IACR Communications in Cryptology
ISSN 3006-5496, Vol. 3, No. 2, 27 pages.

https://doi.org/10.62056/av11fh888

Check for updates

Secure Multi-party Matrix Invertibility Testing over Small Finite Fields

Seungwoo Han, Jooyoung Lee, Seungmin Park and Mincheol Son

KAIST, Daejeon, South Korea

Abstract. Multi-party matrix invertibility testing over finite fields of small order or characteristic is a pivotal operation for thresholdizing multivariate quadratic (MQ) signature schemes. However, achieving perfect privacy in a constant number of rounds remains a challenge: existing solutions either suffer from information leakage, failing to provide perfect privacy, or require high computational and communication overhead, in particular, when $p \leq n$, where p and n denote the characteristic of the underlying field and the matrix size, respectively.

To address this limitation, we propose two protocols for multi-party testing of matrix invertibility. The first protocol extends the Cramer-Damgård protocol to fields of small order by employing the field lifting technique. The second protocol is based on multi-party computation of the Samuelson-Berkowitz algorithm, specifically designed for fields of small characteristic. Both protocols are formalized in the arithmetic black-box (ABB) model with Shamir's secret sharing scheme.

We show that both protocols achieve perfect privacy while allowing for a tradeoff between input-independent offline rounds and input-dependent online rounds, where the input corresponds to the shared matrices. Specifically, the first protocol runs in 7 offline rounds with communication complexity $O(Nn^4)$ and in 3 online rounds with communication complexity $O(n^4)$, and the second protocol runs in 3 offline rounds with communication complexity $O(n^4)$ and in 9 online rounds with communication complexity $O(n^4)$, where n is the matrix size and N is the number of parties.

Keywords: Multi-party Computation · Linear Algebra · Matrix Invertibility Testing

1 Introduction

Secure execution of linear algebra algorithms in the multi-party setting is a cornerstone of privacy-preserving computation with applications including zero-knowledge proofs via MPC-in-the-Head, private set intersection (PSI), privacy-preserving machine learning (PPML), and threshold multivariate quadratic (MQ) signatures. For instance, secure linear regression may require solving a secret-shared normal-equation system or inverting a secret-shared matrix during training. In particular, matrix invertibility testing over finite fields of small order or characteristic serves as a fundamental building block for thresholdizing MQ signature schemes [CEN25, ABCN26], where the underlying MQ schemes include UOV [BCD+24] and MAYO [BCC+23].

In general, an MPC protocol is designed to achieve perfect privacy, ensuring that no private information is leaked to an adversary during the execution of the protocol. However, while several constant-round threshold MQ signature protocols have been proposed [CS19,

Jooyoung Lee was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00399491, Development of Privacy-Preserving Multiparty Computation Techniques for Secure Multiparty Data Integration).

E-mail: swhan@kaist.ac.kr (Seungwoo Han), hicalf@kaist.ac.kr (Jooyoung Lee), smpak@kaist.ac.kr (Seungmin Park), encrypted.def@kaist.ac.kr (Mincheol Son)

This work is licensed under a "CC BY 4.0" license.

Received: 2026-05-03 Accepted: 2026-06-30

CEN25], they often fail to achieve perfect privacy. This information leakage typically originates from the functionality used to test the invertibility of a given matrix A during the computation of its inverse A^{-1}. Recently, Maire and Vergnaud proposed a perfectly private, constant-round protocol for this functionality [MV24]; however, their approach becomes inefficient when $p \leq n$, where p and n denote the characteristic of the underlying field and the size of the matrix, respectively. This constraint is undesirable for typical MQ signature schemes, which rely on computations over a finite field $\mathbb{F}_q$ with a small characteristic such as $p = 2$.

1.1 Related Work

In the plain setting, matrix invertibility testing is typically performed using Gaussian elimination. However, its straightforward application to the multi-party setting poses significant challenges due to its high round complexity and conditional pivoting operations. Instead of employing this sequential algorithm, various constant-round protocols have been proposed [BIB89, CD01, CKP07, MW08, MV24]. In this section, we briefly review these existing protocols and highlight the limitations that our work aims to address.

Bar-Ilan-Beaver Protocol. Bar-Ilan and Beaver [BIB89] proposed several constant-round multi-party protocols for computing inverses in a group, multiplying multiple matrices, and generating random invertible matrices. Given a group $(G, \cdot)$ and a share of an element $g \in G$, denoted $[g]$, the basic idea of computing $[g^{-1}]$ is to mask $[g]$ with a random share $[r]$ and reveal $[r \cdot g]$. In this way, each party can locally compute $(r \cdot g)^{-1}$, and then $[g^{-1}] = (r \cdot g)^{-1} \cdot [r]$. The idea of random masking can be extended to secure matrix invertibility testing: given a matrix $A \in \mathcal{M}_n(\mathbb{F}_q)$, each party generates a share of a random invertible matrix $R \in \mathrm{GL}_n(\mathbb{F}_q)$. Then they jointly compute $A \cdot R$, and hence its determinant. This method is called the one-sided masking protocol. It has been used to construct the first threshold MQ signature [CS19].

Two-sided masking Protocol. Bar-Ilan and Beaver [BIB89] proposed several constant-round multi-party protocols for computing inverses in a group, multiplying multiple matrices, and generating random invertible matrices. Given a group $(G, \cdot)$ and a share of an element $g \in G$, denoted $[g]$, the basic idea of computing $[g^{-1}]$ is to mask $[g]$ with a random share $[r]$ and reveal $[r \cdot g]$. In this way, each party can locally compute $(r \cdot g)^{-1}$, and then $[g^{-1}] = (r \cdot g)^{-1} \cdot [r]$. The idea of random masking can be extended to secure matrix invertibility testing: given a matrix $A \in \mathcal{M}_n(\mathbb{F}_q)$, each party generates a share of a random invertible matrix $R \in \mathrm{GL}_n(\mathbb{F}_q)$. Then they jointly compute $A \cdot R$, and hence its determinant. This method is called the one-sided masking protocol. It has been used to construct the first threshold MQ signature [CS19].

Later, Celi, Escudero, and Niot [CEN25] showed that the one-sided masking protocol potentially reveals the column space of the matrix A, increasing the guessing probability of the oil space. To address this weakness, they proposed a two-sided masking protocol with two random invertible matrices $R, S \in \mathrm{GL}_n(\mathbb{F}_q)$, where the parties jointly compute the determinant of $R \cdot A \cdot S$. The two-sided masking protocol prevents leakage of A's column space, while it still reveals the rank of A. In the context of threshold MQ signatures, leaking the column space of A may reveal information of the secret-key component O. By contrast, although rank leakage means that the protocol does not achieve perfect privacy, there is no known concrete key-recovery attack that exploits only the rank of A.

This two-sided masking protocol is also adopted in [ABCN26]. While their work optimizes the number of rounds after the message is given in threshold MQ signing to single-round, the matrix invertibility test itself is still instantiated with two-sided masking protocol.

In [CEN25], the ideal functionality explicitly outputs the rank of A as leakage. Our protocols instead realize the standard ideal functionality for matrix invertibility testing,

whose only output is the invertibility bit. Since the real execution is perfectly simulatable from this bit alone, the distinguishing advantage from the ideal functionality is zero; in this sense, our protocols achieve perfect privacy.

Cramer-Damgård Protocol. Cramer and Damgård proposed a constant-round protocol for several linear algebra tasks, including determinant computation, rank evaluation, and solving linear systems over finite fields [CD01]. Their protocol for invertibility testing exploits certain properties of the characteristic polynomial, providing strong privacy guarantees.

Given a matrix $A \in \mathcal{M}_n(\mathbb{F}_q)$, let $\chi_A(x)$ denote the characteristic polynomial of A, namely,

$$\chi_A(x) = \det(xI_n - A).$$

Since $\chi_A(x)$ is of degree at most n, it is uniquely determined by its evaluations at $n+1$ distinct field elements. Moreover, the determinant of A corresponds to its constant term, up to a sign:

$$\chi_A(0) = \det(-A) = (-1)^n \det A.$$

In this protocol, $n+1$ distinct public points $z_1, z_2, \ldots, z_{n+1} \in \mathbb{F}_q$ are fixed. For any $z \in \mathbb{F}_q$, the matrix $zI_n - A$ is invertible if and only if z is not an eigenvalue of A. Since A has at most n eigenvalues, a uniformly random z makes $zI_n - A$ invertible with probability at least $1 - n/q$. When q is sufficiently large, matrices $M_i = z_i I_n - A$, $i = 1, \ldots, n+1$, will be invertible with high probability.

The parties jointly compute secret shares of the determinant $[y_i] = [\det(M_i)]$. In this step, the parties are required to share a random invertible matrix and its determinant: this is done by LU decomposition, where a lower-triangular matrix L and an upper-triangular matrix U are sampled uniformly at random and the random matrix is defined as their product, namely $R = L \cdot U$.

Given the pairs $(z_i, [y_i])_{i \in \{1,\ldots,n+1\}}$, the parties reconstruct the shares of $[\chi_A(0)]$ via Lagrange interpolation:

$$[\chi_A(0)] = \sum_{i=1}^{n+1} \lambda_i [y_i] \tag{1}$$

where λ_i's are the public Lagrange coefficients.

This protocol enables the computation of $\det(A)$ regardless of whether A is invertible by evaluating the determinants of $n+1$ matrices in parallel. However, it suffers from several limitations. First, each M_i is singular with a probability approximately n/q. Although the original analysis assumes q to be sufficiently large, this probability becomes non-negligible over small fields. Second, the protocol does not achieve perfect privacy, as the rank of each M_i is leaked during execution. Specifically, M_i being full-rank reveals that z_i is not an eigenvalue of A. When q is small, this information can be exploited to infer certain properties of A. Finally, the distribution of the random mask R, being the product of lower-triangular and upper-triangular matrices, is not uniform over $\mathrm{GL}_n(\mathbb{F}_q)$, which may further compromise privacy.

Maire-Vergnaud Protocol. Maire and Vergnaud [MV24] proposed protocols for secure linear algebra, employing Leverrier's lemma to compute the characteristic polynomial and the determinant of a shared matrix.

Lemma 1 (Leverrier)**.** *Let $A \in \mathcal{M}_n(\mathbb{F}_q)$ be a matrix and let $\chi_A(x) = \det(xI_n - A) = x^n + d_1 x^{n-1} + \cdots + d_n$ be its characteristic polynomial. Let $t_k = \mathrm{Tr}(A^k)$ denote the power*

sums of the eigenvalues of A. Then the coefficients d_k satisfy the following relation:

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ t_1 & 2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ t_{n-1} & t_{n-2} & \cdots & n \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix} = - \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix} \tag{2}$$

If the field characteristic p is greater than n, then the matrix on the left-hand side is invertible. Consequently, the coefficients $d_1, \ldots, d_n$, in particular, $d_n = (-1)^n \det(A)$ can be recovered from $t_1, \ldots, t_n$.

The parties jointly compute the shares of the matrix powers $[A], [A^2], \ldots, [A^n]$, and locally compute the traces $\mathrm{Tr}([A^k]) = [\mathrm{Tr}(A^k)]$. In this way, they obtain a share of M, where M is the lower-triangular matrix from Leverrier's lemma. To solve the system of equations in d_i, $i = 1, \ldots, n$, the parties jointly compute $[M^{-1}]$: they generate secret shares of a random nonzero element r and a random invertible matrix R, compute the shares of a masked matrix $[Z] = [R \cdot M]$, and open $[Z]$. Then each party computes $[M^{-1}] = Z^{-1} \cdot [R]$, from which $[d_n]$ is obtained. Finally, the parties perform the invertibility test by computing $[z] = [r] \cdot [d_n]$ and opening $[z]$: A is invertible if and only if $z \neq 0$.

