Secure and Resilient Control Design for Cloud Enabled Networked Control Systems

Zhiheng Xu and Quanyan Zhu

Department of Electrical and Computer Engineering, Polytechnic School of Engineering
New York University, Brooklyn, 11201, USA. E-mail: zx383, qz494@nyu.edu.

ABSTRACT

Cloud computing enables resource-constrained Networked Control Systems (NCSs) to outsource heavy computations to a cloud server with massive computational resources. However, Cloud-enabled NCSs (CE-NCSs) introduce new challenges arising from the trustworthiness of the cloud and the cyber-physical connections between the control system and the cloud. To address these concerns, this paper presents a secure and resilient mechanism, which employs customized cryptographic tools to encrypt the data of a control problem and develops verification methods to guarantee the integrity of the computational results from the cloud. In addition, our design enables a Switching Mode Mechanism (SMM) to provide resiliency to the NCSs when the system successively fails to receive correct control inputs from the cloud. We demonstrate that the mechanism can achieve the data confidentiality and integrity, guarantee the stability, and enhance the resiliency. Finally, an Unmanned Aerial Vehicle (UAV) example is used to corroborate these properties.

Categories and Subject Descriptors

K.6.5 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Security and Protection—*unauthorized access*; C.3 [**SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS**]: Process control systems

Keywords

Cloud-enabled networked control system, cyber-physical system, security, resiliency

1. INTRODUCTION

Feedback control systems are ubiquitous in modern critical infrastructures (e.g, power grids and transportation) as well as in civil lives (e.g., elevators and robots). In the classical design of control systems, all system components, including sensors, controller, actuator, and plant, are embedded in a local unit. This practice leads to a high cost of installation, communication constraints, and lack of flexibility. The advent of Information and Communication technologies (ICTs) greatly facilitates the integration of advanced technologies into control systems. Feedback loops are integrated with wireless communications between different system components as a Networked Control System (NCS), which enables a broad range of modern control system applications, such as remote surgery [23], Unmanned Aerial Vehicles (UAVs) [25], and smart grids [8].

With an increasing amount of information gathered from networked sensors and a large scale of the system, an NCS requires a significant amount of computational resources to achieve the control objectives. As a result, resource-constrained control systems restrain their control performance using a conventional design. For example, in the case of nano and micro robots, the size of physical part of the robots is often limited. This property constrains the resources of the hardware equipped in these robots, making it impractical for the robots to handle complicated sensing information, such as video and image information. The recent developments of Cloud Computing Technologies (CCTs) makes it possible to resolve resource-constrained issues of NCSs. CCTs provide massive computing resources as well as storage resources to NCSs, and extensively enhance their performance by migrating local computations to a cloud. The integration of CCTs with NCSs leads to Cloud-enabled Networked Control Systems (CE-NCSs), bringing the following revolutionary features to control systems: large-scale real-time data processing [4], massive paralleled computation and signal processing [2], and resource sharing. Another advantage of outsourcing complicated computations to a cloud is that the control system can reduce its energy cost, enabling the system to conduct a mission in a longer duration [27].

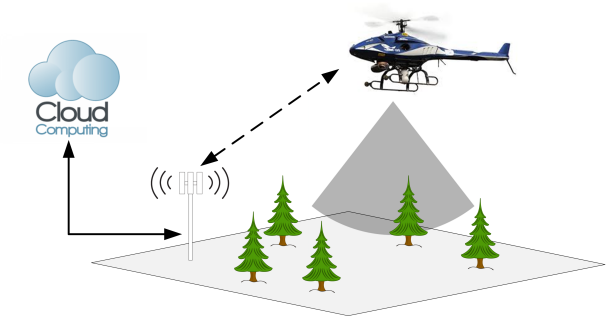

Figure 1: An example of the Cloud-enabled NCSs: An unmanned helicopter conducts a search mission and outsources its computations to a cloud.

Fig. 1 illustrates an example of UAV that conducts a searching mission. As the UAV is resource-constrained, it can improve its performance by outsourcing the computations of the searching and

CPS-SPC'15, October 16, 2015, Denver, Colorado, USA.

DOI: http://dx.doi.org/10.1145/2808705.2808708.

control problems to the cloud. One real application of CE-NCSs is the Google self-driving car [13]. The self-driving car, connected to satellites to collect and update the maps and images, can upload the sensing information to a cloud to achieve accurate positioning. Another example is the cloud-based formation control [27]. The authors have demonstrated that the integrated robotic system with a cloud simplifies the hardware of the robots as the robots outsource heavy computations to a cloud. In [24], Pandey et al. have developed a dynamic mechanism to integrate resource-constrained networked robots with cloud, and Autonomous Underwater Vehicles (AUVs) are used as an example to demonstrate the benefits of collaboration between the AUVs and cloud resources.

1.1 Security Challenges and Solutions

Despite the advantages of CE-NCSs, new security challenges arise from both the cyber and the physical parts of CE-NCSs. The first challenge is the confidentiality and integrity of the data transmitted between a cloud and NCSs. One the one hand, outsourcing computations to a cloud may expose the private information (e.g., parameters of system model and sensor information) of the control system to cyber attackers [29]. On the other hand, cyber adversaries can modify the data between the cloud and the control system to disrupt data integrity. The second one comes from the control system, which requires feedback control inputs at every sampling time to guarantee stability. The lack of control inputs can lead to instability of NCSs. The third one is efficiency. The amount of local computation, including encryption, decryption and verification, performed by the control system, should be less than that of solving the original control problems [18].

To address these challenges, we propose a secure and resilient control design for CE-NCSs. In our design, the NCSs encrypt the data prior to the computation in the cloud, and verify the solutions from the cloud. In standard encryptions (e.g., symmetric or asymmetric encryption), senders and receivers share a public key to encrypt or decrypt the exchanging data [26]. Sharing a public key makes these encryptions infeasible in CE-NCSs because exchanging the key is costly, and exposing the key to an unreliable cloud can be vulnerable. In the proposed encryption, however, only the controllers of NCSs process the encryption and decryption, which avoids exchanging the key between the NCSs and the cloud. To guarantee the stability, and enhance its resilience to cyber attacks, we design a Switch Mode Mechanism (SMM) with three working modes: Cloud Mode, Buffer Mode, and Safe Mode. The controller of the CE-NCSs can switch between these modes when unanticipated events occur.

To verify our mechanism and algorithm, we use a UAV control system as an example to corroborate the analytical results in different attack models. The result demonstrates that without verifying the solution, the adversary deviates the UAV by sending a wrong solution. In addition, without SMM, the attackers can disrupt the UAV by sending sequential wrong solutions. After that, we show that the proposed mechanism can protect the UAV from those attacks.

1.2 Related Works

NCSs have been well investigated for the past two decades. Many techniques, such as distributed Model Predictive Control (MPC) [20], event-trigger control [16], and H_∞ robust control [19], have been used to address the control challenges in NCSs. This work deals with the emerging control issues in CE-NCSs. We can apply the current methods to formulate control problems for CE-NCS, but these methods are not sufficient to meet the security issues in CE-NCSs.

Our work is also related to cloud robotics and automation [13], which investigates object recognition, robot navigation and path planning. These are practical examples of CE-NCSs. Many other people also develop applications by using the architecture of CE-NCSs to enhance efficiency, e.g., cloud-based transportation system [9] and cloud-based grasping system [11]. Nevertheless, very little work has been done to provide the fundamental understanding of CE-NCSs. In addition, the security challenges are often ignored.

Another related body of work is cloud outsourcing computation. Recent years, we have witnessed a significant growth in this field. Recent works have developed secure protocols for outsourcing scientific computations, such as linear programming problems [29], large matrix inversion problems [18], large matrix determinant problems [17]. They develop cryptographic mechanisms to achieve confidentiality and integrity in the outsourcing computation. However, CE-NCSs have unique cyber-physical features, because their physical control components are tightly integrated with other counterparts. The protocols designed for outsourcing computation are not sufficient to address these features. Wang et al. have developed a matrix-disguising method to achieve data privacy in outsourcing computations of a quadratic programming problem to unreliable parties [30]. Our work in addition verifies the solution from the cloud to achieve data integrity.

Many methods have been designed to deal with cyber-physical security issues. One important approach is to apply game theoretic tools. Game theory has shown promise in a wide range of security issues as it can capture and model the strategic interactions between the attackers and defenders [22]. In [32], Zhu et al. apply a game-theoretical framework to improve the resilience of NCSs in the face of attacks. The framework constructs two game models to capture the cyber-physical nature of NCSs. The physical-layer game framework models the interactions between the physical system and disturbance, while the cyber-layer game framework capture the interactions between an attacker and a defender.

Table 1: Related work and the scope of their solutions

References	Applications of Cloud	Cyber Security	Applications of NCSs
Liu et al. [20], Lehmann et al. [16], Li and Shi [19].	No	No	Yes
Kehoe et al. [13, 12], Chen et al. [7].	Yes	No	Yes
Lei et al. [18, 17], Wang and Ren [29], Wang and Li [30].	Yes	Yes	No

In Table I, we summarize the existing literatures and the scope of their solutions. Due to the unique cyber-physical property, the existing approaches cannot be directly applied to tackle the security issues in CE-NCSs. We need to take into account both control and security issues, and design security mechanism in a holistic fashion.

