Encrypted Coordinate Transformation via Parallelized Somewhat Homomorphic Encryption for Robotic Teleoperation

Hyuk Bin Kwon[1], Shane Kosieradzki[1], Jacob Blevins[1], and Jun Ueda[1]

***Abstract*— This paper seeks to understand the viability of encrypted robot control. Controllers are susceptible to malicious attacks unless controller parameters are encrypted; however, homomorphic encryption is necessary in order to allow controller mathematical operations on encrypted text, but is limited due to heavy computational overhead. Encrypted control is accomplished via the implementation of Dyer's somewhat homomorphic encryption scheme on multi and single threaded matrix transformations in order to telecommunicate movement commands between a virtual-reality joystick and a robot arm. Results find that encrypted teleoperation via the user interface is a viable encrypted controller technique, and is optimally produced on multi-threaded systems.**

Index terms: Homomorphic encryption, Teleoperation, Multi-threading, Security parameters, Virtual Reality, Motion Control

I. Introduction

Resilience to cyber attacks is becoming more and more important in many connected robotic systems [1], [2]. Weak cyber security allows adversaries to attain unauthorized access to control systems, resulting in data breaches, falsification, and infrastructure failure [3]–[6]. Control schemes and controller gains are carefully designed and tuned when being implemented into the system. Falsification of real-time control schemes, even of a few parameters, may cause a substantial performance decrease or immediate instability. One of the cybersecurity measures to prevent falsification is encryption. In particular, encrypted control is an emerging concept that encrypts not only signals on communication lines, but also control schemes and controller gains by applying homomorphic encryption algorithms [7]–[10].

Among existing homomorphic encryption algorithms, fully homomorphic encryption (FHE) can perform both addition and multiplication on the ciphertext an unlimited number of times. On the other hand, partially homomorphic encryption (PHE) performs either addition or multiplication an unlimited number of times. Where real-time control of robotic systems is concerned, practically usable FHE has not yet been realized due to high computational load. PHE however has been used in the majority of the existing encrypted control studies for practical reasons; most of these studies are limited to linear and relatively-low dimensional controllers [7], [11]–[13].

[1]G.W.W. School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia, 30332-0405. Email: bin.kwon, skosieradzki3, jacob.blevins, jun.ueda}@gatech.edu.

This work was supported in part by the National Science Foundation under Grant No. 2112793. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Expansion of encrypted control methodologies to general nonlinear and/or time-varying control has not been well studied. Even "text-level" transformation between coordinates, a simple kinematics problem in robotics, involves matrix multiplications, which cannot be performed by PHE. A "somewhat homomorphic encryption" (SHE) algorithm proposed by Dyer et al. has shown promise of real-time controller encryption [14]. SHE is a family of algorithms that can perform both additive and multiplicative homomorphic encryption with a limited number of operations. The authors' group has applied that particular SHE algorithm to dynamic controllers with nonlinear expressions [15]–[17]. Known limitations of SHE require the users to choose appropriate security parameters and quantization levels to balance between the performance and numerical computation stability (i.e., overflow). The original message cannot be recovered in case of overflow, and is a critical issue in leveraging SHE algorithms in systems.

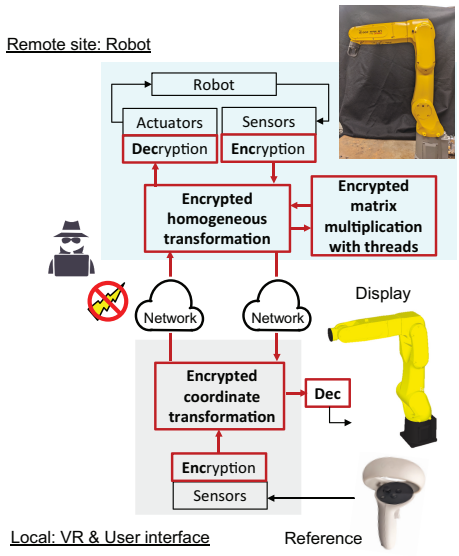


Fig. 1: Encrypted teleoperation of a robot manipulator.

The objective of this paper is to implement and demonstrate encrypted homogeneous transformation with SHE as shown in Fig. 1. Past literature proposing an encrypted controller presents primarily presents a numerical example of new methods. In this paper in addition to a simulation of the proposed methods, a physical implementation of the system is presented. As an experimental platform, a virtual reality (VR) interface, in the form of a joystick, was used for teleoperation of an industrial robot arm. The homogeneous transformation between the user interface and the robot is encrypted by using

Dyer's SHE scheme [18].

