2020 59th IEEE Conference on Decision and Control (CDC)
Jeju Island, Republic of Korea, December 14-18, 2020

Towards Private Data-driven Control

Andreea B. Alexandru Anastasios Tsiamis George J. Pappas

Abstract— **Control as a Service (CaaS) is becoming a reality–particularly in the case of building automation and smart grid management. Often, the control algorithms in CaaS focus on controlling the client's system directly from input-output data, since the system's model might be private or unavailable. Therefore, large quantities of data collected from the client need to be uploaded to a cloud server. This data can be used by a malevolent cloud service provider to infer sensitive information about the client and mount attacks. In this paper, we co-design a solution that interlaces control and privacy. Our goal is to perform online data-driven control on encrypted input-output data, while maintaining the privacy of the client's uploaded data, desired setpoint and control actions. We design our control algorithm based on results from the behavioral framework, which is more encryption-friendly compared to other classical frameworks. We obtain privacy by using a leveled homomorphic encryption scheme to enable the cloud to perform complex computations on the client's encrypted data. Finally, we achieve efficiency by manipulating the tasks required by the control algorithm such that they only involve arithmetic circuits, as well as by leveraging parallelization and ciphertext packing.**

I. Introduction

The shift in paradigm brought by the advent of cloud computing has facilitated more and more services to be offloaded to the cloud. Examples include analytics and machine learning over data collected distributedly, smart grid control and management, and process control. In the area of smart building control, the Internet of Things technology has created numerous opportunities for automation [1], leading to the emergence of Control as a Service (CaaS) businesses. The owners/clients are incentivized to outsource the building infrastructure management to a CaaS business that offers specialized algorithms to optimize a desired energy cost, while achieving the desired levels of comfort. At the same time, the CaaS algorithms need to guarantee cybersecurity and privacy for the client's privacy-sensitive data collected and computed upon at the CaaS cloud server. Otherwise, the large collection of data from the IoT sensors and devices could be used by a malevolent cloud service provider or a hacker to infer sensitive information about the people occupying those buildings or mount attacks on the infrastructure.

A. Related work

In recent years, there has been a surge of interest in private control algorithms that operate on encrypted data [2]–[7]. Most of the works in this area consider the system parameters to be public and compute mainly linear control algorithms, which require only partially homomorphic encryption (PHE) schemes. Nonlinear control algorithms are considered in [4], [5], where PHE is either combined with secure multiparty computation tools or the client partakes in the computation. The exceptions to using PHE in encrypted control are [2], [6], [7], which use either somewhat or (leveled) fully homomorphic encryption (FHE) schemes to guarantee more privacy for the client's system parameters or to achieve more complex control algorithms. The closest work to this paper is [7], where the cloud computes the value function in a reinforcement learning task over multiple time steps in a one-shot way with FHE.

Since its genesis in [8], FHE has been substantially developed, in terms of more efficient leveled constructions [9]–[11], bootstrapping methods [12], [13], computational and hardware optimizations. There are many libraries that implement various FHE schemes and capabilities [14]–[17] and a tremendous number of papers that build on them.

Identification and data-driven control of unknown systems has been extensively studied in classical and recent literature [18]–[23]. Most methods are usually designed with the objectives of sample-efficiency and control performance, without considering privacy requirements. For example, the standard system identification-certainty equivalence control architecture might require solving non-convex problems [24] or involve operations which are not encryption-friendly, e.g. Singular Value Decomposition [19]. Other methods might be more compatible with modern encryption tools, for example the behavioral framework [20]–[23], [25], which has received renewed attention recently. In the behavioral framework, an alternative representation of linear systems is considered: the system can be directly represented in terms of input-output data [20]; the only requirement is that the input should be rich enough (persistently exciting). Although the behavioral representation is less parsimonious than the state space one, it enables us to use less costly encrypted operations.

B. Contributions

Our aim is to give the first solution to the problem of encryption-aware data-driven control. Precisely, we consider that the model parameters of a linear system to be controlled are either not available, variable or privacy-sensitive. Unlike the known-model case, the existing control techniques might not be encryption-friendly. To this end, we co-design the control scheme with respect to both *control performance* and *privacy specifications*. Our controller is based on the behavioral framework and the control performance is captured by the LQR cost. We consider i) an offline version, where the cloud computes an offline feedback control law on precollected encrypted data from the client, and ii) an

The authors are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104. {aandreea,atsiamis,pappasg}@seas.upenn.edu

online version, where the cloud computes the control based on both the offline precollected data and on new encrypted samples received from the client. In both cases, the cloud has to process the received data in a timely manner to send back a control input for the system at every time step. This problem involves nonpolynomial computations, so we leverage approximations of the initial problem to allow the implementation with a leveled homomorphic cryptosystem.

Specifically, our contributions are the following:

- Propose an approximation of a data-predictive control problem that is amenable to encrypted implementation.
- Extend this approximation to allow online collection and incorporation of samples to improve robustness.
- Formulate optimized encrypted algorithms for offline and online feedback data-driven control problems.
- Implement the algorithms and showcase the results for a temperature control problem.

II. Problem formulation

A client owns system (1) and contracts a cloud service to provide the control for this system:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \mathbf{e}^1_t, \quad \mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{e}^2_t, \tag{1}$$

with $\mathbf{x}, \mathbf{e}^1 \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{y}, \mathbf{e}^2 \in \mathbb{R}^p$.

A private CaaS should solve the control problem that minimizes the cost associated to setpoint tracking while satisfying the unknown system's dynamics (1).

Control requirements. Assume we are given a batch of offline input-output data for system (1). At every time t, given all the previous input-output samples, i.e., $\mathbf{u}_{0:t-1}$ and $\mathbf{y}_{0:t-1}$, we want to privately compute the receding horizon control $\mathbf{u}^{*,t} \in \mathbb{R}^{Nm}$ in order to track the reference $\mathbf{r}$ for some costs $\bar{\mathbf{Q}}$ and $\bar{\mathbf{R}}$, which is the solution of the LQR optimization problem:

$$\begin{aligned} \min_{\mathbf{u},\mathbf{y}} \quad & \frac{1}{2}\sum_{k=t}^{N+t-1}\Big(||\mathbf{y}_k - \mathbf{r}_k||^2_{\bar{\mathbf{Q}}} + ||\mathbf{u}_k||^2_{\bar{\mathbf{R}}}\Big) \\ s.t. \quad & \mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k,\ \mathbf{y}_k = \mathbf{C}\mathbf{x}_k, \end{aligned} \tag{2}$$

with no prior knowledge about the system model $\mathbf{A}, \mathbf{B}, \mathbf{C}$. We also want (2) to perform well under (small) process and measurement noise $\mathbf{e}^1_t, \mathbf{e}^2_t$. In Section III, we describe a data-driven reformulation of (2) that is encryption-friendly.

Privacy requirements. In the scenario we consider, the cloud service should not be able to infer anything about the client's private data, which consists of the input signals $\mathbf{u}$, the output signals $\mathbf{y}$, the model $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and state $\mathbf{x}$ (which are unknown in a data-driven control problem), and any intermediate values. The costs $\bar{\mathbf{Q}}, \bar{\mathbf{R}}$ can be chosen by the cloud, as part of the CaaS service or chosen by the client. The cloud service is considered to be *semi-honest*, which means that it does not deviate from the client's specifications. We assume this to be the case because the cloud server is under contract. Furthermore, the cloud service should receive all the data from the client in encrypted form and hence, only perform operations on encrypted data.

The formal privacy definition can be written in the standard simulation paradigm [26, Ch. 7, Def. 7.2.1], i.e., the input-output distribution of the cloud server is computationally indistinguishable from the input-output distribution of a simulator that does not receive any of the client's messages.

Efficiency requirements. The computation time of the control action at the current time step should not exceed the sampling time of the system. We also require the client to be exempt from heavy computation and communication.

