x^*$. For a fixed $n \times \ell n$-matrix $A$, we have

$$\text{DP}(\Delta x^*, \Delta z^*) = \sum_{\Delta y \in \operatorname{Im}(\Delta x^*)} \text{DP}(\Delta x^*, \Delta y) \leq \sum_{\Delta y \in \operatorname{Im}(\Delta x^*)} \min_{1 \leq i \leq \ell} \text{MDP}^\text{Mer}[\varepsilon_i].$$

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

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

where $\mathcal{B} = \operatorname{Bin}(|\operatorname{Im}(\Delta x^*)|, \Pr_A[\Delta y \in A^{-1}(\Delta x^*)])$ is a binomial distribution. The probabilities $\Pr_A[\Delta y \in A^{-1}(\Delta x^*)]$ for $\ell \in \{2, 3\}$ are summarized in Table 8, and the proof is given in Appendix C.

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

Table 8: $\Pr_A[\Delta y \in A^{-1}(\Delta x^*)]$ for $\ell \in \{2, 3\}$.

For a binomial distribution $\mathcal{B}' = \operatorname{Bin}(2^n, 1/2^n + 2/2^{2n})$, we
have
\[
\Pr[A \mid \text{DP}(\Delta x^*, \Delta z) > (1 + \delta)(1 + 2/2^n)\varepsilon] \\
\leq \Pr_{X', \Delta z^*} [X' > (1 + \delta)(1 + 2/2^n)] \\
\leq \Pr_{X', \Delta z^*} [X' > (1 + \delta)(1 + 2/2^n)] \\
\leq \left( \frac{e^\delta}{(1 + \delta)^{1 + \delta}} \right)^{1 + 2/2^n}
\]
by the Chernoff bound. Now, DP(\Delta x^*, \Delta v^*) can be expressed in terms of DP(\Delta x^*, \Delta z^*) as follows.

- If \Delta v^* = \Delta x^*, then
  \[
  \text{DP}(\Delta x^*, \Delta v^*) = \text{DP}^{\Delta x \to \Delta z}(\Delta x^*, 0).
  \]
- Otherwise, for a fixed \( B \in \mathbb{F}_2^n \),
  \[
  \text{DP}(\Delta x^*, \Delta v^*) = \sum_{\Delta z} \text{DP}(\Delta x^*, \Delta z) \\
  \times \Pr_{X, z} \left[ H_b(X \oplus \Delta x^*) \oplus H_b(X) = \Delta v^* \oplus \Delta z \right.
  \left. \quad F(X \oplus \Delta x^*) \oplus F(X) = \Delta z \right]
  \]
where \( F(x) = A \cdot \text{Mer}(e_1, \ldots, e_{\ell}) \cdot (x), G = \text{Mer}[e], \) and \( H_b(x) = G(F(x) \oplus b) \). The vector \( b \in \mathbb{F}_2^n \) is from the constant in affine layers.

We remark that
\[
\mathbb{E}_b[\text{DP}(\Delta x^*, \Delta v^*)] = \sum_{\Delta z} \text{DP}(\Delta x^*, \Delta z) \text{DP}(\Delta z, \Delta v^*)
\]
but we do not use this equation since \( b \) is public. For \( \delta' > 0 \), assuming the independence, we have
\[
\Pr_{B''}[\text{DP}(\Delta x^*, \Delta v^*) > (1 + \delta')(2^{\delta'} - 2) \max_{\Delta z} \text{DP}(\Delta x^*, \Delta z)]
\leq \Pr_{X'' \sim B''} [X'' > (1 + \delta')(2^{\delta'} - 2)]
\leq \left( \frac{e^{\delta'}}{(1 + \delta')^{1 + \delta'}} \right)^{2^{\delta'} - 2}
\]
where \( B'' = \text{Bin}(2^n, \max_{\Delta z \neq 0} \text{DP}(\Delta z, \Delta v^*)) \) is a binomial distribution.