Given the high computational cost of homomorphic operations, the execution time of elementary arithmetic operations (i.e. addition and multiplication) become nontrivial dominating the CPU's resources [17]. Thus to minimize system impact, homomorphic calculations should be run concurrently whenever possible. In the case of matrix multiplication each element in the product matrix is computed by the dot product of a row and column in the factor matrices. Each dot product can be computed independent of each other, thus can be constructed as a distinct thread-safe job.

In order to ensure a real-time response between the user interface and the robot, optimal timing is vital. Thus, these dot product jobs are submitted to a thread manager to accelerate encrypted matrix multiplication and to ensure real-time computation of encrypted transformations as well as communication between multiple software platforms.

This paper is organized as follows: Section II gives the methodology utilized to encrypt and geometrically represent the matrix operations. Section III describes the encrypted, teleoperated system. Section IV presents experimental results and their analyses. Finally, Section V provides concluding remarks.

II. Methodology

A. Geometric representation and its encryption

The positions of the user control device (a hand-held virtual reality (VR) interface) and the robot end-point are represented by the coordinate frames shown in Fig. 2. For a discretized time $t = k\Delta_t$ where Δ_t is a sampling time and k is a counter $(k = 0, 1, \cdots)$, the transformation matrix ${}^{C0}\boldsymbol{T}_C$ transforming the initial user interface state ${}^{0}\boldsymbol{T}_{C0}$ to the current user interface state ${}^{0}\boldsymbol{T}_C(k)$ is given as ${}^{C0}\boldsymbol{T}_C(k) = {}^{0}\boldsymbol{T}_{C0}^{-1}\,{}^{0}\boldsymbol{T}_C(k)$. Frames $C0$ and $R0$ represent the initial states of the user interface and robot, respectively, and C and R are the current states. This transformation is applied to the initial robot state to move the robot's end-point as the desired state, i.e.,

$$ {}^{0}\boldsymbol{T}_R(k) = {}^{0}\,\boldsymbol{T}_{R0}\,{}^{C0}\boldsymbol{T}_C(k) = {}^{0}\,\boldsymbol{T}_R\,{}^{0}\boldsymbol{T}_{C0}^{-1}\,{}^{0}\boldsymbol{T}_C(k). \quad (1) $$

A mathematical representation to evaluate the matrix multiplicative depth (mmd) is introduced. Reducing multiplicative depth optimizes matrix multiplication time and prevents overflow of the encryption scheme. A multiplicative depth of a square $n \times n$ matrix is given by the maximum among all element-wise multiplicative depth, $l^{max}(*)$:

$$ mmd(\mathsf{Enc}(\boldsymbol{T})) := \max_{1\le i,j\le n} l^{max}(\mathsf{Enc}(T_{ij})) \quad (2) $$

For example, multiplication between two encrypted matrices in ciphertext results in a multiplicative depth of one:

$$ mmd(\mathsf{Enc}({}^{0}\boldsymbol{T}_1) \circledast \mathsf{Enc}({}^{1}\boldsymbol{T}_2)) = 1 \quad (3) $$

Consider the encryption of (1) with SHE, $\mathsf{Enc}({}^{0}\boldsymbol{T}_{Ed}(k))$. Note that only the relative displacement from the VR interface's initial state to the current state is used for robot control. The initial states of the robot and VR interface are fixed and stored in the system as constants. For improved security, ideally, the constant matrices, ${}^{0}\boldsymbol{T}_{C0}$ and ${}^{0}\boldsymbol{T}_{R0}$, are encrypted at the beginning (on the robot side and on the user side, respectively) and stored as in ciphertext, not in plaintext:

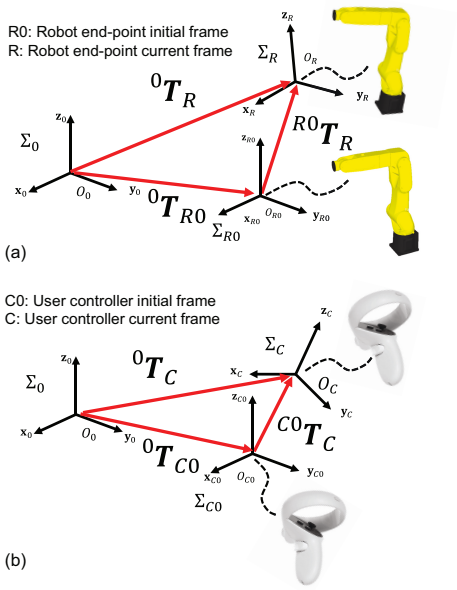


Fig. 2: Coordinate frames; (a) Initial and current robot position, (b) Initial and current input position.

