

Algorithm-Based Secure and Fault Tolerant Outsourcing of Matrix Computations

Amrit Kumar, Jean-Louis Roch

▶ **To cite this version:**

Amrit Kumar, Jean-Louis Roch. Algorithm-Based Secure and Fault Tolerant Outsourcing of Matrix Computations. 2013. hal-00876156

HAL Id: hal-00876156
https://hal.science/hal-00876156

Preprint submitted on 23 Oct 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Algorithm-Based Secure and Fault Tolerant Outsourcing of Matrix Computations

Amrit Kumar and Jean-Louis Roch

MOAIS : LIG-INRIA Grenoble joint team
amrit.kumar@inria.fr,jean-louis.roch@imag.fr

Abstract

We study interactive algorithmic schemes for outsourcing matrix computations on untrusted global computing infrastructures such as clouds or volunteer peer-to-peer platforms. In these schemes the client outsources part of the computation with guaranties on both the inputs' secrecy and output's integrity. For the sake of efficiency, thanks to interaction, the number of operations performed by the client is almost linear in the input/output size, while the number of outsourced operations is of the order of matrix multiplication. Our scheme is based on efficient linear codes (especially evaluation/interpolation version of Reed-Solomon codes). Confidentiality is ensured by encoding the inputs using a secret generator matrix, while fault tolerance is ensured together by using fast probabilistic verification and high correction capability of the code. The scheme can tolerate multiple malicious errors and hence provides an efficient solution beyond resilience against soft errors. These schemes also allow to securely compute multiplication of a secret matrix with a known public matrix. Under reasonable hypotheses, we further prove the non-existence of such unconditionally secure schemes for general matrices.

1 Introduction

Computational power is often asymmetric. On the one hand, global computing platforms such as clouds available through Internet provide a large scale of computational power and resources, but remain susceptible to various kinds of malicious attacks and faults. While on the other hand, a relatively weak local client is secure and reliable. The goal of today's computing infrastructures is to provide solutions to draw the benefits of both of these components while ensuring secrecy of certain inputs.

With the advent of general *Fully-Homomorphic Encryption* (FHE) schemes as in [7], any computation can be securely outsourced. However, such schemes suffer from efficiency issues as the key size and the cipher-text size are too long compared to the plain-text size. This makes large matrix computations impractical to outsource. A more specific symmetric scheme presented in [9] can be extended to operate on matrices. Despite the gain in efficiency the scheme defeats the purpose of outsourcing as it involves matrix-matrix multiplication. Furthermore, these schemes do not address perfect secrecy.

In an outsourcing scenario the client should be able to efficiently verify the correctness of the result returned by the untrusted platform. Solutions for general computations either rely on Probabilistically Checkable Proofs (PCPs) [1], or FHE. Considering their complexity, these constructions are currently beyond practical use. However, authors in [11] propose a new construction that can efficiently verify general computations under cryptographic assumptions. This construction compactly encodes computations as quadratic programs ([6]) which are then encoded as elements of a group equipped with a bi-linear map. The weak client receives a proof (of constant size) along with the computational result. The verification procedure involves group operations. However, this scheme does not provide secrecy of inputs.

Furthermore, on large scale computing systems error resilience is an issue. Error probability increases with the node count ([4]). Algorithm-based Fault Tolerance (ABFT) [8] solutions have been explored for matrix computations without considering privacy. Focusing on a small rate of *soft errors*, an ABFT dense linear system solver is provided in [4] that is based on a low density parity check matrix. *Soft errors* in general are produced in case the computing system is subject to cosmic radiations. The complexity of localising and correcting errors using these techniques however increases with the number of errors.

Yet, on externalised computing platforms, malicious attacks that may corrupt a large number of intermediate computations are of concern. Such massive attacks occur due to Trojan attacks, and more generally orchestrated attacks against widespread vulnerabilities of a specific operating system that may result in the corruption of a large number of resources.

In [12], an ABFT efficient solution for vector-matrix multiplication is proposed for integrity against massive attacks and is based on BCH ([3]) codes. In this work we extend this solution in both directions. First, to provide secrecy, we use a secret code, based on a private Vandermonde generator matrix. Second, we extend the scheme to more general matrix computations such as LU factorisation or matrix powering (with application to connected components in a graph), where the output data is non-linear in the input data.

A major constraint for these constructions is efficiency : trivial lower bounds for the cost of an interactive scheme are the size of the input and the output (memory cost) and the work of the best known algorithm (computational cost).