For any \( \Delta x^* \neq 0, \Delta v^* \), we have
\[
\Pr_{A,B}[\text{DP}(\Delta x^*, \Delta v^*) > (1 + \delta)(1 + 2/2^n)(1 + \delta')(2^{\delta'} - 2)\varepsilon] \\
\leq \Pr_{A,\Delta z} [\max_{\Delta z} \text{DP}(\Delta x^*, \Delta z) > (1 + \delta)(1 + 2/2^n)\varepsilon] \\
\times \left( \frac{e^{\delta'}}{(1 + \delta')^{1 + \delta'}} \right)^{2^{\delta'} - 2} \\
\leq \left( \frac{e^{\delta}}{(1 + \delta)^{1 + \delta}} \right)^{1 + 2/2^n} \left( \frac{e^{\delta'}}{(1 + \delta')^{1 + \delta'}} \right)^{2^{\delta'} - 2} 
\]
We set the bound at
\[
\left( \frac{e^{\delta}}{(1 + \delta)^{1 + \delta}} \right)^{1 + 2/2^n} \left( \frac{e^{\delta'}}{(1 + \delta')^{1 + \delta'}} \right)^{2^{\delta'} - 2} = 2^{-\lambda}
\]
for security parameter \( \lambda \) and summarize the values of \( \log \gamma \) such that
\[
\Pr_{A,B}[\text{MDP}^{\text{AIM}} > \gamma] < 2^{-\lambda}
\]
according to its security level in Table 4. We remark that \( \gamma > 2^{-\lambda} \) for each \( \lambda \) does not imply the feasibility of differential cryptanalysis.

### B 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 \((\alpha, \beta) \in \mathbb{F}_{2^n} \times \mathbb{F}_{2^n}^\ast\), the linear probability of \( \text{Mer}[e] \) is defined by
\[
\text{LP}^{\text{Mer}[e]}(\alpha, \beta) \overset{\text{def}}{=} \frac{2}{2^n} \cdot \left| \{ x \in \mathbb{F}_{2^n} : \alpha \cdot x = \beta \cdot \text{Mer}[e](x) \} \right| - 1.
\]

The maximal linear probability is defined as follows.
\[
\text{MLP}^{\text{Mer}[e]}(\alpha, \beta) \overset{\text{def}}{=} \max_{\alpha, \beta \neq 0} \text{LP}^{\text{Mer}[e]}(\alpha, \beta).
\]

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 \( \text{MLP}^f \leq (d - 1)/2^{n/2} \) [48]. Specifically, the maximum linear probability of a Mersenne S-box is bounded by
\[
\text{MLP}^{\text{Mer}[e]} \leq \frac{2^n - 2}{2^{n/2}}.
\]

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
\[
\text{LP}(\alpha, \gamma) = \text{LP}^{\text{Lin} \circ \text{Mer}[e_1, \ldots, e_{\ell}]}(\alpha, \gamma)
\]
where the function in superscript is omitted if it is obvious (e.g., Lin \( \circ \text{Mer}[e_1, \ldots, e_{\ell}] \) for \( \alpha \to \gamma \)). Then the bias of a trail
\[
\alpha^* \to \beta^* \to \gamma^* \to \delta^* \to \varepsilon^*
\]
is computed as
\[
(LP(\alpha^*, \beta^*) LP(\beta^*, \gamma^*) LP(\gamma^*, \delta^*) LP(\delta^*, \epsilon^*))^2
\leq (LP(\alpha^*, \beta^*) LP(\gamma^*, \delta^*))^2
\leq \left( \min_{1 \leq \ell \leq \ell} MLP_{\text{Mer}(\epsilon^*)} \right)^2
\leq \min_{1 \leq \ell \leq \ell} \frac{(2^n - 2)(2^{\epsilon^*} - 2)}{2^n}
\]
assuming independence of each edge. When
\[
\min_{1 \leq \ell \leq \ell} \frac{(2^n - 2)(2^{\epsilon^*} - 2)}{2^n} < 2^n,
\]
the bias of AIM is smaller $2^{-n}$, and the amount of data required for linear cryptanalysis becomes at least $2^n$.