Unfortunately, this protocol does not work when $p \leq n$ since the matrix in (2) is singular with zero in its diagonal. They claimed that Schönhage's protocol [Sch93], which uses Lemma 1 over an extended domain $\mathbb{F}_q[y]/(y^{n+1})$ with an appropriate companion matrix, can handle this case with higher communication complexity. However, a precise description of this variant and its formal complexity analysis were omitted in their work. We analyze the round and communication complexity of the existing protocols reviewed in this section in Section 4.

1.2 Our contribution

In this paper, we propose two perfect-privacy protocols for multi-party testing of matrix invertibility, each of which addresses a specific limitation of existing protocols. The first protocol extends the Cramer-Damgård protocol [CD01] to fields of small order by employing the field lifting technique. The second protocol is based on multi-party computation of the Samuelson-Berkowitz algorithm [Ber84], which is especially effective for fields with a small characteristic where $p \leq n$. Our protocols are formulated as standalone MPC subroutines for testing the invertibility of a shared matrix, and can be instantiated in any application requiring this functionality.

These protocols are described in the *arithmetic black-box* (ABB) model, and their round and communication complexities are analyzed with Shamir's secret sharing. Consequently, each construction is executed in a constant number of rounds with a tradeoff between (input-dependent) online and (input-independent) offline rounds. In practical executions, the offline phase can often be performed during idle time or instantiated via a trusted-dealer preprocessing model, in which correlated randomness is distributed before the input becomes available. We note that the online/offline terminology used in threshold MQ signatures usually distinguishes the phases before and after the message is given. With respect to this convention, the invertibility-testing subroutine studied in this work is part of the offline phase of threshold signing, since it can be executed before the message is known. Our online/offline distinction is instead internal to the invertibility-testing functionality and separates input-independent preprocessing from the input-dependent computation on the shared matrix A. This distinction is still meaningful in threshold MQ signatures: the offline phase of our subroutine is independent of the signing key and can therefore be prepared without knowing the secret key, for example, by a trusted dealer. In contrast, the online phase involves the secret-shared, key-dependent matrix A, and hence must be carried out by the signers unless key-dependent information is revealed to a third party. We

therefore regard the online phase as the latency-critical part of the protocol and prioritize reducing its round and communication complexity.

Specifically, the first protocol runs in 7 offline rounds with communication complexity $O(Nn^4)$ and in 3 online rounds with communication complexity $O(n^4)$, and the second protocol runs in 3 offline rounds with communication complexity $O(n^4)$ and in 9 online rounds with communication complexity $O(n^4)$, where n is the matrix size and N is the number of parties.

Compared to the invertibility testing protocol from [MV24] combined with Schönhage's algorithm [Sch93], which is applicable to fields of small characteristic, our protocols maintain online communication overhead while minimizing online round complexity. In the threshold implementation of MQ signature schemes such as UOV [BCD+24] and MAYO [BCC+23], matrix invertibility testing is a key online subroutine that directly affects threshold signing performance. Instantiating this subroutine with our protocols results in fewer online rounds than existing perfectly private protocols, leading to lower signing latency. To the best of our knowledge, these protocols are the state-of-the-art perfectly private and efficient MPC protocols for invertibility testing over fields of small order or characteristic. The efficiency of our protocols is summarized in Table 1.

Table 1: Online and offline round complexities and communication cost of invertibility testing protocols. For rounds and communication complexities, online and offline costs are distinguished.

Protocol	Field setting	Rounds		Communication		Perfect privacy
		Online	Offline	Online	Offline	
[BIB89] (Sec. 4.1)	All	2	2	$O(n^2)$	$O(n^2)$	✗
[CEN25] (Sec. 4.1)	All	3	2	$O(n^2)$	$O(n^2)$	✗
[CD01] (Sec. 4.2)	Large order	3	6	$O(n^3)$	$O(n^3)$	✗
[MV24] (Sec. 4.3)	Large char.	8	2	$O(n^3)$	$O(n^3)$	✓
[MV24]+[Sch93] (Sec. 4.4)	Small char.	12	2	$O(n^4)$	$O(n^4)$	✓
isInverseCD(A) (Sec. 5)	Small order	3	7	$O(n^4)$	$O(Nn^4)$	✓
isInverseSB(A) (Sec. 6)	Small char.	9	3	$O(n^4)$	$O(n^4)$	✓

For a concrete comparison over small finite fields, Table 2 reports the online and offline communication costs over the field of order $q = 16$, with matrix dimensions $m \in \{67, 102, 132\}$ and $N = 3$ parties. We include only protocols that are applicable over small finite fields, and exclude the one-sided masking protocol from this comparison because its leakage is generally considered significant. These values of m arise in the threshold signing procedure for the MAYO signature scheme. The constants c_1 and c_2 are chosen so that the failure probability of GenNonzeroElems and GenInvMats is at most 2^{-40}. The results show that the two-sided masking protocol has the lowest communication cost. Among perfectly private approaches, isInverseSB improves over the Schönhage-based protocol in both online and offline communication. The isInverseCD protocol incurs a much larger offline cost, but achieves the smallest online communication among perfectly private protocols, reflecting its different online/offline tradeoff.

Remark. We note that, concurrently and independently of our work, Azevedo-Oliveira, Beraud, and Varjabedian [AOBV26] proposed a similar approach. They proposed two algorithms that securely compute the determinant of a shared matrix. One uses a combination of recent theoretical results on Newton's polynomials and Schönhage's algorithm, while the other is based on the Samuelson–Berkowitz algorithm, similarly to our second approach.

Their first algorithm takes $O(1)$ rounds with $O(n^{3.5})$ communication, and the second takes $O(n^2)$ rounds with $O(n^3)$ communication. Compared to our protocol in Section 6,

Table 2: Concrete communication costs of invertibility testing protocols for the parameter sets $m \in \{67, 102, 132\}$, $q = 16$, and $N = 3$. Online and offline communication costs are reported separately.

m	Protocol	Communication (MB)			Perfect privacy
		Online	Offline	Total	
67	[CEN25] (Sec. 4.1)	0.01	0.33	0.34	✗
	[MV24]+[Sch93] (Sec. 4.4)	193.37	393.57	586.94	✓
	isInverseCD(A) (Sec. 5)	19.80	1514.70	1534.50	✓
	isInverseSB(A) (Sec. 6)	22.85	74.60	97.45	✓
102	[CEN25] (Sec. 4.1)	0.01	0.77	0.79	✗
	[MV24]+[Sch93] (Sec. 4.4)	1077.74	1873.11	2950.85	✓
	isInverseCD(A) (Sec. 5)	105.26	8053.39	8158.65	✓
	isInverseSB(A) (Sec. 6)	124.88	323.37	448.25	✓
132	[CEN25] (Sec. 4.1)	0.02	1.30	1.32	✗
	[MV24]+[Sch93] (Sec. 4.4)	2993.50	4792.80	7786.30	✓
	isInverseCD(A) (Sec. 5)	293.94	22487.33	22781.27	✓
	isInverseSB(A) (Sec. 6)	352.88	806.78	1159.66	✓

their second algorithm avoids the constant-round computation of cumulative powers, thereby reducing communication complexity at the expense of higher round complexity.

2 Preliminaries

2.1 Notation

Let $\mathbb{F}_q$ denote a finite field with q elements, and let $\mathbb{F}_{q^k}$ denote its degree-k extension. For a positive integer n, we write $\mathcal{M}_n(\mathbb{F}_q) = \mathbb{F}_q^{n\times n}$ for the set of $n \times n$ matrices over $\mathbb{F}_q$, and $\mathrm{GL}_n(\mathbb{F}_q) \subseteq \mathcal{M}_n(\mathbb{F}_q)$ for the subset of invertible matrices. We write $r \xleftarrow{\$} S$ to denote sampling r uniformly at random from a finite set S. For a matrix $A \in \mathcal{M}_n(\mathbb{F}_q)$, $\chi_A(x) = \det(xI_n - A)$ denotes its characteristic polynomial.

We describe protocols in the arithmetic black-box (ABB) model (see Section 2.2). In this model, a secret share of $x \in \mathbb{F}_q$ is denoted by $[x]$. When needed, we use $[x]^{(i)}$ to denote the local share held by party P_i. For matrix- and vector-valued data, the same bracket notation is used entry-wise, e.g., $[A] \in \mathbb{F}_q^{n\times n}$ denotes an entry-wise sharing of $A \in \mathbb{F}_q^{n\times n}$.

In Section 5, computations are performed over $\mathbb{F}_{q^{n+1}}$ by *lifting* inputs from $\mathbb{F}_q$. We denote by A^{lift} the natural embedding of $A \in \mathcal{M}_n(\mathbb{F}_q)$ into $\mathcal{M}_n(\mathbb{F}_{q^{n+1}})$ by viewing each entry of A as an element of the extension field. The notation is summarized in Table 3.

2.2 Arithmetic Black-Box Model

We describe our protocol in the *arithmetic black-box* (ABB) model, a standard abstraction for secure multi-party computation. In this model, an ideal functionality $\mathcal{F}_{\mathrm{ABB}}$ takes as input secret shares from a set of parties $\mathcal{P} = \{P_1, \ldots, P_N\}$ and performs specific types of arithmetic operations over these shared values. For $x \in \mathbb{F}_q$, its secret share is denoted $[x]$. In this model, a concrete secret sharing scheme (e.g., Shamir's secret shares or additive shares with MACs) is not specified, treating the model as a black box. The functionality $\mathcal{F}_{\mathrm{ABB}}$ provides the following interfaces:

Table 3: Notations used in this paper.

Notation	Meaning
N	Number of parties.
T	Maximum number of corrupted parties tolerated by the protocol.
p	Field characteristic.
q	Field order.
$\mathbb{F}_q$	Finite field with q elements.
$\mathbb{F}_{q^k}$	Degree-k extension field of $\mathbb{F}_q$.
n	Matrix dimension.
$\mathcal{M}_n(\mathbb{F}_q)$	Set of $n \times n$ matrices over $\mathbb{F}_q$.
$\mathrm{GL}_n(\mathbb{F}_q)$	Set of $n \times n$ invertible matrices over $\mathbb{F}_q$.
A	Input matrix, typically $A \in \mathcal{M}_n(\mathbb{F}_q)$.
I_n	$n \times n$ identity matrix.
$\chi_A(x)$	Characteristic polynomial $\det(xI_n - A)$.
$\xleftarrow{\$}$	Uniform sampling, e.g., $r \xleftarrow{\$} S$.
$[x]$	Sharing of x (secret-shared value).
$[x]^{(i)}$	Local share of $[x]$ held by party P_i.
$\mathsf{dist}(P_i, x)$	Input distribution of x by P_i.
$\mathsf{rand}(\mathbb{F}_q)$	Generation of a random shared element in $\mathbb{F}_q$.
$\mathsf{open}([x])$	Opening (reconstruction) of a shared value.
A^{lift}	Natural embedding of A into the set of matrices over an extension field.

- **Linear operations** ($[z] \leftarrow [x] + [y]$, $[z] \leftarrow a \cdot [x] + b$): Given secret shares $[x], [y]$ and public constants $a, b \in \mathbb{F}_q$, the functionality outputs $[z]$ where $z = x + y$ or $z = ax + b$. For most linear secret sharing schemes (LSSS), these computations are local and do not require interaction.
- **Secure multiplication** ($[z] \leftarrow [x] \cdot [y]$): Given secret shares $[x]$ and $[y]$, the functionality outputs $[z]$ where $z = x \cdot y$. In contrast to linear operations, this procedure typically requires interaction and may consume preprocessed correlated randomness (e.g., Beaver triples) depending on the underlying instantiation.
- **Input distribution** ($[x] \leftarrow \mathsf{dist}(P_i, x)$): A party P_i submits a private input $x \in \mathbb{F}_q$. The functionality stores x and distributes a secret share $[x]$ to each party.
- **Random element generation** ($[r] \leftarrow \mathsf{rand}(\mathbb{F}_q)$): The functionality samples an element r from $\mathbb{F}_q$ uniformly at random and distributes its secret shares $[r]$ to the parties. The value r remains information-theoretically hidden from the parties until it is opened.
- **Value opening** ($x \leftarrow \mathsf{open}([x])$): Given the secret shares $[x]$ of the parties, the functionality reconstructs and reveals the corresponding value x to all the parties.

