Private Anomaly Detection in Linear Controllers: Garbled Circuits vs. Homomorphic Encryption

Andreea B. Alexandru[†] Luis Burbano[‡,*] Mestan F. Çeliktuğ[♯,*] Juanita Gomez[‡]
Alvaro A. Cardenas[‡] Murat Kantarcioglu[♯] Jonathan Katz[†]

***Abstract*— Anomaly detection can ensure the operational *integrity* of control systems by identifying issues such as faulty sensors and false data injection attacks. At the same time, we need *privacy* to protect personal data and limit the information attackers can get about the operation of a system. However, anomaly detection and privacy can sometimes be at odds, as monitoring the system's behavior is impeded by data hiding. Cryptographic tools such as garbled circuits and homomorphic encryption can help, but each of these is best suited for certain different types of computation. Control with anomaly detection requires both types of computations so a naive cryptographic implementation might be inefficient. To address these challenges, we propose and implement protocols for privacy-preserving anomaly detection in a linear control system using garbled circuits, homomorphic encryption, and a combination of the two. In doing so, we show how to distribute private computations between the system and the controller to reduce the amount of computation–in particular at the low-power system. Finally, we systematically compare our proposed protocols in terms of precision, computation, and communication costs.**

I. Introduction

Encrypted control offers provable guarantees on the privacy of the underlying data in a control system [1]. The standard setup for an encrypted control system is that the state, input, and output of a plant are cryptographically protected with schemes that enable a controller to perform estimation and control without learning private information.

Privacy alone, however, cannot guarantee the integrity and fitness of data. To catch potential data injection attacks, we need additional checks such as anomaly detection [2]. Incorporating it within existing privacy-preserving control solutions is challenging since the leading cryptographic tools—garbled circuits and homomorphic encryption—are each suited for different computation types, and anomaly detection requires both types. Here, we study for the first time the challenges and methods for providing both control performance and anomaly detection in a privacy-preserving manner.

A. Related work

Most existing works focus on linear controllers, or split the computations so that the controller only has to perform linear operations [3]–[7]. Partially homomorphic encryption (PHE) is a popular tool in this context because it allows efficient evaluations of linear functions on private data, but is not sufficient to implement private non-linear controllers. Some works augment PHE with secret sharing private data into two shares and using many rounds of interaction to carry out non-linear private computations [8], [9]. Another option is to employ leveled homomorphic encryption, which enables the evaluation of (low-degree) polynomials over private data, to approximate non-linear computations. Existing works that use this approach with a single server for the controller either implement polynomial control without refreshing [10], [11] or approximate non-polynomial control [12], [13].

Circuit garbling is a two-party tool where one party garbles an arbitrary function, along with its own input, and sends these to a second party who evaluates the garbled function once, on its local input. Although a powerful tool, garbled circuits have been used less frequently for encrypted control. In [14], [15], two servers are used at the controller side to implement non-polynomial control and sensor fusion, external to the system. The closest work to ours is [16], which directly employs both homomorphic encryption and garbled circuits to privately compute path intersection, but does not discuss parties' asymmetry or compare to other solutions. In addition, as far as we are aware, no other work in encrypted control has considered anomaly detection.

B. Contributions

As noted above, homomorphic encryption (HE) and garbled circuits (GC) provide two different paradigms of private computation between two mutually distrusting parties. At a high level, HE encrypts the data to be computed upon, while GC garbles the function to be computed. HE is designed to evaluate arithmetic circuits, and is efficient for e.g., matrix-vector multiplications (essential for control computation), while GC is designed to evaluate Boolean circuits, and is efficient for e.g., comparisons (essential for anomaly detection). Most HE schemes use public-key operations, while GC mostly relies on more efficient symmetric-key operations.

Our work sets out to establish, for control and anomaly detection, the best way to use HE and GC to achieve privacy. Specifically, our goal is to design and implement a privacy-preserving anomaly-detection protocol based on cumulative statistics over a Linear Quadratic Gaussian regulator such that the controller cannot infer any information about the values of the state estimates, measurements, or control inputs, but can still identify anomalies and trigger alarms. While the

[†] Department of Computer Science, University of Maryland, College Park, MD 20742, USA {aandreea,jkatz2}@umd.edu. Work supported by NSF award #1837517.

[‡]Department of Computer Science and Engineering, University of California, Santa Cruz, CA 95064, USA {lburbano,jgomez91,alacarde}@ucsc.edu. Work supported by NSF award #1929410

[♯]Department of Computer Science, University of Texas at Dallas, TX 75080, USA {muratk,mestanfirat.celiktug}@utdallas.edu. Work supported by NSF award #1929410

[*] These authors contributed equally.

control part involves linear operations, the alarm computation is non-polynomial. Hence, it is not immediately clear whether it is more efficient to use homomorphic encryption, garbled circuits, or a combination of the two. We design and implement optimized control and anomaly detection protocols for a physical system and a single cloud-controller using:

1) Garbled circuits only. Since the roles of garbling and evaluation are not symmetric, we consider both the cloud as garbler and the cloud as evaluator.
2) Leveled homomorphic encryption for the computation at the cloud, and secret sharing for refreshing the state in order to support repeated computations.
3) Leveled homomorphic encryption for the control part and garbled circuits for anomaly detection.

Finally, we compare all three cases for various system sizes in terms of precision, communication, and computation, and identify efficiency trade-offs. The results can be used to design solutions for other control applications in this setting.

II. Problem Statement

Consider a *system* consisting of sensors, actuators and a physical plant, and a digital controller implemented at a *cloud* server, depicted in Figure 1. Our goals are to (i) control the plant to track a desired reference and (ii) perform anomaly detection to ensure correct functioning. We also want to achieve both objectives in a privacy-preserving manner. First, we describe the intended operation in the absence of privacy.

The physical system is described by a state $\mathbf{x} \in \mathbb{R}^n$ and an output $\mathbf{z} \in \mathbb{R}^p$ that evolve according to difference equations

$$\mathbf{x}[k+1] = \mathbf{A}\,\mathbf{x}[k] + \mathbf{B}\,\mathbf{u}[k] + \mathbf{v}[k] \tag{1}$$

$$\mathbf{z}[k] = \mathbf{C}\,\mathbf{x}[k] + \mathbf{w}[k], \tag{2}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{B} \in \mathbb{R}^{n\times m}, \mathbf{C} \in \mathbb{R}^{p\times n}$ are constant system matrices, and $\mathbf{v} \in \mathbb{R}^n$ is the random process noise. We assume $(\mathbf{A}, \mathbf{B})$ is controllable and $(\mathbf{A}, \mathbf{C})$ is observable.