Our symmetric scheme for matrix multiplication is almost optimal with respect to both the input/output size on the reliable resource (user side) and the best upper bound on the unreliable one (global computing platform side). In section 3 the scheme leads to interactive protocols to efficiently outsource other matrix computations. Finally in section 4 we provide new impossibility results for achieving perfect secrecy for outsourcing multiplication of a secret matrix and a known public matrix. Yet, partially secret schemes are provided that tolerate massive errors.

2 ABFT Dense Matrix Multiplication

As most of the linear algebra computations reduce to dense matrix-matrix multiplication, the design of interactive *zero-knowledge protocols* for matrix computations is based on outsourcing matrix multiplication. For the sake of clarity, we restrict in the sequel to $k \times k$ square matrices. The goal of these protocols is to keep the complexity of the operations almost linear in the size of the input ($O(k^2)$) on the weak client, while on the unreliable cloud, a complexity $\widetilde{O}(n^\omega)$ would be acceptable, where $2 < \omega \leq 3$ denotes the exponent in the matrix multiplication cost.

A standard way for ABFT matrix multiplication consists in encoding the left and right operands by multiplying each by the generator matrix of a linear code ([8]). These codes defined over a base field $\mathbb{F}$ are maximum distance separable. For any n with $\text{card}(\mathbb{F}) \geq n > k$, an (n, k) code is characterized by a $k \times n$ generator matrix G. A source vector x of size k is encoded by $y = x \cdot G$. In particular, to correct very few errors, G is chosen as a LDPC (Low Density Parity Check) code with $\mathcal{O}(k)$ coefficients. To correct higher error rate G can be chosen as Vandermonde matrix (evaluation-interpolation code) that defines a Reed-Solomon (RS) code and corrects any configuration of $(n-k)/2$ errors in y. For RS code we assume that $\mathbb{F}$ is large enough.

The proposed secured ABFT protocol is as follows. The weak client initiates the protocol by generating two (n, k) RS codes, defined by G_1 and G_2. The input $k \times k$ matrices A and B are encoded as : $G_A = tr(G_1) \cdot A$ and $G_B = B \cdot G_2$, where tr denotes the transpose map. G_1

and G_2 are kept secret which eventually renders A and B secret. The client sends G_A and G_B to the global platform that performs $G_A \cdot G_B$ and sends back the result. Various errors may occur during computations or communication. The received matrix $\widetilde{R}$ is seen as a perturbation of the correct encoded result $G_C = G_A \cdot G_B$. Upon decoding $\widetilde{R} = G_C + E$, where $E_{n\times n}$ is the error matrix, client obtains the desired matrix product $C = A \cdot B$. Thanks to unique decoding, the client can correct up to $(n-k)^2/4$ errors in $\widetilde{R}$ to recover C.

The cost of computation on the reliable client sums to the cost of encoding and decoding. The encoding of each row or column reduces to k polynomial evaluations of degree k in n points, each computed in $\widetilde{\mathcal{O}}(n)$ with precomputation, so $\widetilde{\mathcal{O}}(n^2)$ for the full matrices A and B. With fast extended GCD, decoding can be performed in $\widetilde{\mathcal{O}}(n)$ for each row or column, so $\widetilde{\mathcal{O}}(n^2)$ for the matrix $\widetilde{R}$. Multiplication performed by the remote platform costs $O(n^\omega)$.

The client also has the possibility to verify correctness of the result. This verification allows to detect extremely unreliable or collusive workers and even prevents man-in-the-middle (MITM) attacks. Verification allows to abort an interactive protocol while reducing computational overhead. In MITM scenario, an adversary might simply intercept the communication between the client and the platform and replaces the result by some other *good-looking* matrix, ex. the adversary might replace the matrix $\widetilde{R}$ by $G_A \cdot D \cdot G_B$ for a random D. For verification i.e. testing if $C = A \cdot B$, we propose to use probabilistic Freivalds' algorithm ([5]) which runs in $O(n^2)$, and states the correctness with good probability.

The evaluation points in general are randomly chosen and kept secret. However, if secrecy is discarded, the evaluation points can be chosen as $\{1, \alpha, \alpha^2, \ldots, \alpha^{\mathrm{card}(\mathbb{F})-2}\}$, where α is a generator of the multiplicative group of the base field $\mathbb{F}$. With this choice of evaluation points, Fast Fourier Transformation (FFT) allows fast encoding and decoding and provides a logarithmic advantage. We note that a small field (such as $\mathbb{F}_2$) would not provide enough evaluation points.