C Computing $\Pr_A[\Delta y \in A^{-1}(\Delta z^*)]$ 

Let $L_n$ denote a set of $n \times n$ invertible matrices over $\mathbb{F}_2$. Then we have
\[
\Pr_{L \in L_n}[Lx = y] = \frac{1}{2^n - 1}
\]
for nonzero vectors $x, y \in \mathbb{F}_n^*$. 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 $A_1, A_2 \in L_n$. Then
\[
\Pr_A[\Delta y = \Delta z^*] = \Pr_{A_1, A_2}[A_1 \Delta y_1 + A_2 \Delta y_2 = \Delta z^*] = \Pr_{A_1, A_2}[A_2 \Delta y_2 \neq \Delta z^* \land A_1 \Delta y_1 = A_2 \Delta y_2 + \Delta z^*]
\]
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
\[
\Pr_A[\Delta y = 0] = \Pr_A[A_1 \Delta y_1 = A_2 \Delta y_2] = \frac{1}{2^n - 1}.
\]
On the other hand, if $\Delta z^* \neq 0$ then
\[
\Pr_A[\Delta y = \Delta z^*] = \Pr_{A_2}[A_2 \Delta y_2 \neq \Delta z^*] \times \Pr_A[A_1 \Delta y_1 = A_2 \Delta y_2 + \Delta z^*[A_2 \Delta y_2 \neq \Delta z^*] = \frac{2^n - 2}{(2^n - 1)^2}.
\]

CASE $\ell = 3$. The $n \times 3n$ matrix $A$ is written as
\[
A = [A_1 | A_2 | A_3]
\]
where $A_1, A_2, A_3 \in L_n$. Then
\[
\Pr_A[\Delta y = \Delta z^*] = \Pr_{A_1, A_2, A_3}[A_1 \Delta y_1 + A_2 \Delta y_2 + A_3 \Delta y_3 = \Delta z^*] = \Pr_{A_1, A_2, A_3}[A_1 \Delta y_1 = A_2 \Delta y_2 + A_3 \Delta y_3 \lor A_1 \Delta y_1 = A_2 \Delta y_2 + A_3 \Delta y_3 + \Delta z^*]
\]
since $\Delta y_1 \neq 0$ for $\Delta x \neq 0$. If $\Delta z^* = 0$, then we have
\[
\Pr_A[A_2 \Delta y_2 \neq A_3 \Delta y_3] = \frac{2^n - 2}{(2^n - 1)^2}.
\]
For any nonzero $a$, we have
\[
\Pr_A[A_1 \Delta y_1 = a | A_2 \Delta y_2 + A_3 \Delta y_3 = a] = \frac{1}{2^n - 1}.
\]
Combining (4) and (5), we obtain
\[
\Pr_A[\Delta y = 0] = \Pr_{A_2, A_3}[A_2 \Delta y_2 \neq A_3 \Delta y_3] \times \Pr_A[A_1 \Delta y_1 = A_2 \Delta y_2 + A_3 \Delta y_3 | A_2 \Delta y_2 \neq A_3 \Delta y_3]
\]
\[
= \frac{2^n - 2}{(2^n - 1)^2}.
\]
Suppose that $\Delta z^* \neq 0$. Since $\Delta y_2$ is nonzero, we have
\[
\Pr_{A_2, A_3}[A_2 \Delta y_2 + A_3 \Delta y_3 \neq \Delta z^*] = \Pr_{A_2, A_3}[A_2 \Delta y_2 \neq \Delta z^*] + \Pr_{A_3}[A_3 \Delta y_3 = \Delta z^*]
\]
\[
= \frac{2^n - 2}{(2^n - 1)^2} + \frac{1}{2^n - 1}.
\]
Therefore, we obtain
\[
\Pr_A[\Delta y = \Delta z^*] = \frac{(2^n - 2)^2}{(2^n - 1)^3} + \frac{1}{(2^n - 1)^2}.
\]

D Full Description of AlMer

The AlMer 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 $(iv, ct)$ and a secret key $pt$ such that $ct = \text{AIM}(iv, pt)$. The signing algorithm takes as input the pair of secret and public keys $(pt, (iv, ct))$ and a message $m$ and outputs the corresponding signature $\sigma$. The verification algorithm takes as input the public key $(iv, ct)$, a message $m$ and a signature $\sigma$ and outputs either Accept or Reject. We describe the AlMer signing and verification algorithms in Algorithm 1 and 2, respectively.
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 $\alpha$ 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.