III. Encryption-aware reformulation

We can reformulate the control problem (2) inspired by the behavioral framework [20]–[23]. First, we introduce some preliminary concepts. A *block-Hankel matrix* for the input signal $\mathbf{u} = \begin{bmatrix}\mathbf{u}_0^\intercal & \mathbf{u}_1^\intercal & \dots & \mathbf{u}_{T-1}^\intercal\end{bmatrix}^\intercal \in \mathbb{R}^{mT}$ is given by the following, for a positive integer L:

$$\mathbf{H}_L(\mathbf{u}) := \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \dots & \mathbf{u}_{T-L} \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_{T-L+1} \\ \vdots & & \ddots & \vdots \\ \mathbf{u}_{L-1} & \mathbf{u}_L & \dots & \mathbf{u}_{T-1} \end{bmatrix}.$$

By definition, the signal $\mathbf{u}$ is persistently exciting of order L if $\mathbf{H}_L(\mathbf{u}) \in \mathbb{R}^{mL\times(T-L+1)}$ is full row rank.

Let us construct block-Hankel matrices for the "past" and "future" input and output data, $\mathbf{u}^d \in \mathbb{R}^{mT}$ and $\mathbf{y}^d \in \mathbb{R}^{pT}$, for M samples for the past data and N samples for the future data, where $S := T - M - N + 1$ and $\mathbf{U}^p \in \mathbb{R}^{mM\times S}, \mathbf{U}^f \in \mathbb{R}^{mN\times S}, \mathbf{Y}^p \in \mathbb{R}^{pM\times S}, \mathbf{Y}^f \in \mathbb{R}^{pN\times S}$:

$$\mathbf{H}_{M+N}(\mathbf{u}^d) =: \begin{bmatrix}\mathbf{U}^p \\ \mathbf{U}^f\end{bmatrix}, \quad \mathbf{H}_{M+N}(\mathbf{y}^d) =: \begin{bmatrix}\mathbf{Y}^p \\ \mathbf{Y}^f\end{bmatrix}. \tag{3}$$

A. Offline feedback data-driven control problem

Assume we are given precollected input and output data satisfying Assumption 1. Let $\mathbf{U}^p, \mathbf{Y}^p, \mathbf{U}^f, \mathbf{Y}^f$ be the respective past and future Hankel matrices, for some past and future horizons M, N.

Assumption 1 (Data richness): The precollected offline input trajectory $\mathbf{u}^d$ is persistently exciting of order $M+N+n$, where n is the order of the system [20].

Fix a time t and let $\bar{\mathbf{u}}_t = \mathbf{u}_{t-M:t-1}$ be the batch vector of the last M inputs. The batch vector $\bar{\mathbf{y}}_t$ of the last M outputs is defined similarly. If $M \geq n$, the standard LQR problem (2) can be re-formulated as the data-driven control problem [22] in (4), where the state representation is replaced with the precollected data. According to the behavioral framework, an input-output trajectory of a linear system is in the image of the block-Hankel matrices for the precollected data, i.e., the constraint in (4), where $\mathbf{g}$ is a preimage of the trajectory. The first m elements of $\mathbf{u}^{*,t}$ are then input into the system in a receding horizon fashion and $\mathbf{y}^{*,t}$ is the predicted output.

$$\begin{aligned} \min_{\mathbf{g},\mathbf{u},\mathbf{y}} \quad & \frac{1}{2}\sum_{k=t}^{N+t-1}\Big(||\mathbf{y}_k - \mathbf{r}_k||^2_{\bar{\mathbf{Q}}} + ||\mathbf{u}_k||^2_{\bar{\mathbf{R}}}\Big) \\ s.t. \quad & \begin{bmatrix}\mathbf{U}^p \\ \mathbf{Y}^p \\ \mathbf{U}^f \\ \mathbf{Y}^f\end{bmatrix} \cdot \mathbf{g} = \begin{bmatrix}\bar{\mathbf{u}}_t \\ \bar{\mathbf{y}}_t \\ \mathbf{u} \\ \mathbf{y}\end{bmatrix}. \end{aligned} \tag{4}$$

We now depart from the behavioral control problem (4) considered in the existing literature and explore a more encryption-friendly form. First, we rewrite (4) as a minimization problem depending only on $\mathbf{g}$ by enforcing $\mathbf{u} = \mathbf{U}^f\mathbf{g}$ and $\mathbf{y} = \mathbf{Y}^f\mathbf{g}$. Second, in practice, there will be noise affecting the output measurement, as well as precision errors induced by encryption, which might prevent an exact solution to the equality constraint of (4). Hence, we prefer a regularized least-squares approach to the equality constraint in (4) with regularization weights λ_y and λ_u. Finally, to avoid overfitting, we also penalize the magnitude of $\mathbf{g}$ through two-norm regularization. We opt for two-norm regularizations to obtain better efficiency for the encrypted algorithm, as well as more robustness with respect to noise, and uniqueness of solution $\mathbf{g}^{*,t}$. This yields the following formulation, where $\mathbf{Q} = \mathrm{blockdiag}(\bar{\mathbf{Q}}, \dots, \bar{\mathbf{Q}})$ and $\mathbf{R} = \mathrm{blockdiag}(\bar{\mathbf{R}}, \dots, \bar{\mathbf{R}})$:

$$\begin{aligned}\min_{\mathbf{g}} \quad & \frac{1}{2}\left(\|\mathbf{Y}^f\mathbf{g}-\mathbf{r}_t\|_{\mathbf{Q}}^2+\|\mathbf{U}^f\mathbf{g}\|_{\mathbf{R}}^2+\lambda_y\|\mathbf{Y}^p\mathbf{g}-\bar{\mathbf{y}}_t\|_2^2+\right.\\ & \left.+\lambda_u\|\mathbf{U}^p\mathbf{g}-\bar{\mathbf{u}}_t\|_2^2+\lambda_g\|\mathbf{g}\|_2^2\right).\end{aligned} \tag{5}$$

Note that the resulting problem (5) is an approximation of (4). Finally, (5) can be written as a quadratic program:

$$\min_{\mathbf{g}} \quad \frac{1}{2}\mathbf{g}^\intercal\mathbf{M}\mathbf{g}-\mathbf{g}^\intercal\left(\mathbf{Y}^{f\intercal}\mathbf{Q}\mathbf{r}_t+\lambda_y\mathbf{Y}^{p\intercal}\bar{\mathbf{y}}_t+\lambda_u\mathbf{U}^{p\intercal}\bar{\mathbf{u}}_t\right), \tag{6}$$

where $\mathbf{M} \in \mathbb{R}^{S\times S}$ is:

$$\mathbf{M} := \mathbf{Y}^{f\intercal}\mathbf{Q}\mathbf{Y}^f+\mathbf{U}^{f\intercal}\mathbf{R}\mathbf{U}^f+\lambda_y\mathbf{Y}^{p\intercal}\mathbf{Y}^p+\lambda_u\mathbf{U}^{p\intercal}\mathbf{U}^p+\lambda_g\mathbf{I}.$$

Since (6) is strongly convex, we can find the optimal value for $\mathbf{g}^*$ by zeroing the gradient of the objective function:

$$\mathbf{g}^{*,t} = \mathbf{M}^{-1}\left(\mathbf{Y}^{f\intercal}\mathbf{Q}\mathbf{r}_t + \lambda_y\mathbf{Y}^{p\intercal}\bar{\mathbf{y}}_t + \lambda_u\mathbf{U}^{p\intercal}\bar{\mathbf{u}}_t\right). \tag{7}$$

The optimal control input is obtained from (7):

$$\mathbf{u}^{*,t} = \mathbf{U}^f\mathbf{M}^{-1}\left(\mathbf{Y}^{f\intercal}\mathbf{Q}\mathbf{r}_t + \lambda_y\mathbf{Y}^{p\intercal}\bar{\mathbf{y}}_t + \lambda_u\mathbf{U}^{p\intercal}\bar{\mathbf{u}}_t\right), \tag{8}$$

from which we input $\mathbf{u}^{*,t}_{[0:m-1]}$ to the unknown system.