These interfaces can be extended to matrices over $\mathbb{F}_q$ in a straightforward manner. For example, $[R] \leftarrow \mathsf{rand}(\mathbb{F}_q^{m \times m})$ denotes the generation of a secret share of an $m \times m$ matrix whose entries are sampled independently and uniformly at random from $\mathbb{F}_q$. The interfaces for linear operations, input distribution, and value opening can be extended in a component-wise manner. Moreover, secure matrix multiplication can be implemented generically by composing the corresponding scalar multiplications.

However, such a direct extension is not necessarily optimal for performance. Depending on the underlying MPC instantiation, matrix-level primitives and other specialized techniques may reduce the complexity of interaction and communication relative to a naive scalar-based implementation. We defer a detailed discussion of such optimized implementations and their performance to Section 2.4.

2.3 Communication Model

For a set of N parties $\mathcal{P} = \{P_1, \ldots, P_N\}$, we assume the existence of $\binom{N}{2}$ secure point-to-point channels, where every distinct pair of parties shares a dedicated link ensuring perfect confidentiality and authentication. We do not assume the availability of a physical broadcast channel; consequently, all global data dissemination is achieved through multiple unicast transmissions over the established point-to-point links.

This framework is specifically instantiated to facilitate the asymptotic analysis of communication and round complexity under the assumption of an honest-majority ($T < N/2$) facing a semi-honest adversary. By isolating the fundamental transmission cost inherent to the network topology, we establish a baseline for information-theoretic performance. It is important to note that this model is modular: a protocol can be employed for stronger adversarial settings (e.g., malicious behavior) or weaker trust assumptions (e.g., dishonest majority) by augmenting the framework with additional primitives such as broadcast channels or verifiable consistent transfer mechanisms.

To evaluate the efficiency of a protocol independent of the network size, we define the communication cost as the traffic load per secure point-to-point channel. Let $\mathcal{V}_{\text{total}}$ denote the aggregate volume of data (measured in field elements) transmitted throughout the network during the execution of the protocol. When protocols operate over an extension field $\mathbb{F}_{q^k}$, one communicated extension-field element is counted as k base-field elements over $\mathbb{F}_q$. Given a fully connected topology with N parties, the total number of distinct directed links is $N(N-1)$. The communication cost $\mathcal{C}$ is computed by normalizing the total volume by the number of links:

$$\mathcal{C} = \frac{\mathcal{V}_{\text{total}}}{N(N-1)}.$$

With this metric, an "all-to-all" exchange where every party transmits a single field element to every other party yields a normalized communication cost of exactly 1.

2.4 Basic Protocols with Shamir's Secret Sharing

For concrete analysis of complexity, we instantiate the ABB model with Shamir's secret sharing in the semi-honest, honest-majority setting. In this section, we review basic protocols to instantiate the ABB model using Shamir's secret sharing scheme. The efficiency of these protocols is summarized in Table 4 in terms of the number of rounds and communication complexity.

Although our cost analysis is given for this Shamir-sharing instantiation, the protocols themselves are described at the ABB level. Hence, they can be instantiated with any MPC protocol providing the ABB operations. In particular, by employing a maliciously-secure protocol (e.g., SPDZ [DPSZ12] or replicated secret sharing [AFL+16]), the resulting instantiation achieves security against malicious adversaries, albeit with additional overhead.

Shamir's Secret Sharing. To share a secret $s \in \mathbb{F}_q$ in Shamir's scheme, a polynomial $f(x)$ of degree T such that $f(0) = s$ is sampled uniformly at random. Each party i receives a share $[s]^{(i)} = f(i)$, and the secret is reconstructed using Lagrange interpolation from any subset of at least $T + 1$ shares. Basic MPC protocols based on the Shamir's scheme are described in Figure 1.

Computing Inner Products in Parallel. A secure inner product protocol naturally extends to both scalar and matrix multiplication. In particular, since the product of $p \times q$ and $q \times r$ matrices decomposes into pr independent inner products, executing these in parallel ensures that the round complexity remains constant. Consequently, the protocol's communication complexity scales linearly with the output size pr compared to a single inner product invocation.

Algorithm: Addition

Input : Shares $[x]$ and $[y]$
Output : Share $[x+y]$

1 The parties locally compute their respective shares $[x+y]^{(i)} = [x]^{(i)} + [y]^{(i)}$ for $i \in \{1, ..., N\}$.

Algorithm: Multiplication by public constant

Input : Share $[x]$ and public value c
Output : Share $[cx]$

1 The parties locally compute their respective shares $[cx]^{(i)} = c[x]^{(i)}$ for $i \in \{1, ..., N\}$.

Algorithm: Input distribution

Input : A private input x and a party j
Output : Share $[x]$

1 Party j generates random polynomial $f(X)$ of degree at most T with $f(0) = x$.
2 Party j sends $[x]^{(i)} = f(i)$ to party i.
3 Party j sets its share $[x]^{(j)} = f(j)$.

Algorithm: Random Element Generation [CDI05]

Input : None
Output : Share $[r]$ where $r \xleftarrow{\$} \mathbb{F}_q$

1 Let $\mathcal{S} = \left\{S \subset \{P_1, \ldots, P_N\} : |S| = N - T\right\}$ be the collection of all subsets of $N - T$ parties. For each set $S \in \mathcal{S}$, $N - T$ parties share a secret key k_S.
2 Each set S is associated with a (public) polynomial $f_S(X)$ of degree at most t such that $f_S(0) = 1$ and $f_S(j) = 0$ for all $P_j \notin S$.
3 For each subset S, each party $P_j \in S$ computes $r_S = \mathsf{PRF}(k_S, \mathsf{nonce})$ where nonce is a public nonce.
4 For each party i, let $\mathcal{I}_i = \{S \in \mathcal{S} : P_i \in S\}$ denote the family of sets for which P_i holds the key k_S. Then P_i locally computes $[r]^{(i)} = \sum_{S \in \mathcal{I}_i} r_S f_S(i)$.

Algorithm: Value Opening

Input : Share $[x]$
Output : Public x

1 The parties send their shares $[x]^{(i)}$ to all parties.
2 The parties locally reconstruct x using the available shares.

Algorithm: Secure Inner Product (Secure Multiplication for $n = 1$) [BGW88]

Input : Shared vectors $[\mathbf{x}]$ and $[\mathbf{y}]$ of length n
Output : Share $[\mathbf{x} \cdot \mathbf{y}]$

1 Let $\{\lambda_i\}_{i=1}^N$ be Lagrange coefficients, i.e., $\lambda_i = \prod_{j \neq i} \frac{0-j}{i-j} \in \mathbb{F}_q$
2 The parties re-share their local sum $z_i = \sum_{k=1}^n [\mathbf{x}_k]^{(i)} \cdot [\mathbf{y}_k]^{(i)}$ by invoking $\mathsf{dist}(P_i, z_i)$ for $i \in \{1, ..., N\}$.
3 The parties locally compute their respective shares $[\mathbf{x} \cdot \mathbf{y}]^{(i)} = \sum_{j=1}^N \lambda_j [z_j]^{(i)}$ for $i \in \{1, ..., N\}$.

Figure 1: Basic MPC protocols with Shamir's secret sharing.

Table 4: Online round and communication complexity (field elements per channel)

Operation	Rounds	Communication
Addition & Multiplication by Public Value	0	0
Secure Multiplication	1	1
Input Distribution	1	$1/N$
Random Element	$0^\dagger$	$0^\dagger$
Opening	1	1
Inner Product (dimension n)	1	1
Matrix Multiplication ($p \times q$ and $q \times r$)	1	pr

† Zero-communication is achieved by using pseudorandom secret sharing (PRSS), while its scalability is limited as $\binom{N}{T}$ seeds should be pre-distributed offline.

2.5 Perfect Privacy of MPC Protocol

The concept of "revealing no information" (perfect privacy) is formally defined using the real/ideal world paradigm. Proving perfect privacy of a protocol requires that whatever information a corrupted party gathers during the protocol could have been entirely simulated by them, using only their own input and the final output.

In the real world, each party P_i, $i = 1, \ldots, N$, runs the protocol on private input x_i while interacting with the other parties. When a protocol π is executed to compute a certain functionality f on their private inputs $x = (x_1, ..., x_N)$, all the information obtained by P_i is defined as

$$\mathsf{View}_i^\pi(x) = (x_i, r_i, m_1, ..., m_\rho)$$

where r_i is the internal random coin used by P_i during the execution of the protocol, and m_j is the messages received from other parties at the j-th round. For a subset $I \subseteq \{1, \ldots, N\}$ of corrupted parties, we define their joint view as

$$\mathsf{View}_I^\pi(x) = (\mathsf{View}_i^\pi(x))_{i \in I}\,.$$

In the ideal world, a trusted third party (TTP) receives the inputs $x_1, \ldots, x_N$, computes $f(x_1, \ldots, x_N)$, and returns the output. A simulator S is an algorithm that, given only the inputs of the corrupted parties and the output, generates a fake joint view for the corrupted set. Perfect privacy requires that this simulated view be identically distributed to the real joint view of the corrupted parties.

Definition 1 (Perfect Privacy)**.** A protocol π provides perfect privacy against up to T corrupted parties if there exists a probabilistic polynomial-time (PPT) algorithm $\mathcal{S}$ such that

$$\{\mathcal{S}((x_i)_{i \in I}, f(x))\} \equiv \{\mathsf{View}_I^\pi(x)\}$$

for every input vector $x = (x_1, \ldots, x_N)$ and every subset $I \subseteq \{1, \ldots, N\}$ with $|I| \leq T$.

3 Building Blocks

In this section, we describe the building blocks used in our protocols. The descriptions of GenNonzeroElems, GenInvMats, MultScalars, and MultMats are based on [BIB89], and

SecPower is based on [CKP07]. The GenInvMatDet protocol is proposed in this work. All protocols described in this section provide perfect privacy, and their complexities are summarized in Table 5.

3.1 Generating Random Nonzero Elements

Algorithm 1 presents a constant-round protocol for generating m independent and random nonzero elements and their inverses in secret-shared form. This algorithm begins with generating a certain number of pairs of random elements $([u_j], [v_j])$ and computing their products $[w_j] = [u_j] \cdot [v_j]$. The products w_j are made public: if w_j is nonzero, it implies that both u_j and v_j are nonzero. The parties then take u_j as a random nonzero element and compute the share of its inverse as $[u_j^{-1}] = [v_j] \cdot w_j^{-1}$.

When two elements are chosen independently at random from $\mathbb{F}_q$, both elements are nonzero with probability $\alpha = (1 - q^{-1})^2$. Then the number of candidates c is determined by

$$c = \left\lceil \frac{m + \delta\sqrt{m} + \delta^2}{\alpha} \right\rceil$$

for a constant $\delta > 0$. The Chernoff bound guarantees that a single execution of the protocol yields at least m nonzero elements with exponentially high probability in δ^2 as shown in Lemma 2. We note that $c = O(m)$ for fixed q and δ. If the initial execution fails to produce m nonzero elements, then the parties can perform additional iterations until the required number of elements are obtained.

Algorithm 1: GenNonzeroElems($\mathbb{F}_q, m$) - Generate m nonzero elements

Input : Integer m
Output : m pairs of nonzero elements and their inverses $([r_k], [r_k^{-1}])$ where $r_k \in \mathbb{F}_q \setminus \{0\}$

Set constants $\alpha \leftarrow (1 - q^{-1})^2$ and $c \leftarrow \lceil (m + \delta\sqrt{m} + \delta^2)/\alpha \rceil$.
Initialize list $L \leftarrow \emptyset$.
while $|L| < m$ **do**
  The parties locally generate c pairs of random elements $([u_j], [v_j])$ where $u_j \leftarrow \mathsf{rand}(\mathbb{F}_q)$ and $v_j \leftarrow \mathsf{rand}(\mathbb{F}_q)$ for $j \in \{1, \ldots, c\}$.
  The parties compute and open $w_j \leftarrow \mathsf{open}([u_j] \cdot [v_j])$ for $j \in \{1, \ldots, c\}$.
  for $j = 1$ **to** c **do**
    if $w_j \neq 0$ **then**
      Compute $[r] \leftarrow [u_j]$ and $[r^{-1}] \leftarrow [v_j] \cdot w_j^{-1}$.
      Append $([r], [r^{-1}])$ to L.
return first m pairs in L

Efficiency. The protocol runs in a loop that terminates with overwhelming probability in the first iteration. Within the loop, the operations consist of generating random elements (local), followed by multiplication and opening (2 rounds). Assuming that the protocol terminates in the first iteration, the protocol runs in 2 rounds and requires $O(m)$ multiplications and $O(m)$ openings of the elements. Consequently, the total communication complexity amounts to $O(m)$.

