Distributed Simulation of Encrypted Dynamics via Functional Mockup Units

Xiaofeng Zhao, Shane Kosieradzki, Jun Ueda
George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology
Atlanta, GA, 30332, USA
Email: {zhaoxiaofeng, skosieradzki3, jueda3}@gatech.edu

***Abstract*—The research presented in this paper aims to establish functional mockup units (FMU) co-simulation methods to simulate and evaluate encrypted dynamic systems using somewhat homomorphic encryption (SHE). The proposed approach encrypts the entire dynamic system expressions, including: model parameters, state variables, feedback gains, and sensor signals, and perform computation in the ciphertext space to simulate dynamic behaviors or generate motion commands to servo systems. The developed FMU co-simulation helps analyze the relationship between security parameters and performance. Two illustrative examples are presented and analyzed: 1) encrypted Duffing oscillator and 2) encrypted teleoperation. How the time delay due to FMU co-simulation affects the refresh rate is also reported.**

***Index Terms*—Encrypted control, Cybersecurity, Somewhat homomorphic encryption, Functional mockup units**

I. Introduction

Encrypted control is an emerging topic in control engineering to realize secured dynamic controllers in a networked control system by applying homomorphic encryption methods [1] [2]. This approach aims to ensure that sensitive information, such as controller gains and signals is always encrypted in the cloud. As a proactive measure for unauthorized login and falsification, the secret key is possessed only in the local end, not in the cloud controller; encrypted signals and feedback gains are used to compute motion commands in the ciphertext directly. Due to practical limitations in existing homomorphic encryption algorithms, which usually perform either additions or multiplications, termed "partially homomorphic encryption" or PHE, the majority of the existing studies used linear controllers [3]–[6]. In some cases, nonlinear plant dynamics must be evaluated in real-time for model-based compensation, which increases the complexity of the control scheme. Practically usable fully homomorphic encryption (FHE) has not yet been developed. Due to this reason, expansion of homomorphic control encryption methodologies to nonlinear and/or time-varying control has not been well studied.

A "somewhat homomorphic encryption" (SHE) algorithm proposed by Dyer et al. [7] has shown promise of online encryption upon which this study will develop new realization procedures. SHE is a family of algorithms that can perform both additive and multiplicative homomorphic encryption with a limited number of operations. Teranishi et al. showed it possible to use this SHE algorithm for real-time control [8]. Note that inappropriate selections of security parameters and signal quantization levels in SHE will cause overflow and system instability. Due to the known limitations of SHE, there is a need for a simulation environment that can investigate the relationship between security parameters and performance.

This paper presents a Functional Mock-up Unit (FMU) [9] co-simulation environment to simulate and evaluate encrypted dynamic systems with SHE. FMUs have been chosen as the simulation technology. Unlike commercial solutions (e.g. Ansys, Simulink, etc.) the specification is open-source and designed to be as widely compatible as possible. As such, interfacing with external codes is much simpler than commercial solutions. Furthermore, vendors are increasingly choosing to allow their models to be exported as an FMU, which significantly simplifies the construction of FMUs. With this technology, users can "mix-and-match" FMUs to construct simulations of arbitrary systems. The addition of external cryptographic methods allows us to test the feasibility of encrypted control of said systems, see Fig. 1. As case studies, an encrypted duffing oscillator and teleoperated robotics system are analyzed in terms of quantization and overflow.

II. Simulation Environment

A. Functional Mock-up Interface

Functional Mock-up Interface (FMI) is a tool independent standard for the Model Exchange (ME) and for Co-Simulation (CS) between different tools in a standardized format [10]. In the FMI nomenclature, a Functional Mock-up Unit (FMU) model implements one or two of FMIs. FMU is a promising candidate to become the industry standard and cross-company collaboration as it allows co-simulation of various FMUs components generated from different tools to develop complex cyber-physical systems [11]. Compared to Simulink (MATLAB, Mathworks) that is designed to model dynamic systems, FMU has the advantage of supporting more data types and language features. FMI also addresses the disadvantage of the Simulink S-function as it is easier to integrate with other simulators and takes less memory overhead [9].