As seen from (8), the controller has the form of a dynamic output-feedback law, where the feedback terms are computed using only the offline precollected data.

B. Online feedback data-driven control problem

To satisfy Assumption 1, it is necessary that the precollected input signal has length at least $(m+1)(M+N+n)-1$, cf. [22]. In practice, Assumption 1 might be violated if, for example, less precollected data is available. Precollected data can also be affected by perturbations, e.g. measurement noise. To alleviate these issues, we prefer an online algorithm, where the Hankel matrices $\mathbf{H}_{M+N}(\mathbf{u}^d)$ and $\mathbf{H}_{M+N}(\mathbf{y}^d)$ are updated at each time step with the used control input and the corresponding output measured. Collecting a number of new samples empirically ensures richness of the data and robustness to perturbations.

Note that the precollected and the online data in such an online feedback algorithm belong to different trajectories. For this reason, we must compute the Hankel matrix for each data set separately, then append them in a single matrix [27]. This means that the online adaptation of the matrices $\mathbf{U}^p, \mathbf{U}^f, \mathbf{Y}^p, \mathbf{Y}^f$ can only start at time $t = M+N-1$, after we obtained enough data $\mathbf{u}_0, \mathbf{y}_0, \dots, \mathbf{u}_{M+N-1}, \mathbf{y}_{M+N-1}$ to fill the first Hankel matrix column for the online data set.

The data-driven LQR algorithm is given in Algorithm 1.

Algorithm 1 Online data-driven control algorithm

Input: $\bar{\mathbf{u}}_0, \bar{\mathbf{y}}_0, \mathbf{U}_0^{p,f}, \mathbf{Y}_0^{p,f}, \mathbf{Q}, \mathbf{R}, \lambda_y, \lambda_u, \lambda_g, S = T-M-N+1$.
Output: $\mathbf{u}_t$ for $t = 0, 1, \dots$.

```
1: for $t = 0, 1, \dots, M-1$ do
2:     Randomly select and input $\mathbf{u}_t$ then measure $\mathbf{y}_t$.
3: end for
4: Construct $\bar{\mathbf{u}}_t = \mathbf{u}_{0:M-1}$ and $\bar{\mathbf{y}}_t = \mathbf{y}_{0:M-1}$.
5: for $t = M, M+1, \dots$ do
6:     Solve (6) for $\mathbf{g}^{*,t}$ and obtain (7).
7:     Compute $\mathbf{u}^{*,t} = \mathbf{U}_t^f\mathbf{g}^{*,t}$ and obtain (8).
8:     Input to the system $\mathbf{u}_t = \mathbf{u}^{*,t}_{[0:m-1]}$.
9:     Measure the output $\mathbf{y}_t$.
10:    Update $\bar{\mathbf{u}}_t$ and $\bar{\mathbf{y}}_t$ to be the last $M$ components of $\begin{bmatrix}\bar{\mathbf{u}}_t^\intercal & \mathbf{u}_t^\intercal\end{bmatrix}^\intercal$ and $\begin{bmatrix}\bar{\mathbf{y}}_t^\intercal & \mathbf{y}_t^\intercal\end{bmatrix}^\intercal$, respectively.
11:    if $t = M+N-1$ then
12:        Add $\mathbf{u}_{0:t}$ and $\mathbf{y}_{0:t}$ to the $S+1$'th column of the trajectory Hankel matrices.
13:    else if $t > M+N-1$ then
14:        Set $S = S+1$. Use $\mathbf{u}_t$ to add a new column to both $\mathbf{U}_{t+1}^p$ and $\mathbf{U}_{t+1}^f$ as in (3). Perform the corresponding operations for $\mathbf{Y}_{t+1}^p$ and $\mathbf{Y}_{t+1}^f$ using $\mathbf{y}_t$.
15:    end if
16: end for
```

IV. Homomorphic encryption preliminaries

The decryption primitive of a homomorphic encryption scheme is a homomorphism from the space of encrypted messages, or *ciphertexts*, to the space of unencrypted messages, or *plaintexts*. Homomorphic encryption allows one server (in contrast to multiple servers in secure multiparty computation tools) to evaluate multivariate polynomial functionalities over the encrypted data of the client, while keeping control over the accuracy of the result at a desired security level (in contrast to differential privacy, where there is an accuracy-privacy trade-off). An encryption scheme is called partially homomorphic if it supports the encrypted evaluation of either a linear polynomial or a monomial, somewhat or leveled homomorphic if it supports the encrypted evaluation of a polynomial with a finite degree and fully homomorphic if it supports the encrypted evaluation of arbitrary polynomials. Leveled homomorphic schemes can be turned into fully homomorphic schemes by a bootstrapping operation. The common term for such a multivariate polynomial functionality is *arithmetic circuit* and the logarithm of the degree of the polynomial determines the multiplicative depth of the circuit. We call *multiplicative budget* the multiplicative depth of the deepest circuit that can be evaluated by a specific instance of a leveled homomorphic encryption.

In this paper, we work with a leveled homomorphic encryption scheme. Specifically, we use the version of the CKKS scheme [11], optimized to run on machine word size of 64-bit integer arithmetic [28], [29]. We chose this scheme because it can perform operations on encrypted real numbers with a smaller error than other leveled homomorphic schemes. Each real number is multiplied by a positive integer scaling factor and truncated, as commonly done, but the real advantage of this scheme is that one can remove the extra scaling factor occurring in the result after a multiplication, through a rescaling procedure, at very little error. This manages the magnitude of the underlying plaintexts, which could otherwise cause overflow in a large depth circuit.

A ciphertext's size grows with the number of sequential multiplications it supports. Each ciphertext, respectively plaintext, is characterized by a level and a number of moduli. A new ciphertext (freshly encrypted, rather than obtained as the result of operations on other ciphertexts) has level 0 and has as many moduli as the multiplicative budget. After one multiplication followed by one rescaling procedure, the number of levels increases by 1 and one modulus is dropped.

To avoid some technicalities and parallel the notion of circuit depth, we refer to the multiplicative depth of a ciphertext as being the multiplicative depth of the circuit from which that ciphertext is obtained as a result. Intuitively, the multiplicative depth $d(x)$ of a ciphertext x obtained as a result of a circuit is equal to the number of levels consumed by evaluating that circuit from a fresh ciphertext plus 1.

In the CKKS scheme, each plaintext is a polynomial in the ring of integers of a cyclotomic field with dimension $\mathrm{ringDim}$. This enables the encoding of multiple scalars in a plaintext/ciphertext and performing single instruction multiple data (SIMD) operations, which can bring major computation and storage improvements when evaluating an arithmetic circuit. We can pack up to $\mathrm{ringDim}/2$ values in one plaintext/ciphertext using a Discrete Fourier Transform [11]. Packing can be thought of as the ciphertext having $\mathrm{ringDim}/2$ independent data slots. Abstracting the details away, the SIMD operations that can be supported are addition, element-wise multiplication by a plaintext or ciphertext and data slot permutations that can achieve ciphertext rotations (e.g., used for summing up the values in every slot). In what follows, we will use $+$ and $\odot$ for SIMD addition and multiplication and $\rho(\mathbf{x}, i)$ to denote the row vector $\mathbf{x}$ rotated to the left by i positions ($i < 0$ means rotation to the right).

A scheme has a *security parameter* κ if all known attacks against the scheme take 2^κ bit operations. For CKKS, the security parameter is determined according to the Decisional Ring Learning with Errors problem hardness [30]. We are also interested in the *semantic security* of this encryption scheme, which, at a high level, states that any two ciphertexts are computationally indistinguishable to an adversary that does not have the secret key.