Lemma 2. *Let X be the number of nonzero elements found in a single execution of Algorithm 1. Then we have*

$$P[X < m] \leq e^{-\delta^2/2}.$$

Proof. Let $X = \sum_{j=1}^{c} X_j$ be the sum of independent Bernoulli random variables where $X_j = 1$ with probability α. The expected number of successes is $\mu = \mathbb{E}[X] = c\alpha$. From

our choice of c, we have the lower bound:

$$\mu \geq m + \delta\sqrt{m} + \delta^2.$$

We analyze the failure probability $P[X < m]$ using the Chernoff bound for the lower tail. For any deviation $\Delta > 0$ such that $\mu = m + \Delta$, the bound states:

$$P[X < m] \leq \exp\left(-\frac{(\mu - m)^2}{2\mu}\right) = \exp\left(-\frac{\Delta^2}{2(m+\Delta)}\right).$$

From our choice of c, we have $\Delta \geq \delta\sqrt{m} + \delta^2$. To prove the lemma, it suffices to show that the exponent term satisfies:

$$\frac{\Delta^2}{2(m+\Delta)} \geq \frac{\delta^2}{2} \iff \Delta^2 \geq \delta^2(m+\Delta).$$

We substitute $\Delta = \delta\sqrt{m} + \delta^2$ into the expression $\Delta^2 - \delta^2\Delta - \delta^2 m$:

$$\begin{aligned}\Delta^2 - \delta^2\Delta - \delta^2 m &= (\delta\sqrt{m} + \delta^2)^2 - \delta^2(\delta\sqrt{m} + \delta^2) - \delta^2 m \\ &= (\delta^2 m + 2\delta^3\sqrt{m} + \delta^4) - (\delta^3\sqrt{m} + \delta^4) - \delta^2 m \\ &= \delta^3\sqrt{m}.\end{aligned}$$

Since $\delta > 0$ and $m \geq 1$, we have $\delta^3\sqrt{m} \geq 0$. It follows that $\frac{\Delta^2}{2(m+\Delta)} \geq \frac{\delta^2}{2}$, and hence

$$P[X < m] \leq e^{-\delta^2/2}.$$ □

3.2 Generating Random Invertible Matrices

Algorithm 2 generates m independent and random invertible matrices over a finite field in secret-shared form using a constant number of rounds. This protocol is a straightforward extension of Algorithm 1 to the matrix setting: random matrices U_j and V_j are generated, and $[W_j] = [U_j] \cdot [V_j]$ are computed and made public.

As in Section 3.1, the number of candidates c is determined by

$$c = \left\lceil \frac{m + \delta\sqrt{m} + \delta^2}{\alpha} \right\rceil$$

for a constant $\delta > 0$, where

$$\alpha = \left(\frac{|\mathrm{GL}_n(\mathbb{F}_q)|}{|\mathcal{M}_n(\mathbb{F}_q)|}\right)^2 = \left(\prod_{j=1}^{n}(1 - q^{-j})\right)^2$$

is the probability that two independently sampled matrices in $\mathcal{M}_n(\mathbb{F}_q)$ are both invertible. Similarly to Lemma 2, we can prove that a single execution of the protocol yields at least m invertible matrices with exponentially high probability in δ^2, where $c = O(m)$ for fixed q and δ.

Lemma 3. *Let X be the number of invertible matrices found in a single execution of Algorithm 2. Then we have*

$$P[X < m] \leq e^{-\delta^2/2}.$$

Algorithm 2: $\mathsf{GenInvMats}(\mathbb{F}_q, m, n)$ - Generate m Invertible Matrices of size n

Input : Integer m, n
Output : m pairs of shared matrices $([R_k], [R_k^{-1}])$ where $R_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Set constants $\alpha \leftarrow (|\mathrm{GL}_n(\mathbb{F}_q)|/|\mathcal{M}_n(\mathbb{F}_q)|)^2$ and $c \leftarrow \lceil (m + \delta\sqrt{m} + \delta^2)/\alpha \rceil$.
Initialize list $L \leftarrow \emptyset$.
while $|L| < m$ **do**
  The parties locally generate c pairs of random matrices $([U_j], [V_j])$ where $U_j \leftarrow \mathsf{rand}(\mathcal{M}_n(\mathbb{F}_q))$ and $V_j \leftarrow \mathsf{rand}(\mathcal{M}_n(\mathbb{F}_q))$ for $j \in \{1, \ldots, c\}$.
  The parties compute and open $W_j \leftarrow \mathsf{open}([U_j] \cdot [V_j])$ for $j \in \{1, \ldots, c\}$.
  for $j = 1$ **to** c **do**
    if W_j *is invertible* **then**
      Compute $[R] \leftarrow [U_j]$ and $[R^{-1}] \leftarrow [V_j] \cdot W_j^{-1}$.
      Append $([R], [R^{-1}])$ to L.
return first m pairs in L

Efficiency. The protocol runs in a loop that terminates with overwhelming probability in the first iteration. Within the loop, the operations consist of generating random shared matrices (local), followed by matrix multiplication and opening (2 rounds). Assuming that the protocol terminates in the first iteration and that a sufficient number of random invertible matrices are available from the first iteration, the protocol runs in 2 rounds and requires $O(m)$ matrix multiplications and $O(m)$ openings of the matrices. Consequently, the total communication complexity amounts to $O(mn^2)$.

3.3 Multiplying Nonzero Scalar Elements

Algorithm 3 presents a protocol for computing the product of t secret-shared nonzero scalar elements, $[a_1], \ldots, [a_t]$. In the first phase, the parties generate t random nonzero values $r_1, \ldots, r_t$ along with their inverses $r_1^{-1}, \ldots, r_t^{-1}$; this step requires 2 rounds and can be performed offline. Subsequently, the parties compute and open the values $c_i = r_i \cdot a_i \cdot r_{i+1}^{-1}$ for each $i \in \{1, \ldots, t-1\}$, and $c_t = r_t \cdot a_t$. Finally, each share of the product is computed as $[b] = [r_1^{-1}] \cdot (c_1 \cdot \cdots \cdot c_t)$.

Algorithm 3: $\mathsf{MultScalars}(\mathbb{F}_q, [a_1], \ldots, [a_t])$ - Multiply t Nonzero Scalar Elements

Input : $[a_1], \cdots, [a_t]$, where $a_i \in \mathbb{F}_q \setminus \{0\}$ for all $i \in [t]$
Output : $[b] = [a_1 a_2 \cdots a_t]$

The parties generate t random pairs $([r_i], [r_i^{-1}])_{i \in [t]} \leftarrow \mathsf{GenNonzeroElems}(\mathbb{F}_q, t)$.
for $i = 1$ **to** t **do**
  3 The parties compute and open $c_i \leftarrow \begin{cases} \mathsf{open}([r_i] \cdot [a_i] \cdot [r_{i+1}^{-1}]) & \text{if } i < t \\ \mathsf{open}([r_t] \cdot [a_t]) & \text{if } i = t. \end{cases}$
The parties compute $[b] \leftarrow [r_1^{-1}] \cdot (c_1 \cdot \cdots \cdot c_t)$.
return $[b]$

Efficiency. The generation of random pairs $([r_i], [r_i^{-1}])$ is independent of the input, so they can be precomputed offline. Thus, the offline phase takes 2 rounds with $O(t)$ communication. On the other hand, Line 3 is executed in 3 online rounds to perform $2t - 1$ multiplications and t openings in parallel. Consequently, the online communication complexity amounts to $O(t)$.

3.4 Multiplying Invertible Matrices

Algorithm 4 describes a protocol for computing the product of t secret-shared invertible matrices, $[A_1], \ldots, [A_t]$. This protocol begins with the generation of t random invertible matrices $R_1, \ldots, R_t$ and their corresponding inverses $R_1^{-1}, \ldots, R_t^{-1}$ in secret-shared form using Algorithm 2. Then the parties compute and reveal $C_i = R_i \cdot A_i \cdot R_{i+1}^{-1}$, $i = 1, \ldots, t-1$, and $C_t = R_t \cdot A_t$. Each party computes $[B] = [R_1^{-1}] \cdot (C_1 \cdot \cdots \cdot C_t)$, where

$$B = R_1^{-1} \cdot (R_1 A_1 R_2^{-1}) \cdot (R_2 A_2 R_3^{-1}) \cdots (R_{t-1} A_{t-1} R_t^{-1}) \cdot (R_t A_t) = A_1 A_2 \cdots A_t.$$

Algorithm 4: $\mathsf{MultMats}(\mathbb{F}_q, [A_1], \ldots, [A_t])$ - Multiply t Invertible Matrices

Input : $[A_1], \ldots, [A_t]$, where $A_i \in \mathrm{GL}_n(\mathbb{F}_q)$ for all $i \in [t]$
Output : $[B] = [A_1 A_2 \cdots A_t]$

1 The parties generate t random pairs $([R_i], [R_i^{-1}])_{i \in [t]} \leftarrow \mathsf{GenInvMats}(\mathbb{F}_q, t, n)$.
2 **for** $i = 1$ **to** t **do**
3 The parties compute and open $C_i \leftarrow \begin{cases} \mathsf{open}([R_i] \cdot [A_i] \cdot [R_{i+1}^{-1}]) & \text{if } i < t \\ \mathsf{open}([R_t] \cdot [A_t]) & \text{if } i = t. \end{cases}$
4 The parties compute $[B] \leftarrow [R_1^{-1}] \cdot (C_1 \cdot \cdots \cdot C_t)$.
5 **return** $[B]$

Efficiency. Similar to Section 3.3, it takes 2 offline rounds with $O(tn^2)$ communication and 3 online rounds with $O(tn^2)$ communication.

3.5 Generating a Random Invertible Matrix R and $\det(R)^{-1}$

Algorithm 5 describes a protocol to jointly sample a uniformly random invertible matrix and compute the determinant of its inverse, both in secret-shared form. Unlike the protocol of Cramer and Damgård [CD01, Section 6.1], our algorithm provides perfect privacy, as it consists only of input distributions via dist and internal invocations of $\mathsf{MultMats}$ and $\mathsf{MultScalars}$ without revealing any information beyond what is inherently disclosed by the sub-protocols.

Each party P_i locally samples a random invertible matrix $R_i \in \mathrm{GL}_n(\mathbb{F}_q)$ and secret shares of R_i and $\det(R_i)^{-1}$. The parties then compute $[R] = \prod_{i=1}^{N} [R_i]$ and $[\det(R)^{-1}] = \prod_{i=1}^{N} [\det(R_i)^{-1}]$ using Algorithm 4 and Algorithm 3, respectively, where $R = R_1 R_2 \cdots R_N$. Since each R_i is uniform over $\mathrm{GL}_n(\mathbb{F}_q)$, R is also uniformly distributed in $\mathrm{GL}_n(\mathbb{F}_q)$.