$$ \mathsf{Enc}({}^{0}\boldsymbol{T}_R(k)) = \mathsf{Enc}({}^{0}\boldsymbol{T}_{R0}) \circledast \mathsf{Enc}({}^{0}\boldsymbol{T}_{C0}^{-1}) \circledast \mathsf{Enc}({}^{0}\boldsymbol{T}_C(k)) \quad (4) $$

As a result, $mmd(\mathsf{Enc}({}^{0}\boldsymbol{T}_R(k)) = 2$. If a user wishes to reduce the multiplicative depth by one, bootstrapping can be applied on the robot side to replace the first two terms with $\mathsf{Enc}(\mathsf{Dec}(\mathsf{Enc}({}^{0}\boldsymbol{T}_{R0}) \circledast \mathsf{Enc}({}^{0}\boldsymbol{T}_{C0}^{-1})))$ without risking revealing either ${}^{0}\boldsymbol{T}_{R0}$ or ${}^{0}\boldsymbol{T}_{C0}^{-1}$ as it is performed only once.

B. Encrypted matrix multiplication with threads

Consider the matrices

$$ \begin{aligned} \mathbf{A} &= [\alpha \times \gamma] \\ \mathbf{B} &= [\gamma \times \beta] \\ \mathbf{\Gamma} &= \mathbf{AB} = [\alpha \times \beta] \end{aligned} \quad (5) $$

To calculate $\mathbf{\Gamma}$, $\alpha\beta$ calculations of the form:

$$ \begin{aligned} &\mathsf{Enc}\left(\boldsymbol{AB}\right)_{ij} = \tilde{\mathbf{\Gamma}}_{ij} = \\ &\left(\tilde{A}_{i1} \otimes \tilde{B}_{j1}\right) \oplus \left(\tilde{A}_{i2} \otimes \tilde{B}_{j2}\right) \oplus ... \oplus \left(\tilde{A}_{i\gamma} \otimes \tilde{B}_{j\gamma}\right) \end{aligned} \quad (6) $$

must be performed, where $\tilde{x} = \mathsf{Enc}(x)$. Each calculation considers γ encrypted multiplications with average time μ, and $\gamma - 1$ encrypted additions with average time σ. Then the computation time for a single entry Γ_{ij} is given by $\xi_\gamma = \gamma\mu + (\gamma - 1)\sigma$. However, in general, homomorphic multiplicative operations take significantly longer than homomorphic additive operations, i.e., $\mu \gg \sigma$. As such we will make the following simplification $\xi_\gamma \approx \gamma\mu$.

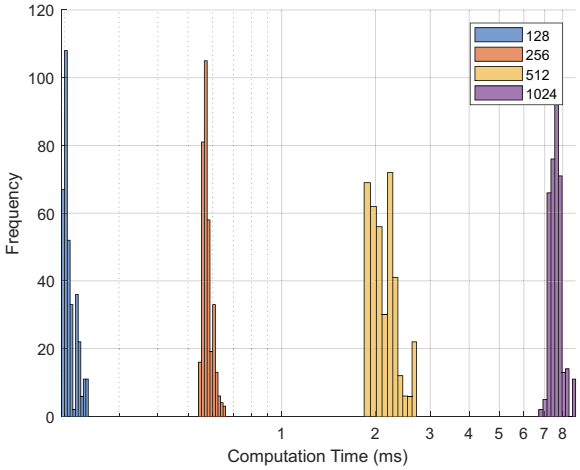


Fig. 3: Computation time histogram for Method 1 (series), showing distribution of computation times for for security parameter λ of varying bit lengths with semi log scale. Each set is represented with 10 bins across its range.