We will denote by $\mathrm{e}_{\mathrm{v}0}(\mathbf{x})$ the encoding of the vector $\mathbf{x}$ followed by trailing zeros into a plaintext and by $\mathrm{e}_{\mathrm{v}*}(\mathbf{x})$ the encoding of the vector $\mathbf{x}$ followed by junk elements (elements whose value we do not care about). We denote by $\mathrm{e}_{\mathrm{vv}}(\mathbf{x})$ the encoding of the repeated vector $\mathbf{x}$: $[\mathbf{x}\,\mathbf{x}\,\mathbf{x}\,\ldots]$. When constructing ciphertexts through encryption, we use similar notations for the encryptions of the corresponding plaintexts encoding vectors: $\mathrm{E}_{\mathrm{v}0}(\mathbf{x})$, $\mathrm{E}_{\mathrm{v}*}(\mathbf{x})$ and $\mathrm{E}_{\mathrm{vv}}(\mathbf{x})$.

V. Co-design of Encrypted Controller

According to the requirements in Section II, the challenges for the encrypted data-driven control can be summarized as:

- The computations are iterative and not readily formulated as (low-depth) arithmetic circuits.
- The problem is computationally intensive: it requires large storage, large matrix inversions and many consecutive matrix multiplications.
- The precision loss due to the private computations should not affect the control performance.

To deal with these challenges, we design an encrypted version of the closed-form solution (7) of the control problem stated in Section II and manipulate the computations involving matrix inverses to reduce the multiplicative depth needed. We employ the CKKS homomorphic scheme described in Section IV to address the precision challenge.

As hinted in the problem reformulation, we approximate problem (4) into problem (6) to simplify the encrypted computations. We prefer the closed-form solution (8) to an iterative solution that would increase the depth with each iteration. However, this closed-form solution involves the encrypted inversion of a matrix, which cannot be generally written as a low-degree polynomial. In the offline feedback problem (6), the complex computations can be all performed offline, leaving only three encrypted matrix-vector multiplications to be performed at each time step, as shown in Section VI. But in the online feedback algorithm, a matrix inversion and many consecutive matrix-vector multiplications are required at every time step. In Section VII, we leverage the special structure of the matrix to be inverted by using Schur's complement [31] to reformulate the inverse computation as some lower multiplicative depth matrix-vector multiplications and one scalar division. To avoid performing the division on encrypted data, the server sends to the client the numerator and denominator of the result, and the division is performed after decryption by the client.

For simplicity, we drop the time t indices in the rest of this section. For step $t = 0$, all the values involved in computing $\mathbf{u}^*$ in (8) are precollected and can be computed offline. This includes the inverse $\mathbf{M}^{-1}$ and other matrix products. However, at the next time step, cf. line 14 in Algorithm 1:

$$\mathbf{U}^{p'} := \begin{bmatrix}\mathbf{U}^p & \mathbf{u}^p\end{bmatrix}, \quad \mathbf{U}^{f'} := \begin{bmatrix}\mathbf{U}^f & \mathbf{u}^f\end{bmatrix}.$$

The last m blocks in $\mathbf{u}^f$ are the values of $\mathbf{u}_t$ at the previous time step. Analogous equations can be written for $\mathbf{Y}^{p'} := \begin{bmatrix}\mathbf{Y}^p & \mathbf{y}^p\end{bmatrix}$ and $\mathbf{Y}^{f'} := \begin{bmatrix}\mathbf{Y}^f & \mathbf{y}^f\end{bmatrix}$. Notice that the matrix:

$$\mathbf{M}' := (\mathbf{Y}^{f'})^\intercal\mathbf{Q}\mathbf{Y}^{f'} + (\mathbf{U}^{f'})^\intercal\mathbf{R}\mathbf{U}^{f'} + \lambda_y(\mathbf{Y}^{p'})^\intercal\mathbf{Y}^{p'} + \\ + \lambda_u(\mathbf{U}^{p'})^\intercal\mathbf{U}^{p'} + \lambda_g\mathbf{I} \quad \in \mathbb{R}^{(S+1)\times(S+1)}$$

is a *rank-1 update* of matrix $\mathbf{M}$. Let

$$\begin{aligned}\mu &:= {\mathbf{y}^f}^\intercal\mathbf{Q}\mathbf{y}^f + {\mathbf{u}^f}^\intercal\mathbf{R}\mathbf{u}^f + \lambda_y{\mathbf{y}^p}^\intercal\mathbf{y}^p + \lambda_u{\mathbf{u}^p}^\intercal\mathbf{u}^p + \lambda_g \\ \mathbf{m} &:= {\mathbf{y}^f}^\intercal\mathbf{Q}\mathbf{Y}^f + {\mathbf{u}^f}^\intercal\mathbf{R}\mathbf{U}^f + \lambda_y{\mathbf{y}^p}^\intercal\mathbf{Y}^p + \lambda_u{\mathbf{u}^p}^\intercal\mathbf{U}^p.\end{aligned} \tag{9}$$

Specifically, $\mathbf{M}'$ will have the following form:

$$\mathbf{M}' = \begin{bmatrix}\mathbf{M} & \mathbf{m}^\intercal \\ \mathbf{m} & \mu\end{bmatrix}.$$

Schur's complement [31] gives an efficient way of computing $\mathbf{M}'^{-1}$ from $\mathbf{M}^{-1}$, by inverting a single scalar m_S:

$$m_S := \mu - \mathbf{m}\mathbf{M}^{-1}\mathbf{m}^\intercal, \tag{10}$$

$$\mathbf{M}'^{-1} = \frac{1}{m_S}\begin{bmatrix} m_S\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{m}^\intercal\mathbf{m}\mathbf{M}^{-1} & -\mathbf{M}^{-1}\mathbf{m}^\intercal \\ -\mathbf{m}\mathbf{M}^{-1} & 1 \end{bmatrix}. \tag{11}$$

Finally, the depth of the arithmetic circuit computing $\mathbf{u}^*_{[0:m-1]}$ can be reduced through reordering of the intermediate operations. Given two circuits f and g with multiplicative depth $d(f)$ and $d(g)$, the multiplicative depth of their product is: $d(fg) = \max(d(f), d(g)) + 1$. The multiplicative depth should not be confounded with the number of multiplications. Judiciously choosing the order in which to perform the multiplications can reduce the multiplicative depth of the result. For example, consider we want to compute $y = x_1x_2x_3x_4$, where $d(x_i) = 1$. If we sequentially perform the multiplications, we obtain $d(y) = 4$. However, if we perform $y = (x_1x_2)(x_3x_4)$, we obtain $d(y) = 3$. We will use this trick in Section VII.

VI. Offline feedback encrypted solution

Recall the closed-form solution of the optimization problem (6). We now compute the multiplicative depth of the ciphertexts of interest for consecutive time steps. First, the ciphertexts corresponding to the precollected input and output measurements, the reference signal and measurements, have a multiplicative depth of 1. This means, $\forall t \geq 0$: $d(\mathbf{U}^{p,f}) = d(\mathbf{Y}^{p,f}) = d(\bar{\mathbf{y}}_t) = d(\mathbf{r}_t) = 1$. Second, we assume that all the quantities obtained offline will have multiplicative depth 1. Since these computations depend only on the offline data and are one-time, expensive secure solutions can be used, e.g., the cloud could perform the encrypted computations, then perform bootstrapping to refresh the ciphertexts [12], [13]. We do not address the offline computations in this paper. This means that the cloud has fresh encryptions of following products: $\mathbf{A}_r := \mathbf{E}\mathbf{U}^f\mathbf{M}^{-1}\mathbf{Y}^{f\intercal}\mathbf{Q} \in \mathbb{R}^{m\times pN}$, $\mathbf{A}_y := \mathbf{E}\mathbf{U}^f\mathbf{M}^{-1}\lambda_u\mathbf{Y}^{p\intercal} \in \mathbb{R}^{m\times pM}$, $\mathbf{A}_u := \mathbf{E}\mathbf{U}^f\mathbf{M}^{-1}\lambda_u\mathbf{U}^{p\intercal} \in \mathbb{R}^{m\times mM}$, so they have multiplicative depth 1, where $\mathbf{E} := \begin{bmatrix}\mathbf{I}_m & \mathbf{0}_{(N-1)m}\end{bmatrix}$. Then:

$$\mathbf{u}_t = \mathbf{A}_r\mathbf{r}_t + \mathbf{A}_y\bar{\mathbf{y}}_t + \mathbf{A}_u\bar{\mathbf{u}}_t. \tag{12}$$