Algorithm 5: $\mathsf{GenInvMatDet}(\mathbb{F}_q, n)$ - Generate invertible $[R]$ and $[\det(R)^{-1}]$

Input : None
Output : $([R], [\det(R)^{-1}])$ where $R \xleftarrow{\$} \mathrm{GL}_n(\mathbb{F}_q)$

1 The parties uniformly sample their respective invertible matrices $R_i \xleftarrow{\$} \mathrm{GL}_n(\mathbb{F}_q)$ for $i \in \{1, ..., N\}$.
2 The parties locally compute $\det(R_i)$ for $i \in \{1, ..., N\}$.
3 The parties distribute their matrices $[R_i] \leftarrow \mathsf{dist}(P_i, R_i)$ and their determinants $[\det(R_i)^{-1}] \leftarrow \mathsf{dist}(P_i, \det(R_i)^{-1})$ for $i \in \{1, ..., N\}$.
4 The parties compute $[R] \leftarrow \mathsf{MultMats}(\mathbb{F}_q, [R_1], \ldots, [R_N])$.
5 The parties compute $[\det(R)^{-1}] \leftarrow \mathsf{MultScalars}(\mathbb{F}_q, [\det(R_1)^{-1}], \cdots, [\det(R_N)^{-1}])$.
6 **return** $[R]$, $[\det(R)^{-1}]$

Efficiency. $\mathsf{MultMats}$ and $\mathsf{MultScalars}$ consume $O(Nn^2)$ communication in 2 offline rounds. On the other hand, the online phase requires $O(n^2)$ communication for Line 3 (in 1 round) and $O(Nn^2)$ communication for Lines 4–5 (in 3 rounds). In total, the online overhead amounts to 4 rounds and $O(Nn^2)$ communication.

3.6 Computing Matrix Powers

Algorithm 6 computes successive powers of a secret-shared matrix $A \in \mathcal{M}_n(\mathbb{F}_q)$. To achieve perfect privacy, it uses the augmented matrix

$$A^+ = \begin{pmatrix} A & -I_n \\ I_n & 0 \end{pmatrix} \in \mathbb{F}_q^{2n \times 2n}$$

where $\det(A^+) = 1$ regardless of the invertibility of A. Let B_i denote the $n \times n$ top-left block of $(A^+)^i$. Once $[(A^+)^i]$ are computed for $i = 0, \ldots, m$, $[A^i]$ are obtained by locally solving the following system of equations (in secret shared form):

$$\begin{pmatrix} \alpha_{0,0} & & \\ \vdots & \ddots & \\ \alpha_{m,0} & \cdots & \alpha_{m,m} \end{pmatrix} \begin{pmatrix} I_n \\ A \\ A^2 \\ \vdots \\ A^m \end{pmatrix} = \begin{pmatrix} I_n \\ B_1 \\ B_2 \\ \vdots \\ B_m \end{pmatrix} \tag{3}$$

where $(\alpha_{i,j})$ is a lower-triangular matrix such that

- $\alpha_{0,0} = 1, \alpha_{1,0} = 0, \alpha_{1,1} = 1$,
- $\alpha_{i,j} = 0$ if $j < 0$ or $i < j$,
- $\alpha_{i,j} = \alpha_{i-1,j-1} - \alpha_{i-2,j}$ for $0 \le j \le i$ and $i \ge 2$.

Algorithm 6: $\mathsf{SecPower}(\mathbb{F}_q, A, m)$ - Compute cumulative powers of matrix

Input : $[A] \in \mathcal{M}_n(\mathbb{F}_q), m \in \mathbb{N}$
Output : $[A^2], [A^3], \cdots, [A^m]$

1 The parties locally generate a share of the augmented matrix $[A^+] = \begin{pmatrix} [A] & -I_n \\ I_n & 0 \end{pmatrix}$.
2 The parties generate m pairs of random matrices $([R_i], [R_i^{-1}])_{i \in [m]} \leftarrow \mathsf{GenInvMats}(\mathbb{F}_q, m, 2n)$.
3 The parties compute and open $N_i \leftarrow \begin{cases} \mathsf{open}([R_{i-1}] \cdot [A^+] \cdot [R_i^{-1}]) & \text{if } 2 \le i \le m \\ \mathsf{open}([A^+] \cdot [R_1^{-1}]) & \text{if } i = 1. \end{cases}$
4 The parties locally compute $P_j \leftarrow \prod_{i=1}^{j} N_i$ for $1 \le j \le m$.
5 The parties locally compute $[(A^+)^j] \leftarrow P_j[R_j]$ for $1 \le j \le m$.
6 The parties locally solve the system in equation 3 and get $[A^j]$.
7 **return** $[A^2], \cdots, [A^m]$

Efficiency. Line 2 is executed offline, where its complexity is 2 rounds with $O(mn^2)$ communication. The online complexity is 3 rounds with $O(mn^2)$ communication.

4 Cost Analysis of Existing Protocols

In this section, we analyze the round and communication complexities of existing matrix-invertibility testing protocols within the ABB framework, separating input-independent offline costs from input-dependent online costs. Although these protocols were proposed in prior work, expressing their costs in this common model enables a precise comparison.

4.1 Masking-Based Protocols

We analyze the communication and round complexity of the matrix invertibility testing methods of [BIB89], [CEN25] (i.e., one-sided and two-sided masking methods). These protocols impose no constraint relating the field characteristic to the matrix dimension.

Table 5: Online and offline round complexities and communication cost of building block protocols. For rounds and communication complexities, online and offline costs are distinguished.

Protocol	Rounds		Communication	
	Online	Offline	Online	Offline
$\mathsf{GenNonzeroElems}(\mathbb{F}_q, m)$	2	0	$O(m)$	0
$\mathsf{GenInvMats}(\mathbb{F}_q, m, n)$	2	0	$O(mn^2)$	0
$\mathsf{MultScalars}(\mathbb{F}_q, [a_1], \ldots, [a_t])$	3	2	$O(t)$	$O(t)$
$\mathsf{MultMats}(\mathbb{F}_q, [A_1], \ldots, [A_t])$	3	2	$O(tn^2)$	$O(tn^2)$
$\mathsf{GenInvMatDet}(\mathbb{F}_q, n)$	4	2	$O(Nn^2)$	$O(Nn^2)$
$\mathsf{SecPower}(\mathbb{F}_q, A, m)$	3	2	$O(mn^2)$	$O(mn^2)$

Algorithm 7: Invertibility testing protocol of [BIB89]

Input : $A \in \mathcal{M}_n(\mathbb{F}_q)$
Output : A bit $b \in \{0, 1\}$ such that $b = 1$ iff A is invertible
1 The parties generate $([R], [R^{-1}]) \leftarrow \mathsf{GenInvMats}(\mathbb{F}_q, 1, n)$.
2 The parties open $T \leftarrow \mathsf{open}([A][R])$.
3 **return** $\begin{cases} 0 & \text{if } \det(T) = 0 \\ 1 & \text{otherwise.} \end{cases}$

Efficiency. We estimate the round and communication complexity step by step.

- **Line 1** requires 2 offline rounds with $O(n^2)$ communication.
- **Line 2** requires 2 online rounds with $O(n^2)$ communication.

Algorithm 8: Invertibility testing protocol of [CEN25]

Input : $A \in \mathcal{M}_n(\mathbb{F}_q)$
Output : A bit $b \in \{0, 1\}$ such that $b = 1$ iff A is invertible
1 The parties generate $([R], [R^{-1}]), ([S], [S^{-1}]) \leftarrow \mathsf{GenInvMats}(\mathbb{F}_q, 2, n)$.
2 The parties open $T \leftarrow \mathsf{open}([S][A][R])$.
3 **return** $\begin{cases} 0 & \text{if } \det(T) = 0 \\ 1 & \text{otherwise.} \end{cases}$

Efficiency. We estimate the round and communication complexity step by step.

- **Line 1** requires 2 offline rounds with $O(n^2)$ communication.
- **Line 2** requires 3 online rounds with $O(n^2)$ communication.

Therefore, both protocols take 2 offline rounds with $O(n^2)$ communications and 2 or 3 online rounds with $O(n^2)$ communications depending on the number of masking side.

4.2 Cramer-Damgård Protocol

In this section, we analyze the round and communication complexity of the determinant-based invertibility test obtained from [CD01]. In the protocol, $z_i \in \mathbb{F}_q$ are public constants. In Algorithm 9, for a shared matrix $[A]$, we denote by $[A]_{j,k}$ the sharing of its (j, k)-th entry.

Algorithm 9: Invertibility testing protocol of [CD01]

Input : $A \in \mathcal{M}_n(\mathbb{F}_q)$
Output : A bit $b \in \{0,1\}$ such that $b = 1$ iff A is invertible

1 The parties generate $([r], [r^{-1}]) \leftarrow \mathsf{GenNonzeroElems}(\mathbb{F}_q, 1)$
2 **for** $i = 1$ **to** $n+1$ **do**
3 Define a shared lower-triangular matrix L_i and the upper-triangular matrix U_i.
4 The parties generate $2n$ random pairs $([l_{i,j}], [l_{i,j}^{-1}])_{j\in\{1,\dots,n\}} \leftarrow \mathsf{GenNonzeroElems}(\mathbb{F}_q, n)$ and $([u_{i,j}], [u_{i,j}^{-1}])_{j\in\{1,\dots,n\}} \leftarrow \mathsf{GenNonzeroElems}(\mathbb{F}_q, n)$, and set $[L_i]_{j,j} \leftarrow [l_{i,j}]$ and $[U_i]_{j,j} \leftarrow [u_{i,j}]$.
5 The parties locally generate $[L_i]_{j,k} \leftarrow \mathsf{rand}(\mathbb{F}_q)$ and $[U_i]_{k,j} \leftarrow \mathsf{rand}(\mathbb{F}_q)$ for $1 \le k < j \le n$.
6 The parties compute $[R_i] \leftarrow [L_i] \cdot [U_i]$.
7 The parties compute $[\det(R_i)^{-1}] \leftarrow \mathsf{MultScalars}(\mathbb{F}_q, [l_{i,1}^{-1}], \dots, [l_{i,n}^{-1}], [u_{i,1}^{-1}], \dots, [u_{i,n}^{-1}])$.
8 The parties compute $[r_i] = [r] \cdot [\det(R_i)^{-1}]$
9 The parties set $[C_i] \leftarrow z_i \cdot I_n - [A]$.
10 The parties compute and open $[B_i] \leftarrow [C_i] \cdot [R_i]$.
11 The parties locally compute $\det(B_i)$.
12 The parties locally compute $[r \cdot \chi_A(0)] = \sum_{i=1}^{n+1} \lambda_i \cdot \det(B_i) \cdot r_i$.
13 The parties open $r \cdot \chi_A(0) \leftarrow \mathsf{open}([r \cdot \chi_A(0)])$.
14 **return** $\begin{cases} 0 & \text{if } r \cdot \chi_A(0) = 0 \\ 1 & \text{otherwise.} \end{cases}$

Efficiency. We estimate the round and communication complexity step by step.

- **Line 1** requires 2 offline rounds and $O(1)$ communication.
- **Line 4** invokes $\mathsf{GenNonzeroElems}(\mathbb{F}_q, n)$ twice for each $i \in \{1, \dots, n+1\}$. The total communication is $O(n^2)$ elements. Although this line requires 2 rounds, these can be executed in parallel with Line 1, and thus do not increase the overall offline round number.
- **Line 6** requires one matrix multiplication for each $i \in \{1, \dots, n+1\}$. Hence, it takes 1 offline round with $O(n^3)$ communication.
- **Line 7** invokes $\mathsf{MultScalars}$ with $2n$ inputs for each $i \in \{1, \dots, n+1\}$. The total communication complexity is $O(n^2)$. Although $\mathsf{MultScalars}$ consists of 2 offline rounds and 3 online rounds, this computation is independent of A and thus can be performed offline. Moreover, its offline rounds can be parallelized with Lines 1 and 6, so this step contributes effectively 2 additional offline rounds.
- **Line 8** requires one secure multiplication for each $i \in \{1, \dots, n+1\}$. Thus, it takes 1 offline round with $O(n)$ communication.
- **Line 10** performs one matrix multiplication and one opening of an $n \times n$ matrix for each $i \in \{1, \dots, n+1\}$. Hence, it takes 2 online rounds with $O(n^3)$ communication.
- **Line 13** requires one opening of a scalar, hence 1 online round with $O(1)$ communication.