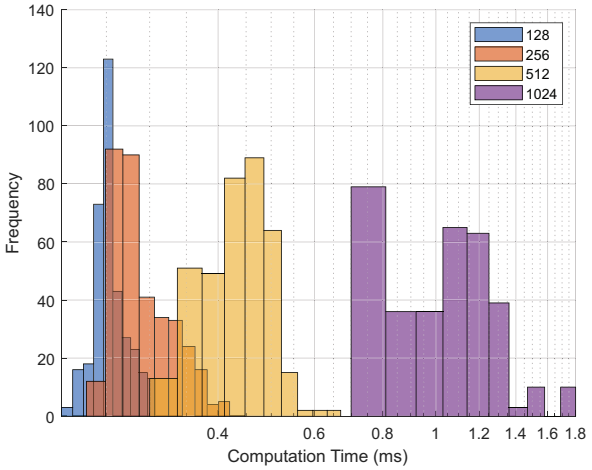


Fig. 4: Computation time histogram for Method 2 (parallel), showing distribution of computation times for security parameter λ of varying bit lengths with semi log scale. Each set is represented with 10 bins across its range.

Using ξ_γ we can construct the time complexity for a single-threaded and multi-threaded implementation of the matrix product. For a single threaded implementation, each Γ_{ij} must be sequentially processed, thus $\mathcal{O}(\alpha\beta\xi_\gamma)$. While it is not typical to consider the runtime of "elementary operations," in big O analysis, it is justified in this setting as homomorphic multiplication significantly impact performance [19].

Speedup can be achieved by delegating the computation of each $\mathbf{\Gamma}_{ij}$ to its own thread. In the limit that the number of system cores N, approaches the number of elements to compute i.e. $N \to \alpha\beta$, then the complexity reduces to $\mathcal{O}(\xi_\gamma)$.

The effectiveness of parallelism is tested by implementing two methods, each tested separately and analyzed for efficiency in timing. Figure 6 shows the following methods: Method 1 (series) executes on a single thread, carrying out standard matrix multiplication in which each dot product awaits a preceding operation to complete before performing its operation. Method 2 (parallel) separates each dot product of a matrix multiplication into its own thread, allowing each dot product to be calculated in parallel with the others.

The performance of each method was evaluated during system operation with a timer. A total of 500 samples were taken from each operation, and its distribution was plotted as a histogram for varying λ between 128 to 1024. The histogram in Fig. 4 shows that across an increasing range of λ, the average computation time only increased to 1.8 ms. In contrast the histogram of Method 1 in Fig. 3 shows the computation time without parallelism. Here we see clearly that the distribution increases exponentially with increasing λ.

III. Encrypted Teleoperation System

A. Implementation

Point-To-Point (PTP) direct control of a robot manipulator (FANUC LR Mate 200iD/7L) requires homogeneous matrix transformations. These transformations are implemented with SHE and threading at various levels to enhance performance.

A virtual reality (VR) headset (Meta Quest 2) is the user input device in this system. The VR joystick was attached to a controllable Universal Robots 6R Manipulator in order to repeat the experimental path consistently as shown in Fig. 5. The position of the controller was acquired on the local computer. OpenVR API (Valve Corporation) was used in this implementation. The current pose of the user interface ${}^{0}\boldsymbol{T}_{C}(k)$ is acquired, and the homogeneous transformation matrix ${}^{C0}\boldsymbol{T}_{C}$ from the initial pose of the controller to the current pose is processed, and then encrypted to be processed in the operator module. This portion of the system is considered local, with full access to all keys as in Fig. 7.

The operator module was implemented with C#, The first part is the cipher, which implements the Dyer's SHE algorithm. The second part of the operator module implemented matrix multiplication in cipherspace. This part is representative of a "cloud" controller shown in Fig. 7 , and will not access any information needed for encryption and decryption. To that end, the two matrices ${}^{0}\boldsymbol{T}_{R0}$ and ${}^{C0}\boldsymbol{T}_{C}(k)$ are encrypted when it is received by the thread on which operation are being carried out. A client on the remote side will receive the encrypted message from the cloud. The final command is

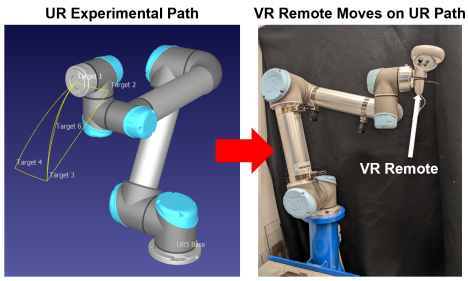


Fig. 5: VR experimental path for UR-VR Mount.