The sensors obtain measurements $\mathbf{y} \in \mathbb{R}^p$ that can differ from the true output $\mathbf{z}$, both because of random measurement noise $\mathbf{w} \in \mathbb{R}^p$ and data injection attacks $\mathbf{y}_a \in \mathbb{R}^p$. This can happen when an attacker uses a physical phenomenon

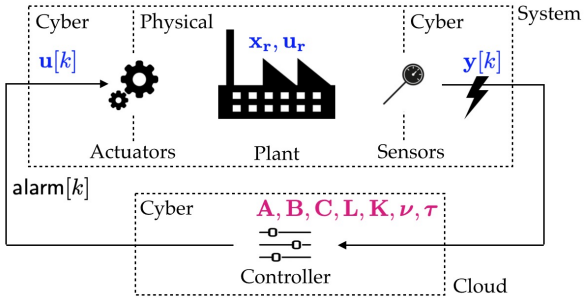


Fig. 1: Two-party setting for control computation and anomaly detection. Only the system should know the blue quantities in the system's box, only the cloud should know the magenta quantities in the cloud's box, and both should know the alarm. Not shown quantities should be hidden from both.

affecting the measurement, e.g., electromagnetic interference that changes the analog to digital conversion of the sensed value [17], or digitally injects noise. Note that an attack signal $\mathbf{y}_a$ injected outside of the system is easier to detect than one injected at the sensors (e.g., by using message authentication codes). Therefore, we consider the more challenging scenario and model this false data injection attack at the sensors as an additional signal distorting the measurements:

$$\mathbf{y}[k] = \mathbf{z}[k] + \mathbf{w}[k] + \mathbf{y}_a[k]. \tag{3}$$

Sensors then send these measurements to the cloud controller, which computes an estimate of the state $\hat{\mathbf{x}}_e \in \mathbb{R}^n$. We assume the cloud has an initial estimate $\hat{\mathbf{x}}_e[0] = \mathbf{x}[0]$. In the following steps, the estimate is computed as

$$\begin{aligned}\hat{\mathbf{x}}_e[k] = \mathbf{A}\,\hat{\mathbf{x}}_e[k-1] + \mathbf{B}\,\mathbf{u}[k-1] \\ + \mathbf{L}\left(\mathbf{y}[k] - \mathbf{C}\left(\mathbf{A}\,\hat{\mathbf{x}}_e[k-1] + \mathbf{B}\,\mathbf{u}[k-1]\right)\right),\end{aligned} \tag{4}$$

where $\mathbf{L} \in \mathbb{R}^{p\times n}$ is the estimation gain. Given the control gain $\mathbf{K} \in \mathbb{R}^{m\times n}$ and the references $\mathbf{x_r} \in \mathbb{R}^n, \mathbf{u_r} \in \mathbb{R}^m$, the control action $\mathbf{u}$ is computed as

$$\mathbf{u}[k] = -\mathbf{K}\left(\hat{\mathbf{x}}_e[k] - \mathbf{x_r}\right) + \mathbf{u_r}. \tag{5}$$

Furthermore, the cloud also has the role of an *anomaly-detection system* performing two steps: model-based prediction and anomaly detection. The predicted system state $\hat{\mathbf{x}}_p \in \mathbb{R}^n$ is computed as:

$$\hat{\mathbf{x}}_p[k] = \mathbf{A}\,\hat{\mathbf{x}}_e[k-1] + \mathbf{B}\,\mathbf{u}[k-1]. \tag{6}$$

We assume the cloud has an initial estimation $\hat{\mathbf{x}}_p[0] = \mathbf{x}[0]$.

For anomaly detection, we define the residues $\mathbf{r} \in \mathbb{R}^p$ as the distance between the measured output and the expected output, where the squaring is done element-wise:

$$\mathbf{r}[k] = \left(\mathbf{y}[k] - \mathbf{C}\,\hat{\mathbf{x}}_p[k]\right)^2. \tag{7}$$

Following [18], we define an additional statistic $\mathbf{s} \in \mathbb{R}^p$, called the nonparametric cumulative sum (CUSUM), which is initialized to zero and computed recursively as shown below. The CUSUM requires two constant design vectors $\boldsymbol{\nu} \in \mathbb{R}^p$ and $\boldsymbol{\tau} \in \mathbb{R}^p$, whose elements are strictly positive. An alarm is triggered in sensor $i \in \{1, \ldots, p\}$ if $\mathbf{s}^{(i)}[k] > \tau^{(i)}$ and the CUSUM is then reset to zero: $\mathbf{s}^{(i)}[k+1] = 0$. Otherwise, for $i \in \{1, \ldots, p\}$, the CUSUM takes the value:

$$\mathbf{s}^{(i)}[k+1] = \max\{0,\ \mathbf{s}^{(i)}[k] + \mathbf{r}^{(i)}[k] - \boldsymbol{\nu}^{(i)}\}. \tag{8}$$

Equivalently, the cloud has to set:

$$\mathsf{alarm}[k] = \mathcal{I}_{(0,\infty)}(\mathbf{s}[k] - \boldsymbol{\tau}) \tag{9}$$

$$\mathbf{s}[k+1] = \max\{\mathbf{0}, \mathbf{s}[k] + \mathbf{r}[k] - \boldsymbol{\nu}\} \odot (\mathbf{1} - \mathsf{alarm}[k]), \tag{10}$$

where $\mathcal{I}_{(0,\infty)}$ is the indicator function on the positive axis and $\odot$ represents element-wise product. $\mathcal{I}_{(0,\infty)}$ and max are applied element-wise. The CUSUM parameters are tuned based on the simulation of the physical system [2] in order to strike a balance between maintaining a low false alarm rate, and a high probability of detecting an attack.

The cloud logs the alarm, then sends the control input and alarm, $\mathbf{u}[k]$ and $\mathsf{alarm}[k]$, to the system.