FMU is essentially an archive (i.e. a .zip file), containing a `modelDescription.xml` file in the root, and either

binary or source files. In model exchange, FMU does not come with its numerical solver. FMU only provides functions to set the states and inputs and compute the state derivatives. It requires the solver in the host environment/import tool to query the derivatives and update the states of the FMU. In co-simulation, a specialized numerical solver is embedded in the FMU. The host environment only sets the inputs and time steps, and reads the outputs [12].

Since encrypted control usually requires a complex model, the authors adopted the co-simulation option. FMU can be run with `FMPy`, a free Python library, to execute FMUs that support Co-Simulation and Model Exchange and run on Windows and Linux [13]. Interested readers can visit the authors' GitHub to find relevant implementations [14].

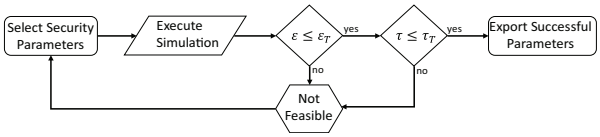
B. FMU/Cypher Interface

1) Cypher Independent Interface: A system has been created by which critical parts of the FMU's calculations can be encrypted via external codes. To achieve this, we defined an interface to access cryptographic methods. FMUs can be created from vendor tools such as Simulink. Co-simulation standalone FMUs can be exported from Simulink after setting the system target file and fixed-step. By using the `S-Function` block, we can insert calls to our interface methods via function pointers and system functions (`LoadLibrary` on Windows, `dlopen` on Linux).

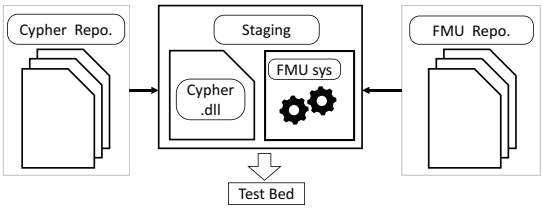
By separating the cryptographic routines into a standalone `.dll`, the implementation of the interface is isolated from the FMU's internal operation. This architecture enables the user to exchange a particular `cypher.dll` for a `cypher.dll` that implements the same interface for testing different cryptographic systems. The interface serves as a convenient test bed for evaluating the feasibility of different encrypted cyber-physical systems.

2) FMU plug-and-play setup: The modular structure of FMU may be used in conjunction with the aforementioned cypher-interface to create an encrypted cyber-physical test-bed. A particular FMU system is paired with different implementations of `cypher.dll` to test which cypher/security-parameters are best compatible with the particular system. An example of this workflow can be seen in Fig. 1.

3) Data type constraints: FMI was designed to be widely compatible with as many hardware architectures and operating systems as possible; as such, data is constrained to be "standard C types" [15]. This poses challenges when trying to pass encrypted values to FMUs, as cipher-text values usually cannot be represented by standard types alone. To work around these constraints, in the following case studies, the authors have elected to move all cryptographic methods into `controller.fmu`. It is usually not ideal to allow encryption and decryption to occur in a single controller. This particular configuration has been chosen to be acceptable as the same computations are being performed, just by different owning processes. As such, timing results are unlikely to be noticeably affected.



(a) Testbed Flow: Evaluation scheme to find successful security parameters, where ε is the measured error, ε_T the error tolerance, τ the measured simulation-cycle time, and τ_T the simulation-cycle time tolerance.



(b) Testbed staging: Construct system from FMU composition, and select the cypher to be tested. Here the "Test Bed" block refers to Fig. 1a.

Fig. 1: FMU/Cypher test-bed: System simulation is constructed by linking FMUs chosen from a remote repository. A cypher is then selected to be tested for compatibility with the given FMU system.

III. Case Studies

A. Limiting Equation

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

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

where d is the degrees of polynomial, p is the λ-bit prime, κ is the ν-bit prime–the scheme's security parameters [8], [16]. Note the left-handed ($\mathcal{L}$) and right-handed ($\mathcal{R}$) arguments of (1) are given by: $\mathcal{L} = \frac{\sqrt[d]{\kappa}}{n+1}$, $\mathcal{R} = \frac{\sqrt[d]{p}}{n+1} - \kappa^2$.

B. Objectives

By running the simulations in FMUs, this study simulates the systems presented in the case studies similar to a real-world scenario. The simulations will validate the relationship of the security parameters in (1) for different systems. Also, the simulation will investigate the quantization and overflow patterns as well as the choices of security parameters when $\mathcal{L}$ or $\mathcal{R}$ of (1) dominates.