For $t = 0$, we obtain $d(\mathbf{u}_0) = \max(d(\mathbf{r}_0) + 1, d(\bar{\mathbf{y}}_0) + 1, d(\bar{\mathbf{u}}_0) + 1) = 2$. Generalizing:

$$d(\mathbf{u}_t) = d(\bar{\mathbf{u}}_t) + 1, \quad t \geq 0. \tag{13}$$

At time $t+1$, $\bar{\mathbf{u}}_{t+1}$ will be updated by $\mathbf{u}_t$.

If we individually encrypt each element of the quantities in (12) in a separate ciphertext, $\bar{\mathbf{u}}_{t+1}$ will have the multiplicative depth of $\mathbf{u}_t$ and $d(\mathbf{u}_t) = t + 2$. However, it is more efficient from both storage and computation points of view to encode a vector instead of a scalar in a ciphertext and perform the matrix-vector multiplications by a diagonal method, using SIMD operations. This requires a different analysis.

We assume that at the onset of time step t, the cloud server has $\mathrm{E}_{\mathrm{vv}}(\bar{\mathbf{u}}_t), \mathrm{E}_{\mathrm{vv}}(\bar{\mathbf{y}}_t), \mathrm{E}_{\mathrm{vv}}(\mathbf{r}_t)$. It also has $\mathrm{E}_{\mathrm{v0}}(\mathrm{diag}_i\mathbf{A}_r)$, $\mathrm{E}_{\mathrm{v0}}(\mathrm{diag}_i\mathbf{A}_y)$, $\mathrm{E}_{\mathrm{v0}}(\mathrm{diag}_i\mathbf{A}_u)$, i.e., each extended diagonal of the matrices $\mathbf{A}_r, \mathbf{A}_y, \mathbf{A}_u$ is encrypted in a separate ciphertext. From these quantities, the cloud server computes and sends back to the client one ciphertext containing $\mathrm{E}_{\mathrm{v}*}(\mathbf{u}_t)$, such that the client performs only one decryption. After that, the cloud service has to create $\mathrm{E}_{\mathrm{vv}}(\bar{\mathbf{u}}_{t+1})$ from $\mathrm{E}_{\mathrm{vv}}(\bar{\mathbf{u}}_t)$ and $\mathrm{E}_{\mathrm{v}*}(\mathbf{u}_t)$. In order to update $\bar{\mathbf{u}}_{t+1}$, we:

- rotate $\mathrm{E}_{\mathrm{vv}}(\bar{\mathbf{u}}_t)$ by m positions to the left;
- apply a mask that extracts the first $(M-1)m$ positions;
- add the masked and rotated $\mathrm{E}_{\mathrm{v}*}(\mathbf{u}_t)$ by $(M-1)m$ positions to the right such that $\mathbf{u}_t$ lands in the last m positions of $\bar{\mathbf{u}}_{t+1}$;
- repeatedly rotate and add to obtain $\mathrm{E}_{\mathrm{vv}}(\bar{\mathbf{u}}_{t+1})$.

Because of the CKKS encoding through the Discrete Fourier Transform, the masking operation needs to consume a level in order to preserve precision [11]. For $t \geq 1$, equation (13) becomes: $d(\mathbf{u}_t) = d(\bar{\mathbf{u}}_t) + 1 = d(\mathbf{u}_{t-1}) + 2 = 2t + 2$. $\mathrm{E}_{\mathrm{vv}}(\bar{\mathbf{y}}_{t+1})$ can be updated the same way $\mathrm{E}_{\mathrm{vv}}(\bar{\mathbf{u}}_{t+1})$ is updated. However, it is the same cost for the client to encrypt $\mathrm{E}_{\mathrm{vv}}(\mathbf{y}_t)$ and $\mathrm{E}_{\mathrm{vv}}(\bar{\mathbf{y}}_{t+1})$, so it can encrypt and send the latter.

Whenever the allocated multiplicative budget is exhausted, the server can ask the client to send a fresh encryption of $\bar{\mathbf{u}}_t$, at little extra cost (encryption of one packed ciphertext).

Nevertheless, a reasonable and inexpensive option is to ask the client to send along with the encryption of $\mathbf{y}_t$ a fresh encryption of $\mathbf{u}_t$ or of $\bar{\mathbf{u}}_t$ at every time step. This would imply that $d(\mathbf{u}_t) = 2$, i.e., a multiplication budget of only 2 is required for computing the control input for no matter how many time steps.

Proposition 1: Assuming the usage of a semantically secure leveled homomorphic encryption scheme, the encrypted offline data-driven control algorithm achieves privacy with respect to the server.

The proof immediately follows from the fact that the server only receives fresh ciphertexts from the client. Using a standard simulation argument [26], we can construct a simulator for the server that replaces the true messages from the client by random encryptions of the same size. The input-output distribution of such a simulator will be indistinguishable from the input-output distribution of the true protocol.

VII. Online feedback encrypted solution

There are several ways of implementing:

$$\mathbf{u}_t = \mathbf{E}\mathbf{U}^f_t\mathbf{M}^{-1}_t\left((\mathbf{Y}^f_t)^\intercal\mathbf{Q}\mathbf{r}_t + \lambda_y(\mathbf{Y}^p_t)^\intercal\bar{\mathbf{y}}_t + \lambda_u(\mathbf{U}^p_t)^\intercal\bar{\mathbf{u}}_t\right).$$

Some variants optimize multiplicative depth, while others optimize memory consumption. We chose to optimize the multiplicative depth of the circuit, as long as it does not affect precision. This involves more computations at the server, compared to a version with a higher multiplicative depth, but reduces the encryption load at the client, since ciphertexts will be smaller. Moreover, we noticed that having a redundancy in the stored ciphertexts (a separate ciphertext for each column of the Hankel matrices) yields a better run time than storing fewer ciphertexts (one ciphertext for all the unique elements in the Hankel matrix) and processing them repeatedly to extract the relevant information. The main difficulty in making the computations tractable is to astutely pack the ciphertexts and order the products in order to reduce depth, number of operations and storage.

In summary, the outline of the encrypted protocol is as follows. We use the shorter notation $\mathbf{H}(\mathbf{u})_t$ and $\mathbf{H}(\mathbf{y})_t$ for the corresponding block Hankel matrices at time t. Assume without loss of generality that we shift the time axis to the left by $M+N$, such that the trajectory concatenation mentioned in Section III-B is performed before $t=0$. This will simplify the circuit depth expression. For further simplicity, we consider diagonal cost matrices $\mathbf{Q}$ and $\mathbf{R}$ (otherwise the multiplication by these matrices would require more complicated encrypted operations).

For $t=-M-N+1:0$, follow the solution for the offline feedback solution, cf. Section VI.