Security requirements. In this work, we assume the cloud has previously performed system identification to obtain the system model, computed the control gains, and designed the CUSUM vectors using proprietary knowledge, so it wants to keep those values secret. In future work, we will consider the case where this information is given encrypted to the cloud. The system's owner wants to conceal the measurements, references, states and control actions from the cloud. However, there might be physical or cyber injection attacks against the sensors. These can impact computations and decisions both at the system and cloud controller, so both parties wish to know whether attacks have been detected and alarms raised. Thus, the two parties want a protocol that efficiently computes reference-tracking controls and accurately detects attacks on the sensors. In short, our goal is to design protocols which ensure *control performance*, *precise anomaly detection*, *privacy of sensitive data*, and *real-time efficiency*.

Notice that we are defending against (i) an external adversary that can perform injection attacks, and (ii) an internal adversary that wants to access the data at the system or at the cloud. We want our anomaly detection system to quickly and accurately identify attacks by the first adversary. The second adversary is computationally bounded and *semi-honest*, i.e., correctly follows the steps of the protocol but may store the transcript of the messages exchanged and process it to try to learn more information than allowed. A two-party protocol between the system and the cloud is defined to be *private* if all information obtained by a party during execution of the protocol, including intermediate computations, can be deduced only from the allowed inputs and outputs of that party [19, Ch. 7]. In this work, we use this privacy definition to develop privacy-preserving protocols.

III. Cryptographic Preliminaries

A. Oblivious transfer (OT)

Oblivious transfer allows a *receiver* to get a secret σ_b out of a pair σ_0, σ_1 held by a *sender*. The sender learns nothing about $b \in \{0,1\}$ and the receiver learns nothing about σ_{1-b}.

B. Garbled circuits (GC)

Garbled circuits [20] allow two parties to evaluate an arbitrary function represented as a Boolean circuit on private inputs from both parties, leaking nothing beyond the output of the function. A garbling scheme is a tuple of algorithms $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval}, \mathsf{Decode})$. During Garble, the *garbler* party creates a "garbled" representation of a Boolean circuit in which each wire is associated with two cryptographic keys, called *labels*, one for 0 and one for 1. In the garbled representation, the gates of the circuit are tables of ciphertexts encrypted using the wire labels (e.g., via AES). The garbled circuit consisting of all garbled gates is sent to the *evaluator* party. Then, the garbler provides the input garbled values to the evaluator. For its own input bits, the garbler directly sends the corresponding garbled labels. For the evaluator's input bits, the two parties run OT protocols, enabling the evaluator to obtain the garbled labels without revealing its inputs. During Eval, the evaluator evaluates the garbled circuit on the received garbled labels. Finally, the garbled output is decoded using the corresponding output wire labels in Decode.

Formally, the GC algorithms are:

- $\mathsf{Garble}(f, x) \to (\mathcal{C}, X, d)$: on input function f and input bits $x \in \{0,1\}^n$, outputs garbled representation $\mathcal{C}$, garbled input labels pairs $X = \{\mathsf{lab}_i^0, \mathsf{lab}_i^1\}_{i\in[n]}$, decoding map d.
- $\mathsf{Eval}(\mathcal{C}, \bar{X}) \to Y$: on input garbled representation $\mathcal{C}$ and a set of garbled input labels $\bar{X} = \{\mathsf{lab}_i^{x_i}\}_{i\in[n]}$ associated to input $x \in \{0,1\}^n$, output garbled output labels Y.
- $\mathsf{Decode}(d, Y) \to y$: on input decoding map d and garbled output labels Y, output the circuit's output y.

A garbling scheme is *correct* if Decode outputs $y = f(x)$ for the desired function f and input x. A garbling scheme is *private* if, given $\mathcal{C}$, the garbled input labels $\bar{X}$ of the circuit wires and the decoding map d, an adversary learns nothing apart from the number of inputs, the size of the circuit and y.

C. Leveled homomorphic encryption (HE)

A *leveled homomorphic encryption scheme* supports the encrypted evaluation of bounded-degree polynomials or bounded-depth arithmetic circuits. In HE schemes based on the Learning with Errors hardness problem, each operation evaluated on ciphertexts introduces some noise, which can cause incorrect decryption if it overflows. Multiplications introduce the most noise, therefore we want to evaluate low-depth circuits, i.e., few sequential multiplications.

Plaintexts and ciphertexts can encode a vector of scalars; an operation applied on a ciphertext is applied component-wise on the encrypted vector. This allows us to perform component-wise *additions* and *multiplications* between ciphertexts or between a plaintext and a ciphertext, and vector *rotations* (cyclic permutations of the encrypted vector).

Formally, a leveled homomorphic encryption scheme is a tuple of algorithms $\mathsf{LHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$:

- $\mathsf{KeyGen} \to (\mathsf{pk}, \mathsf{sk}, \mathsf{evk})$: outputs a public key pk, a secret key sk, and an evaluation key evk.
- $\mathsf{Enc}_{\mathsf{pk}}(m) \to c$: on input public key pk and a message m, output a ciphertext c.
- $\mathsf{Dec}_{\mathsf{sk}}(c) \to m$: on input secret key sk and a ciphertext c, output a message m.
- $\mathsf{Eval}_{\mathsf{evk}}(f, c_1, c_2, m_3) \to c$: on input evaluation key evk, ciphertexts c_1, c_2 encrypting messages m_1, m_2, a message m_3 and a depth-d arithmetic circuit f, output a new ciphertext c. The inputs c_2, m_3 are optional.

Briefly, LHE is *correct* if $\mathsf{Dec}_{sk}(\mathsf{Enc}_{\mathsf{pk}}(m)) = m$ and $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}_{\mathsf{evk}}(f, \mathsf{Enc}_{\mathsf{pk}}(m), \mathsf{Enc}_{\mathsf{pk}}(m'))) = f(m, m')$. LHE is *IND-CPA secure* if the encryptions of any two messages are computationally indistinguishable, and *function private* if a ciphertext obtained from the homomorphic evaluation of a function, together with the private key, reveals only the result of the evaluation and nothing else about the function.

D. Statistical additive blinding

In additive blinding, a secret m of l bits is split into two shares: a uniform random value s of $l + \lambda$ bits, and $m + s$. This guarantees that both shares together allow recovery of m yet each share on its own reveals nothing about m.

Both GC and HE schemes are κ-computationally secure, i.e., an adversary has to do $O(1/2^{\kappa})$ operations in order to break the underlying cryptographic problem. Additive blinding is λ-statistically secure, i.e., an adversary has $O(1/2^{\lambda})$ probability of guessing the secret based on the given shares.