C. Duffing Oscillator

This paper will apply the SHE approach to the Duffing oscillator that includes a third degree of polynomials term. The Duffing equation is regarded as one of the prototypes for systems of nonlinear dynamics. In mechanics, it represents a class of single-degree-of-freedom systems with nonlinear stiffness.

Let the coefficients $\delta, \alpha, \beta, \gamma, \omega$ denote system damping, linear stiffness, amount of non-linearity in the restoring force,

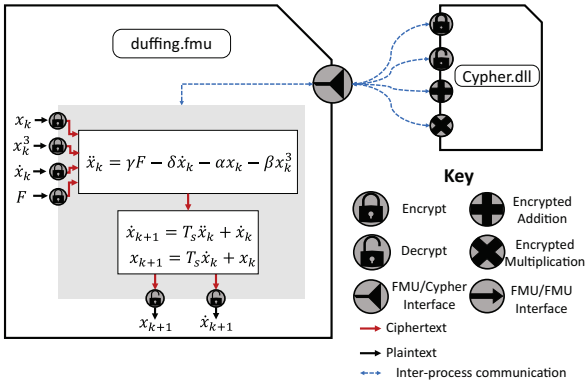


Fig. 2: All parameters in the duffing equations were encrypted and ran in FMU, where $F = \cos(\omega t)$ is the forcing function, and $x_k = x(kT_s)$.

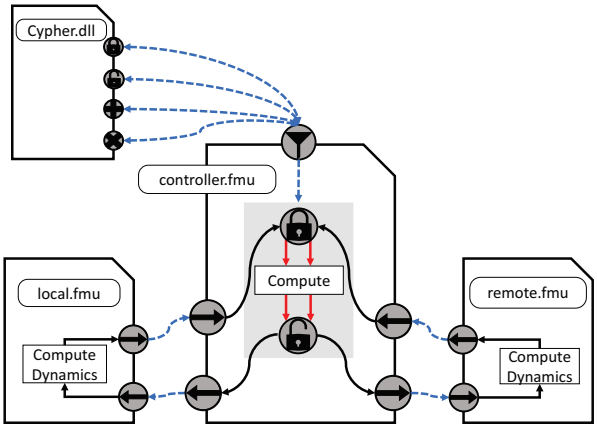


Fig. 3: The teleoperation system consists of three separate FMU: local plant, controller, and remote plant. The controller makes calls to `cypher.dll` to perform cryptographic operations.

amplitude of the driving force, and angular frequency of the force:

$$\ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 = \gamma\cos(\omega t). \tag{2}$$

Equation (2) is discretized with a step time T_s for the current time $t = kT_s (k = 0, 1, 2, \cdots)$ and encrypted by using SHE, as shown in Fig. 2. The main purpose of the encryption is to treat the parameters, $\alpha, \beta, \gamma, \delta$, as well as $\ddot{x}$, in ciphertext. On the other hand, we assume that the state variables, x and $\dot{x}$, are of interest of the user and observed in plaintext.

$$\begin{aligned}\mathsf{Enc}(\ddot{x}_k) = {} & \mathsf{Enc}(\gamma) \otimes \mathsf{Enc}(\cos(\omega t)) \\ & \oplus \mathsf{Enc}(-\delta) \otimes \mathsf{Enc}(\dot{x}_k) \\ & \oplus \mathsf{Enc}(-\alpha) \otimes \mathsf{Enc}(x_k) \\ & \oplus \mathsf{Enc}(-\beta) \otimes \mathsf{Enc}(x_k^3)\end{aligned} \tag{3}$$

$$\mathsf{Enc}(\dot{x}_{k+1}) = \mathsf{Enc}(T_s) \otimes \mathsf{Enc}(\ddot{x}_k) \oplus \mathsf{Enc}(\dot{x}_k) \tag{4}$$

$$\mathsf{Enc}(x_{k+1}) = \mathsf{Enc}(T_s) \otimes \mathsf{Enc}(\dot{x}_k) \oplus \mathsf{Enc}(x_k) \tag{5}$$

The simulation model will then be exported into FMU and run in FMPy by setting the time step and initial conditions. Note that if the hardware resources permit in terms of multiplicative depth [17], $\mathsf{Enc}(x_k^3)$ may be replaced with $\mathsf{Enc}(x_k) \otimes \mathsf{Enc}(x_k) \otimes \mathsf{Enc}(x_k)$.