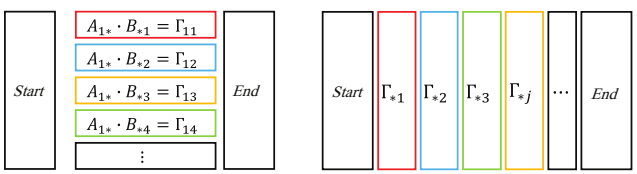
decrypted and sent to the robot controller (RoboDK). Once RoboDK receives this command, a trajectory is sent to the FANUC Manipulator, resulting in a translation and rotation to the desired robot pose as shown in Fig. 7. It is noted that this implementation uses threads in favor of processes for homomorphic operations. An inter-process model would be a closer representation of an ideal teleoperation system, as there is extra communication overhead. While there is no inter-process communication, threads are being executed in an asynchronous manner as would be expected of a multi-process system. For the purposes of this study, a multi-threaded architecture can sufficiently represent a successful physical system that only operates on encrypted information.

B. Threading

As established in Section II-B, encrypted matrix operations can be sped up by distributing computational overhead across multiple processors. Multi-threading has been chosen to implement the parallelism proposed in Method 2 in Fig. 6. Performance of a multi-threaded program is dependent on the size and scheduling of the tasks put on each thread. The size of the task should not be smaller than the overhead to start threads. If this occurs, implementing parallelism could lead to performance degradation. Computation time of homomorphic operations has been evaluated in detail in [17], including that of Dyer's SHE algorithm. Based on these findings, it is expected that for low security messages with lighter computational load, gains from parallelism will be comparable to the threading scheduling overhead. However, with increase in message length, parallel computing gains as discussed in will dominate as discussed in Section II-B.

C. Simulation

To motivate our choice of security parameters we ran simulation of single-threaded vs multi-threaded execution. To simplify this choice we define all security parameters to be in terms of one parameter λ by $\rho = 10\log_2(\lambda)$ and $\rho' = 2\rho$. This parameterization ensures that the cypher has sufficient entropy to prevent a cyphertext attack [18]. Furthermore, by parameterizing ρ and ρ' by λ, we have collapsed the parameter space to a single dimension, thus making a sweep of parameter space far less computationally burdensome. Ultimately, the security of the cyphertext against brute force attacks will be determined by the bitlength of the encrypted values. With the above parameterization we can see that bitlength is related to λ as shown in Fig. 8. The computation time to complete one matrix multiplication with



(a) Method 1: Series Matrix Operations (b) Method 2: Parallel Matrix Operations

Fig. 6: Threading methods

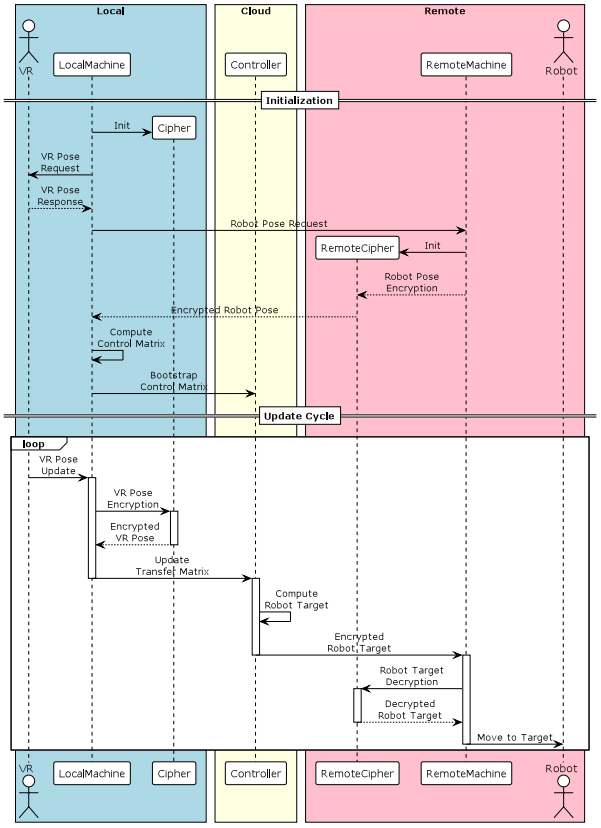


Fig. 7: Sequential order of implementation

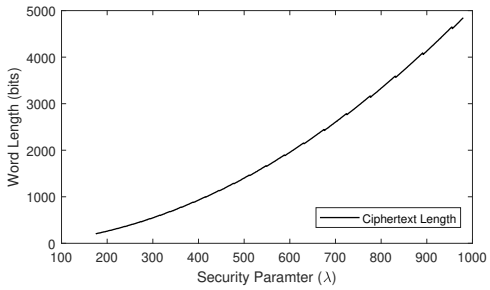


Fig. 8: Ciphertext bitlength with respect to lambda

and without threading was simulated on a 11th Gen Intel(R) Core(TM) i7-1165G7 CPU. Computation was averaged over 10 runs for each choice of λ. The results of this simulation can be seen in Fig. 9.