1.3 Main Contributions

As CE-NCSs lie at the intersection of cyber and control systems, this paper develops a secure mechanism by using both cryptographic and control tools. The unique physical nature of control systems differentiate our work from other secure protocols for cloud outsourcing computation. We summarize our contributions as follows:

a) We first apply Model Predictive Control (MPC) to formulate a control problem for NCSs. By developing crypto-

graphic mechanisms, we encrypt the quadratic problem associated with MPC prior to the computations in the cloud, and propose an efficient method to verify its solutions from the cloud. We show that no false solutions can succeed in the verification with a non-negligible probability.

b) Secondly, to guarantee stability and enhance resiliency, we design a switching mechanism SMM using an event-triggered MPC scheme and H_∞-optimal control. Under this mechanism, the NCSs switch to a Buffer Mode when control inputs are unavailable from the cloud for a short duration, or switch to a Safe Mode when control inputs are unavailable for a long duration. We demonstrate that the SMM guarantees the stability of CE-NCSs.

c) Finally, both analytical and experimental results show that our mechanism can boost efficiency for NCSs. The experimental results also corroborate that the SMM can guarantee stability for CE-NCSs when control inputs are unavailable for either a short or long period of time.

1.4 Organization

The rest of the paper is organized as follows: Section 2 presents the problem formulation, three cloud attack models, the design goals, and the framework of the proposed mechanism. In Section 3, we develop specific strategies and techniques to achieve data confidentiality and integrity. In Section 4, we propose a switching mechanism to deal with availability issues. Both theoretical analysis and experiment results are presented in Section 5. Finally, Section 6 concludes this paper.

2. PROBLEM STATEMENTS

Each CE-NCS has two layers: cyber layer and physical layer. The cyber layer consists of wireless communications and a cloud, while the physical layer incorporates a plant, actuators, and sensors, and a controller. The integration of the controller with the cloud constitutes a cloud-based controller. Fig. 2 illustrates a feedback architecture of a CE-NCS. The controller of a CE-NCS is structurally different from other general NCSs. Instead of solving an off-line control problem, the controller of a CE-NCS formulates a dynamic optimization problem based on the sensor data, and outsources the computations of the control decisions to a cloud. After receiving the solutions from the cloud, the controller sends them to the actuators of the physical system.

To describe in detail the architecture of a CE-NCS, we first introduce the physical layer control problem. In this work, we present a discrete-time linear system to capture the dynamics of the physical plant and use Model Predictive Control (MPC) framework to design optimal control to stabilize the system. The computations of the MPC control inputs are outsourced to the cloud, which can be subject to adversarial attacks. To this end, the second part of this section presents three attack models on the CE-NCS, and outline the design goals and framework of the proposed mechanism.

2.1 System Dynamics and MPC Algorithm

MPC has been widely used in many domains of industries and civil applications, such as process control of chemical plants and oil refineries, energy systems in building, and autonomous vehicles [1, 31]. MPC is a model-based control strategy that uses a system model to predict the future behaviors of the system to establish appropriate control inputs [28]. To achieve prediction, MPC strategy provides a moving finite-horizon problem based on the system model, and control inputs can be computed by solving this problem at each sampling instant. One advantage of MPC is that it

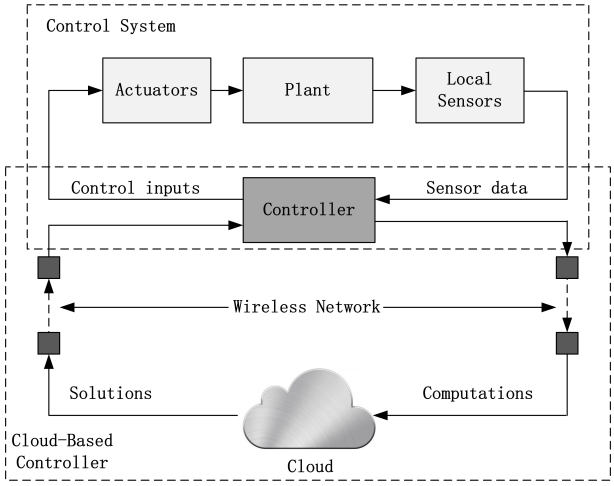

Figure 2: Architecture of a CE-NCS: The integrated system consists of a plant, sensors, actuators, and a cloud-based controller. The cloud-based controller is composed of a local controller, a wireless network, and a cloud. The network is deployed to transmit data of computations to the cloud and the control solutions from the cloud. The controller aims to stabilize the plant and achieve system objectives.

allows the system to operate constraints on control inputs or system states [15], while it is difficult for the other control strategies, e.g., Linear Quadratic Regulator (LQR) and H_∞-optimal control, to handle constraints. The second advantage is that MPC enables an on-line design, which can deal with disturbance in real time, while LQR and H_∞-optimal control are off-line designs. In addition, H_∞-optimal control considers the worst-case disturbance, leading to more conservative and expansive control inputs. A main challenge of MPC is that the controller needs to solve a optimization problem at each sampling time, resulting in a higher computational complexity. This motivates us to introduce a CE-NCS framework to outsource the computations of the MPC problem to the cloud.

To apply MPC, we use a discrete-time linear system model to describe the system dynamics of a control system, given by

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \tag{1}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^{n\times 1}$ is the state vector of the NCS, $\boldsymbol{x}(0) = \boldsymbol{x}_0 \in \mathbb{R}^{n\times 1}$ is the given initial state value, $\boldsymbol{u}(k) \in \mathbb{R}^{l\times 1}$ is the control input, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times l}$ are constant matrices.

At each sampling time k, a finite-horizon MPC problem $\mathcal{P}$ is a minimization problem, which is given as follows:

$$\begin{aligned}\mathcal{P}: \ & \min_{\boldsymbol{U}(k)} J(\boldsymbol{x}(k), \boldsymbol{U}(k)) \\ & = \sum_{\tau=0}^{N-1} \Big(\|\hat{\boldsymbol{x}}(k+\tau|k)\|^2 + \eta \|\hat{\boldsymbol{u}}(k+\tau|k)\|^2 \Big), \end{aligned} \tag{2}$$

subject to

$$\begin{aligned}&\hat{\boldsymbol{x}}(k+\tau+1|k) = \boldsymbol{A}\hat{\boldsymbol{x}}(k+\tau|k) + \boldsymbol{B}\hat{\boldsymbol{u}}(k+\tau|k), \\ &\hat{\boldsymbol{x}}(k|k) = \boldsymbol{x}(k), \ \hat{\boldsymbol{u}}(k+\tau|k) \in \boldsymbol{\mathcal{U}}, \\ &\boldsymbol{U}(k) = [\hat{\boldsymbol{u}}^T(k|k), \hat{\boldsymbol{u}}^T(k+1|k), \ldots, \hat{\boldsymbol{u}}^T(k+N-1|k)]^T,\end{aligned}$$

where $J : \mathbb{R}^{n\times 1} \times \mathbb{R}^{lN\times 1} \to \mathbb{R}$ is the objective function, η is a tuning parameter for the desired control performance, and $\boldsymbol{U}(k) \in \mathbb{R}^{lN\times 1}$ is the solution sequence of the problem $\mathcal{P}$. $\hat{\boldsymbol{x}}(k+\tau|k)$ denotes an estimate value of $\boldsymbol{x}(k+\tau)$, based on the feedback state $\boldsymbol{x}(k)$. Through out this paper, $\|\cdot\|$ represents the Euclidean norm.

Due to the existence of noise and disturbances, an error between $\hat{\boldsymbol{x}}(k+\tau|k)$ and $\boldsymbol{x}(k)$ always exists. For this reason, the NCS solves the problem $\mathcal{P}$ for each k, and only takes the first element $\hat{\boldsymbol{u}}(k|k)$ of $\boldsymbol{U}(k)$ as the control input at time k.

2.2 The Standard form of Quadratic Problem

It is convenient to transform $\mathcal{P}$ into a standard quadratic form. To do this, we rewrite (2) into a matrix form,

$$\hat{\boldsymbol{X}} = \boldsymbol{\Gamma}\boldsymbol{x}(k) + \boldsymbol{H}\boldsymbol{U}(k), \tag{3}$$

where

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{A^2} \\ \vdots \\ \boldsymbol{A^N} \end{bmatrix}, \ \boldsymbol{H} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{AB} & \boldsymbol{B} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A^{N-1}B} & \boldsymbol{A^{N-2}B} & \cdots & \boldsymbol{B} \end{bmatrix}, \tag{4}$$

$$\hat{\boldsymbol{X}} = [\hat{\boldsymbol{x}}^T(k+1|k), \hat{\boldsymbol{x}}^T(k+2|k) \cdots, \hat{\boldsymbol{x}}^T(k+N|k)]^T,$$
$$\boldsymbol{\Gamma} \in \mathbb{R}^{nN\times n}, \ \boldsymbol{H} \in \mathbb{R}^{nN\times lN}, \ \hat{\boldsymbol{X}} \in \mathbb{R}^{nN\times 1}.$$

Then, with (3) and (4), the optimization problem (2) becomes

$$\begin{aligned} \min_{\boldsymbol{U}(k)} \quad & J(\boldsymbol{x}(k), \boldsymbol{U}(k)) = \hat{\boldsymbol{X}}^T\hat{\boldsymbol{X}} + \eta\boldsymbol{U}(k)^T\boldsymbol{U}(k) \\ & = (\boldsymbol{\Gamma}\boldsymbol{x}(k) + \boldsymbol{H}\boldsymbol{U}(k))^T(\boldsymbol{\Gamma}\boldsymbol{x}(k) + \boldsymbol{H}\boldsymbol{U}(k)) \\ & +\eta\boldsymbol{U}(k)^T\boldsymbol{U}(k). \end{aligned}$$