$$pt \cdot t_i = pt^{2^i}$$

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

$$pt \cdot \text{Lin}[iv](t) = (\text{Lin}[iv](t))^{2^e} + ct \cdot \text{Lin}[iv](t)$$

where $t_i, i = 1, 2, \ldots, \ell$, is the output of the $i$-th S-box and $t \equiv [t_1| \ldots |t_\ell]$. Since every multiplication triple shares the same multiplier $pt$, a single value of $\alpha$ 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 $\Delta z$ in the signature. This technique is similar with multiplications with public output, suggested in BN++.

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

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

where $t^{(i)} \in \mathbb{F}_2^{\ell}$ is the output shares of the first $\ell$ S-boxes for the $i$-th party: $t^{(i)} = [t^{(i)}_1| \ldots |t^{(i)}_\ell]$.

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

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

**Other Symmetric Primitives In Use.** The SHAKE128 (resp. SHAKE256) XOF is used to instantiate hash functions Commit, $H_1$, $H_2$ and pseudorandom generators Expand and ExpandTape in the signature scheme for $\lambda = 128$ (resp. $\lambda \in \{192, 256\}$). Sample(tape) samples an element from a random tape tape, which is an output of ExpandTape, tracking the current position of the tape.
Algorithm 1: Sign(pt, (iv, ct), m) - A1Mer signature scheme, signing algorithm.

// Phase 1: Committing to the seeds and the execution views of the parties.
1. Sample a random salt $\sigma \leftarrow \{0, 1\}^\lambda$.
2. Compute the first $\ell$ S-boxes’ outputs $t_1, \ldots, t_\ell$.
3. Derive the binary matrix $A_{iv} \in \mathbb{F}_2^{n \times \ell}$ and the vector $b_{iv} \in \mathbb{F}_2^n$ from the initial vector iv.
4. For each parallel execution $k \in [\ell]$ do
   5. Sample a root seed $\sigma_k \leftarrow \{0, 1\}^\lambda$.
   6. Compute parties’ seeds $\sigma_k^{(1)}, \ldots, \sigma_k^{(N)}$ as leaves of binary tree from $\sigma_k$.
   7. For each party $i \in [N]$ do
      8. Commit to seed: $\text{com}_k^{(i)} \leftarrow \text{Commit}(\text{salt}, i, \sigma_k^{(i)})$.
      9. Expand random tape: $\text{tape}_k^{(i)} \leftarrow \text{ExpandTape}(\text{salt}, i, \sigma_k^{(i)})$.
      10. Sample witness share: $\text{pt}_k^{(i)} \leftarrow \text{Sample}(\text{tape}_k^{(i)})$.
   8. Compute witness offset and adjust first witness: $\Delta \text{pt}_k \leftarrow \text{pt} - \sum_i \text{pt}_k^{(i)}$, $\text{pt}_k^{(1)} \leftarrow \text{pt}_k^{(1)} + \Delta \text{pt}_k$.
   9. For each S-box with index $j$ do
      10. If $j \leq \ell$ then
          11. For each party $i$, sample a S-box output: $i_{k,j}^{(i)} \leftarrow \text{Sample}(\text{tape}_k^{(i)})$.
          12. Compute output offset and adjust first share: $\Delta t_{k,j} = t_j - \sum_j i_{k,j}^{(i)} - i_{k,j}^{(1)} - \Delta t_{k,j}$.
          13. For each party $i$, set $x_{k,j}^{(i)} = i_{k,j}^{(i)}$ and $z_{k,j}^{(i)} = (\text{pt}_k^{(i)})^2 + \Delta t_{k,j}$.
      14. If $j = \ell + 1$ then
          15. For $i = 1$, set $x_{k,j}^{(i)} = A_{iv} \cdot i_{k,j}^{(i)} + b_{iv}$ where $i_{k,j}^{(i)} = [i_{k,j}^{(i)}]_{k=1}^{\ell}$ is the output shares of the first $\ell$ S-boxes.
          16. For each party $i \in [N] \{1\}$, set $x_{k,j}^{(i)} = A_{iv} \cdot i_{k,j}^{(i)}$.
          17. For each party $i$, set $z_{k,j}^{(i)} = (x_{k,j}^{(i)})^2 + \text{ct} \cdot x_{k,j}^{(i)}$.
   18. For each party $i$, set $a_k^{(i)} \leftarrow \text{Sample}(\text{tape}_k^{(i)})$.
   19. Compute $a_k = \sum_{i=1}^{N} a_k^{(i)}$.
   20. Set $c_k = a_k \cdot \text{pt}$.
   21. For each party $i$, set $c_k^{(i)} \leftarrow \text{Sample}(\text{tape}_k^{(i)})$.
   22. Compute offset and adjust first share: $\Delta c_k = c_k - \sum c_k^{(i)}$, $c_k^{(1)} \leftarrow c_k^{(1)} + \Delta c_k$.
   23. Set $\sigma_1 \leftarrow (\text{salt}, ((\text{com}_k^{(i)})_{i \in [N]}), \Delta \text{pt}_k, \Delta c_k, (\Delta t_{k,j})_{j \in [\ell]} k \in [\ell])$.