Overall, the protocol uses 6 offline rounds and 3 online rounds. The total offline communication is $O(n^3)$, and the total online communication is also $O(n^3)$.

4.3 Maire–Vergnaud Protocol

In this section, we analyze round and communication complexity of matrix invertibility testing with perfect privacy for $n < p$ [MV24]. Note that we can safely use one-sided masking to compute matrix inverse in Lines 5–6 since the coefficient matrix in the equation system is invertible.

Algorithm 10: Invertibility testing protocol of [MV24]

Input : $A \in \mathcal{M}_n(\mathbb{F}_q)$
Output: A bit $b \in \{0,1\}$ such that $b = 1$ iff A is invertible

1 The parties generate $([r], [r^{-1}]) \leftarrow \mathsf{GenNonzeroElems}(\mathbb{F}_q, 1)$
2 The parties generate $([R], [R^{-1}]) \leftarrow \mathsf{GenInvMats}(\mathbb{F}_q, 1, n)$.
3 The parties compute $[A^2], \cdots, [A^n] \leftarrow \mathsf{SecPower}(\mathbb{F}_q, A, n)$.
4 The parties construct equation system $[T][d] = [t]$ of Lemma 1 with $[\mathrm{Tr}(A^j)]$.
5 The parties compute $S \leftarrow \mathsf{open}([T][R])$.
6 The parties compute $[d] \leftarrow [R]S^{-1}[t]$.
7 The parties compute $d'_n \leftarrow \mathsf{open}([d_n] \cdot [r])$ from $[d]$.
8 **return** $\begin{cases} 0 & \text{if } d'_n = 0 \\ 1 & \text{otherwise.} \end{cases}$

Efficiency. We estimate the round and communication complexity step by step.

- **Lines 1–2** require 2 offline rounds with $O(n^2)$ communication. Both lines can be executed in parallel without increasing the overall offline rounds.
- **Line 3** requires 2 offline rounds with $O(n^3)$ communication and 3 online rounds with $O(n^3)$ communication. The offline rounds in this line can be executed in parallel with Line 1 without increasing the overall offline rounds.
- **Line 5** requires 2 online rounds with $O(n^2)$ communication.
- **Line 6** requires 1 online round with $O(n)$ communication.
- **Line 7** requires 2 online rounds with $O(1)$ communication.

The overall complexity of Algorithm 10 is 2 offline rounds with $O(n^3)$ communication and 8 online rounds with $O(n^3)$ communication.

4.4 Maire–Vergnaud Protocol with the Schönhage Algorithm

In this section, we analyze the detailed round and communication complexity of the invertibility testing algorithm from [MV24] adapted with the Schönhage algorithm [Sch93]. Although Maire and Vergnaud suggested this substitution to handle small characteristic fields, a concrete complexity analysis for this modified construction was omitted.

Let $J(p, n)$ be the set of the first n positive integers that are not divisible by p. Next, we define constants l, m, r, μ, ρ as follows:

$$l = \max(J(p, n)) = m \cdot p + r \text{ where } 1 \leq r \leq p - 1,$$

$$n = \mu \cdot p + \rho \text{ where } 0 \leq \rho \leq p - 1.$$

For formal definitions of matrices D, Q, vectors u, w and companion matrix C used in Algorithm 11, we direct the reader to [Sch93].

Algorithm 11: Modified invertibility testing protocol from [MV24] adapted with the Schönhage algorithm [Sch93] for small characteristic fields

Input : $A \in \mathcal{M}_n(\mathbb{F}_q)$
Output : 0 if $\det(A) = 0$, 1 otherwise.

1 The parties appropriately construct zero-padded $n' \times n'$ matrix $[A'] = \mathsf{Pad}([A])$ where n' is the smallest integer satisfying $n \leq n'$ and $n' \equiv 1 \bmod p$.
2 The parties generate $([r], [r^{-1}]) \leftarrow \mathsf{GenNonzeroElems}(\mathbb{F}_q, 1)$
3 The parties locally compute the constants m, r, l, μ, ρ with respect to n', public companion matrix C of polynomial $\hat{f}(t)$ given in Proposition 3 of [Sch93] and set $[B] := C - yC[A']$.
▷ Perform further steps in $R_q = \mathbb{F}_q[y]/(y^{n'+1})$.
4 The parties generate 2 pairs of random matrices $([S_1], [S_1^{-1}]) \leftarrow \mathsf{GenInvMats}(R_q, 1, n')$, and $([S_2], [S_2^{-1}]) \leftarrow \mathsf{GenInvMats}(R_q, 1, \mu)$.
5 The parties invoke $\mathsf{SecPower}(R_q, B, l)$ and get $[s_j] := \mathrm{Tr}([B^j])$ for $1 \leq j \leq l$.
6 The parties locally compute $[D] \in R_q^{(n' \times n')}$ and $[u] \in R_q^{n'}$ from $\{[s_j]\}_{j \in \{1,\ldots,l\}}$.
7 The parties solve $[v] = [(I - D)^{-1}][u]$ using $([S_1], [S_1^{-1}])$.
8 The parties locally compute $[Q] \in R_q^{(\mu \times \mu)}$ and $[w] \in R_q^{\mu}$ from $[v]$.
9 The parties solve $[([a_1], [a_2], ..., [a_\mu])] = [a] := -[Q^{-1}][w]$ using $([S_2], [S_2^{-1}])$.
10 The parties compute $[\det(I - yA')] = [h] := \sum_{i+j=\mu} [a_i] \cdot [v_{1,j}]$, where $[v_{1,j}]$ are some entries of $[v]$.
11 The parties open $\gamma'_n \leftarrow \mathsf{open}([\gamma_n] \cdot [r])$, where γ_n is coefficient of y^n in $[h] \in R_q$.
12 **return** $\begin{cases} 0 & \text{if } \gamma'_n = 0 \\ 1 & \text{otherwise.} \end{cases}$

Note that every operation of Lines 4–10 is performed over truncated polynomial ring R_q. An element g of R_q is represented as

$$g = c_0 + c_1 y + c_2 y^2 + \cdots + c_{n'} y^{n'}, \{c_i\}_{\{0,\ldots,n'\}} \in \mathbb{F}_q,$$

or equivalently, vector of length $(n' + 1)$ over $\mathbb{F}_q$. In this ring, an element becomes unit if and only if $c_0 \neq 0$, and a matrix is invertible if and only if the matrix that consists only constant term of each entry is invertible over $\mathbb{F}_q$.

Regarding the perfect privacy of this protocol, we first have to analyze perfect privacy of sub-protocols GenInvMats and SecPower over R_q. Let $U, V \in \mathcal{M}_n(R_q)$. From the above invertibility condition, if $M = UV$ is invertible, then U, V are also invertible thus GenInvMats over R_q generates uniform random invertible matrices. Next, for $A, B, C \in \mathrm{GL}_n(R_q)$ such that $A \neq B$, we have $AC \neq BC$. Hence the opened values in SecPower over R_q become uniform random. Consequently, changing the domain from $\mathbb{F}_q$ to R_q preserves perfect-privacy of such sub-protocols.

In Lines 7 and 9, computing an inverse of matrix is done by one-sided masking (i.e., $[A^{-1}] = [R] \cdot \mathsf{open}([A] \cdot [R])^{-1}$). These procedures does not degrade the perfect privacy of this protocol, since matrices $(I - D)$ and Q are invertible over R_q, thus reveal no rank information. Specifically, a matrix D is nilpotent lower triangular matrix such that $D^n = 0$, and $Q = I - yM$ for some $\mu \times \mu$ matrix M, where yM is also nilpotent matrix such that $(yM)^{n+1} = 0$. Then

$$(I - D)(I + D + D^2 + \cdots + D^{n-1}) = I,$$

and

$$(I - yM)(I + yM + (yM)^2 + \cdots + (yM)^n) = I,$$

holds, indicating that they are invertible over R_q. With other basic operations of Shamir's sharing and GenNonzeroElems over $\mathbb{F}_q$, we can conclude that Algorithm 11 has the perfect-privacy.

In Lines 10–11, since A' is zero-padded matrix, we have $\det(I - yA') = \det(I - yA)$ hence the coefficient of y^n (not $y^{n'}$) represents $\det(A)$.

Efficiency. We estimate the round and communication complexity step by step. Since all arithmetic operations are done in R_q for Lines 4–10, the communicated field elements for those lines are multiplied by $n' + 1$.

- **Line 2** requires 2 offline rounds with $O(1)$ communication.
- **Line 4** requires 2 offline rounds with $O(n'^3 + \mu^2 n')$ communication.
- **Line 5** requires 2 offline rounds with $O(ln'^3)$ communication, and 3 online rounds with $O(ln'^3)$ communication.
- **Line 7** requires 3 online rounds with $O(n'^3)$ communication.
- **Line 9** requires 3 online rounds with $O(n'\mu^2)$ communication.
- **Line 10** requires 1 online round with $O(\mu n')$ communication.
- **Line 11** requires 2 online rounds with $O(1)$ communication.

For fields with characteristic 2, which are widely used in threshold MQ signatures, n' is the smallest odd integer satisfying $n' \geq n$, $l = 2n' - 1$, and $\mu = \lfloor n'/2 \rfloor$. More generally, for characteristic p, $n' = n + O(p)$, $l = O(n')$ and $\mu = \lfloor n'/p \rfloor$ hence $\mu = O(n'/p)$. Therefore, for fixed p, n', l, μ become all $O(n)$. Overall, as offline rounds of Lines 2, 4, and 5 allow concurrent execution, this protocol takes 2 offline rounds with $O(n^4)$ and 12 online rounds with $O(n^4)$ communication complexity.

5 Cramer-Damgård Protocol over Extension Fields

In this section, we present protocols with perfect privacy for secure matrix invertibility testing over fields of small order by lifting the Cramer-Damgård protocol to extension fields. Although the original work briefly notes this lifting idea [CD01], it does not provide a concrete analysis. We therefore give a detailed treatment, including a proof of perfect privacy as well as explicit bounds on round and communication complexity.

In the original Cramer-Damgård protocol, two limitations make it difficult to achieve perfect privacy. First, the protocol generates a random invertible matrix $[R]$ together with $[\det(R)]$ via an LU-decomposition-based procedure and uses R to mask the target matrix. However, the distribution of matrix R is not uniform in $\mathrm{GL}_n(\mathbb{F}_q)$. Second, the protocol computes $\det(M_i)$, where $M_i := z_i I_n - A$ for fixed and distinct points $z_1, \ldots, z_{n+1} \in \mathbb{F}_q$. During this computation, the rank of M_i is made public, which leaks whether z_i is an eigenvalue of A.

To address the first limitation, we employ Algorithm 5 to generate $([R], [\det(R)^{-1}])$, ensuring that R is uniformly distributed over $\mathrm{GL}_n(\mathbb{F}_q)$. To address the second limitation, we perform the computation in the extension field $\mathbb{F}_{q^{n+1}}$. Let A^{lift} denote the natural embedding of A into $\mathcal{M}_n(\mathbb{F}_{q^{n+1}})$. We sample $z_i \in \mathbb{F}_{q^{n+1}}$ such that its minimal polynomial over $\mathbb{F}_q$ has degree $n + 1$. Then z_i cannot be a root of $\chi_{A^{\mathsf{lift}}}(X)$, since any eigenvalue of A^{lift} is algebraic over $\mathbb{F}_q$ of degree at most n. Equivalently, for every i, the matrix $M_i := z_i I_n - A^{\mathsf{lift}}$ is nonsingular over $\mathbb{F}_{q^{n+1}}$. Consequently, n is always revealed as the rank of M_i over the extension field without any leakage of information on A.

5.1 Protocol Description

Our first protocol based on the approach of Cramer and Damgård is described in Algorithm 12, where $z_i \in \mathbb{F}_{q^{n+1}}$ are public constants whose minimal polynomials over $\mathbb{F}_q$ have degree $n + 1$.