Notice in Fig. 9 that for low security parameters the series method performs better than the multi-threaded method. This is due to the overhead of creating and managing multiple threads, and indicates the threaded tasks require so little computational effort that the threading overhead is actually detrimental. Typically, we want our security parameters to be as large as possible, so such a situation is unlikely to occur in a production system.

IV. Experimental Results & Analysis

In order to verify encrypted operation, experimental data was collected with security parameter λ chosen from the

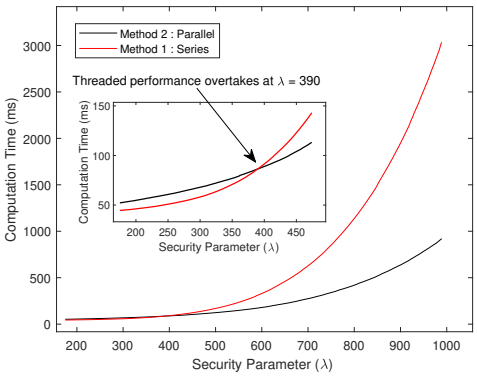


Fig. 9: Computation time for given lambda

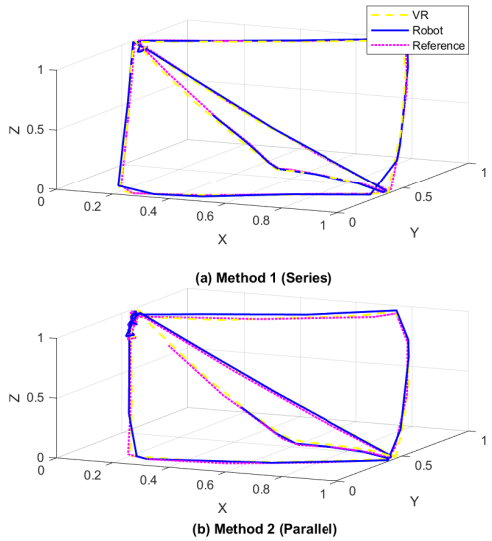


Fig. 10: Position comparison between VR, reference , and robot trajectories for λ = 1024.

set $\{128, 256, 512, 1024\}$. Figure 10 presents translation data of the end effector for a test case with λ = 1024. These results show that the robot manipulator closely tracked the VR remote while command calculations were performed in cipherspace. For example, at the lower left corner, the robot tracked better with Method 2 (parallel) since the reference command was generated faster than Method 1 (series). This observation is expected to hold for complex path tracking.

Table I shows the median deviation of the robot end effector from the ideal path in millimeters at varying security parameters. The deviation showed non-normal distribution for all trials tested with the Shapiro-Wilks normality test with 95% confidence. One tailed Mann Whitney U test was performed at the significance level of 0.0125 to determine if deviation of the system was significantly smaller when method 2 is used over method 1. Results showed that there was a significant difference in deviation between methods for cases $\lambda = 512$ and 1024.

All λ values show similar results, confirming the viability of encrypted robot control through the VR system. Additionally

TABLE I: Median Robot - VR path deviation in mm at varying security parameter λ for Method 1 and Method 2, and Mann Whitney U test score U by λ ($^*U < 0.0125$)

λ	Method 1: Series	Method 2: Parallel	U
128	16.41	16.16	.342
256	21.64	23.26	.953
512	31.60	18.48	$< .001^*$
1024	42.27	18.87	$< .001^*$

tested was computation time again with varying λ chosen from the same set $\{128, 256, 512, 1024\}$. Results of this test can be seen in Table II. Observe that Method 2 gains benefit as the λ parameter increases, approximately seven times faster than that of the series implementation in the $\lambda = 1024$ case.