Since the constraints are often given in the algebraic forms of inequalities and equalities, we can transform the constraints $\hat{\boldsymbol{u}}(k+\tau|k) \in \boldsymbol{\mathcal{U}}$ into the following inequality and equality constraints, $\boldsymbol{M}\boldsymbol{U}(k) \leq \boldsymbol{c}$, $\boldsymbol{E}\boldsymbol{U}(k) = \boldsymbol{d}$, where $\boldsymbol{M} \in \mathbb{R}^{m\times lN}$, $\boldsymbol{E} \in \mathbb{R}^{p\times lN}$, $\boldsymbol{c} \in \mathbb{R}^{m\times 1}$, and $\boldsymbol{d} \in \mathbb{R}^{p\times 1}$ are constant matrices and vectors. Then, we define

$$\boldsymbol{Q} = \boldsymbol{H}^T\boldsymbol{H} + \eta\boldsymbol{I}, \ \boldsymbol{b}^T = 2\boldsymbol{x}^T(k)\boldsymbol{\Gamma}^T\boldsymbol{H}, \tag{5}$$

where $\boldsymbol{I}$ is an $lN\times lN$ identity matrix, the size of $\boldsymbol{Q}$ is $lN\times lN$, and the length of vector $\boldsymbol{b}$ is lN. By eliminating the terms independent of the $\boldsymbol{U}(k)$, we transform $\mathcal{P}$ into a standard quadratic form:

$$\begin{aligned} \mathcal{QP}: \ \min_{\boldsymbol{U}(k)} \quad & J(\boldsymbol{x}(k), \boldsymbol{U}(k)) = \boldsymbol{U}^T(k)\boldsymbol{Q}\boldsymbol{U}(k) + \boldsymbol{b}^T\boldsymbol{U}(k) \\ \textbf{s.t.} \quad & \boldsymbol{M}\boldsymbol{U}(k) \leq \boldsymbol{c}, \ \boldsymbol{E}\boldsymbol{U}(k) = \boldsymbol{d}. \end{aligned}$$

The $\mathcal{QP}$ can thus be defined by the tuple,

$$\boldsymbol{\Psi} \triangleq \{\boldsymbol{Q}, \boldsymbol{M}, \boldsymbol{E}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}\}. \tag{6}$$

2.3 Cloud Attack Models

The security threats mainly come from the cloud and the networked communications between the cloud and control system. To capture the possible behaviors of adversaries, we present three attack models:

a) **Ciphered-only attack**: The attacker observes the encrypted information between the cloud and the control system, and attempts to determine the original information. For example, the attacker can simply record all the information it can access in the cloud, and use this to learn the sensor values and control inputs that should remain private [18].

b) **Message modification attack**: The attacker may either intrude the cloud or use spoofing techniques [14] to modify the solution that is sent to the control system. The fake solution can mislead or disrupt the control system, leading to catastrophical consequences.

c) **Availability attack**: The attacker blocks the communication between the cloud and the control systems so that the control systems cannot receive correct solutions from the cloud for either a short or long duration. As control systems require the feedback control inputs to stabilize themselves, this type of attack can lead to instability of the systems.

2.4 Design Goal

Based on the three attack models, we present five design goals for the proposed mechanism:

- **Confidentiality**: No sensitive information in the original control problems can be obtained by the cloud server or adversaries during the computation of the problem.
- **Correctness**: Any cloud server that faithfully runs an application should generate an output that can be decrypted to a correct solution for the control system.
- **Integrity**: The correct results from a faithful cloud should be verified successfully by using a proof from the cloud. No false result can succeed in the verification.
- **Efficiency**: The local computation (e.g., encryption, verification, and decryption) processed by the control system should be less heavy and cheaper than that of solving the original problem.
- **Resiliency**: The control system can maintain stability when no verifiable solution are available from the cloud. The control system can recovery online after an *availability attack*.

2.5 The Framework of the proposed mechanism

Due to the design goals, we present the framework of the proposed mechanism as follows:

i) $\mathsf{KeyGen}(1^\alpha) \rightarrow \{\boldsymbol{K}\}$: *This algorithm randomly generates a key $\boldsymbol{K}$ with a random security parameter α*;

ii) $\mathsf{ProbEnc}(\boldsymbol{K}, \boldsymbol{\Psi}) \rightarrow \{\boldsymbol{\Psi_K}\}$: *This algorithm encrypts $\boldsymbol{\Psi}$ into $\boldsymbol{\Psi_K}$ with the secret key $\boldsymbol{K}$*;

iii) $\mathsf{ProbDec}(\boldsymbol{y}, \boldsymbol{K}) \rightarrow \{\boldsymbol{U}\}$: *If $\Pi_v = 1$, the control system runs this algorithm to decrypt $\boldsymbol{y}$ to obtain the solution $\boldsymbol{U}$ using the key $\boldsymbol{K}$*;

iv) $\mathsf{ProbSolve}(\boldsymbol{\Psi_K}) \rightarrow \{\boldsymbol{y}, \boldsymbol{\Omega}\}$: *This algorithm solves the problem $\boldsymbol{\Psi_K}$ to produce an output $\boldsymbol{y}$ and a proof $\boldsymbol{\Omega}$.*;

v) $\mathsf{ResVer}(\boldsymbol{\Psi_K}, \boldsymbol{y}, \boldsymbol{\Omega}) \rightarrow \{\Pi_v\}$: *This algorithm verify y via the proof $\boldsymbol{\Omega}$. The algorithm outputs $\Pi_v = 0$ if the verification fails, and outputs $\Pi_v = 1$ otherwise*;

vi) $\mathsf{SwitchChec}(\boldsymbol{x}) \rightarrow \{\Pi_s\}$. *If $\Pi_v = 0$, the control system runs this algorithm to check a switching condition. It outputs $\Pi_s = 1$ if the switching condition is satisfied, and outputs $\Pi_s = 0$ otherwise.*

In this framework, the cloud processes the algorithm ProbSolve, while the control system processes KeyGen, ProbEnc, ResVer, ProbDec and SwitchChec.

REMARK 1. *One significant difference between this framework and other encryption methods, such as symmetric and asymmetric encryption, is that the proposed encryption only needs the controllers to process the encryption and decryption, whereas other encryptions require the NCS and cloud to share a public key [26].*

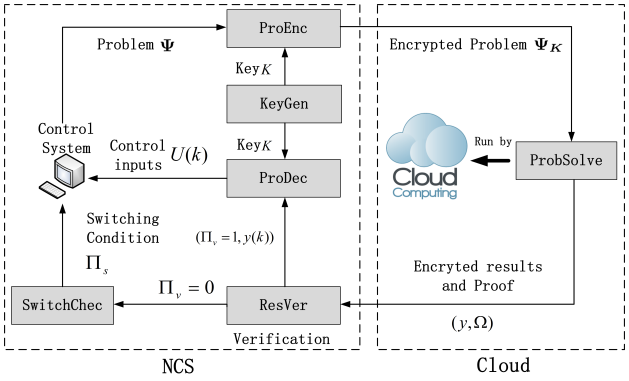

Figure 3: The proposed CE-NCS mechanism interacts between NCS and the cloud.

This property removes the expensive public key exchange in standard encryption, avoids exposing the secret key to unreliable cloud, and also provides additional layer of defense if standard encryption infrastructure is built in for the cloud.

Fig. 3 illustrates the structure of framework. At every sampling time k, the control system first formulates a control problem $\boldsymbol{\Psi}$, encrypts it with a key $\boldsymbol{K}$ to obtain $\boldsymbol{\Psi_K}$, and sends it to the cloud. The cloud runs ProbSolve to solve $\boldsymbol{\Psi_K}$, and returns a solution $\boldsymbol{y}$ together with a proof $\boldsymbol{\Omega}$. The NCS verifies $\boldsymbol{y}$ with $\boldsymbol{\Omega}$. If the verification succeeds, the algorithm ResVer output $\Pi_v = 1$, and the NCS decrypts $\boldsymbol{y}$ to obtain the control input $\boldsymbol{U}(k)$. If the verification fails ($\Pi_v = 0$), then the NCS switches to a Buffer Mode, and checks a switching condition Π_s to determine whether the NCS needs to switch to a Safe Mode. These modes will be designed later in Section 4.

3. CONFIDENTIALITY AND INTEGRITY

The first two attack models, *ciphertext-only attack* and *message modification attack*, are related to data confidentiality and integrity. To protect the CE-NCS from these attacks, we introduce a matrix-disguising method to encrypt the $\mathcal{QP}$ problem and design an efficient way to verify its solution from the cloud. The encryption, decryption, and verification are processed in the control systems.

3.1 Encryption Methods

The first part of the mechanism is to encrypt the constraints of $\mathcal{QP}$. Let $\boldsymbol{G} \in \mathbb{R}^{p\times p}$ and $\boldsymbol{F} \in \mathbb{R}^{m\times p}$ be random non-singular matrices. We perform the following transformation,

$$\begin{cases} \boldsymbol{MU}(k) \le \boldsymbol{c}, \\ \boldsymbol{EU}(k) = \boldsymbol{d}, \end{cases} \Rightarrow \begin{cases} \boldsymbol{(M+FE)U}(k) \le \boldsymbol{c+Fd}, \\ \boldsymbol{GEU}(k) = \boldsymbol{Gd}. \end{cases}$$

Secondly, we encrypt the decision variable $\boldsymbol{U}(k)$. Let $\boldsymbol{P} \in \mathbb{R}^{lN\times lN}$ be a random non-singular matrix, i.e., $\boldsymbol{P}^{-1}$ exists, and $\boldsymbol{g} \in \mathbb{R}^{lN\times 1}$ be a random non-zero vector. Then, we transform $\boldsymbol{U}(k)$ into $\boldsymbol{y}(k)$ by using a one-to-one mapping $\boldsymbol{y}(k) \triangleq \boldsymbol{P}^{-1}(\boldsymbol{U}(k)-\boldsymbol{g})$, and $\boldsymbol{y}(k)$ has the same size as $\boldsymbol{U}(k)$.