IV. Overview of our solutions

Based on the theoretical preliminaries from Section III, we design efficient solutions for the privacy-preserving control and anomaly detection problem, using garbled circuits in Section V and homomorphic encryption in Section VI.

Our first step is to process the functionality of equations (4)–(10) in a Boolean circuit representation for GC and in an arithmetic circuit for HE, and push computations on constant matrices in a one-time preprocessing phase. For the GC solutions, we want to build a circuit with few multiplications, which require fewer AND gates—the gates most expensive to transmit. For the HE solution, we want to build a low-depth circuit, since fewer sequential multiplications imply smaller ciphertexts. Further, we need to transform the comparison-based computations into polynomials for HE; to this end, we use the Chebyshev interpolation and select the lowest polynomial degree that maintains sufficient accuracy.

The second step is to reduce the data size dependence. In the GC approach, we reuse some already generated garbled labels while still preserving privacy, in order to reduce the overhead associated with data transmission, which is exacerbated as the data size increases. In the HE approach, we use vector packing, which makes computations independent or only logarithmically dependent on the data size.

Noting the advantages and disadvantages of the GC and HE solutions, in Section VII, we design a hybrid solution that computes (4)–(5) as an arithmetic circuit, evaluated with HE, and (9), (10) as a Boolean circuit, evaluated with GC.

The system and the cloud are asymmetric from a computational power perspective; we expect a low-power microcontroller at the system side and a more powerful server at the cloud side. Therefore, in Section VIII, we run experiments to determine which of the four protocols (two GC, one HE and one hybrid) guarantees the lowest computation and communication overhead for the system, while offering good precision. Our experimental observations match the theoretical predictions, but also illustrate quantifiable differences between the protocols' performance.

V. A solution based on garbled circuits

We first modify equation (4) to have fewer multiplications and precompute the matrix products at the cloud. Then, we transform it, along with (5)–(10), in a Boolean circuit denoted by $\mathcal{C}$. Below, we explore two possible scenarios: 1) the cloud is the garbler and the system is the evaluator, and 2) the system is the garbler and the cloud is the evaluator.

A. Cloud is the garbler

Preprocessing: Cloud generates pairs of labels for the input wires of $\mathcal{C}$. It stores the labels corresponding to its secrets (all are constant) and sends them to the system. It then performs OT with the system for $\mathbf{x}_r$, $\mathbf{u}_r$.

Online step k

1) System measures $\mathbf{y}[k]$ after inputting $\mathbf{u}[k-1]$.
2) Cloud performs OT with the system for the labels corresponding to $\mathbf{y}[k]$.
3) Cloud computes circuit $\mathcal{C}$ and sends it to the system, along with the decoding scheme for $\mathbf{u}[k]$ and $\mathsf{alarm}[k]$.
4) System evaluates $\mathcal{C}$ and decodes $\mathbf{u}[k]$ and $\mathsf{alarm}[k]$.
5) System sends $\mathsf{alarm}[k]$ to the cloud.

B. System is the garbler

Preprocessing: System generates pairs of labels for the input wires of $\mathcal{C}$. It stores the labels, sends the labels corresponding to the references $\mathbf{x}_r, \mathbf{u}_r$ and performs OT with the cloud for the cloud's secrets (all are constant).

Online step k

1) System measures $\mathbf{y}[k]$ after inputting $\mathbf{u}[k-1]$.
2) System sends to the cloud the labels for $\mathbf{y}[k]$.
3) System computes $\mathcal{C}$ and sends it to the cloud, along with the decoding scheme for $\mathsf{alarm}[k]$.
4) Cloud evaluates $\mathcal{C}$ and decodes $\mathsf{alarm}[k]$.
5) Cloud sends $\mathsf{alarm}[k]$ and the encoded $\mathbf{u}[k]$ to the system.
6) System decodes $\mathbf{u}[k]$.

C. Optimizations

We use the Efficient Multiparty (EMP) toolkit [21] to implement our solution. EMP applies several improvements for GC: i) OT extension: push public-key operations to the preprocessing phase, ii) free XOR: an encoding such that evaluating XOR gates does not require transmission of the garbled gate, iii) garbled row reduction: only three rows are computed and sent for a non-XOR gate, iv) reduced number of non-XOR gates in the Boolean circuit, v) pipelined circuit execution: the entire circuit is not fully stored in memory, but computed as needed based on the available wires.

Further, we use the following optimizations for the circuit:

- Reuse the input wire labels for constant inputs (e.g., model, references) across circuits for different iterations.
- Hardwire zeros for the CUSUM statistic $\mathbf{s}[k+1]$ corresponding to the raised alarms.
- Do not decode $\hat{\mathbf{x}}_e[k]$ and $\mathbf{s}[k]$ between iterations. The garbler uses the output wire labels corresponding to $\hat{\mathbf{x}}_e[k]$ and $\mathbf{s}[k]$ from the circuit at time k as input wire labels for circuit at time $k+1$. This avoids performing online OT for those labels, since the evaluator already has them.
- If the garbler has enough memory, it pregenerates the circuits for consecutive iterations and stores them, which saves time in the online iterations.

Neither optimization causes any privacy leakage. To provide privacy, we need to generate new labels for the non-constant inputs of the circuits at every time step, but we can reuse the labels for constant inputs as long as the evaluator cannot decode them. Moreover, feeding output labels as input labels in the circuit at the subsequent time step is also private, since the evaluator does not have their decoding scheme and the garbler does not receive these labels from the evaluator.

D. Communication and computation

When the cloud is the garbler, in the preprocessing phase it sends to the system the labels for the model and they perform OT for the references. At each time step k, the cloud sends to the system the circuit and the decoding scheme, the system sends the alarm back, and the two parties perform OT for labels for $\mathbf{y}[k]$. When the system is the garbler, the above are reversed, with the difference that the cloud sends back the encoded $\mathbf{u}[k]$ and there is no online OT for $\mathbf{y}[k]$, but many more OTs for the model at the cloud in preprocessing phase.

When the cloud is the garbler, it has to perform the label generation and circuit computation, and is the sender in the OTs, while the system is the receiver in the OTs, evaluates the circuit and performs decoding. When the system is the garbler, these are reversed.

Circuit garbling is more intensive than circuit evaluation, because i) it involves computing cryptographic primitives for each entry of the truth table, but the evaluation only computes them for one entry and ii) it requires sampling many random values, which can be expensive on low-power platforms.