// Phase 2: Challenging the checking protocol.
24. Compute challenge hash: $h_1 \leftarrow H_1(m, iv, ct, \sigma_1)$.
25. Expand hash: $((\epsilon_{k,j})_{j \in [\ell + 1]} k \in [\ell] \} \leftarrow \text{Expand}(h_1)$ where $\epsilon_{k,j} \in \mathbb{F}_2$.

// Phase 3. Commit to the simulation of the checking protocol.
26. For each repetition $k$ do
   27. Simulate the triple checking protocol as in Section 2.3 for all parties with challenge $\epsilon_{k,j}$. The inputs are
   28. $(x_{k,j}^{(i)}, p_{k,j}^{(i)}, x_{k,j}^{(i)})_{j \in [\ell + 1]}, d_{k,j}^{(i)}, b_{k,j}^{(i)}, c_{k,j}^{(i)}$, where $b_{k,j}^{(i)} = p_{k,j}^{(i)}$, and let $c_{k,j}^{(i)}$ and $v_{k,j}^{(i)}$ be the broadcast values.
   29. Set $\sigma_2 \leftarrow (\text{salt}, ((\text{com}_k^{(i)})_{i \in [N]} k \in [\ell])$.

30. Compute challenge hash: $h_2 \leftarrow H_2(h_1, \sigma_2)$.
31. Expand hash: $((\tilde{i}_k) k \in [\ell] \} \leftarrow \text{Expand}(h_2)$ where $\tilde{i}_k \in [N]$.

// Phase 5. Opening the views of the MPC and checking protocols.
32. For each repetition $k$ do
   33. Set seeds $\tilde{k} \leftarrow \{\log_2(N)\}$ nodes to compute $\text{seed}_{\tilde{k}}^{(i)}$ for $i \in [N] \{\tilde{i}\}$.
   34. Output $\sigma \leftarrow (\text{salt}, h_1, h_2, (\text{seeds}_{\tilde{k}}, \text{com}_{\tilde{k}}^{(i)})_{\tilde{i} \in [\ell]}, (\Delta t_{\tilde{i}, j})_{j \in [\ell]}, (\text{com}_{\tilde{k}}^{(i)})_{k \in [\ell]}$).
Algorithm 2: Verify((iv, ct), m, σ) - AIMer signature scheme, verification algorithm.