Using the key $\boldsymbol{K} \triangleq (\boldsymbol{G}, \boldsymbol{F}, \boldsymbol{P}, \boldsymbol{g})$, $\mathcal{QP}$ can be transformed into the following encrypted problem $\mathcal{QP}_K$ by

$$\begin{aligned} \mathcal{QP}_K : \min_{\boldsymbol{y}(k)} \quad & J_K(\boldsymbol{y}(k)) = \boldsymbol{y}^T(k)\tilde{\boldsymbol{Q}}\boldsymbol{y}(k) + \tilde{\boldsymbol{b}}^{\boldsymbol{T}}\boldsymbol{y}(k) \\ \textbf{s.t.} \quad & \tilde{\boldsymbol{M}}\boldsymbol{y}(k) \le \tilde{\boldsymbol{c}},\ \tilde{\boldsymbol{E}}\boldsymbol{y}(k) = \tilde{\boldsymbol{d}}, \end{aligned}$$

where $\tilde{\boldsymbol{Q}}$, $\tilde{\boldsymbol{M}}$, $\tilde{\boldsymbol{E}}$, $\tilde{\boldsymbol{b}}$, $\tilde{\boldsymbol{c}}$, and $\tilde{\boldsymbol{d}}$ are matrices and vectors of appropriate dimensions given as follows:

$$\begin{cases} \tilde{\boldsymbol{Q}} \triangleq \boldsymbol{P^T QP}, \\ \tilde{\boldsymbol{M}} \triangleq \boldsymbol{(M+FE)P}, \\ \tilde{\boldsymbol{E}} \triangleq \boldsymbol{GEP}, \\ \tilde{\boldsymbol{b}} \triangleq \boldsymbol{(2g^T QP + b^T P)^T}, \\ \tilde{\boldsymbol{c}} \triangleq \boldsymbol{c+Fd-(M+FE)g}, \\ \tilde{\boldsymbol{d}} \triangleq \boldsymbol{G(d-Eg)}. \end{cases} \tag{7}$$

We define $\mathcal{QP}_K$ by the tuple $\boldsymbol{\Psi_K} \triangleq (\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{M}}, \tilde{\boldsymbol{E}}, \tilde{\boldsymbol{b}}, \tilde{\boldsymbol{c}}, \tilde{\boldsymbol{d}})$, which has the same size as $\boldsymbol{\Psi}$ defined by (6).

THEOREM 1. *If $\boldsymbol{y}$ is the solution for $\mathcal{QP}_K$, then $\boldsymbol{U}(k) = \boldsymbol{Py}(k)+\boldsymbol{g}$ must be the solution for $\mathcal{QP}$.*

PROOF. Suppose $\boldsymbol{U}(k) = \boldsymbol{Py}(k) + \boldsymbol{g}$ is not the optimal solution for $\mathcal{QP}$, but $\boldsymbol{y}$ is the solution for $\mathcal{QP}_K$. Then, a $\boldsymbol{U}^*(k)$ exists such that

$$J(\boldsymbol{U}^*(k)) < J(\boldsymbol{U}(k), \boldsymbol{x}(k)), \tag{8}$$

where $J(\boldsymbol{U}(k))$ is a shorthand notation of $J(\boldsymbol{U}(k), \boldsymbol{x}(k))$ when $\boldsymbol{x}(k)$ is fixed. Eq. (8) implies that a $\boldsymbol{y}^* = \boldsymbol{P}^{-1}(\boldsymbol{U}^*(k) - \boldsymbol{g})$ satisfies that

$$\tilde{J}(\boldsymbol{y}^*(k)) = J(\boldsymbol{U}^*(k)) < J(\boldsymbol{U}(k)) = \tilde{J}(\boldsymbol{y}(k)). \tag{9}$$

Therefore, $\boldsymbol{y}^*(k)$ is the solution of $\mathcal{QP}_K$, which contradicts our assumption that $\boldsymbol{y}(k)$ is the solution. This completes the proof. □

The algorithms KeyGen, ProbEnc, and ProbDec are presented as follow:

Algorithm 1: KeyGen$(1^\alpha) \to \{\boldsymbol{K}\}$

1. The input security parameter α specifies a key space $\mathcal{K}_\alpha$
2. Randomly select a key $\boldsymbol{K} = (\boldsymbol{G}, \boldsymbol{F}, \boldsymbol{P}, \boldsymbol{g})$ in the key space $\mathcal{K}_\alpha$
3. Later, use the key $\boldsymbol{K} = (\boldsymbol{G}, \boldsymbol{F}, \boldsymbol{P}, \boldsymbol{g})$ to encrypt the problem.

Algorithm 2: ProbEnc$(\boldsymbol{K}, \boldsymbol{\Psi}) \to \{\boldsymbol{\Psi_K}\}$

1. On the input $\boldsymbol{K}$, compute (7) to get $\boldsymbol{\Psi_K} = (\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{M}}, \tilde{\boldsymbol{E}}, \tilde{\boldsymbol{b}}, \tilde{\boldsymbol{c}}, \tilde{\boldsymbol{d}})$
2. Later, $\boldsymbol{\Psi_K}$ will be sent to the cloud

Algorithm 3: ProbDec$(\boldsymbol{y}, \boldsymbol{K}) \to \{\boldsymbol{U}\}$

1. On the input $(\boldsymbol{y}, \boldsymbol{K})$, compute $\boldsymbol{U} = \boldsymbol{Py} + \boldsymbol{g}$
2. Send $\boldsymbol{U}$ to the actuator of the system.

The time consuming part in these two algorithm is the matrix multiplication, whose time complexity is $O(n^3)$.

3.2 Verification Methods

Due to the *message modification attack*, we need to verify the solution $\boldsymbol{y}$ from the cloud to achieve data integrity. To reduce the complexity of the verification, we find an easily verifiable sufficient and necessary condition for the solution.

Let $\mathcal{D}_K$ be the dual problem of $\mathcal{QP}_K$, and $d_K(\boldsymbol{\lambda}, \boldsymbol{\nu})$ be the value of $\mathcal{D}_K$, where $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ are the dual decision variables. Since the Slater's condition is satisfied, so the strong duality of $\mathcal{QP}_K$ holds, i.e., $d_K(\boldsymbol{\lambda}, \boldsymbol{\nu}) = J_K(\boldsymbol{y})$, when $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ are solutions to $\mathcal{D}_K$, and $\boldsymbol{y}$ is a solution to $\mathcal{QP}_K$ [5]. Then, the cloud can solve $\mathcal{QP}_K$ using the Karush-Kuhn-Tucker (KKT) condition [5], which states that $\boldsymbol{y}$ is a solution to $\mathcal{QP}_K$ if and only if $\boldsymbol{y}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\nu}$ satisfy

$$\tilde{\boldsymbol{E}}\boldsymbol{y} - \tilde{\boldsymbol{d}} = \boldsymbol{0},\ \tilde{\boldsymbol{M}}\boldsymbol{y} - \tilde{\boldsymbol{c}} \leq \boldsymbol{0}, \tag{10}$$

$$\boldsymbol{\lambda} \geq \boldsymbol{0}, \quad \boldsymbol{\lambda}^{\boldsymbol{T}}(\tilde{\boldsymbol{M}}\boldsymbol{y} - \tilde{\boldsymbol{c}}) = \boldsymbol{0}, \tag{11}$$

$$\frac{\partial J_K}{\partial \boldsymbol{y}} = 2\tilde{\boldsymbol{Q}}\boldsymbol{y} + \tilde{\boldsymbol{b}} + \tilde{\boldsymbol{M}}^{\boldsymbol{T}}\boldsymbol{\lambda} + \tilde{\boldsymbol{E}}^{\boldsymbol{T}}\boldsymbol{\nu} = \boldsymbol{0}. \tag{12}$$

Based on the above results, we design a proof $\boldsymbol{\Omega} \triangleq (\boldsymbol{\lambda}, \boldsymbol{\nu})$ to be sent to the control system together with the solution $\boldsymbol{y}$. The control system can verify $\boldsymbol{y}$ using (10)-(12).

A direct verification using (12) can be inefficient when N is large, because the computation required to check a $lN \times 1$ vector $\partial J_K/\partial \boldsymbol{y}$ can be costly. To improve the efficiency, we can randomly check the elements in $\partial J_K/\partial \boldsymbol{y}$. To this end, we define an $lN \times 1$ random vector $\boldsymbol{r} = \{r_1, \ldots, r_{lN}\}$, where $r_i \sim$ Bernoulli$(1/2)$, i.e.,

$$\Pr[r_i = 0] = \Pr[r_i = 1] = \frac{1}{2},\ \text{for } i = 1, \ldots, lN.$$

Using (12), we arrive at

$$\boldsymbol{r}^{\boldsymbol{T}}\left(2\tilde{\boldsymbol{Q}}\boldsymbol{y} + \tilde{\boldsymbol{b}} + \tilde{\boldsymbol{M}}^{\boldsymbol{T}}\boldsymbol{\lambda} + \tilde{\boldsymbol{E}}^{\boldsymbol{T}}\boldsymbol{\nu}\right) = 0. \tag{13}$$

Hence, instead of checking (12) directly, we can check (13), which is a scalar equality. This verification can be performed q times by generating the random vector $\boldsymbol{r}$ q times.