3 Illustrative Examples

Interactive matrix inversion. We use relation (T) where $R = (A - B \cdot D^{-1} \cdot C)$ and suppose that D and R are invertible. Interactive matrix inversion requires to recursively calculate D^{-1} and R^{-1}. $\begin{bmatrix} A_{n\times n} & B_{n\times n} \\ C_{n\times n} & D_{n\times n} \end{bmatrix}^{-1} = \begin{bmatrix} R^{-1} & -R^{-1} \cdot B \cdot D^{-1} \\ -D^{-1} \cdot C \cdot R^{-1} & D^{-1} + D^{-1} \cdot C \cdot R^{-1} \cdot B \cdot D^{-1} \end{bmatrix}$ (T)
Let $T^{\text{client}}(n)$ be the client's cost of outsourcing while $T^{\text{worker}}(n)$ be the cost on the worker's side. Then $T^{\text{client}}(2n) = 2T^{\text{client}}(n) + \Theta(\text{Cost of outsourcing multiplication})$ and $T^{\text{worker}}(2n) = 2T^{\text{worker}}(n) + \mathcal{O}(n^w)$. Hence, inverse can be outsourced with $T^{\text{worker}}(n) = \widetilde{\mathcal{O}}(n^w)$ and $T^{\text{client}}(n) = \widetilde{\mathcal{O}}(\text{Cost of outsourcing multiplication})$.

Interactive block LU decomposition. To outsource block LU we use the relation : $\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} I & 0 \\ C \cdot A^{-1} & I \end{bmatrix} \times \begin{bmatrix} A & B \\ 0 & S \end{bmatrix}$ where $S = D - C \cdot A^{-1} \cdot B$. Computation of A^{-1} is outsourced and so are the multiplications involved in computing S. Recursively outsourcing of decomposition of S leads to the same client-cost as in outsourcing multiplication.

Linear system solving. Efficient Block LU decomposition leads to a $\widetilde{\mathcal{O}}(n^2)$ algorithm for solving linear system of equations : $Ax = b$ with A and b secret. The client interactively retrieves the LU decomposition of A and outsources inversion of L and U, and finally computes the product $U^{-1} \cdot L^{-1} \cdot b$ locally in $\mathcal{O}(n^2)$.

4 On Information-Theoretic Security

Considering the fact that an RS code is parametrized using only n coefficients, the secrecy provided by the scheme to encrypt an $n \times n$ matrix is very limited. We analyse the existential possibility of similar schemes which could provide perfect secrecy. We focus our search to only those schemes which allow to outsource multiplication of a secret matrix with a public matrix. We consider a symmetric encryption scheme defined using three algorithms $\mathcal{KG}, \mathcal{E}$, and $\mathcal{D}$, denoting key-generation, encryption and decryption algorithms respectively. We consider the spaces $\mathcal{K}, \mathcal{M}$ and $\mathcal{C}$ denoting the key-space, clear-text space and the cipher-text space respectively. In the sequel we consider encryption schemes of type $\mathcal{E}(K, M) = K \cdot M$ for suitable $\mathcal{K}$ and $\mathcal{M}$, we call it *product-based encryption*. In the following we write *linear time* cost to denote $\widetilde{\mathcal{O}}(n^2)$ complexity.

4.1 Impossibility Results

Lemma 4.1. *If there exists a product-based encryption with cost $\mathcal{O}(f(n))$ for invertible matrices which achieves perfect secrecy then the cost of matrix multiplication is $\mathcal{O}(f(n))$.*

Proof. Perfect secrecy is defined as: $\forall A$ invertible $\forall K \in \mathcal{K}$, $\forall A'$ invertible $\exists K'$ verifying $A' \cdot K' = A \cdot K$, so $K' = (A')^{-1} \cdot A \cdot K$. Thus $\mathcal{K} \supset \{M_1 \cdot M_2 \cdot K \mid M_1, M_2$ invertible and $K \in \mathcal{K}\}$. Now consider two invertible matrices A and B. We choose $K_1 \in \mathcal{K}$ and compute $K_2 = \mathcal{E}(K_1, B)$ in $\mathcal{O}(f(n))$. As $K_2 = B \cdot K_1 = I_n \cdot B \cdot K_1$, K_2 belongs to $\mathcal{K}$. Hence $A \cdot B = \mathcal{D}(K_1, \mathcal{E}(K_2, A))$ can be computed in $\mathcal{O}(f(n))$. □

Corollary 1. *Unless there exists a linear time algorithm for matrix multiplication, there does not exist a product-based matrix encryption scheme for invertible matrices which ensures perfect secrecy and runs with linear cost.*

The following lemma characterises perfectly secure encryption schemes which allow a user to export matrix multiplication of a secret matrix with a known public matrix. For lemma 4.2 we consider an encryption scheme defined over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$ where $\mathcal{M}$ is the set of all $n \times p$ matrices and $\mathcal{C}$ the set of all $m \times n$.