One tailed Mann Whitney U test was performed to determine if the positive shift in the overall distribution was statistically significant. It was found that reduction in computational time is significant from $\lambda = 256$ and above. Although there is significant time increase for $\lambda = 256$ case, amount of reduction is suspected to be too small to result in performance increase. For $\lambda = 512$ and 1024 case, time reduction and performance increase were consistent.

TABLE II: Median and standard deviation, computation time of reference command generation (ms), and Mann Whitney U test score U by λ ($^*U < 0.0125$) .

λ	Method 1: Series		Method 2: Parallel		U
	Mdn	σ	Mdn	σ	
128	0.204	0.001	0.252	0.023	1
256	0.574	0.026	0.285	0.042	$< .001^*$
512	2.098	0.193	0.441	0.056	$< .001^*$
1024	7.661	0.383	1.068	0.236	$< .001^*$

V. Conclusions

This paper presented a physical system implementing a somewhat homomorphic encryption algorithm to secure communication between distinct systems in the form of a VR hand-held remote controller and a 6DOF manipulator. Messages between the local and remote systems were in the form of a homogeneous transformation matrix. A method to accelerate computationally heavy, encrypted calculations exploiting the parallel nature of matrices was devised and evaluated for algorithmic complexity. The new method was shown to significantly reduce update times once security parameters were large enough to offset communication overhead inherent in parallel computing. Among the tested security parameters the median computation time was 7.22 times faster compared to when parallel computation was not applied. This improvement in computational time was shown to meaningfully decrease positional deviation of the robot end effector from the ideal path up to 2.25 times. Usage of a larger security parameter will result in even larger performance gains. Further research would implement filtering in cipherspace in the form

of an extended Kalman filter and model predictive controllers leveraging improved performance of matrix operations.

APPENDIX

A. Cryptographic Operations

This study adopts the SHE algorithm proposed in [18] that can be summarized as follows:

Gen: Set security parameters λ, ρ, ρ'. Let:

$$\nu = \rho' - \rho \tag{7}$$

$$\eta = \frac{\lambda^2}{\rho'} - \lambda \tag{8}$$

Randomly choose a λ-bit prime p, a ν-bit prime κ, and an η-bit prime q. Generate a key $k = (\kappa, p)$ and publish $N = pq$. In range of plaintext integer numbers: $\mathcal{M} = \{0, 1, 2, ..., M-1\}$, to compute any polynomial expression: $P(m_1, m_2, ..., m_n)$ and $P(m_1 + s_1\kappa, m_2 + s_2\kappa, ..., m_n + s_n\kappa)$ up to the degree of d, key lengths κ and p are lower-bounded by the power of d given by:

$$\kappa > (n+1)^d M^d \tag{9}$$

$$p > (n+1)^d (M + \kappa^2)^d \tag{10}$$

where $s_i \in \{0, 1, ..., \kappa - 1\} (i = 1, ..., n)$ are random integers.

Enc: Plaintext $m \in \mathcal{M}$ is encrypted by:

$$c = m + s\kappa + rp \mod N \tag{11}$$

where $s \in \{0, 1, ..., \kappa - 1\}$ and $r \in \{0, 1, ..., q - 1\}$ are random noise.

Dec: Ciphertext $c \in \mathcal{C}$ is decrypted by:

$$m = (c \mod p) \mod \kappa \tag{12}$$

Add: Additive homomorphism $\mathsf{Enc}(m) \oplus \mathsf{Enc}(m') \mod N = \mathsf{Enc}(m + m')$, $\forall m, m' \in \mathcal{M}$ is realized if:

$$m + m' < \kappa \tag{13}$$

$$(m + m') + (s + s')\kappa < p \tag{14}$$

where s' is random noise corresponding to m'.

Mult: Multiplicative homomorphism $\mathsf{Enc}(m) \otimes \mathsf{Enc}(m') \mod N = \mathsf{Enc}(mm')$, $\forall m, m' \in \mathcal{M}$ is realized if:

$$mm' < \kappa \tag{15}$$

$$mm' + (ms' + m's + ss'\kappa)\kappa < p \tag{16}$$

Equations (9), (10), (13), (14), (15), and (16) are conditions that must be satisfied at all times.