THEOREM 2. *The proposed mechanism ensures that a false solution cannot succeed in the verification mechanism with a probability greater than* $(1/2)^q$.

PROOF. We define that

$$\boldsymbol{\beta} \triangleq 2\tilde{\boldsymbol{Q}}\boldsymbol{y} + \tilde{\boldsymbol{b}} + \tilde{\boldsymbol{M}}^{\boldsymbol{T}}\boldsymbol{\lambda} + \tilde{\boldsymbol{E}}^{\boldsymbol{T}}\boldsymbol{\nu},\ v \triangleq \boldsymbol{r}^{\boldsymbol{T}}\boldsymbol{\beta},$$

where $\boldsymbol{\beta} \in \mathbb{R}^{lN \times 1}$. If the solution $\boldsymbol{y}$ is not correct, then at least one element of $\boldsymbol{\beta}$ is nonzero. Suppose that $\beta_j \neq 0$ where $j \in \{1, \cdots, lN\}$. According to (14), we obtain

$$v = \sum_{i=1}^{lN} \beta_i r_i = r_j\beta_j + s,$$

where $s = \sum_{i=1}^{lN} \beta_i r_i - r_j\beta_j$. Using Bayes' rule, we obtain

$$\begin{aligned} Pr[v = 0] \ =\ & Pr[v = 0 | s = 0] Pr[s = 0] \\ +\ & Pr[v = 0 | s \neq 0] Pr[s \neq 0] \end{aligned} \tag{14}$$

Note that

$$\begin{cases} Pr[v = 0 | s = 0] = Pr[r_j = 0] = 1/2, \\ Pr[v = 0 | s \neq 0] \leq Pr[r_j = 1] = 1/2. \end{cases} \tag{15}$$

Combining (14) and (15), we arrive at

$$Pr[v = 0] \leq (1/2)Pr[s = 0] + (1/2)Pr[s \neq 0] = 1/2.$$

Accordingly, if we verify (13) for q times, then error probability is given by

$$P_e \leq \left(Pr[v = 0]\right)^q \leq (1/2)^q.$$

This completes the proof. □

REMARK 2. *Theorem 2 indicates that we can achieve different security levels by choosing the parameter q. In general, we choose q much smaller than lN so that the verification can be more efficient than directly checking (12). However, a small q will lead to a low security level. Therefore, there is a fundamental tradeoff between efficiency and security level.*

The cloud processes the algorithm ProbSolve to generate a solution $\boldsymbol{y}$ and a proof $\boldsymbol{\Omega}$. The algorithm is given as follow.

Algorithm 4: ProbSolve$(\boldsymbol{\Psi_K}) \rightarrow \{(\boldsymbol{y}, \boldsymbol{\Omega})\}$

1. The cloud use $\boldsymbol{\Psi_K}$ to formulate the dual problem $\mathcal{D}_K$.
2. Solve the dual problem to get $\boldsymbol{\Omega} = (\boldsymbol{\lambda}, \boldsymbol{\nu})$.
3. Based on the dual solutions $(\boldsymbol{\lambda}, \boldsymbol{\nu})$, solve the primal problem $\mathcal{QP}_K$ to get $\boldsymbol{y}$.
4. Later, send $\boldsymbol{y}$ and $\boldsymbol{\Omega} = (\boldsymbol{\lambda}, \boldsymbol{\nu})$ to the control system.

The time complexity of this algorithm is greater than $O(n^3)$ as it solves a quadratic programming problem [21].

The following algorithm ResVer is used to verify the results. ResVer will output a flag Π_v to identify the verification result. The time complexity of this algorithm is $O(n^2)$ as it runs matrix-vector multiplication.

Algorithm 5: ResVer$(\boldsymbol{\Psi_K}, \boldsymbol{y}, \boldsymbol{\Omega}) \rightarrow \{\Pi_v\}$.

1. Check (10) and (11), **if** one of them is not satisfied, then **stop here** and **return** $\Pi_v = 0$.
2. **for** $i = 1 : q$, (q is the running times of the random check from step 3 to step 5)
3. Selects a $lN \times 1$ random vector 0/1 $\boldsymbol{r}$ and check (13)
4. **if** (13) is not satisfied, **stop here** and **return** $\Pi_v = 0$.
5. **end for**
6. The solution is correct and **return** $\Pi_v = 1$;

4. AVAILABILITY ISSUES

The third attack, *availability attack*, highlights the cyber-physical nature of CE-NCSs and can cause a serious problem for the stability of CE-NCSs as the control system requires feedback control inputs at every sampling time to stabilize itself. It can be easily implemented by an attacker who continuously sends erroneous control solutions from the cloud to the NCS. This time-critical property makes NCSs vulnerable to this type of threat.

To this end, we design a Switching Mode Mechanism (SMM) as part of our control design to guarantee the stability of NCSs and enhance their resilience to this attack even when no control inputs are available from the cloud either in a short or long duration. In addition, SMM allows the NCS to maintain a certain level of control performance, and recover on-line when an attacker succeeds in the attack.

4.1 Switching Mode Mechanism

In the SMM, we design three modes for the NCSs: Cloud Mode $\mathcal{N}_c$, Buffer Mode $\mathcal{N}_b$, and Safe Mode $\mathcal{N}_s$. Fig. 4 illustrates how the SMM works for the CE-NCS. $\Pi_v \in \{0, 1\}$ is the output of the verification: $\Pi_v = 1$ means that the verification is successful, while $\Pi_v = 0$ means otherwise. $\Pi_s \in \{0, 1\}$ is the output of a

switching condition, and $\Pi_s = 1$ means that a switching condition is satisfied, while $\Pi_s = 0$ means otherwise. Each working mode is described as follows:

1. **Cloud Mode** $\mathcal{N}_c$: In $\mathcal{N}_c$, the NCS uses a cloud-based controller, which outsources the computation of $\mathcal{QP}_K$ to the cloud. When $\Pi_v = 1$, the NCS stays in the $\mathcal{N}_c$.

2. **Buffer Mode** $\mathcal{N}_b$: In $\mathcal{N}_b$, the NCS uses the control inputs generated in the previous stage. For instance, if the NCS receives the last control input at time k, and no inputs are available from $k+1$ to $k+\tau$, then at time $k+\tau$, the NCS uses control inputs $\hat{\boldsymbol{u}}(k+\tau|k)$ generated at time k when $\Pi_s = 0$.

3. **Safe Mode** $\mathcal{N}_s$: In $\mathcal{N}_s$, the NCS uses a local controller to stabilize itself. For example, if the NCS stays in Buffer Mode, and a switching condition is satisfied ($\Pi_s = 1$) at time $k+\tau$, indicating that the control inputs $\hat{\boldsymbol{u}}(k+\tau|k)$ cannot guarantee the stability for the NCS, then NCS switches to $\mathcal{N}_s$. This switching condition will be designed later in Section 4.2.

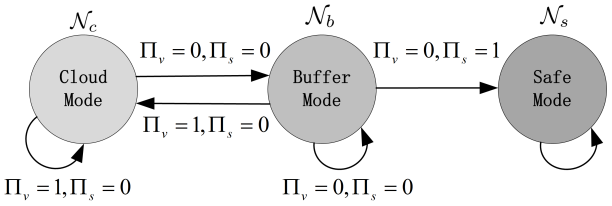

Figure 4: The Switching Mode Mechanism: (1) Cloud Mode $\mathcal{N}_c$: if $\Pi_v = 1$, the NCS stays in this mode; if $\Pi_v = 0$, the NCS switches to Buffer mode. (2) Buffer Mode $\mathcal{N}_b$: if $\Pi_v = 1$ and $\Pi_s = 0$, then system switches back to the Cloud Mode. (3) Safe Mode $\mathcal{N}_s$: if $\Pi_s = 1$, then the NCS switches to the Safe Mode, and never switch back.

Here, we do not allow the NCS to switch back from the Safe Mode to the Cloud Mode. This is due to the fact that "switching back" may result in unanticipated oscillations, which can lead to a new threat.

4.2 The Buffer Mode and the Switching Condition

The design goal of the Buffer Mode is to guarantee the stability of the NCSs when the *availability attack* happens for a sufficiently short period of time. One simple way to tackle this problem is to use the remaining control inputs $\{\hat{\boldsymbol{u}}(k+1|k), \ldots, \hat{\boldsymbol{u}}(k+\tau|k)\}$ generated at time k, when the attack occurs from $k+1$ to $k+\tau$. However, this solution is not ideal, since in real applications, disturbances and noises are ubiquitous in NCSs, and we cannot ensure that these control inputs can still stabilize the NCSs.

One sophisticated way to make use of the remaining control inputs is to apply an event-triggered MPC scheme [16], and find a switching condition, as illustrated in Fig. 3, that can stabilize the system through switching between $\mathcal{N}_c$ and $\mathcal{N}_b$. In addition, to capture the effect of disturbances, a modified model is given as follows:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{D}\boldsymbol{w}(k), \tag{16}$$

where $\boldsymbol{w}(k) \in \mathbb{R}^{n\times 1}$ is a disturbance vector bounded by $||\boldsymbol{w}(k)|| \leq \bar{w}$, and $\boldsymbol{D} \in \mathcal{R}^{n\times n}$ is a constant matrix.

Based on (16) and the event-triggered MPC scheme, the following theorem presents a switching condition for the SMM.