The preprocessing phase requires two rounds of communication between the parties, while an online step requires one round of communication such that the evaluator obtains the output and one round of communication for the evaluator to communicate the alarm back.

VI. A SOLUTION BASED ON HOMOMORPHIC ENCRYPTION

The parameters of a leveled HE scheme are set to support a fixed multiplicative depth—the lower the depth, the smaller the ciphertext size. We design a low-depth circuit to compute the functionality of equations (4)–(10), by precomputing matrix multiplications at the cloud and minimizing the amount of sequential online multiplications. We then use the CKKS leveled approximate homomorphic scheme [22] to homomorphically evaluate the circuit, which is designed to have better precision for real-valued inputs[1].

Since our computations continue over an unspecified number of time steps, we need to support circuits deeper than a fixed depth, so we have to refresh the noise in the ciphertexts. We do this by having the system, who owns the secret key, decrypt and re-encrypt ciphertexts. In order to preserve privacy of the intermediate results, the cloud first applies additive blinding on the ciphertexts that need to be refreshed.

A. Packed-matrix multiplication

We compute products such as $\mathbf{q} = \mathbf{M}\mathbf{p}$ between a plaintext matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ and an encrypted vector $\mathbf{p} \in \mathbb{R}^n$. We take advantage of the encrypted vector encoding to perform this efficiently, with only a logarithmic dependence on the data size. Note that $\mathbf{q}_i = \langle \mathbf{M}_i, \mathbf{p} \rangle = \langle \mathbf{1}_n, \mathbf{M}_i \odot \mathbf{p} \rangle$, $i \in \{1, \ldots, m\}$, where $\mathbf{M}_i$ is the i-th row of $\mathbf{M}$, $\langle , \rangle$ denotes the inner product, $\odot$ denotes element-wise multiplication and $\mathbf{1}_n$ is the vector of n ones. We encode the matrix $\mathbf{M}$ row-by-row in one plaintext. We encode m copies of $\mathbf{p}$ in a ciphertext and perform only one homomorphic multiplication to get a ciphertext of $[\mathbf{M}_1 \odot \mathbf{p}, \mathbf{M}_2 \odot \mathbf{p}, \ldots, \mathbf{M}_m \odot \mathbf{p}]$. Summing elements of an encrypted vector involves repeatedly adding copies of that ciphertext rotated to the left accordingly. Specifically, we perform $O(\log n)$ homomorphic rotations and $O(\log n)$ homomorphic additions to obtain the encryption of $\mathbf{q} = [\mathbf{q}_1, *, \ldots, *, \mathbf{q}_2, *, \ldots, *, \mathbf{q}_m, *]$, where the elements of the result are every n positions and by $*$ we denote partial sums. We also ensure that the ciphertexts in the subsequent operations involving $\mathbf{q}$ have the same encoding (the relevant values are in positions which are multiples of n).

B. Polynomial approximation for comparisons

Leveled HE can evaluate polynomials on encrypted values, but comparisons are not natively supported. Thus, we evaluate polynomial approximations of the non-polynomial functions. The Chebyshev series polynomial interpolation [23] is a near-minimax approximation of a continuous function on the interval $[-1, 1]$. If the input interval is not $[-1, 1]$, we first apply a linear transformation on the inputs. The quality of the approximation increases with the degree of the polynomial. Nevertheless, polynomial approximations of functions with discontinuities will have larger approximation errors compared to smooth functions for the same degree.

Both the CUSUM statistic and the alarm are non-polynomial functions, so we use Chebyshev interpolation to evaluate them on encrypted data. We choose 16 as the lowest degree that provides good accuracy. Define $\bar{\mathbf{s}}[k] := \max\{\mathbf{0}, \mathbf{s}[k] + \mathbf{r}[k] - \boldsymbol{\nu}\}$, where the maximum is applied element-wise. Importantly, we will compute $\mathsf{Enc}(\bar{\mathbf{s}}[k])$ and $\mathsf{Enc}(\mathsf{alarm}[k])$ with only one polynomial evaluation each (recall that the ciphertexts encode vectors and a function can be homomorphically applied element-wise).

C. Workflow

We minimize the depth of the computed arithmetic circuit by writing $\hat{\mathbf{x}}_e[k], \mathbf{u}[k], \mathbf{x}_p[k]$ as depth-1 circuits, $\mathbf{r}[k]$ as a depth-2 circuit, the evaluation of the Chebyshev approximation of degree-16 as a depth-6 circuit, and thus $\mathrm{alarm}[k]$ as a depth-6 circuit and $\mathbf{s}[k+1]$ as a depth-8 circuit.

To avoid excess notation, we implicitly assume the cloud additively blinds all slots containing partial information when it sends $\mathsf{Enc}(\mathbf{u}[k])$, $\mathsf{Enc}(\mathsf{alarm}[k])$ to the system. We denote the additive blinding of a vector $\mathbf{m}$ by $(\langle \mathbf{m} \rangle_1, \langle \mathbf{m} \rangle_2)$.

Preprocessing

1) System generates keys $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk})$ and encrypts $\mathbf{x}_r, \mathbf{u}_r$ under the appropriate encoding for packed-matrix multiplication. It sends $\mathsf{pk}, \mathsf{evk}, \mathsf{Enc}(\mathbf{x}_r), \mathsf{Enc}(\mathbf{u}_r)$ to the cloud.
2) Cloud encodes matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{L}, \mathbf{K}$ and the relevant products between them for packed-matrix multiplication.

Online step k

1) System measures $\mathbf{y}[k]$ after inputting $\mathbf{u}[k-1]$.
2) System encrypts and sends $\mathsf{Enc}(\mathbf{y}[k])$, $\mathsf{Enc}(\langle \hat{\mathbf{x}}_e[k-1] \rangle_1)$ under the appropriate encodings.
3) Cloud recovers $\mathsf{Enc}(\hat{\mathbf{x}}_e[k-1])$ and computes $\mathsf{Enc}(\hat{\mathbf{x}}_e[k])$, $\mathsf{Enc}(\mathbf{u}[k])$, $\mathsf{Enc}(\hat{\mathbf{x}}_p[k])$, $\mathsf{Enc}(\mathbf{r}[k])$ as in equations (4)–(7).

[1] In the CKKS scheme, quantifying the leakage from the noise about the inputs, i.e., function privacy, is an open research problem.