For $t \geq 1$, the following plaintexts and ciphertexts are stored at the cloud: λ_y and $\mathbf{Q}$ encoded as $\mathrm{e}_{\mathrm{v}0}(\lambda\mathbf{Q}) := \mathrm{e}_{\mathrm{v}0}\left(\begin{bmatrix}\lambda_{y,1} & \dots & \lambda_{y,pM} & q_1 & \dots & q_{pN}\end{bmatrix}\right)$, λ_u and $\mathbf{R}$ as $\mathrm{e}_{\mathrm{v}0}(\lambda\mathbf{R}) := \mathrm{e}_{\mathrm{v}0}(\begin{bmatrix}\lambda_{u,1} & \dots & \lambda_{u,mM} & r_1 & \dots & r_{mN}\end{bmatrix})$, $\mathrm{E}_{\mathrm{v}0}((\mathbf{M}_{t-1}^{-1})_{ij})$ for $j \geq i$, i.e., each entry on and above the diagonal of M_{t-1}^{-1} is separately encrypted in a ciphertext, $\mathrm{E}_{\mathrm{v}*}(\mathrm{col}_i(\mathbf{H}(\mathbf{y})_t))_{i\in\{0,S+t-1\}}$, $\mathrm{E}_{\mathrm{v}*}(\mathrm{col}_i(\mathbf{H}(\mathbf{u})_t))_{i\{0,S+t-1\}}$, i.e., each column of $\mathbf{H}(\mathbf{y})_t$ and $\mathbf{H}(\mathbf{u})_t$, respectively, is separately encrypted in a ciphertext, $\mathrm{E}_{\mathrm{v}0}(\mathbf{y}_t)$, $\mathrm{E}_{\mathrm{v}0}(\mathbf{u}_t)$, following the same reasoning as in the last part of Section VI, and $\mathrm{E}_{\mathrm{v}0}(\bar{\mathbf{y}}_{t-1})$, $\mathrm{E}_{\mathrm{v}0}(\bar{\mathbf{u}}_{t-1})$. To avoid some masking operations, the cloud also stores $\mathrm{E}_{\mathrm{v}*}((\mathbf{U}_t^f)_{ij})$ for $i \in \{0,\dots,m-1\}$ and $j \in \{0,\dots,S+t-1\}$.

Following (9)–(11), the cloud service performs the encrypted computations shown in the Appendix in equations (15) and (16). The cloud obtains $\mathrm{E}_{\mathrm{v}*}(D\mathbf{u}_t)$ and $\mathrm{E}_{\mathrm{v}*}((N\mathbf{u}_t)$, where $D\mathbf{u}_t$ is the denominator and $N\mathbf{u}_t$ the numerator of $\mathbf{u}_t = \mathbf{u}_{[0:m-1]}^{*,t}$. To further reduce the total multiplicative depth, the server asks the client to send back an encryption of the inverse of the denominator: $\mathrm{E}_{\mathrm{v}0}(1/D\mathbf{u}_t)$, so that the computation at the next time step is started with denominator 1. From (15), (16), we obtain that the multiplicative depth of computing the control input is given by:

$$d(D\mathbf{u}_t) = 2(t-1)+5, \quad d((N\mathbf{u}_t) = 2(t-1)+7.$$

Considerations for continuous running. An important aspect to be clarified is what happens after the multiplication budget is exhausted. The options are:

(i) Restore the initial precollected Hankel matrices which bypasses a refreshing step altogether. Advantages: no extra computations needed. Disadvantages: this causes oscillations in the control actions.

(ii) Stop adding new information to the matrices. Advantages: no extra computations needed. Disadvantages: the multiplicative budget has to be large enough such that *enough* samples are collected.

(iii) Pack the matrix into a single ciphertext as in (14) ($S' = S+t-1$) and ask the client to refresh it. Advantages: the server can continue collecting values for any desired time, with no extra depth. Disadvantages: the client has to decrypt, encrypt and send another ciphertext; the rotation keys necessary for packing can occupy a lot of storage.

$$\begin{aligned}\mathrm{E}_{\mathrm{v}0}(\mathbf{M}_t^{-1}) &= \mathrm{E}_{\mathrm{v}0}(\begin{bmatrix}(\mathbf{M}_t^{-1})_{00} & \dots & (\mathbf{M}_t^{-1})_{S'S'}\end{bmatrix}) \\ &= \sum_{i=0}^{S'}\sum_{j=i}^{S'} \rho\Big(\mathrm{E}_{\mathrm{v}*}(\mathbf{M}_t^{-1})_{ij} \odot \mathbf{e}_0, iS' - \frac{i(i-1)}{2} + j\Big).\end{aligned} \tag{14}$$

(iv) Bootstrap the ciphertext of $\mathbf{M}_t^{-1}$. The computation advancements regarding the bootstrapping procedure suggest that it is likely to locally resolve the refreshing step. Advantages: the server can continue collecting values for any desired time without the client's intervention. Disadvantages: the initial multiplication budget has to be larger to also allow for the bootstrapping circuit.

In the solution we implemented, we chose option (iii). This gives us flexibility on the maximum multiplicative depth of the circuit, which will now be $2(t_{refresh}-1)+7$, at little extra cost for the client. Since $\mathbf{M}_t^{-1}$ is symmetric, we keep $(S+t-1)(S+t)/2$ elements encrypted. The number of ciphertexts the server can pack $\mathbf{M}_t^{-1}$ into is $\lceil (S+t-1)(S+t)/\mathrm{ringDim} \rceil$. The client only has to decrypt, re-encrypt and send back this number of ciphertexts. The server then uses one extra level to perform the reverse of (14) to unpack $\mathrm{E}_{\mathrm{v}0}(\mathbf{M}_t^{-1})$ into ciphertexts $\mathrm{E}_{\mathrm{v}0}((\mathbf{M}_t^{-1})_{ij})$. These multiplications can be absorbed in the same initial multiplicative depth.

Proposition 2: Assuming the usage of a semantically secure leveled homomorphic encryption scheme, the encrypted online data-driven control algorithm achieves privacy with respect to the server.

The proof is similar to the proof of Proposition 1, since the server only receives fresh encryptions of the private data.

VIII. Numerical results

We considered a zone temperature control problem derived from [32], with sampling time of $T_s = 420$ seconds. The system parameters are: $n=4, m=1, p=1, M=4, N=10, T=40$. We add a zero mean Gaussian process noise with covariance $0.001\mathbf{I}$ and a zero mean Gaussian measurement noise with covariance $0.01\mathbf{I}$. We choose the cost matrices and regularization terms $\mathbf{Q}=\mathbf{I}, \mathbf{R}=10^{-5}\mathbf{I}, \lambda_g = 1, \lambda_y = \lambda_u = 10$. For the offline data collection, we assume a different initial point than for the online computation, and randomly sample the offline input trajectory so that the corresponding output trajectory lies in the interval $[12,16]$. The M, N and T parameters mean that we start from 27 columns in the Hankel matrices, which, due to noise, results in suboptimal performance and slow convergence for the offline feedback control–see Figure 1. We collect data online for 15 more time steps, which means adding another 15 columns to the Hankel matrices. Afterwards, we run the system with fixed gains. Figure 1 shows the performance of the setpoint tracking with these parameters.

We implemented[1] the encrypted solutions proposed in this paper using the PALISADE library [16]. We use a ring

[1]https://github.com/andreea-alexandru/private-data-driven-control

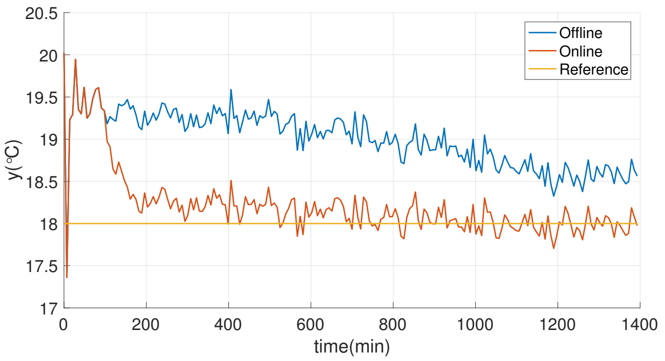

Fig. 1: Tracking performance of the unencrypted online versus offline feedback in the presence of small noise. The performance is for one typical random sample for one day of simulation. To make the comparison meaningful, we use the same noise sequence for both schemes.

dimension of 2^{12} and 21 moduli (for a refresh step after 8 collected samples). The resulting ciphertext modulus of a fresh ciphertext is of 1060 bits. We chose this ring dimension such that the computation is contained in the 8 GB of RAM of a standard laptop with Intel Core i7 on which the simulation was run using 8 threads. We note that this toy-sized problem gives a security parameter of 47 bits, according to the online estimator [30], which is not a security parameter recommended in practice. For a typical security parameter of 80 bits, a ring dimension of 2^{15} is required. For the fully secure implementation, we recommend using a more powerful machine.