1. Parse σ as \(\text{salt}, h_1, h_2, (\text{seeds}_k, \text{com}_k, c_k, \Delta p_k, c_k, (\Delta h_k, j)_{j \in [\ell]}, a_k^{(i)})_{k \in [\ell]}\).
2. Derive the binary matrix \(A_{iv} \in \mathbb{F}_2^{N \times \ell}\) and the vector \(b_{iv} \in \mathbb{F}_2^\ell\) from the initial vector iv.
3. Expand hashes: \((\langle \varepsilon, k, j \rangle_{j \in [\ell+1]} \rangle_{k \in [\ell]} \leftarrow \text{Expand}(h_1)\) and \((\tilde{h}_k)_{k \in [\ell]} \leftarrow \text{Expand}(h_2)\).
4. For each parallel repetition \(k \in \{\ell\}\)
   5. Uses seeds\(_k\) to recompute \(seed_k^{(i)}\) for \(i \in [N] \setminus \{\tilde{h}_k\}\).
   6. For each party \(i \in [N] \setminus \{\tilde{h}_k\}\)
      7. Recompute \(\text{com}_k^{(i)} \leftarrow \text{Commit}(\text{salt}, k, i, seed_k^{(i)})\).
      8. \(pt_k^{(i)} \leftarrow \text{ExpandTape}(\text{salt}, k, i, seed_k^{(i)})\) and \(tape_k^{(i)} \leftarrow \text{Sample}(\text{tape}_k^{(i)})\).
      9. If \(i = 1\)
         10. Adjust first share: \(pt_k^{(i)} \leftarrow pt_k^{(i)} + \Delta p_k\)
   11. For each S-box with index \(j\)
      12. If \(j \leq \ell\)
         13. Sample a S-box output: \(t_{k,j}^{(i)} \leftarrow \text{Sample}(\text{tape}_k^{(i)})\).
         14. If \(i = 1\)
            15. Adjust first share: \(t_{k,j}^{(i)} \leftarrow t_{k,j}^{(i)} + \Delta t_{k,j}\).
            16. Set \(x_{k,j}^{(i)} = t_{k,j}^{(i)}\) and \(s_{k,j}^{(i)} = (pt_k^{(i)})^2\).
         17. If \(j = \ell + 1\)
            18. If \(i = 1\)
               19. Set \(x_{k,j}^{(i)} = A_{iv} \cdot t_{k,s}^{(i)} + b_{iv}\) where \(t_{k,s} = [t_{k,1}^{(i)}, \ldots, t_{k,s}^{(i)}]\) is the output shares of the first \(\ell\) S-boxes.
            20. Else
               21. Set \(x_{k,j}^{(i)} = A_{iv} \cdot t_{k,s}^{(i)}\).
               22. Set \(z_{k,j}^{(i)} = (x_{k,j}^{(i)})^2 + \text{ct} \cdot x_{k,j}^{(i)}\).
      23. Set \(a_k^{(i)} \leftarrow \text{Sample}(\text{tape}_k^{(i)})\) and \(c_k^{(i)} \leftarrow \text{Sample}(\text{tape}_k^{(i)})\).
      24. If \(i = 1\)
         25. Adjust first share: \(c_k^{(i)} \leftarrow c_k^{(i)} + \Delta c_k\).
   26. Set \(\sigma_1 \leftarrow \left(\text{salt}, \left(\left(\text{com}_k^{(i)}\right)_{i \in [N]}, \Delta p_k, \Delta c_k, (\Delta h_k, j)_{j \in [\ell]}\right)_{k \in [\ell]}\right)\).
   27. Set \(h_1' \leftarrow H_1(m, iv, ct, \sigma_1)\).
   28. For each parallel execution \(k \in \{\ell\}\)
      29. For each party \(i \in [N] \setminus \{\tilde{h}_k\}\)
         30. Simulate the triple checking protocol as defined in Section 2.3 for all parties with challenge \(\varepsilon_{k,j}\). The inputs are \((x_{k,j}^{(i)}, pt_k^{(i)} \cdot z_{k,j}^{(i)}_{j \in [\ell+1]} a_k^{(i)} b_k^{(i)} c_k^{(i)}), where b_k^{(i)} = pt_k^{(i)}\), and let \(\alpha_k^{(i)}\) and \(\nu_k^{(i)}\) be the broadcast values.
      31. Compute \(v_k^{(i)} = 0 - \sum_{j \neq k} v_j^{(i)}\).
      32. Set \(\sigma_2 \leftarrow \left(\text{salt}, \left(\left(\alpha_k^{(i)}, \nu_k^{(i)}\right)_{i \in [N]}\right)_{k \in [\ell]}\right)\).
      33. Set \(h_2' = H_2(h_1', \sigma_2)\).
   34. Output Accept if \(h_1 = h_1'\) and \(h_2 = h_2'\).
   35. Otherwise, output Reject.