4) Cloud computes $\mathsf{Enc}(\mathsf{alarm}[k])$ via a polynomial approximation for $\mathsf{Enc}(\mathcal{I}_{[0,\infty]}(\mathbf{s}[k]-\boldsymbol{\tau}))$.
5) Cloud computes $\mathsf{Enc}(\bar{\mathbf{s}}[k+1])$ via a polynomial approximation for $\mathsf{Enc}(\max\{\mathbf{0}, \mathbf{s}[k]+\mathbf{r}[k]-\boldsymbol{\nu}\})$.
6) Cloud samples random shares and sends $\mathsf{Enc}(\langle\bar{\mathbf{s}}[k+1]\rangle_1)$, $\mathsf{Enc}(\langle\hat{\mathbf{x}}_e[k]\rangle_1)$, $\mathsf{Enc}(\mathbf{u}[k])$, $\mathsf{Enc}(\mathsf{alarm}[k])$ to the system.
7) System decrypts all ciphertexts. It rounds the alarm and resets the elements in $\mathbf{s}[k+1]$ corresponding to a triggered alarm. It sends to the cloud fresh encryptions $\mathsf{Enc}(\langle\hat{\mathbf{x}}_e[k]\rangle_1)$, $\mathsf{Enc}(\langle\mathbf{s}[k+1]\rangle_1)$ along with the $\mathsf{alarm}[k]$.
8) Cloud obtains a fresh encryption $\mathsf{Enc}(\langle\hat{\mathbf{x}}_e[k]\rangle_1)$ by adding $\langle\hat{\mathbf{x}}_e[k]\rangle_2$. Then, the cloud obtains a fresh encryption of $\mathsf{Enc}(\mathbf{s}[k+1])$ by subtracting $\langle\mathbf{s}[k+1]\rangle_2 \odot (1-\mathsf{alarm}[k])$.

D. Communication and computation

In the preprocessing phase, the system sends the public and evaluation keys to the cloud, which stores them. The system can erase the evaluation keys afterwards.

At every iteration k the system sends four fresh ciphertexts: $\mathsf{Enc}(\langle\hat{\mathbf{x}}_e[k-1]\rangle_1), \mathsf{Enc}(\langle\mathbf{s}[k]\rangle)$, two encodings of $\mathsf{Enc}(\mathbf{y}[k])$, and a plaintext $\mathsf{alarm}[k-1]$, and receives four ciphertexts: $\mathsf{Enc}(\langle\hat{\mathbf{x}}_e[k]\rangle_1)$, $\mathsf{Enc}(\mathbf{u}[k])$, $\mathsf{Enc}(\mathrm{alarm}[k])$ and $\mathsf{Enc}(\langle\bar{\mathbf{s}}[k+1]\rangle_1)$. Computation-wise, the system only performs four encryptions and four decryptions. On the other hand, the cloud needs to evaluate high degree polynomials and perform rotations over ciphertexts.

The preprocessing requires one communication round from the system to the cloud, while an online time step requires one round from the system to the cloud to send the initial ciphertexts and one round from the cloud to the system to send the encrypted output and states.

VII. A hybrid solution

The problem of private anomaly detection involves both arithmetic and Boolean circuits. Therefore, we aim to combine the best of both worlds in a hybrid approach, where the computation of the state estimates, control input and residues, which involve matrix-vector multiplications, is done via HE, and the computation of the cumulative sum and alarm, which involve non-polynomial operations, is done via GC. Here, we consider the cloud to be the garbler. The garbled circuit for equations (9), (10) is denoted by $\mathcal{C}'$. The optimizations described for GC and HE should be applied here as well.

Preprocessing

1) Perform steps 1) and 2) of the HE preprocessing.
2) Cloud generates pairs of labels for the input wires of $\mathcal{C}'$. It stores the labels corresponding to constant secrets $\boldsymbol{\tau}$ and $\boldsymbol{\nu}$ and sends them to the system.

Online step k

1) Perform steps 1)–3) of the HE online step k.
2) Cloud sends $\mathsf{Enc}(\mathbf{u}[k])$, $\mathsf{Enc}(\langle\hat{\mathbf{x}}_e[k]\rangle_1)$ and $\mathsf{Enc}(\langle\mathbf{r}[k]\rangle_1)$.
3) System decrypts all ciphertexts.
4) Cloud performs OT with the system for the labels corresponding to $\langle\mathbf{r}[k]\rangle_1$ and sends the labels for $\langle\mathbf{r}[k]\rangle_2$.
5) Cloud computes circuit $\mathcal{C}'$ and sends it to the system, along with the decoding scheme for $\mathrm{alarm}[k]$.
6) System evaluates $\mathcal{C}'$ and decodes $\mathsf{alarm}[k]$.
7) System sends $\mathsf{alarm}[k]$ to the cloud.

The preprocessing phase requires two rounds of communication between the parties, while an online step requires one round of communication from the system to the cloud for the HE-based computation, one round of communication from the cloud to the system for the OT and circuit transfer, and finally, another round of communication from the system to the cloud to send back the alarm.

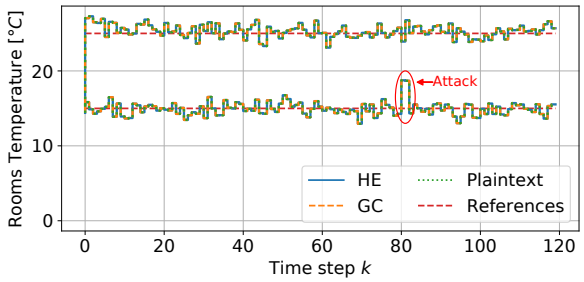
VIII. Implementation and comparison

We illustrate the performance of the proposed protocols on the problem of temperature control in a building with two zones described in [7], that has $p = 10$ sensors and $m = 2$ actuators. The sampling time is 420 seconds. We set the reference for the indoors air temperature to be 15°C and 25°C, respectively. The CUSUM discount parameter $\boldsymbol{\nu}$ and threshold parameter $\boldsymbol{\tau}$ are designed in order to obtain a 0.01% false alarm rate in nominal conditions. To see how performance scales with data size, we also consider two other systems of size $(p = 20, m = 4)$ and $(p = 50, m = 10)$.