The run times for the encrypted computation of the online feedback algorithm are given in Figure 2, where three different phases are depicted. The offline time (for key generation and encryption) for the whole simulation is 87 seconds.

The first online phase is the trajectory concatenation phase, as described in Section III-B, from time step 0 to time step $t = M + N - 1 = 13$, where the server computes the encrypted control action only with the precollected data, which is very efficient, despite computing on the maximum number of moduli, resulting in a total computation time for one time step of under 1.2 seconds.

The second phase is the online collection of new input-output samples, which implies modifying the Hankel matrices and computing the inverse matrix via the Schur complement trick at every time step $\mathbf{M}_t^{-1}$ from step 14. This phase is also split in two parts depending on the refreshing time. At the established refresh time ($t = 20$), the server packs the matrix $\mathbf{M}_t^{-1}$ into one ciphertext, sends it to the client to re-encrypt it with the maximum number of moduli, and unpacks it back into component-wise ciphertexts. This justifies the increase in the client and server computation time at step 21 compared to the previous time step. The substantial increase in computation time from at 21 (308.8 seconds) compared to step 14 (173.7 seconds) is given by the fact that the server has to deal with more collected samples ciphertexts than in the beginning of phase two. The computation time decreases as the number of levels increases (the ciphertext

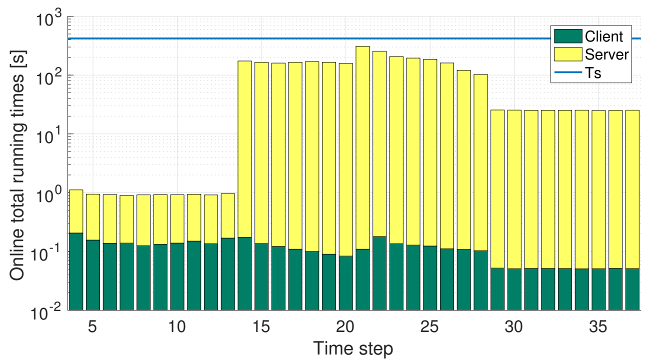

Fig. 2: Running times for the computations performed at the client and the cloud server for the encrypted online control algorithm. The plot is semi-logarithmic and the amounts of time required by the client and the cloud server are stacked. The maximum time required for the total computations for a time step is less than the sampling time $Ts = 420$ seconds.

size decreases).

The third phase corresponds to the computations after stopping the collection of new samples, which starts from time step 29 and can go for the rest of the desired simulation time. The third phase is more computationally intensive than the first, because in the first phase, we use an offline generated diagonal packing with corresponding SIMD operations. This packing is more expensive to achieve online, at the onset of phase three, so we instead use less efficient matrix-vector multiplications. Nevertheless, the running time required for computing $\mathbf{u}_t$ at one time step is around 25 seconds.

By scaling the $\mathbf{M}_0^{-1}$ matrix at the client and appropriately modifying the computations so that this scaling does not affect the result, we improve the precision of the encrypted result. Specifically, we obtain a maximum magnitude of 0.012°C and an average of 0.005°C for the difference between the unencrypted output and the encrypted output, and 0.17 kWatts, respectively 0.07 kWatts, for the output, during the online simulation from Figure 2. This difference is introduced by the accumulation of errors due to the encrypted computations. One can obtain bounds on these errors, but these bounds are overly conservative. Increasing the plaintext modulus offers a solution to precision increase.

IX. Conclusions and Future Work

We proposed a privacy-preserving online data-driven control algorithm designed to be encryption-friendly, i.e., allow both accurate setpoint tracking and efficient encrypted computations. We achieve this by choosing a regularized convex formulation of the control optimization problem with a closed form solution for the control input; this solution can be computed using a low-depth arithmetic circuit, using the Schur's complement formula for matrix inversion. The client encrypts its measurements with a leveled homomorphic encryption scheme and sends them to a cloud server, which returns the encrypted control input.

For future research, we aim to prove closeness of the approximation problem that we proposed in Section III to the classic LQR problem and to introduce hard constraints. We also note that the behavioral framework might not be the only encryption-friendly method and we would like to explore other methods as well, e.g. adaptive control techniques. Finally, we plan to further optimize the encrypted code to reduce the storage and amount of computations performed.

References

[1] S. Kejriwal and S. Mahajan, "Smart buildings: How iot technology aims to add value for real estate companies," *Deloitte Center for Financial Services*, 2016.

[2] 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.

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

[4] M. Schulze 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, 2018.

[5] A. B. Alexandru, M. Morari, and G. J. Pappas, "Cloud-based MPC with encrypted data," in *Proceedings of the 57th Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 5014–5019.

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

[7] J. Suh and T. Tanaka, "SARSA (0) reinforcement learning over fully homomorphic encryption," *arXiv preprint arXiv:2002.00506*, 2020.

[8] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Department of Computer Science, Stanford University, 2009.

[9] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Intl. Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2011, pp. 505–524.

[10] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Annual Cryptology Conference*. Springer, 2013, pp. 75–92.

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

[12] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *Intl. Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 360–384.

[13] H. Chen, I. Chillotti, and Y. Song, "Improved bootstrapping for approximate homomorphic encryption," in *Intl. Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 34–54.

[14] S. Halevi and V. Shoup, "Algorithms in HElib," in *Annual Cryptology Conference*. Springer, 2014, pp. 554–571.

[15] "Microsoft SEAL," https://github.com/Microsoft/SEAL, Oct. 2019, Microsoft Research, Redmond, WA.

[16] "PALISADE Lattice Cryptography Library (release 1.9.1)," https://palisade-crypto.org/, Mar. 2020.

[17] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption library," August 2016, https://tfhe.github.io/tfhe/.

[18] L. Ljung, *System Identification: Theory for the User*. Prentice Hall, 1999.

[19] P. Van Overschee and B. De Moor, *Subspace identification for linear systems: Theory–Implementation–Applications*. Springer Science & Business Media, 2012.

[20] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. De Moor, "A note on persistency of excitation," *Systems & Control Letters*, vol. 54, no. 4, pp. 325–329, 2005.

[21] C. De Persis and P. Tesi, "On persistency of excitation and formulas for data-driven control," in *Proceedings of the 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 873–878.

[22] J. Coulson, J. Lygeros, and F. Dörfler, "Data-enabled predictive control: In the shallows of the DeePC," in *Proceedings of the 18th European Control Conference (ECC)*. IEEE, 2019, pp. 307–312.

[23] J. Berberich, J. Köhler, M. A. Muller, and F. Allgower, "Data-driven model predictive control with stability and robustness guarantees," *IEEE Transactions on Automatic Control*, 2020.

[24] C. Yu, L. Ljung, and M. Verhaegen, "Identification of structured state-space models," *Automatica*, vol. 90, pp. 54–61, 2018.

[25] I. Markovsky and P. Rapisarda, "Data-driven simulation and control," *Intl. Journal of Control*, vol. 81, no. 12, pp. 1946–1959, 2008.

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

[27] H. J. van Waarde, C. De Persis, M. K. Camlibel, and P. Tesi, "Willems fundamental lemma for state-space systems and its extension to multiple datasets," *IEEE Control Systems Letters*, vol. 4, no. 3, pp. 602–607, 2020.