D. Teleoperation System

The SHE approach is also applied to an encrypted teleoperation system [18] where two control loops of the local and remote plants are intertwined. Let the coefficients m, b, μ, τ, and f denote system mass, damping, friction coefficient, actuator force, and external force; furthermore, let the subscript m and s denote the local and remote system. The plant dynamics is modeled as:

$$m_m\ddot{x}_m + b_m\dot{x}_m + \mu_m\text{sign}(\dot{x}_m) = \tau_m + f_m \tag{6}$$

$$m_s\ddot{x}_s + b_s\dot{x}_s + \mu_s\text{sign}(\dot{x}_s) = \tau_s - f_s \tag{7}$$

Evaluation of both linear and nonlinear terms must be performed in an increased number of encoding blocks. Figure 3 shows a possible configuration of an encrypted teleoperation system. The main concept is to encrypt shared information using a private encryption key known *only* to the plants. Both the local and remote plants are responsible for system output measurements and encryption by using the keys. The networked controller stores encrypted system parameters, as well as encrypted output measurements received from both plants.

While there are a variety of control schemes to realize bilateral teleportation, a representative symmetric control scheme utilizing PD feedback with inertial and friction compensation is considered in this case study:

$$\begin{aligned}\tau_m = {} & (m_m - m_{ms})\ddot{x}_m + k_p(x_s - x_m) + \\ & k_d(\dot{x}_s - \dot{x}_m) + 0.9\mu_m\text{sign}(\dot{x}_m)\end{aligned} \tag{8}$$

$$\begin{aligned}\tau_s = {} & (m_s - m_{ms})\ddot{x}_s + k_p(x_m - x_s) + \\ & k_d(\dot{x}_m - \dot{x}_s) + 0.9\mu_s\text{sign}(\dot{x}_s)\end{aligned} \tag{9}$$

Inside the controller, the SHE algorithm encrypts the dynamics outputs from the local and remote plants including: accelerations, velocities, displacements, as well as gains. Then, the controller will do the computation in encryption and output the decrypted forces back to the plants.

The teleoperation system consists of three separate FMUs: local plant, controller, and remote plant. Each component is exported from the Simulink model and run in `FMPy`. During the FMU simulation, the components in the co-simulation establish communications with each other. A component publishes a specific output variable that is subscribed by other components as input. Two cycles exist in the simulations: feedback from the controller to the local plant and from the remote plant to the controller. Therefore, both plants and controllers have the same priority but cannot run in parallel.

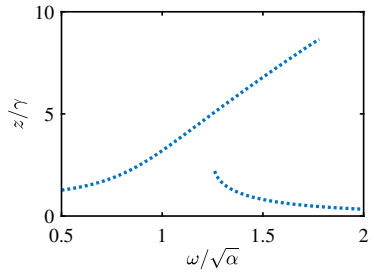


Fig. 4: Duffing's parameter $(\alpha = \delta = 1, \gamma = 0.1, \beta = 0.04)$. Dyer's encrypted signal $(\lambda = 256, \rho = 1, \nu = 35, \Delta = 0.005)$. Frequency response overhangs to the high-frequency side in a hardening spring oscillator.

IV. Results and Discussion

A. Duffing Oscillator

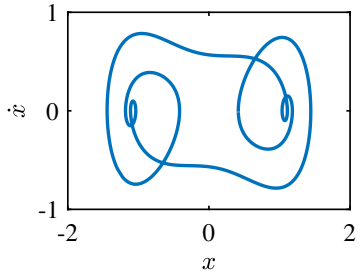
The nonlinear phenomenon is evident in hysteresis, which is induced by x^3 in the Duffing equation. When α and β have the same signs, the stiffness characteristic is hardening. For a hardening spring oscillator, the frequency response overhangs to the high-frequency side, as shown in Fig. 4.