THEOREM 3. *Let k be the last sampling time that the NCS receives a correct solution from the cloud. At the time $k+\tau$ ($1 \leq \tau \leq N-1$), the NCS cannot be guaranteed stability using the input $\hat{\boldsymbol{u}}(k+\tau|k)$ if*

$$\Pi_s : \sum_{i=0}^{N-1} \|\boldsymbol{A}^i\boldsymbol{D}\|^2\bar{\omega} > \sigma h(\boldsymbol{x}(k+\tau-1), \hat{\boldsymbol{u}}(k+\tau-1|k)), \tag{17}$$

where $h(\boldsymbol{x}, \boldsymbol{u}) = \|\boldsymbol{x}\|^2 + \eta\|\boldsymbol{u}\|^2$, and $\sigma \in (0,1)$, is a tuning parameter. Otherwise, $\hat{\boldsymbol{u}}(k+\tau|k)$ can be used to stabilize the system, when no inputs are available from the cloud.

REMARK 3. *The proof of Theorem 3 is presented in Appendix A. The idea of proving the stability of a control system is to use Lyapunov theory to demonstrate that the system energy is decreasing in time k when using the designed control inputs. To describe the system energy, we need to find an appropriate Lyapunov function. The details are given in the proof.*

REMARK 4. *Since $\boldsymbol{A}$, $\boldsymbol{D}$, $\bar{\omega}$, and σ are given, the NCS only needs to compute $h(\boldsymbol{x}(k+\tau-1|k), \hat{\boldsymbol{u}}(k+\tau-1|k))$ to check (17). Therefore, the computations of checking this switching condition is not heavy for the control system.*

The algorithm SwitchChec is used to check the switching condition (17) and decrypt the results from the cloud. In this algorithm, the time-consuming part is the matrix-vector multiplication in the decryption and the computation of switching condition. Therefore, the time complexity of this algorithm is $O(n^2)$.

Algorithm 6: SwitchChec$(\boldsymbol{x}) \rightarrow \{\Pi_s\}$

1. **if** $\Pi_v = 0$, check the switching condition (17).

   if (17) is satisfied, output $\Pi_s = 1$. Use (19) to compute the control input $\boldsymbol{u}_s$.

   else Output $\Pi_s = 0$. Use the remaining control inputs $\hat{\boldsymbol{u}}(k+\tau|k)$.

   end if

2. **end if**

4.3 The Local Controller for the Safe Mode

If (17) holds, the NCS switches to the Safe Mode $\mathcal{N}_s$ to use a local controller. Since the NCSs have no other information in $\mathcal{N}_s$, we use a perfect-state-feedback H_∞-optimal controller, which deals with the worst-case disturbance, through an off-line design.

Let $k+\tau$ be the time that the switching condition (17) holds ($\Pi_s = 1$), and the NCS switches to $\mathcal{N}_s$, where k is the last sampling time that the NCS receives a correct $\boldsymbol{y}$ from the cloud. We denote $\boldsymbol{x}_s(\hat{k})$, $\boldsymbol{u}_s(\hat{k})$ and $\boldsymbol{w}_s(\hat{k})$ as the state values, the control inputs, and the disturbances in $\mathcal{N}_s$, respectively, where $\hat{k} \geq k+\tau$. The system model in $\mathcal{N}_s$ is given by

$$\begin{aligned} \boldsymbol{x}_s(\hat{k}+1) &= \boldsymbol{A}\boldsymbol{x}_s(\hat{k}) + \boldsymbol{B}\boldsymbol{u}_s(\hat{k}) + \boldsymbol{D}\boldsymbol{w}_s(\hat{k}), \\ \boldsymbol{z}_s(\hat{k}) &= \boldsymbol{C}x_s(\hat{k}), \end{aligned}$$

where $\boldsymbol{z}_s \in \mathbb{R}^n$ is the system output, $\boldsymbol{C} = \boldsymbol{I} \in \mathbb{R}^{n\times n}$ is an identity matrix, and the initial condition of the state is $\boldsymbol{x}_s(k+\tau) = \boldsymbol{x}(k+\tau)$, which is the last state in $\mathcal{N}_b$.

The goal of the H_∞-optimal controller is to find an optimal control policy $\boldsymbol{u}_s^*(\hat{k}) = \mu(\boldsymbol{x}(\hat{k}))$ such that

$$\sup_{\boldsymbol{w}_s} \frac{\|\boldsymbol{z}_s\|}{\|\boldsymbol{w}_s\|} \leq \gamma, \tag{18}$$

where $\gamma > 0$ is a given constant.

We assume that the pair $(\boldsymbol{A}, \boldsymbol{B})$ is stabilizable. As stated in [3], with a given parameter $\gamma > 0$, the H_∞-optimal controller for the infinite-horizon case is given by

$$\boldsymbol{u}_s^*(\hat{k}) = \mu^*(\boldsymbol{x}_s(\hat{k})) = \boldsymbol{K}_s \boldsymbol{x}_s(\hat{k}), \tag{19}$$

where $\boldsymbol{K}_s = -\boldsymbol{B}^{\boldsymbol{T}}\boldsymbol{\Lambda}(\boldsymbol{I} + (\boldsymbol{B}\boldsymbol{B}^{\boldsymbol{T}} - \gamma^{-2}\boldsymbol{D}\boldsymbol{D}^{\boldsymbol{T}})\boldsymbol{\Lambda})^{-1}\boldsymbol{A}$, and $\boldsymbol{\Lambda}$ satisfies the following discrete-time algebraic Riccati equation,

$$\boldsymbol{\Lambda} = \boldsymbol{Q}_x + \boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{\Lambda}(\boldsymbol{I} + (\boldsymbol{B}\boldsymbol{B}^{\boldsymbol{T}} - \gamma^{-2}\boldsymbol{D}\boldsymbol{D}^{\boldsymbol{T}})\boldsymbol{\Lambda})^{-1}\boldsymbol{A},$$

where $\boldsymbol{Q}_x = \boldsymbol{C}^T\boldsymbol{C}$. The existence condition of the controller (19) is given by

$$\gamma^2\boldsymbol{I} - \boldsymbol{D}^T\boldsymbol{\Lambda}\boldsymbol{D} > 0. \tag{20}$$

If we can find the smallest $\bar{\gamma}$ satisfying (20), then for all $\gamma \geq \bar{\gamma}$, the NCS can be guaranteed bounded-input bounded-state (BIBS) stable [3].

The control gain $\boldsymbol{K}_s$ is computed off-line based on the knowledge of $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{D}$, and γ. Therefore, the NCSs only need to compute (19) in $\mathcal{N}_s$. The H_∞-optimal controller is more costly in comparison to MPC because it deals with the worst-case disturbance $\boldsymbol{w}_s^*$, which leads to more conservative and expensive control inputs. Hence, the H_∞ controller is suitable for the Safe Mode $\mathcal{N}_s$, when the cloud is not available.

5. ANALYSIS AND EXPERIMENTS

In this section, we present theoretical analysis for the security and efficiency of the mechanism in Fig. 3. Then, we use a UAV as an example to demonstrate that the mechanism can enhance its efficiency and guarantee the stability for CE-NCSs.

5.1 Theoretical Analysis

1. Security analysis: In the first attack model, *ciphertext-only attack*, since we encrypt the problem with a randomly generated key $\boldsymbol{K}$, the knowledge of the encrypted information $\boldsymbol{\Psi}_{\boldsymbol{K}}$ brings no benefit to the attacker in guessing the original information $\boldsymbol{\Psi}$. Secondly, in the *message modification attack*, the verification ensures that no such attack can succeed with a non-negligible probability. Hence, the proposed mechanism can achieve data confidentiality and integrity for CE-NCSs.

2. Efficiency analysis: The overhead on the control system comes from the computations of the encryption, decryption, and verification. The most time-consuming operations are matrix multiplications. The worst time complexity of the proposed mechanism is $O(n^3)$, while the time complexity of solving the original problem $\mathcal{QP}$ is $O(n^\rho)$ for $\rho > 3$ [21]. Therefore, the proposed mechanism can increase the efficiency for the control system.

5.2 Experimental Results

The framework of CE-NCSs plays an important roles in applications of robotic and cyber-physical systems. For example, the Amazon drones for package delivery can be designed based on the proposed framework to achieve better performances. Here, in this subsection, we use UAVs as an example in our experiments. The computations of the UAV and the cloud are performed on the different blocks in the Simulink of MATLAB 2014b, running on the workstation with an Intel Core i7-4770 processor and a 12-G RAM.

The UAV is a small-scale unmanned helicopter, whose dynamic model is linearized at its equilibrium point (hovering point), and the state $\boldsymbol{x}$ of the UAV is defined as

$$\boldsymbol{x} = \left[\, p_x,\ p_y,\ p_z,\ v_x,\ v_y,\ v_z,\ \phi,\ \theta,\ \psi,\ p_a,\ q_a,\ r_a,\ a_s,\ b_s \,\right],$$

where (p_x, p_y, p_z) are the position on (X, Y, Z)-axis of the north-east-down (NED) frame, (v_x, v_y, v_z) are the velocity on (X, Y, Z)-axis of the body frame, (ϕ, θ, ψ) are Euler angles, (p_a, q_a, r_a) are roll, pitch, and yaw angular rates, (a_s, b_s) are the longitudinal and lateral flapping angle of the main rotor. The elements of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ in (1) are given in [6].