B. Numerical Limitations

The primary limiting factor of Dyer's SHE is the divergence of noise introduced into the ciphertext, primarily by multiplication. The largest possible value among all encrypted signals, parameters, and products should be smaller than:

$$M(n, d, \lambda, \nu) := \left\lfloor \min \left\{ \frac{\sqrt[d]{\kappa}}{n+1}, \frac{\sqrt[d]{p}}{n+1} - \kappa^2 \right\} \right\rfloor \tag{17}$$

where d is the degrees of polynomial, p is the λ-bit prime, κ is the ν-bit prime–the scheme's security parameters [15], [18].

REFERENCES

[1] F. Farokhi, *Privacy in Dynamical Systems*. Springer, 2020.

[2] M. Schulze Darup, A. B. Alexandru, D. E. Quevedo, and G. J. Pappas, "Encrypted control for networked systems: An illustrative introduction and current challenges," *IEEE Control Systems Magazine*, vol. 41, no. 3, pp. 58–78, 2021.

[3] J. E. Sullivan and D. Kamensky, "How cyber-attacks in ukraine show the vulnerability of the us power grid," *The Electricity Journal*, vol. 30, no. 3, pp. 30–35, 2017.

[4] A. Hobbs, "The colonial pipeline hack: Exposing vulnerabilities in us cybersecurity," in *SAGE Business Cases*. SAGE Publications: SAGE Business Cases Originals, 2021.

[5] H. Alemzadeh, D. Chen, X. Li, T. Kesavadas, Z. T. Kalbarczyk, and R. K. Iyer, "Targeted attacks on teleoperated surgical robots: Dynamic model-based detection and mitigation," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2016, pp. 395–406.

[6] S. Gorman, Y. J. Dreazen, and A. Cole, "Insurgents hack us drones," https://www.wsj.com/articles/SB126102247889095011, 2009.

[7] K. Kogiso and T. Fujita, "Cyber-security enhancement of networked control systems using homomorphic encryption," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, Conference Proceedings, pp. 6836–6843.

[8] J. Kim, C. Lee, H. Shim, J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Encrypting controller using fully homomorphic encryption for security of cyber-physical systems," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 175–180, 2016.

[9] Y. Lin, F. Farokhi, I. Shames, and D. Nešić, "Secure control of nonlinear systems using semi-homomorphic encryption," in *IEEE Conference on Decision and Control*, Miami Beach, FL, 2018, pp. 5002–5007.

[10] A. B. Alexandru and G. J. Pappas, "Encrypted LQG using labeled homomorphic encryption," in *ACM/IEEE International Conference on Cyber-Physical Systems*, Montreal, 2019, pp. 129–140.

[11] K. Teranishi, M. Kusaka, N. Shimada, J. Ueda, and K. Kogiso, "Secure observer-based motion control based on controller encryption," in *2019 American Control Conference (ACC)*. IEEE, 2019, Conference Proceedings, pp. 2978–2983.

[12] Y. Qiu and J. Ueda, "Encrypted motion control of a teleoperation system with security-enhanced controller by deception," in *Dynamic Systems and Control Conference*, vol. 59148. American Society of Mechanical Engineers, 2019, Conference Proceedings, p. V001T07A006.

[13] Y. Z. Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, and M. D. Di Benedetto, "State of the art of cyber-physical systems security: An automatic control perspective," *Journal of Systems and Software*, vol. 149, pp. 174–216, 2019.

[14] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–35, 2018.

[15] K. Teranishi, K. Kogiso, and J. Ueda, "Encrypted feedback linearization and motion control for manipulator with somewhat homomorphic encryption," in *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2020, Conference Proceedings, pp. 613–618.

[16] S. Kosieradzki, Y. Qiu, K. Kogiso, and J. Ueda, "Rewrite rules for automated depth reduction of encrypted control expressions with somewhat homomorphic encryption," in *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2022, pp. 804–809.

[17] S. Kosieradzki, X. Zhao, H. Kawase, Y. Qiu, K. Kogiso, and J. Ueda, "Secure teleoperation control using somewhat homomorphic encryption," in *Modeling, Estimation and Control Conference 2022*, 2022, Conference Proceedings, pp. 6836–6843.

[18] J. Dyer, M. Dyer, and J. Xu, "Practical homomorphic encryption over the integers for secure computation in the cloud," *International Journal of Information Security*, vol. 18, pp. 549–579, 2019.

[19] X. Cao, C. Moore, M. O'Neill, E. O'Sullivan, and N. Hanley, "Optimised multiplication architectures for accelerating fully homomorphic encryption," *IEEE Transactions on Computers*, vol. 65, no. 9, pp. 2794–2806, 2016.