[28] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," in *Intl. Conference on Selected Areas in Cryptography*. Springer, 2018, pp. 347–368.

[29] S. Halevi, Y. Polyakov, and V. Shoup, "An improved RNS variant of the BFV homomorphic encryption scheme," in *Cryptographers Track at the RSA Conference*. Springer, 2019, pp. 83–105.

[30] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *Journal of Mathematical Cryptology*, vol. 9, pp. 169–203, 2015, https://lwe-estimator.readthedocs.io/en/latest.

[31] F. Zhang, *The Schur complement and its applications*. Springer Science & Business Media, 2006, vol. 4.

[32] A. W. M. van Schijndel, "Integrated heat air and moisture modeling and simulation," Ph.D. dissertation, T. U. Eindhoven, 2007.

Appendix

The cloud service locally obtains the quantities in (15) and (16). EvalSum sums the slots inside a ciphertext. Let $\mathbf{v}_t := (\mathbf{m}_t\mathbf{M}_{t-1}^{-1})^\intercal$ and $S' := S + t - 1$, for $t \geq 1$.

$$
\begin{aligned}
&\mathrm{E}_{\mathrm{v}0}(\bar{\mathbf{y}}_t)=\rho(\mathrm{E}_{\mathrm{v}0}(\bar{\mathbf{y}}_{t-1}),p)+\rho(\mathrm{E}_{\mathrm{v}0}(\mathbf{y}_t),-p(M-1))\\
&\mathrm{E}_{\mathrm{v}0}(\bar{\mathbf{u}}_t)=\rho(\mathrm{E}_{\mathrm{v}0}(\bar{\mathbf{u}}_{t-1}),m)+\rho(\mathrm{E}_{\mathrm{v}0}(\mathbf{u}_t),-m(M-1))\\
&\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_{S'}\mathbf{H}(\mathbf{y})_t)=\rho(\mathrm{col}_{S'-1}\mathbf{H}(\mathbf{y})_t,p)+\\
&\quad+\rho(\mathrm{E}_{\mathrm{v}0}(\mathbf{y}_t),-p(M+N-1))\\
&\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_{S'}\mathbf{H}(\mathbf{u})_t)=\rho(\mathrm{col}_{S'-1}\mathbf{H}(\mathbf{u})_t,m)+\\
&\quad+\rho(\mathrm{E}_{\mathrm{v}0}(\mathbf{u}_t),-m(M+N-1))
\end{aligned}
\tag{15}
$$

$$
\begin{aligned}
&\mathrm{E}_{\mathrm{v}*}((\mathbf{U}_t^f)_{iS'})=\rho(\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_{S'}\mathbf{H}(\mathbf{u})_t),mM+i),\ i\in\{0,\dots,m-1\}\\
&\mathrm{E}_{\mathrm{v}0}(\bar{\mathbf{y}}_t,\mathbf{r}_t)=\mathrm{E}_{\mathrm{v}0}(\bar{\mathbf{y}}_t)+\rho(\mathrm{E}_{\mathrm{v}0}(\mathbf{r}_t),-pM).\\
&\mathrm{E}_{\mathrm{v}*}(\mu_t)=\mathrm{EvalSum}(\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_{S'}\mathbf{H}(\mathbf{y})_t\odot\mathrm{e}_{\mathrm{v}0}(\lambda\mathbf{Q})\odot\\
&\quad\odot\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_{S'}\mathbf{H}(\mathbf{y})_t)+\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_{S'}\mathbf{H}(\mathbf{u})_t\odot\mathrm{e}_{\mathrm{v}0}(\lambda\mathbf{R})\odot\\
&\quad\odot\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_{S'+1}\mathbf{H}(\mathbf{u})_t)+\lambda_g\\
&\mathrm{E}_{\mathrm{v}*}(\mathbf{m}_t)_i=\mathrm{EvalSum}(\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_{S'}\mathbf{H}(\mathbf{y})_t\odot\mathrm{e}_{\mathrm{v}0}(\lambda\mathbf{Q})\odot\\
&\quad\odot\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_i\mathbf{H}(\mathbf{y})_t)+\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_{S'}\mathbf{H}(\mathbf{u})_t)\odot\mathrm{e}_{\mathrm{v}0}(\lambda\mathbf{R})\odot\\
&\quad\odot\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_i\mathbf{H}(\mathbf{u})_t),\quad i\in\{0,\dots,S'-1\}\\
&\mathrm{E}_{\mathrm{v}*}(D\mathbf{u}_t)=\mathrm{E}_{\mathrm{v}*}(m_{S,t})=\mathrm{E}_{\mathrm{v}*}(\mu_t)-\\
&\quad-\sum_{i=0}^{S'-1}\sum_{j=0}^{S'-1}\mathrm{E}_{\mathrm{v}*}(\mathbf{M}_{t-1}^{-1})_{ij}\odot(\mathrm{E}_{\mathrm{v}*}(\mathbf{m}_t)_i\odot\mathrm{E}_{\mathrm{v}*}(\mathbf{m}_t)_j)\\
&\mathrm{E}_{\mathrm{v}*}(\mathbf{v}_t)_j=\sum_{i=0}^{S'-1}\mathrm{E}_{\mathrm{v}*}(\mathbf{m}_t)_i\odot\mathrm{E}_{\mathrm{v}*}(\mathbf{M}_{t-1}^{-1})_{ji}\\
&\mathrm{E}_{\mathrm{v}*}(\mathbf{v}_t\mathbf{v}_t^\intercal)_{ij}=\mathrm{E}_{\mathrm{v}*}(\mathbf{v}_t)_i\odot\mathrm{E}_{\mathrm{v}*}(\mathbf{v}_t)_j,\quad i,j\in\{0,\dots,S'-1\}\\
&\mathrm{E}_{\mathrm{v}*}(N\mathbf{M}_t^{-1})=\begin{bmatrix}\mathrm{E}_{\mathrm{v}*}(m_{S,t}\mathbf{M}_{t-1}^{-1}+\mathbf{v}_t\mathbf{v}_t^\intercal & \mathrm{E}_{\mathrm{v}*}(-\mathbf{v}_t)\\ \mathrm{E}_{\mathrm{v}*}(-\mathbf{v}_t^\intercal) & 1\end{bmatrix}\\
&\mathrm{E}_{\mathrm{v}0}(\mathbf{z}_t)_i=\mathbf{e}_0\odot\mathrm{EvalSum}(\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_i\mathbf{H}(\mathbf{y})_t)\odot\mathrm{e}_{\mathrm{v}0}(\lambda\mathbf{Q})\odot\\
&\quad\odot\mathrm{E}_{v0}(\bar{\mathbf{y}}_t,\mathbf{r}_t)+\mathrm{E}_{\mathrm{v}0}(\mathrm{col}_i\mathbf{H}(\mathbf{u})_t)\odot\mathrm{e}_{\mathrm{v}0}(\lambda\mathbf{R})\odot\mathrm{E}_{\mathrm{v}0}(\bar{\mathbf{u}}_t))\\
&\quad i\in\{0,\dots,S'\}\\
&\mathrm{E}_{\mathrm{v}*}(N\mathbf{u}_t)=\sum_{k=0}^{m-1}\rho\Big(\sum_{i=0}^{S'}\sum_{j=0}^{S'}\Big(\mathrm{E}_{\mathrm{v}*}(\mathbf{U}_t^f)_{ki}\odot\mathrm{E}_{\mathrm{v}0}(\mathbf{z}_t)_j\Big)\odot\\
&\quad\odot\mathrm{E}_{\mathrm{v}*}(N\mathbf{M}_t^{-1})_{ij},-k\Big)=\mathrm{E}_{\mathrm{v}*}((\mathbf{U}_t^fN\mathbf{M}_t^{-1}\mathbf{z}_t)_{[0:m-1]}).
\end{aligned}
\tag{16}
$$