1. Efficiency Tests: The length of the horizon window N determines the size of matrix $\boldsymbol{Q}$, which is related to the complexity of $\mathcal{QP}$, and hence, we increase N from 10 to 500 to observe the relationship between N and computational time. The verification parameter q is 20. Table I summarizes the experimental results, where each entry in the table is the average computational time of 10 identical feedback control experiments. In Table I, the computational time to solve $\mathcal{QP}$, t_{orig}, is given in the third column. The time to solve $\mathcal{QP}_K$, t_{cloud}, is given in the fifth column. The time of the cryptographic overhead, t_{cry}, including the key generation, encryption, decryption, and verification, is given in the fourth column. From the results, t_{cry} is mush less than t_{orig}, that is, outsourcing computations to a cloud can enhance efficiency of the UAV. In addition, the cloud efficiency, calculated as t_{orig}/t_{cloud}, is given in the seventh column. Normally, the encryption should not increase the time to solve the quadratic problem. Thus, the cloud efficiency should be close to 1.

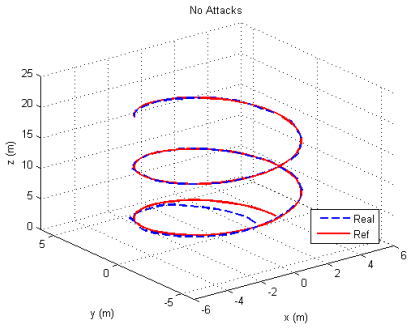

Figure 5: No Attack: The UAV tracks the trajectory smoothly when no attack occurs.

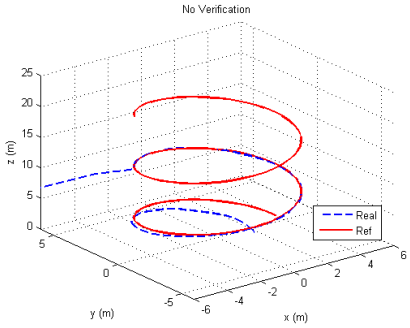

Figure 6: Case 1: No Verification for the solutions.

2. Stability and Resiliency Tests: The next numerical experiment is used to test the stability and resiliency of the CE-NCS in the *message modification attack* and *availability attack*. The control objective of the UAV is to track an ascending helix trajectory under a disturbance $\boldsymbol{w}(k)$ bounded by 0.5. We set the parameter σ in (17) as 0.8. We first show that when no attack occurs, the UAV succeeds in tracking the desired trajectory, and the result is illustrated

Benchmark		Original Problem	Cryptographic Overhead	Encrypted Problem	Cloud Efficiency
#	Length	t_{orig}(sec)	t_{cry}(sec)	t_{cloud}(sec)	t_{orig}/t_{cloud}(sec)
1	$N = 10$	0.006757	0.000159	0.007001	0.965153
2	$N = 50$	0.147428	0.003251	0.164543	0.895984
3	$N = 100$	0.254544	0.008299	0.269742	0.943657
4	$N = 300$	0.634718	0.014426	0.768352	0.826088
5	$N = 500$	1.066516	0.028876	1.609127	0.664027

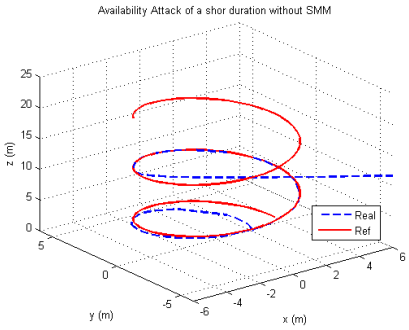

Figure 7: Case 2: Availability Attack of a Short Duration without SMM.

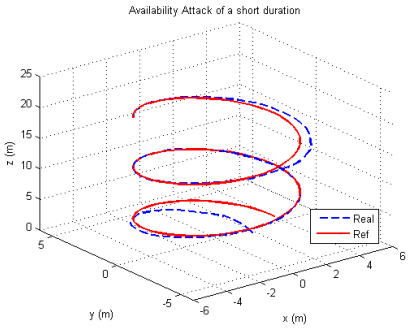

Figure 8: Case 3: Availability Attack of a Short Duration with SMM.

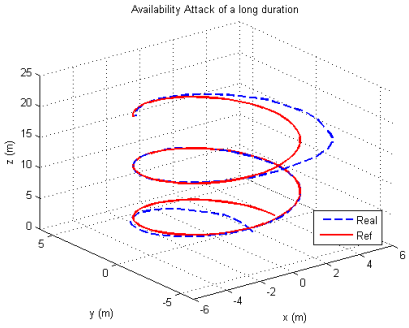

Figure 9: Case 4: Availability Attack of a Long Duration with SMM.

in Fig. 5, where the red solid curve is the reference trajectory, and the blue dash curve is the real trajectory. Then, we conduct other the experiments in four cases to verify our mechanism:

- Case 1 (No Verification): verification algorithms in Section 3.2 are ignored for the solutions. The UAV receives a wrong solution from the cloud at time $k = 60$.
- Case 2 (*Availability attack* of a short duration without SMM): the control inputs from $k = 60$ to $k = 65$ are not available due to the attack. The UAV without SMM have no control inputs, i.e., $\boldsymbol{u}(k) = \mathbf{0}$, in this duration.
- Case 3 (*Availability attack* of a short duration with SMM): the control inputs from $k = 60$ to $k = 65$ are unavailable due to the attack.
- Case 4 (*Availability attack* of a long duration with SMM): the control inputs are unavailable from $k = 60$ to the end of the experiment.

Fig. 6-9 illustrate experimental results for the four cases. In Fig. 6, we can see that without verification, one wrong solution from the cloud deviates the UAV from the reference trajectory. Fig. 7 demonstrates that, without SMM, the UAV deviates from the reference trajectory in a short-duration availability attack, even though it rejects the wrong solutions via verification. Fig. 8 and 9 show that the proposed SMM mechanism guarantees that the UAV tracks reference trajectory successfully, despite the unavailability of the control inputs from the cloud for either a long or a short duration.

To better understand when the UAV switches from the cloud mode to other modes and the effect of σ on the switching condition (17), we present the tracking errors in Fig. 10 and 11 under different scenarios. Fig. 10 illustrates the tracking errors in a short-duration attack with $\sigma = 0.8$. The UAV switches to the buffer mode $\mathcal{N}_b$ at time $k = 60$, and switches back to the cloud mode $\mathcal{N}_c$ at time $k = 66$. Fig. 11 shows the tracking errors in long-duration attacks with $\sigma = 0.6$ and 0.8. When $\sigma = 0.6$, the UAV switches to the buffer mode $\mathcal{N}_b$ at time $k = 60$, and switches to the safe mode $\mathcal{N}_s$ at time $k = 68$. When $\sigma = 0.8$, the UAV switches to the buffer mode $\mathcal{N}_b$ at time $k = 60$, and switches to the safe mode $\mathcal{N}_s$ at time $k = 70$. It is clear that the smaller the σ is, the more quickly the system switches to the safe mode, allowing the system to recover faster. However, when σ is small, the switching mechanism is more sensitive to disturbance $\boldsymbol{\omega}$, i.e., the system may switch to the safe mode inaccurately due to a large disturbance while no attack occurs.

We also quantify the loss of performance when attacks occur by calculating the area between the tracking errors of the no-attack case and attack case. Fig. 12 shows the loss of performance in different scenarios. These results also demonstrate that a smaller σ leads to a lower loss of performance.

The experimental results above highlight that traditional cryptographical tools are not sufficient to meet the security requirements of the CE-NCS due to its cyber-physical nature. Encryption and

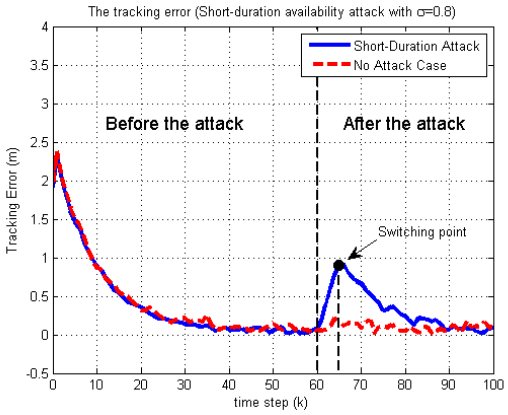

Figure 10: The tracking errors in a short-duration attack with $\sigma = 0.8$.

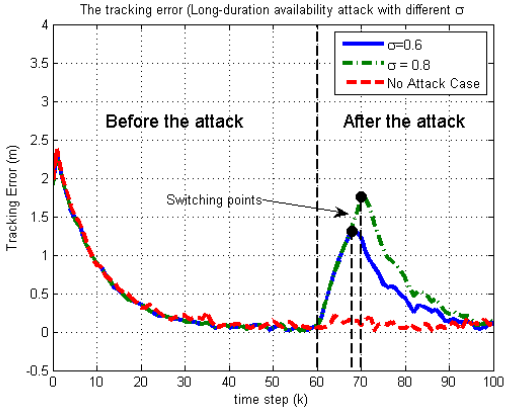

Figure 11: The tracking errors in a long-duration attack with $\sigma = 0.8$ and 0.6.

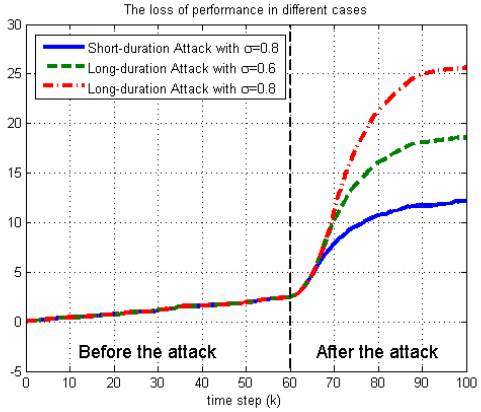

Figure 12: The loss of performance in different cases.

verification of traditional cryptography can protect the data confidentiality and integrity, but they fail to tackle the stability issue of CE-NCSs. Therefore, we incorporate control tools into our mechanism to solve this problem. Our results demonstrate that the proposed mechanism can not only achieve data confidentiality and integrity, but also guarantee the stability and enhance the resiliency of the CE-NCS.