Algorithm 12: $\mathsf{isInverseCD}(A)$ - Invertibility test by lifting the Cramer-Damgård protocol over extension fields

Input : $A \in \mathcal{M}_n(\mathbb{F}_q)$
Output : A bit $b \in \{0,1\}$ such that $b = 1$ iff A is invertible

1 The parties generate a nonzero random element $([r], [r^{-1}]) \leftarrow \mathsf{GenNonzeroElems}(\mathbb{F}_{q^{n+1}}, 1)$.
2 The parties lift $[A^{\mathsf{lift}}] \leftarrow \mathsf{lift}([A], \mathbb{F}_{q^{n+1}})$.
3 **for** $i = 1$ **to** $n+1$ **do**
4 The parties generate $([R_i], [\det(R_i)^{-1}]) \leftarrow \mathsf{GenInvMatDet}(\mathbb{F}_{q^{n+1}}, n)$.
5 The parties compute $[r_i] = [r] \cdot [\det(R_i)^{-1}]$.
6 The parties set $[C_i] \leftarrow z_i \cdot I_n - [A^{\mathsf{lift}}]$.
7 The parties compute and open $B_i \leftarrow [C_i] \cdot [R_i]$.
8 The parties locally compute $\det(B_i)$.
9 The parties locally compute $[r \cdot \chi_{A^{\mathsf{lift}}}(0)] = \sum_{i=1}^{n+1} \lambda_i \cdot \det(B_i) \cdot [r_i]$.
10 The parties open $r \cdot \chi_{A^{\mathsf{lift}}}(0) \leftarrow \mathsf{open}([r \cdot \chi_{A^{\mathsf{lift}}}(0)])$.
11 **return** $\begin{cases} 0 & \text{if } r \cdot \chi_{A^{\mathsf{lift}}}(0) = 0 \\ 1 & \text{otherwise.} \end{cases}$

Theorem 1 (Correctness)**.** *For any matrix $A \in \mathcal{M}_n(\mathbb{F}_q)$, $\mathsf{isInverseCD}(A)$ outputs 1 if and only if A is invertible over $\mathbb{F}_q$.*

Proof. Since z_i has the minimal polynomial of degree $n+1$ over $\mathbb{F}_q$ and every eigenvalue of A^{lift} is algebraic over $\mathbb{F}_q$ with degree at most n, the matrix $C_i = z_i \cdot I_n - A^{\mathsf{lift}}$ is invertible. For $i \in \{1, \ldots, n+1\}$, $B_i = C_i \cdot R_i$ and $r_i = r \cdot \det(R_i)^{-1}$. Therefore, we have

$$\det(B_i) \cdot r_i = \det(C_i) \cdot \det(R_i) \cdot r \cdot \det(R_i)^{-1} = r \cdot \det(C_i). \tag{4}$$

By Lagrange interpolation, Line 9 computes $[r \cdot \chi_{A^{\mathsf{lift}}}(0)]$. Since $r \neq 0$, the opened value in Line 10 is 0 if and only if $\chi_{A^{\mathsf{lift}}}(0) = 0$, equivalently $\det(A) = 0$. □

Theorem 2 (Perfect Privacy)**.** *For any matrix $A \in \mathcal{M}_n(\mathbb{F}_q)$, $\mathsf{isInverseCD}(A)$ reveals no information beyond the output bit indicating whether A is invertible.*

Proof. Let $b \in \{0,1\}$ denote the output of $\mathsf{isInverseCD}(A)$, where $b = 1$ if and only if A is invertible. We construct a simulator $\mathcal{S}$ that, given only b and the inputs of the corrupted parties, produces a distribution identical to $\mathsf{View}_I^{\mathsf{isInverseCD}}(A)$.

We use the perfect privacy of the underlying building blocks stated in Section 3. Their internal executions can therefore be simulated from the information revealed by their functionalities. It remains to analyze only the values explicitly opened by $\mathsf{isInverseCD}$ itself.

The only values opened during the protocol are the matrices B_i in Line 7 and the final value $r \cdot \chi_{A^{\mathsf{lift}}}(0)$ in Line 10. All other intermediate values remain secret-shared.

We first analyze the distribution of the opened matrices. Since z_i has minimal polynomial of degree $n+1$ over $\mathbb{F}_q$, while every eigenvalue of A^{lift} is algebraic over $\mathbb{F}_q$ of degree at most n, the matrix $C_i = z_i \cdot I_n - A^{\mathsf{lift}}$ is invertible. The protocol opens $B_i = C_i \cdot R_i$ where R_i is sampled uniformly from $\mathrm{GL}_n(\mathbb{F}_{q^{n+1}})$. Since left multiplication by the fixed invertible matrix C_i is a bijection on $\mathrm{GL}_n(\mathbb{F}_{q^{n+1}})$, the distribution of B_i is uniform over $\mathrm{GL}_n(\mathbb{F}_{q^{n+1}})$ and is independent of A. Therefore, the jointly opened matrices $(B_1, \ldots, B_{n+1})$ reveal no information about A.

It remains to analyze the final opening. As in (4), we have

$$\det(B_i) \cdot r_i = r \cdot \det(C_i).$$

Therefore, by Lagrange interpolation, Line 9 computes $[r \cdot \chi_{A^{\mathsf{lift}}}(0)]$.

If $b = 0$, then A is singular, and hence $\chi_{A^{\mathsf{lift}}}(0) = 0$. Therefore, the value opened in Line 10 is equal to 0. If $b = 1$, then A is invertible, so $\chi_{A^{\mathsf{lift}}}(0) \neq 0$. Since r is sampled uniformly from $\mathbb{F}_{q^{n+1}} \setminus \{0\}$, the product $r\chi_{A^{\mathsf{lift}}}(0)$ is also uniformly distributed over $\mathbb{F}_{q^{n+1}} \setminus \{0\}$. Thus the final opened value depends only on the output bit b.

We now construct a simulator $\mathcal{S}$. Given b and the inputs of the corrupted parties, it samples independent uniformly random matrices $\widetilde{B}_1, \ldots, \widetilde{B}_{n+1} \xleftarrow{\$} \mathrm{GL}_n(\mathbb{F}_{q^{n+1}})$. If $b = 0$, it sets $\widetilde{u} = 0$. If $b = 1$, it samples $\widetilde{u} \xleftarrow{\$} \mathbb{F}_{q^{n+1}} \setminus \{0\}$. These values simulate the opened transcript

$$(\widetilde{B}_1, \ldots, \widetilde{B}_{n+1}, \widetilde{u}).$$

It remains to simulate the shares and messages seen by the corrupted parties for all values that are not opened. Let $I \subseteq \{1, \ldots, N\}$ be an arbitrary corrupted set with $|I| \leq T$. Since the protocol uses degree-T Shamir's secret sharing and $|I| \leq T$, the shares held by the corrupted parties for any unopened shared value are perfectly simulatable from no information about the underlying secret. Equivalently, for any two possible values of an unopened secret, the induced distributions on the shares indexed by I are identical. Therefore, conditioned on the simulated opened values above, $\mathcal{S}$ can sample the corrupted parties' shares and local randomness with the same distribution as in a real execution.

Consequently, the simulated view is identically distributed to the real view:

$$\{\mathcal{S}(([A]^{(i)})_{i \in I}, b)\} \equiv \{\mathsf{View}_I^{\mathsf{isInverseCD}}(A)\}.$$

Hence, for every corrupted set $I \subseteq \{1, \ldots, N\}$ with $|I| \leq T$, the view of the corrupted parties reveals no information beyond the output bit b. Therefore, $\mathsf{isInverseCD}(A)$ provides perfect privacy. □

5.2 Round and Communication Complexity

We estimate the round and communication complexity step by step.

- **Lines 1 and 4** require 6 offline rounds with $O(Nn^4)$ communication. Although Algorithm 5 includes both offline and online phases, in our setting these computations can be performed prior to receiving A and are therefore classified as offline. There are $(n+1)$ calls of $\mathsf{GenInvMatDet}(\mathbb{F}_{q^{n+1}}, n)$ each of which requires $O(Nn^3)$ communication. Moreover, by sampling one additional random element during the execution of $\mathsf{GenInvMatDet}$, Lines 1 and 4 can be executed in parallel.
- **Line 5** requires 1 offline round with $O(n^2)$ communication.
- **Line 7** requires 2 online rounds with $O(n^4)$ communication.
- **Line 10** requires 1 online round with $O(n)$ communication.

Overall, the protocol consists of 7 offline rounds with communication complexity $O(Nn^4)$ and 3 online rounds with communication complexity $O(n^4)$.

Local computation. We also estimate the local computation since this protocol performs a relatively large amount of local linear-algebra work over the extension field. Let ω denote the matrix multiplication exponent such that the product and the determinant of $n \times n$ matrices can be computed in $O(n^\omega)$ field operations. Each invocation of $\mathsf{GenInvMatDet}$ requires $O(Nn^\omega)$ over $\mathbb{F}_{q^{n+1}}$ local operations per party, dominated by the multiplication of N shared $n \times n$ matrices in $\mathsf{MultMats}$. Since Algorithm 12 invokes this subroutine $n+1$ times, the complexity of local computation in offline rounds is $O(Nn^{\omega+1})$ over $\mathbb{F}_{q^{n+1}}$. The online phase requires $n+1$ multiplications of $[C_i] \cdot [R_i]$ and $n+1$ determinant computations of the public matrices B_i with $O(n^{\omega+1})$ local operations. Thus, the total per-party local computation in online rounds is $O(n^{\omega+1})$ over $\mathbb{F}_{q^{n+1}}$.

Multiplication in $\mathbb{F}_{q^{n+1}}$ amounts to multiplying two polynomials of degree at most n over $\mathbb{F}_q$. With schoolbook polynomial multiplication, it requires $O(n^2)$ operations over $\mathbb{F}_q$. Therefore, with $\omega = 3$, the offline and online rounds require $O(Nn^6)$ and $O(n^6)$ local operations, respectively. If FFT/NTT-based multiplication is available, the multiplication cost is reduced to $O(n \log n)$ base-field operations. In this case, the offline and online complexity becomes $O(Nn^5 \log n)$ and $O(n^5 \log n)$, respectively.

6 Samuelson-Berkowitz in MPC

As noted in Section 1.1, the combination of the Maire-Vergnaud and Schönhage protocols is claimed to handle cases where $p \leq n$. From our analysis in Section 4.4, the multi-party version of this protocol takes 2 offline rounds with $O(n^4)$ communication and 12 online rounds with $O(n^4)$ communication. In this section, we propose our second invertibility testing protocol, achieving fewer online rounds by utilizing the Samuelson-Berkowitz algorithm [Ber84] in an MPC setting.

For each $k \in \{1, \ldots, n\}$, let A_k be the $(n-k+1) \times (n-k+1)$ trailing principal submatrix of A, obtained by taking the last $n-k+1$ rows and columns of A. We decompose A_k as in Equation 5, where $a_{k,k}$ is the upper-left entry of A_k, R_k is the first row of A_k with its first entry removed, S_k is the first column of A_k with its first entry removed, and M_k is the trailing principal submatrix of A_k obtained by deleting its first row and column. For $k = n$, we regard R_n, S_n, and M_n as empty matrices of compatible dimensions.

$$A_k = \left[\begin{array}{c|c} a_{k,k} & R_k \\ \hline S_k & M_k \end{array}\right] \tag{5}$$

Using the above submatrices, we can construct $(n-k+2) \times (n-k+1)$ Toeplitz matrices C_k for $k = 1, \ldots, n$ as follows.

$$C_k := \begin{bmatrix} 1 & 0 & 0 & \cdots & & \\ -a_{k,k} & 1 & 0 & \cdots & & \\ -R_k S_k & -a_{k,k} & 1 & \cdots & & \\ -R_k M_k S_k & -R_k S_k & -a_{k,k} & 1 & \cdots & \\ \vdots & \vdots & \ddots & \ddots & \ddots & \\ -R_k M_k^{n-k-2} S_k & \cdots & \cdots & -R_k S_k & -a_{k,k} & 1 \\ -R_k M_k^{n-k-1} S_k & \cdots & \cdots & \cdots & -R_k S_k & -a_{k,k} \end{bmatrix}$$

The Samuelson-Berkowitz algorithm computes column vector $v = C_1 C_2 \cdots C_n \in \mathbb{F}_q^{n+1}$, where v consists of the coefficients of the characteristic polynomial $\chi_A(x)$. From the vector v, one can directly test the invertibility of A by checking whether the last entry, corresponding to the constant coefficient $(-1)^n \det(A)$, is zero.