For encrypted signals, the time step (T_s) could not be smaller than 10 ms since quantization with a higher resolution is needed for a smaller T_s. Therefore, there was a balance between Δ and T_s to capture accurate data while not overflowing. When T_s is too large, the system fails to capture accurate data points. When T_s is too small, Δ needs to be smaller, resulting in numerical overflow. As shown in Fig. 5, encryption impacted the performance compared to the plaintext equation due to the resolution of the Duffing equation. The percentage of error by comparing the L2-norm of the variables was 6.93%. $\nu = 32$ where $\lambda = 192$ was the minimum security parameter to prevent numerical overflow, as shown in Fig. 5c, which had a maximum velocity threshold of 0.45.

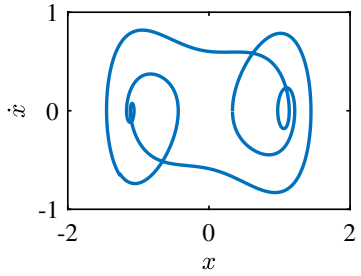
The success of the simulation result is determined by the largest encrypted number M. To investigate the $\mathcal{R}$ argument of (1), the system was parameterised by the security parameter λ and ν. Pass/fail analysis was performed by comparing the L2-norm of the encrypted data. As shown in Fig. 6, the $\mathcal{L}$ argument dominates and $\mathcal{R} = M$. As for a constant λ, increasing ν results in numerical overflow and failure. d, the degree of polynomial, equals to 3 because of the cubic power term results from quantization. At the boundary of pass/fail, there is a positive linear relationship between the two parameters, and λ is about as six times as large as ν.

B. Teleoperation System

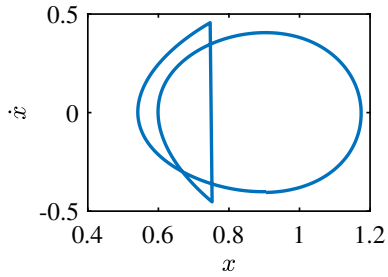
Fig. 7 shows simulated displacements of the teleoperation system. Compared to the plaintext simulation shown in Fig. 7a, oscillations in displacements were observed in the encrypted simulation due to the encoder as shown in Fig. 7b. $\nu = 31$ and $\lambda = 125$ are the smallest security parameters to prevent numerical overflow as shown in Fig. 7c.



(a) Plaintext trajectory. Success phase plot (T_s =10 ms).



(b) SHE trajectory with security parameters: $(\lambda = 256, \rho = 1, \nu = 35, \Delta = 0.005)$. Success phase plot ($T_s = 10$ ms).



(c) SHE trajectory with security parameters: $(\lambda = 256, \rho = 1, \nu = 35, \Delta = 0.004)$. Failure phase plot ($T_s = 10$ ms).

Fig. 5: FMU trajectory of duffing oscillator.

Compared to the plaintext Simulink simulation that could run at a maximum time step of 20 ms for this particular system, running in FMU required a time step of at most 5 ms (Fig. 7b). This is primarily caused by the delay in the co-simulation architecture. The local solver takes one extra time step to transfer the output of one FMU to the input of the other FMU. For example, FMU A transfers data to FMU B. FMU B takes the output from FMU A in the last time step as its input, which represents a unit of time delay between them. There is a communication delay equivalent to four time steps between the local plant's input and output for the signals through the remote plant. The time delays include: one from the local plan to the controller, from the controller to the remote plant, from the remote plant back to the controller, and from the controller back to the local plant. Nevertheless, Dyer's SHE scheme has much faster refresh rate than 200

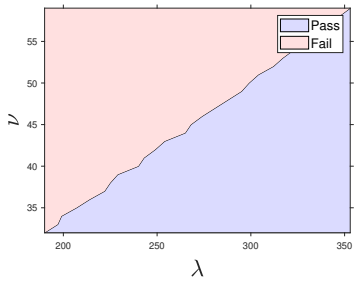


Fig. 6: Pass/fail results of Duffing system simulations using Dyer's SHE. The system was parameterised by the security parameter λ and the encoder resolution ν. Remaining security parameters well held constant at $\rho = 1$, $\Delta = 0.005$. The system starts working from $\nu = 32$ and $\lambda = 192$.

Hz to realize real-time control system.