Lemma 4.2. *If I_n $(p = n)$ belongs to $\mathcal{M}$ and the scheme satisfies : $\mathcal{E}(K, M_1) \cdot M_2 = \mathcal{E}(K, M_1 \cdot M_2)$ for all M_1, M_2 in the message space $\mathcal{M}$ and for each key $K \in \mathcal{K}$ then, $\mathcal{E}(K, M) = S(K) \cdot M$ for some $S(K)$ in $\mathcal{C}$.*

Lemma 4.2 proves that any such encryption scheme with identity matrix in the message space must be a product-based matrix encryption. The key-space $\mathcal{K}$ is the set of encryptions of the identity matrix.

Main theorem. We now present our main theorem which builds on a weaker and more general property than the one used in Lemma 4.2 : $\mathcal{E}(K, M_1) \cdot M_2 = \mathcal{E}(K, M_1 \cdot M_2) \quad \forall M_1,\ M_2 \in \mathcal{M}$ and $\forall K \in \mathcal{K}$.

For this purpose, we suppose that there are two functions $\phi_{\text{Eval}} : \mathcal{C} \times \mathcal{M} \to \mathcal{I}$ and $\psi : \mathcal{K} \times \mathcal{I} \to \mathcal{M}$ where $\mathcal{I}$ is an intermediate subspace of the space containing all $m \times n$ matrices.

Theorem 4.3. *If I_n belongs to $\mathcal{M}$, and $\forall M_1, M_2 \in \mathcal{M}$ $\psi(K, \phi_{Eval}(\mathcal{E}(K, M_1), M_2)) = M_1 \cdot M_2$, with $\phi_{Eval}(C, M) = C \cdot M$ $\forall C \in \mathcal{C}$ and $\forall M \in \mathcal{M}$, then the encryption scheme is product-based.*

Proof. We have, $\forall M_1, M_2 \in \mathcal{M}$ and $\forall K \in \mathcal{K}$ $\psi(K, \mathcal{E}(K, M_1) \cdot M_2) = M_1 \cdot M_2$ (1). Choosing $M_2 = I_n$ in (1), we obtain $\forall M_1 \in \mathcal{M}$, $\forall K \in \mathcal{K}$ $\psi(K, \mathcal{E}(K, M_1)) = M_1$ which implies that $\psi(K)$ is the decryption function for K. If the encryption algorithm was deterministic, encryption of equation (1) using K gives $\mathcal{E}(K, M_1) \cdot M_2 = \mathcal{E}(K, M_1 \cdot M_2)$ $\forall M_1, M_2 \in \mathcal{M}$. The same can be obtained for randomized encryption if the same random coins are used for encryption as the one used in the inner encryption in LHS of (1). Lemma 4.2 now applies and we prove that the scheme is product-based. □

Combining Theorem 4.3 and corollary 1, we conclude that no matrix encryption scheme exists which runs in linear time and which can evaluate the product of an encrypted and a public matrix. The special case in which Theorem 4.3 has been proven assumes a reasonable hypothesis knowing the fact that, we are only interested in characterising schemes which yield matrix product. We further claim that no efficient (running in linear time) multiplicative homomorphic encryption over matrices exists. This follows from the characterization of automorphism ([10]) maps over the general linear group of degree n (i.e. the set of all $n \times n$ invertible matrices over a ring).

4.2 Alternatives for Partial Secrecy

In subsection 4.1, we proved that no outsourcing scheme for matrix multiplication exists which could attain perfect secrecy. For many applications perfect secrecy can be an overkill, a scheme ensuring quantified secrecy would be enough.

Noisy shares in Shamir's protocol. A Shamir based $(n, (n-1)/2)$ threshold scheme ([13]) is described in [2]. We can design a Shamir's protocol based encryption scheme for single worker with quantified and partial secrecy. The client wishes to outsource multiplication of two matrices A and B. She chooses polynomials of degree $k-1$ to hide the entries of the matrices and $r+k$ shares of points for each one of them. The first r values are noises $(r_i, P(r_i))$ i.e. false values, and the remaining $(k_i, P(k_i))$ for the correct values, where P is the polynomial for each entry of A and B.