6.1 Protocol Description

Our second protocol using the multi-party version of the Samuelson-Berkowitz algorithm is described in Algorithm 13. We also provide the proof of correctness and perfect privacy of Algorithm 13 in Theorem 3 and 4, respectively.

Algorithm 13: $\mathsf{isInverseSB}(A)$ - Invertibility test using Samuelson-Berkowitz algorithm

Input : $A \in \mathcal{M}_n(\mathbb{F}_q)$
Output: A bit $b \in \{0,1\}$ such that $b = 1$ iff A is invertible

1 The parties obtain $[M_k], [R_k]$, and $[S_k]$ for $k \in \{1, ..., n\}$ as described above.
2 The parties generate $([T_k], [T_k^{-1}]) \leftarrow \mathsf{GenInvMats}(\mathbb{F}_q, 1, n-k+2)$ for $k \in \{1, ..., n\}$.
3 The parties generate random field element $([r], [r^{-1}]) \leftarrow \mathsf{GenNonzeroElems}(\mathbb{F}_q, 1)$.
4 The parties compute $[T_1'] = [r] \cdot [T_1]$.
5 **for** $k = 1$ **to** n **do**
6 The parties compute $[M_k^2], \cdots, [M_k^{n-k-1}] \leftarrow \mathsf{SecPower}(\mathbb{F}_q, M_k, n-k-1)$ if $k \leq n-3$. Otherwise, no higher powers are needed.
7 **if** $k \neq n$ **then**
8 The parties obtain $[C_k]$ by computing $[R_k][M_k^0][S_k], \cdots, [R_k][M_k^{n-k-1}][S_k]$.
9 The parties compute and open $U_k \leftarrow \mathsf{open}([T_k^{-1}][C_k][T_{k+1}])$.
10 **else**
11 The parties locally compute $[C_n] \leftarrow (1, -[a_{n,n}])^\top$
12 The parties compute and open $U_n \leftarrow \mathsf{open}([T_n^{-1}][C_n])$.
13 The parties locally compute $\left[\begin{pmatrix} p_0 & p_1 & \cdots & p_n \end{pmatrix}^\top\right] \leftarrow [T_1'] \cdot U_1 U_2 \cdots U_n$ and open $p_n \leftarrow \mathsf{open}([p_n])$.
14 **return** $\begin{cases} 0 & \text{if } p_n = 0 \\ 1 & \text{otherwise.} \end{cases}$

Theorem 3 (Correctness)**.** *For any matrix $A \in \mathcal{M}_n(\mathbb{F}_q)$, $\mathsf{isInverseSB}(A)$ outputs 1 if and only if A is invertible over $\mathbb{F}_q$.*

Proof. From the correctness of the Samuelson–Berkowitz algorithm, the expression in Line 13

$$\left[\begin{pmatrix} p_0 & p_1 & \cdots & p_n \end{pmatrix}^\top\right] = [T_1'] \cdot U_1 U_2 \cdots U_n = [r \cdot T_1] \cdot T_1^{-1} C_1 \cdots C_n$$

denotes a secret share of $r \cdot v$, where the vector $v = \begin{pmatrix} v_0 & v_1 & \cdots & v_n \end{pmatrix}^\top$ contains the coefficients of the characteristic polynomial of A. Since r is nonzero, $p_n = 0$ if and only if $v_n = (-1)^n \det(A) = 0$, which implies that A is singular. □

Theorem 4 (Perfect Privacy)**.** *For any matrix $A \in \mathcal{M}_n(\mathbb{F}_q)$, $\mathsf{isInverseSB}(A)$ reveals no information beyond the output bit indicating whether A is invertible.*

Proof. Let $b \in \{0,1\}$ denote the output of $\mathsf{isInverseSB}(A)$, where $b = 1$ if and only if A is invertible. We construct a simulator $\mathcal{S}$ that, given only b and the inputs of the corrupted parties, produces a distribution identical to $\mathsf{View}_I^{\mathsf{isInverseSB}}(A)$.

We use the perfect privacy of the underlying building blocks stated in Section 3. Their internal executions can therefore be simulated from the information revealed by their functionalities. It remains to analyze only the values explicitly opened by $\mathsf{isInverseSB}$ itself.

During the execution of $\mathsf{isInverseSB}(A)$, the only opened values are the matrices $U_1, \ldots, U_n$ (Lines 9 and 12) and the final scalar p_n (in Line 13).

Since the Toeplitz matrices C_k have full column rank and the masking matrices T_k are chosen uniformly and independently from $\mathrm{GL}_{n-k+2}(\mathbb{F}_q)$, we can establish both the uniformity and mutual independence of $\{U_k\}_{k \in \{1,\ldots,n\}}$ by analyzing them in reverse order.

First, the matrix $U_n = T_n^{-1} C_n$ is uniformly distributed over the set of full-column-rank 2×1 matrices over $\mathbb{F}_q$. For any preceding matrix $U_k = T_k^{-1} C_k T_{k+1}$ for $k < n$, even if the mask T_{k+1} is conditioned on the subsequent matrix U_{k+1}, left-multiplying the full-column-rank matrix $C_k T_{k+1}$ by the independent and uniform mask T_k^{-1} perfectly hides this term.

Thus, U_k becomes completely uniform and independent of all $\{U_j\}_{j>k}$. Consequently, by backward induction, the sequence $\{U_k\}_{k\in\{1,\ldots,n\}}$ consists of mutually independent random matrices, each uniformly distributed over the set of full-column-rank $(n-k+2)\times(n-k+1)$ matrices over $\mathbb{F}_q$, revealing no information about A.

Regarding the final opening, Theorem 3 establishes that $p_n = r\cdot(-1)^n \det(A)$. Therefore, if $b = 0$ (i.e., A is singular), $p_n = 0$. Otherwise, p_n is uniformly distributed over $\mathbb{F}_q \setminus \{0\}$ since r is a uniform non-zero element.

To simulate the transcript, $\mathcal{S}$ samples independent uniformly random full-column-rank matrices $\widetilde{U}_k$ from the set of full-column-rank matrices of size $(n-k+2)\times(n-k+1)$ over $\mathbb{F}_q$. Next, it sets $\widetilde{u} = 0$ if $b = 0$, and samples $\widetilde{u} \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$ otherwise. Because the underlying degree-T Shamir's secret sharing and the procedures for unopened intermediate values are identical to those in $\mathsf{isInverseCD}(A)$, $\mathcal{S}$ can simulate the unopened shares and local randomness exactly as described in the previous proof. This yields a simulated view that is identically distributed to the real execution. Consequently, $\mathsf{isInverseSB}(A)$ provides perfect privacy. □

6.2 Round and Communication Complexity

We estimate the round and communication complexity step by step.

- **Lines 2–3** require 2 offline rounds with $O(n^3)$ communication.
- **Line 4** requires 1 offline round and $O(n^2)$ communication.
- **Line 6** requires 2 offline rounds with $O(n^4)$ communication, and 3 online rounds with $O(n^4)$ communication.
- **Line 8** requires 2 online rounds with $O(n^3)$ communication.
- **Lines 9 and 12** require 3 online rounds with $O(n^3)$ communication.
- **Line 13** requires 1 online round with $O(1)$ communication.

For Lines 6–12, the total communication complexity is summed over $k \in \{1,\ldots,n\}$ with the computations for different k performed in parallel. Lines 2, 3, and offline rounds of $\mathsf{SecPower}$ in Line 6 can be done in parallel, while Line 4 must be performed after the completion of Lines 2 and 3. As a result, the overall complexity of this protocol is 3 offline rounds with communication complexity $O(n^4)$ and 9 online rounds with communication complexity $O(n^4)$.

7 Conclusion

In this paper, we addressed the challenge of privately testing matrix invertibility over fields with small order or characteristic in the multi-party setting. Operating under the arithmetic black-box model, we proposed two protocols specifically tailored for such fields that achieve perfect privacy in constant rounds.

While this work establishes a robust foundation, several avenues for further research remain. Practically, integrating these protocols into concrete MPC frameworks such as SPDZ or RSS is essential to assess their performance under realistic network constraints. Theoretically, reducing extension-field overhead and refining preprocessing constants remain promising directions for enhancing overall efficiency.

References

[ABCN26] Diego F Aranha, Giacomo Borin, Sofia Celi, and Guilhem Niot. Optimizing and implementing threshold mayo. *Cryptology ePrint Archive*, 2026. URL: https://eprint.iacr.org/2026/710.

[AFL+16] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 805–817. ACM Press, October 2016. doi:10.1145/2976749.2978331.

[AOBV26] Paco Azevedo-Oliveira, Jordan Beraud, and Pierre Varjabedian. Threshold linear solving in small fields and application to uov. *Cryptology ePrint Archive*, 2026. URL: https://eprint.iacr.org/2026/189.

[BCC+23] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. MAYO. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

[BCD+24] Ward Beullens, Ming-Shing Chen, Jintai Ding, Boru Gong, Matthias J. Kannwischer, Jacques Patarin, Bo-Yuan Peng, Dieter Schmidt, Cheng-Jhih Shih, Chengdong Tao, and Bo-Yin Yang. UOV — Unbalanced Oil and Vinegar. Technical report, National Institute of Standards and Technology, 2024. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures.

[Ber84] Stuart J Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information processing letters*, 18(3):147–150, 1984. doi:10.1016/0020-0190(84)90018-8.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988. doi:10.1145/62212.62213.

[BIB89] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In Piotr Rudnicki, editor, *8th ACM PODC*, pages 201–209. ACM, August 1989. doi:10.1145/72981.72995.

[CD01] Ronald Cramer and Ivan Damgård. Secure distributed linear algebra in a constant number of rounds. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 119–136. Springer, Berlin, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_7.

[CDI05] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 342–362. Springer, Berlin, Heidelberg, February 2005. doi:10.1007/978-3-540-30576-7_19.

[CEN25] Sofía Celi, Daniel Escudero, and Guilhem Niot. Share the MAYO: Thresholdizing MAYO. In Ruben Niederhagen and Markku-Juhani O. Saarinen, editors, *Post-Quantum Cryptography - 16th International Workshop, PQCrypto 2025, Part I*, pages 165–198. Springer, Cham, April 2025. doi:10.1007/978-3-031-86599-2_6.

[CKP07] Ronald Cramer, Eike Kiltz, and Carles Padró. A note on secure computation of the Moore-Penrose pseudoinverse and its application to secure linear algebra. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 613–630. Springer, Berlin, Heidelberg, August 2007. doi:10.1007/978-3-540-74143-5_34.

[CS19] Daniele Cozzo and Nigel P. Smart. Sharing the LUOV: Threshold post-quantum signatures. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *LNCS*, pages 128–153. Springer, Cham, December 2019. doi:10.1007/978-3-030-35199-1_7.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Berlin, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_38.

[MV24] Jules Maire and Damien Vergnaud. Secure multi-party linear algebra with perfect correctness. *CiC*, 1(1):29, 2024. doi:10.62056/avzojbkrz.

[MW08] Payman Mohassel and Enav Weinreb. Efficient secure linear algebra in the presence of covert or computationally unbounded adversaries. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 481–496. Springer, Berlin, Heidelberg, August 2008. doi:10.1007/978-3-540-85174-5_27.

[Sch93] Arnold Schönhage. Fast parallel computation of characteristic polynomials by leverrier's power sum method adapted to fields of finite characteristic. In *International Colloquium on Automata, Languages, and Programming*, pages 410–417. Springer, 1993. doi:10.1007/3-540-56939-1_90.