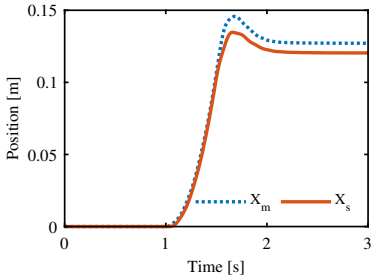
Fig. 8 depicts that the $\mathcal{L}$ argument dominates and $\mathcal{R} = M$. Compared to the duffing system, d decreases from 3 to 2 in the teleoperation system. The minimum start value of $\nu = 32$ is decreased by one, and the minimum value of $\lambda = 125$ is decreased by about 33%. So, the decrease of ν is due to the increase of Δ and the decrease of λ is mainly due to the decrease of d. There is a linear relationship between λ and ν. λ is about four times as large as ν, which is the product of the number of multiplication and d.

To investigate the $\mathcal{L}$ argument of (1), the system was parameterised by the security parameter λ and the encoder resolution Δ, as shown in Fig. 9. When Δ becomes smaller, the size of the encoded number and M increases. As shown in Fig. 9, increases ν can increase M while ν is smaller than 36. Increasing ν can increase the $\mathcal{L}$ argument of (1) while significantly decreasing $\mathcal{R}$. Therefore, for $\lambda = 256$, the $\mathcal{R}$ argument dominates and $\mathcal{L} = M$ while ν is smaller than 36. Otherwise, the $\mathcal{L}$ argument dominates and $\mathcal{R} = M$.

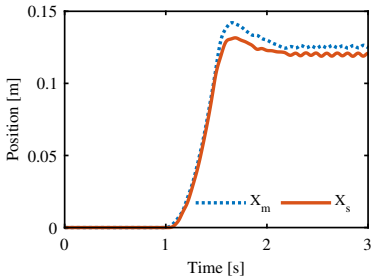
To sum up, the FMU simulations find the quantization and overflow pattern of encrypted control using Dyer's SHE scheme: if $\nu \geq \frac{\lambda}{2d}$, the system fails; if $\frac{\lambda}{3d} \leq \nu \leq \frac{\lambda}{2d}$, the $\mathcal{L}$ argument dominates and $\mathcal{R} = M$; if $\nu \leq \frac{\lambda}{3d}$, then the $\mathcal{R}$ argument dominates and $\mathcal{L} = M$. These inequality relationships are observed patterns found from the simulations and may not hold in general.

V. Conclusion

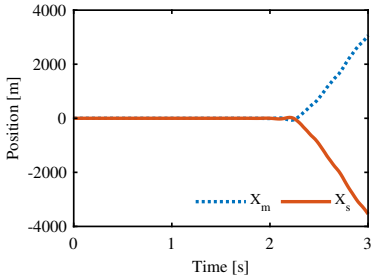
This paper presented a FMU co-simulation environment for various components in an encrypted dynamic system using SHE. The architecture consists of external codes that implement the encrypted calculations and FMU's dynamic systems. The feasibility of performing the co-simulation in FMU was demonstrated in two case studies. The FMU co-simulation presented the success/fail scenarios for both systems and showed the choices of security parameters when $\mathcal{L}$ or $\mathcal{R}$ of (1) dominates. This paper discussed how the relationship among the security parameters and time delay in co-simulation impacts the simulation performance. The



(a) Plaintext control ($T_s = 20$ ms)



(b) SHE control ($\lambda = 128, \rho = 1, \rho' = 32, \Delta = 0.005, T_s = 4$ ms). Oscillations are shown due to encoder.



(c) SHE control ($\lambda = 128, \rho = 1, \rho' = 32, \Delta = 0.005, T_s = 20$ ms). The step size is not small enough to permit correct computations.

Fig. 7: FMU teleoperation results.

developed interface may serve as a convenient test bed for evaluating the numerical stability of different cryptographic systems.

This study was supported in part by National Science Foundation Grant Nos. CMMI 2112793 and ERC 2124319.

This study adopts the SHE algorithm proposed in [16] that can be summarized as follows:

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

$$\nu = \rho' - \rho \tag{10}$$

$$\eta = \frac{\lambda^2}{\rho'} - \lambda \tag{11}$$

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 $N = pq$. Within a range of plaintext integer numbers: $\mathcal{M} = \{0, 1, 2, ..., M - 1\}$, to compute

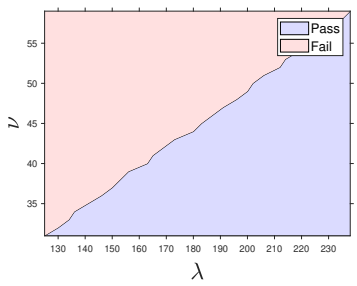


Fig. 8: Pass/fail results of teleoperated control simulations using Dyer's SHE. The system was parameterised by the security parameter λ and the encoder resolution ν. Remaining security parameters well held constant at $\rho = 1$, $\delta = 0.01$. The system starts working from $\nu = 31$ and $\lambda = 125$.