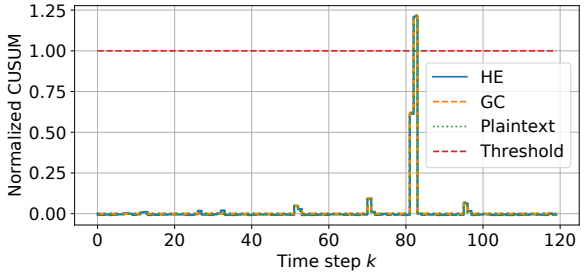
The GC protocols were implemented using EMP [21] and the HE protocol using SEAL [24]. The parameters for all protocols are selected for 128-bit computational security and 20-bit statistical security for the secret refresh.

A. Precision

We show the simulation results for the above system in Figure 2. To capture the quality of the anomaly detection, we add both process and measurement noise, as well as a



(a) System measurements $\mathbf{y}[k]$ for air temperature in the two rooms.



(b) Normalized CUSUM statistic $\mathbf{s}[k]$ for the sensor associated to the temperature in the first room.

Fig. 2: Comparison between the simulation obtained from the plaintext, GC and HE implementations. There is an attack on the first sensor, which can be observed both in (a) and (b).

targeted attack on the sensor for the indoor temperature for the first room, between time steps 80 and 82, which can be observed in Figure 2a. The behavior of the CUSUM statistic shown in Figure 2b is as expected, increasing at time step 81 and triggering an alarm at step 82, then resetting.

The HE implementation perfectly matches the plaintext results for the outputs $\mathbf{y}, \mathbf{u}, \hat{\mathbf{x}}_e, \hat{\mathbf{x}}_p, \mathbf{r}$. However, the CUSUM results slightly differ due to the errors in the polynomial approximation of non-smooth functions. Nevertheless, the true values are tracked closely, thus raising no false alarms.

We implement the garbled circuit protocols using a fixed-point representation of 48-bit with half of the bits used for the integer part and the rest for the fractional part, which also track the plaintext results for all quantities.

The hybrid output (not shown) is identical to HE output shown in Figure 2a and identical to GC output in Figure 2b.

B. Computation

To understand which solution is the most appropriate for a system with computational constraints, we run and compare on the same mid-range machine with Intel Core i5 and 8GB of RAM the amount of work that each party has to do in the proposed protocols. All numerical results are averaged over 120 instances. The cloud platform is usually more powerful, while the system platform is usually lower-power.

We approximate the work of a party in GC by the number of calls to the cryptographic hash function used—the most computationally expensive part. Figure 3 shows that the garbler performs roughly twice as much computation work as the evaluator. For this reason, in the hybrid protocol, we use the cloud as garbler and the system as evaluator.

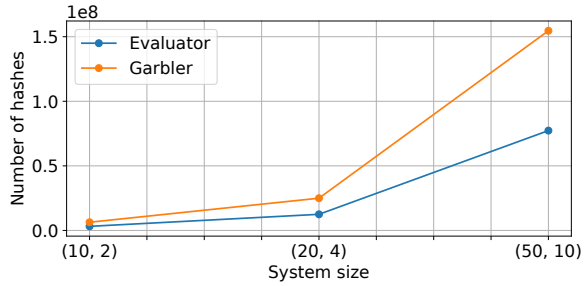


Fig. 3: Computation overhead for garbling and evaluating.

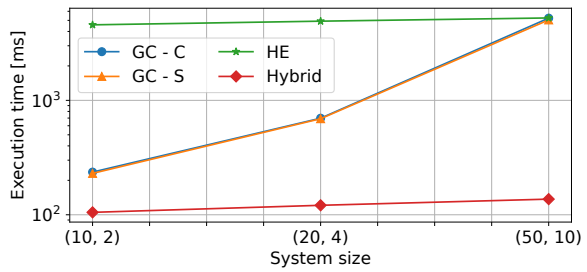


Fig. 4: Average execution time for one online iteration in the four protocols. In the legend, C means that the cloud is the garbler and S means that the system is the garbler.

In the HE protocol, the system's work is negligible compared to the cloud's work: the system's runtime in one step (< 1 second) is 17% of the cloud's runtime, hence also negligible compared to the system runtime in GC.

Figure 4 depicts the total runtime per time step obtained with the two GC protocols, the HE and hybrid protocols. We observe that the runtime for the two GC instances is almost equal, showing that circuit garbling by far dominates the OTs and decoding. For the small and medium system sizes, the GC solution outperforms the HE approach, but is matched by it at the largest system. This happens because in GC, a larger system size implies more (cheaper) symmetric-key operations, whereas the HE scheme supports encoding a large vector in one ciphertext and performing only one (expensive) public-key encrypted operation corresponding to a vectorized operation. The hybrid approach is always the most efficient.

C. Communication

The GC protocols are communication intensive (Figure 5) due to the transfer of the garbled tables and the input wire labels, each of 128 bits. During preprocessing, the communication between parties is approximately half of quantity for one online time step. The communication pattern between the garbler and evaluator is almost identical regardless of which party is the garbler, so we only depict the case where the cloud is the garbler and the system is the evaluator.

Ensuring 128-bits of security also comes at a high communication cost in the HE protocol. Moreover, to allow additive shares of 20 bits, we need to add an extra level. During preprocessing, the system sends the keys and the reference ciphertexts to the cloud, amounting to 40.07 MiB for the $(10, 2)$ and $(20, 4)$ systems, and 56.87 MiB for the $(50, 10)$ system. The online iterations are communication intensive, as shown in Figure 5, but remain constant with the system size, again because of the packing capabilities of HE. Since the hybrid solution is designed to use the advantages of both GC and HE, it has the most efficient communication overhead both for the online iterations and for the prepreprocessing (roughly a 10x improvement compared to HE). However, the hybrid solution requires one additional communication round per time step compared to pure GC and pure HE, and extra interaction is undesirable in low-latency networks.

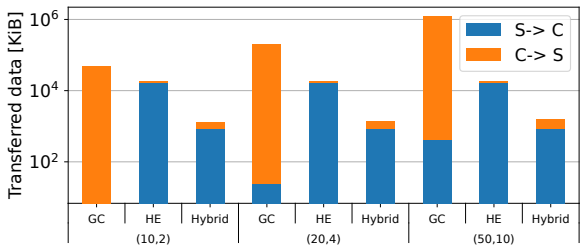


Fig. 5: Communication between the cloud C and the system S. The arrow in the legend shows the data flow. The GC implementation is with the cloud being the garbler.