The client then permutes these shares and sends them to the worker. The worker computes all the matrix multiplications and returns the result, the client upon reception applies the inverse permutation and filters out the correct matrices and by interpolation retrieves the result.

The security of the schemes relies on the secret permutation and hence the worker has to find the correct permutation from $(r+k)!$ choices.

Using product-based encryption. An intermediate solution would be to design a product-based scheme where the set of coding matrices $\mathcal{G}$ is the set of matrices defined using $\mathcal{O}(n)$ parameters. Such a scheme can be easily extended to design a multiplicative homomorphic encryption scheme. The idea is to encrypt A as $C_A = tr(G_1) \cdot S_1 \cdot A$ and B as $C_B = B \cdot S_2 \cdot G_2$ where S_1 and S_2 are defined using $\mathcal{O}(n)$ parameters, and G_1, G_2 are generator matrices. The matrix product $A \cdot B$ can be retrieved by first decoding and then multiplying left by S_1^{-1} and right by S_2^{-1}.

Clearly, the security of the encryption scheme relies on the size of the key-space. We denote $\mathcal{G}$ to be the set of all matrices over a finite field $\mathbb{F}$ of size $|\mathbb{F}|$ *with* $\mathcal{O}(n)$ coefficients and set the key-space $\mathcal{K}$ to be the set of all matrices which can be written as an arithmetic expression over matrices of constant size say λ. In other terms a key k is a finite sum of finite product of matrices over $\mathcal{F}$. For example $k = K_1 \cdot K_2 \cdot K_3 + K_4 \cdot K_5 + K_6$ where $K_i \in \mathcal{F}$.

As the matrices in $\mathcal{G}$ have only $\mathcal{O}(n)$ coefficients, $|\mathcal{G}| = |\mathbb{F}|^{\mathcal{O}(n)} \cdot \binom{n^2}{\mathcal{O}(n)}$. Furthermore, the permissible arithmetic operations are either $\{+, \cdot, \{\cdot\}^{-1}, \{\cdot\}^1\}$ i.e. addition, multiplication, inverse or identity, $|\mathcal{K}| = |\mathcal{F}|^\lambda \cdot 4^\lambda$.

5 Conclusion

In this paper, we design an efficient alternative to FHE based outsourcing with practicality and acceptable security (for certain applications). Our ABFT solution is resilient against malicious errors and hence goes beyond the correction of soft errors and can even handle MITM attacks. The scheme computes matrix operations such as matrix-matrix multiplication and can be extended to interactive protocols performing more complex operations on matrices.

The intriguing question that remains is whether problems with known polynomial time algorithms admit unconditionally secure outsourcing schemes. This question is even more difficult to answer when the gap between the targeted cost of the outsourcing scheme and the best known algorithm is too narrow. We addressed the problem of matrix-matrix multiplication, with $\mathcal{O}(n^3)$ vs. $\widetilde{\mathcal{O}}(n^2)$ cost gap. Another perspective is to extend the ABFT scheme to work on floating point numbers and measure the stability of computations.

References

[1] Sanjeev Arora. Probabilistic checking of proofs: a new characterization of np. In *Journal of the ACM*, pages 2–13, 1998.

[2] Mikhail J. Atallah and Keith B. Frikken. Securely outsourcing linear algebra computations. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 48–59, New York, NY, USA, 2010. ACM.

[3] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, March 1960.

[4] Peng Du, Piotr Luszczek, and Jack Dongarra. High performance dense linear system solver with resilience to multiple soft errors. In *ICCS*, pages 216–225, 2012.

[5] Rusins Freivalds. Probabilistic machines can use less running time. In *IFIP Congress*, pages 839–842, 1977.

[6] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT*, pages 626–645, 2013.

[7] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[8] Kuang-Hua Huang and J. A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Trans. Comput.*, 33(6):518–528, June 1984.

[9] Aviad Kipnis and Eliphaz Hibshoosh. Efficient methods for practical fully homomorphic symmetric-key encrypton, randomization and verification. Cryptology ePrint Archive, Report 2012/637, 2012. `http://eprint.iacr.org/`.

[10] Bernard R. McDonald. Automorphisms of gl_n(r). *Transactions of the American Mathematical Society*, 215:145–159, 1976.

[11] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 238–252, 2013.

[12] Jean-Louis Roch and Sebastien Varrette. Probabilistic certification of divide & conquer algorithms on global computing platforms. application to fault-tolerant exact matrix-vector product. In ACM publishing, editor, *Parallel Symbolic Computation'07 (PASCO'07)*, London, Ontario, Canada, July 2007.

[13] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.