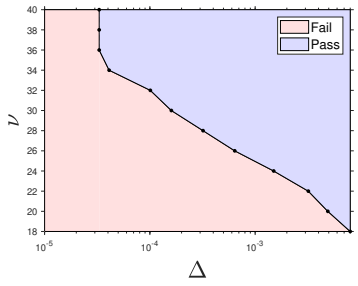


Fig. 9: Pass/fail results of teleoperated control simulations using Dyer's SHE. The system was parameterised by the security parameter ν and the encoder resolution Δ. Remaining security parameters well held constant at $\rho = 1$, $\lambda = 256$.

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:

$$\kappa > (n+1)^d M^d \tag{12}$$

$$p > (n+1)^d (M + \kappa^2)^d \tag{13}$$

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

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

$$c = m + s\kappa + rp \mod N \tag{14}$$

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:

$$m = (c \mod p) \mod \kappa \tag{15}$$

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:

$$m + m' < \kappa \tag{16}$$

$$(m + s) + (m' + s')\kappa < p \tag{17}$$

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:

$$mm' < \kappa \tag{18}$$

$$mm' + (ms' + m's + ss'\kappa)\kappa < p \tag{19}$$

Equations (12), (13), (16), (17), (18), and (19) are conditions that must be satisfied at all times.

REFERENCES

[1] M. Schulze Darup *et al.*, "Encrypted control for networked systems: An illustrative introduction and current challenges," *IEEE Control Systems Magazine*, vol. 41, no. 3, pp. 58–78, 2021.

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

[3] 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.

[4] K. Teranishi *et al.*, "Secure observer-based motion control based on controller encryption," in *2019 American Control Conference (ACC)*. IEEE, 2019, Conference Proceedings, pp. 2978–2983.

[5] 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.

[6] Y. Z. Lun *et al.*, "State of the art of cyber-physical systems security: An automatic control perspective," *Journal of Systems and Software*, vol. 149, pp. 174–216, 2019.

[7] A. Acar *et al.*, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–35, 2018.

[8] 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.

[9] "Functional mock-up interface," https://fmi-standard.org/docs/3.0/, May 2022.

[10] T. Blochwitz *et al.*, "Functional mockup interface 2.0: The standard for tool independent exchange of simulation models," in *Proceedings of the 9th International Modelica Conference*. The Modelica Association, 2012, Conference Proceedings, pp. 173–184.

[11] C. Bertsch *et al.*, "Fmi for physical models on automotive embedded targets," in *Proceedings of the 11th International Modelica Conference*. The Modelica Association, 2015, Conference Proceedings.

[12] C. Gomes, G. Abbiati, and P. G. Larsen, "Seismic hybrid testing using fmi-based co-simulation," in *Proceedings of 14th Modelica Conference 2021*. The Modelica Association, 2021, Conference Proceedings.

[13] "Fmpy (version 0.3.12)," https://github.com/CATIA-Systems/FMPy, Aug. 2022.

[14] X. Zhao, S. Kosieradzki, and J. Ueda, " Encrypted Simulation Research 2022." [Online]. Available: https://github.com/xzhao391/Encrypted-Simulation-Research-2022

[15] T. Blockwitz *et al.*, "Functional mockup interface 2.0: The standard for tool independent exchange of simulation models," in *9th International Modelica Conference*, 2012. [Online]. Available: https://elib.dlr.de/78486/

[16] J. Dyer, M. Dyer, and J. Xu, "Practical homomorphic encryption over the integers for secure computation in the cloud," in *IMA International Conference on Cryptography and Coding*. Springer, 2019, Conference Proceedings, pp. 44–76.

[17] S. Kosieradzki *et al.*, "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.

[18] S. Kosieradzki *et al.*, "Secure teleoperation control using somewhat homomorphic encryption," in *Modeling, Estimation and Control Conference 2022*, 2022, Conference Proceedings, pp. 6836–6843.