IX. Conclusions

After the theoretical and practical analyses of our baseline protocols, our results indicate the following conclusions.

GC. The garbled circuit generation is more intensive than the evaluation so it is preferable for the cloud to garble and the system to evaluate. In this setting, we need fewer OTs for preprocessing (since the cloud has more data), but we need an OT for $\mathbf{y}[k]$ at each online step; however, these are negligible compared to the circuit transmission.

Runtime. Using GC is faster than using HE for medium-sized systems, since symmetric-key operations are faster than public-key operations. However, the number of operations done by the system in the HE protocol is much smaller than in the GC protocol; the system is mostly idle during a time step in the former. Further, the runtime of the HE protocol is largely constant with the data size up until very large matrices, thanks to packing, whereas the runtime for GC scales quadratically with the matrices' size difference.

Communication. The GC protocols are more expensive than the HE protocol, since they require the transmission of the garbled tables at every time step, but have cheaper preprocessing. If the network latency is significant and the data size is large, the communication overhead of the GC can be prohibitive. The HE protocol has an intensive preprocessing, where the evaluation keys need to be transmitted and requires a lower communication overhead for each time step that is mostly constant with the data size, but the bulk of the communication originates at the system.

Precision. All solutions offer very good precision. Decreasing the precision requirements would improve the computation and communication overheads. It is worth noting that the particular application of anomaly detection is forgiving of polynomial approximation errors: the alarm is decrypted and rounded at the system and the outputs of the polynomial approximations are not used in subsequent multiplications. In general, a GC approach is more precise for non-polynomials than the HE approach, since the operations are performed exactly, rather than approximated.

Privacy. The GC implementation leaks no information about the private data of each party and its security guarantees are well understood. The hardness of the cryptographic problem underlying the HE scheme is still actively studied.

Hybrid. The hybrid protocol proves to be the most efficient solution, by combining the best parts of HE and GC. The circuit size and the number of OTs are much smaller, as the matrix-vector multiplications are not garbled, thus removing the communication overhead. The comparisons are not done over encrypted data, thus significantly reducing the depth of the encrypted arithmetic circuit and improving the precision. These improvements come at the cost of one more communication round compared to the HE and GC protocols.

We note that the main bottleneck of the proposed protocols is the communication overhead. We plan to further improve the baseline implementations of the protocols we presented here, by reducing the number of levels needed in the HE solution via more efficient polynomial evaluation, and to reduce the garbled circuit size by a customized design.

References

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

[2] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *Proc. of the 6th ACM Symposium on Information, Computer and Communications Security*, 2011, pp. 355–366.

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

[4] F. Farokhi, I. Shames, and N. Batterham, "Secure and private control using semi-homomorphic encryption," *Control Engineering Practice*, vol. 67, pp. 13–20, 2017.

[5] M. S. Darup, A. Redder, I. Shames, F. Farokhi, and D. Quevedo, "Towards encrypted MPC for linear constrained systems," *IEEE Control Systems Letters*, vol. 2, no. 2, pp. 195–200, 2017.

[6] C. Murguia, F. Farokhi, and I. Shames, "Secure and private implementation of dynamic controllers using semihomomorphic encryption," *IEEE Transactions on Automatic Control*, vol. 65, no. 9, pp. 3950–3957, 2020.

[7] A. B. Alexandru and G. J. Pappas, "Encrypted LQG using labeled homomorphic encryption," in *Proc. of the 10th ACM/IEEE International conference on cyber-physical systems*, 2019, pp. 129–140.

[8] F. J. Gonzalez-Serrano, A. Amor-Martın, and J. Casamayon-Anton, "State estimation using an extended kalman filter with privacy-protected observed inputs," in *International Workshop on Information Forensics and Security*. IEEE, 2014, pp. 54–59.

[9] A. B. Alexandru, M. Morari, and G. J. Pappas, "Cloud-based MPC with encrypted data," in *Conference on Decision and Control*. IEEE, 2018, pp. 5014–5019.

[10] N. Schlüter, M. Neuhaus, and M. S. Darup, "Encrypted dynamic control with unlimited operating time via FIR filters," in *European Control Conference*. IEEE, 2021, pp. 952–957.

[11] J. Kim, H. Shim, and K. Han, "Dynamic controller that operates over homomorphically encrypted data for infinite time horizon," *IEEE Transactions on Automatic Control*, 2022.

[12] A. B. Alexandru, A. Tsiamis, and G. J. Pappas, "Towards private data-driven control," in *Conference on Decision and Control*. IEEE, 2020, pp. 5449–5456.

[13] J. Suh and T. Tanaka, "SARSA (0) reinforcement learning over fully homomorphic encryption," in *SICE International Symposium on Control Systems*. IEEE, 2021, pp. 1–7.

[14] K. Tjell, N. Schlüter, P. Binfet, and M. S. Darup, "Secure learning-based MPC via garbled circuit," in *Conference on Decision and Control*. IEEE, 2021, pp. 4901–4908.

[15] W. E. Curran, C. A. Rojas, L. Bobadilla, and D. A. Shell, "Oblivious sensor fusion via secure multi-party combinatorial filter evaluation," in *Conference on Decision and Control*. IEEE, 2021, pp. 5613–5620.

[16] L. Li, A. Bayuelo, L. Bobadilla, T. Alam, and D. A. Shell, "Coordinated multi-robot planning while preserving individual privacy," in *International Conference on Robotics and Automation*. IEEE, 2019, pp. 2188–2194.

[17] C. Yan, H. Shin, C. Bolton, W. Xu, Y. Kim, and K. Fu, "Sok: A minimalist approach to formalizing analog sensor security," in *Symposium on Security and Privacy*. IEEE, 2020, pp. 233–248.

[18] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–36, 2018.

[19] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.

[20] A. C.-C. Yao, "How to generate and exchange secrets," in *Symposium on Foundations of Computer Science*. IEEE, 1986, pp. 162–167.

[21] X. Wang, A. J. Malozemoff, and J. Katz, "EMP-toolkit: Efficient MultiParty computation toolkit," https://github.com/emp-toolkit, 2016.

[22] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 409–437.

[23] J. C. Mason and D. C. Handscomb, *Chebyshev polynomials*. CRC press, 2002.

[24] "Microsoft SEAL (release 3.6.2)," https://github.com/Microsoft/SEAL, Feb. 2021, Microsoft Research, Redmond, WA.