6. CONCLUSIONS

In this paper, we have designed security and resiliency mechanisms for CE-NCSs with the presence of adversaries. We have formulated a cloud-enabled model predictive control problem for NCSs, and presented three attack models. To achieve confidentiality and integrity, we have developed efficient methods to encrypt the problem and verify the solution from the cloud. To protect the system from the availability attack, a switching mechanism has been designed to ensure the stability and resiliency despite the unavailability of control inputs from the cloud. In the end, a UAV example has been used to demonstrate that the proposed mechanisms can significantly protect CE-NCSs from cyber threats and enhance resiliency in an efficient way. A future work of interest is to apply the mechanism to multi-agent NCSs, which will bring new challenges in resource sharing and availability.

7. REFERENCES

[1] M. Arnold and G. Andersson. Model predictive control of energy storage including uncertain forecasts. In *Power Systems Computation Conference (PSCC), Stockholm, Sweden*, 2011.

[2] R. Arumugam, V. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. Kong, A. Kumar, K. Meng, and G.Kit. Davinci: A cloud computing framework for service robots. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3084–3089, 2010.

[3] T. Başar and P. Bernhard. *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer, 2008.

[4] D. Berenson, P. Abbeel, and K. Goldberg. A robot path planning framework that learns from experience. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3671–3678, 2012.

[5] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2009.

[6] G. Cai, B. M. Chen, X. Dong, and T. H. Lee. Design and implementation of a robust and nonlinear flight control system for an unmanned helicopter. *Mechatronics*, 21(5):803–820, 2011.

[7] Y. Chen, Z. Du, and M. García-Acosta. Robot as a service in cloud computing. In *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*, pages 151–158. IEEE, 2010.

[8] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke. Smart grid technologies: Communication technologies and standards. *IEEE transactions on Industrial informatics*, 7(4):529–539, 2011.

[9] T. Hunter, T. Moldovan, M. Zaharia, S. Merzgui, J. Ma, M. J. Franklin, P. Abbeel, and A. M. Bayen. Scaling the mobile millennium system in the cloud. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, page 28, 2011.

[10] Z.-P. Jiang and Y. Wang. Input-to-state stability for discrete-time nonlinear systems. *Automatica*, 37(6):857–869, 2001.

[11] B. Kehoe, D. Berenson, and K. Goldberg. Toward cloud-based grasping with uncertainty in shape: Estimating lower bounds on achieving force closure with zero-slip push grasps. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 576–583, 2012.

[12] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg. Cloud-based robot grasping with the google object recognition engine. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4263–4270. IEEE, 2013.

[13] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. A survey of research on cloud robotics and automation. *IEEE Transaction on Automation Science and Engineering*, 2015.

[14] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys. Unmanned aircraft capture and control via gps spoofing. *Journal of Field Robotics*, 31(4):617–636, 2014.

[15] J.-S. Kim, T.-W. Yoon, A. Jadbabaie, and C. De Persis. Input-to-state stabilizing mpc for neutrally stable linear systems subject to input constraints. In *43rd IEEE Conference on Decision and Control (CDC)*, volume 5, pages 5041–5046, 2004.

[16] D. Lehmann, E. Henriksson, and K. H. Johansson. Event-triggered model predictive control of discrete-time linear systems subject to disturbances. In *European Control Conference (ECC)*, pages 1156–1161, 2013.

[17] X. Lei, X. Liao, T. Huang, and H. Li. Cloud computing service: the case of large matrix determinant computation. *IEEE Transactions on Services Computing*, PP, 2014.

[18] X. Lei, X. Liao, T. Huang, H. Li, and C. Hu. Outsourcing large matrix inversion computation to a public cloud. *IEEE Transactions on Cloud Computing*, 1, 2013.

[19] H. Li and Y. Shi. Networked min–max model predictive control of constrained nonlinear systems with delays and packet dropouts. *International Journal of Control*, 86(4):610–624, 2013.

[20] G. Liu, J. Mu, D. Rees, and S. Chai. Design and stability analysis of networked control systems with random communication time delay using the modified mpc. *International Journal of Control*, 79(4):288–297, 2006.

[21] D. G. Luenberger and Y. Ye. *Linear and nonlinear programming*, volume 116. Springer, 2008.

[22] M. H. Manshaei, Q. Zhu, T. Alpcan, T. Bacşar, and J.-P. Hubaux. Game theory meets network security and privacy. *ACM Computing Surveys (CSUR)*, 45(3):25, 2013.

[23] C. Meng, T. Wang, W. Chou, S. Luan, Y. Zhang, and Z. Tian. Remote surgery case: robot-assisted teleneurosurgery. In *IEEE Int. Conf. on Robotics and Automation (ICAR)*, pages 819–823, 2004.

[24] P. Pandey, D. Pompili, and J. Yi. Dynamic collaboration between networked robots and clouds in resource-constrained environments. *Automation Science and Engineering, IEEE Transactions on*, 12(2):471–480, 2015.

[25] P. Seiler and R. Sengupta. Analysis of communication losses in vehicle control problems. In *Proceedings of the 2001 American Control Conference*, volume 2, pages 1491–1496, 2001.

[26] G. J. Simmons. Symmetric and asymmetric encryption. *ACM Computing Surveys (CSUR)*, 11(4):305–330, 1979.

[27] L. Turnbull and B. Samanta. Cloud robotics: Formation control of a multi robot system utilizing cloud infrastructure. In *Southeastcon, 2013 Proceedings of IEEE*, 2013.

[28] A. N. Venkat, I. Hiskens, J. B. Rawlings, S. J. Wright, et al. Distributed mpc strategies with application to power system automatic generation control. *Control Systems Technology, IEEE Transactions on*, 16(6):1192–1206, 2008.

[29] C. Wang, K. Ren, and J. Wang. Secure and practical outsourcing of linear programming in cloud computing. In *2011 Proceedings IEEE INFOCOM*, pages 820–828, 2011.

[30] Y. Wang and Y. Li. An efficient and tunable matrix-disguising method toward privacy-preserving computation. *Security and Communication Networks*, 2015.

[31] M. Zanon, J. V. Frasch, M. Vukov, S. Sager, and M. Diehl. Model predictive control of autonomous vehicles. In *Optimization and Optimal Control in Automotive Systems*, pages 41–57. Springer, 2014.

[32] Q. Zhu and T. Basar. Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems: Games-in-games principle for optimal cross-layer resilient control systems. *Control Systems, IEEE*, 35(1):46–65, 2015.

APPENDIX

A. PROOF OF THEOREM 3

To prove Theorem 3, we need the following lemmas.

LEMMA 1. *Given the system dynamics (16) with disturbances, the following inequality holds, which is*

$$\|\hat{\boldsymbol{x}}(k+\tau|k+1) - \hat{\boldsymbol{x}}(k+\tau|k)\| \leq \|\boldsymbol{A}^{\tau-1}\boldsymbol{D}\|\bar{\omega}, \tag{21}$$

where $\hat{\boldsymbol{x}}(k+\tau|k+1)$ and $\hat{\boldsymbol{x}}(k+\tau|k)$ are the estimate state value at time $k+\tau$ based on the feedback information at time k and $k+1$, respectively.

PROOF. Note that

$$\begin{aligned}\|\hat{\boldsymbol{x}}(k+1|k+1) - \hat{\boldsymbol{x}}(k+1|k)\| &= \\ \|\boldsymbol{x}(k+1) - \hat{\boldsymbol{x}}(k+\tau|k)\| &\leq \|\boldsymbol{D}\|\bar{\omega}.\end{aligned} \tag{22}$$

Given (16) and (22), we can get

$$\begin{aligned}&||\hat{\boldsymbol{x}}(k+\tau|k+1) - \hat{\boldsymbol{x}}(k+\tau|k)|| \\ &= \|\boldsymbol{A}[\hat{\boldsymbol{x}}(k+\tau-1|k+1) - \hat{\boldsymbol{x}}(k+\tau-1|k)]\| \\ &= \cdots = \|\boldsymbol{A}^{\tau-1}[\hat{\boldsymbol{x}}(k+1|k+1) - \hat{\boldsymbol{x}}(k+1|k)]\| \\ &\leq \|\boldsymbol{A}^{\tau-1}\boldsymbol{D}\boldsymbol{\omega}(k)\| \leq \|\boldsymbol{A}^{\tau-1}\boldsymbol{D}\|\bar{\omega}.\end{aligned}$$

This completes the proof of Lemma 1. □

LEMMA 2. *(Discrete-time Lyapunov Stability)* The system (16) is input-to-state stable (ISS) if and only if there exists a Lyapunov function $V : \mathbb{R}^n \mapsto \mathbb{R}_+$ such that for functions $\alpha_1, \alpha_2, \alpha_3 \in \mathcal{K}_\infty$, $\delta \in \mathcal{K}$, V satisfies

$$\alpha_1(||\boldsymbol{x}(k)||) \leq V(\boldsymbol{x}(k)) \leq \alpha_2(||\boldsymbol{x}(k)||),$$

$$V(\boldsymbol{x}(k+1)) - V(\boldsymbol{x}(k)) \leq -\alpha_3(||\boldsymbol{x}(k)||) + \delta(||\boldsymbol{\omega}(k)||).$$

REMARK 5. *The proof of Lemma 2 is given in [10]. The basic idea of Lyapunov theory is to guarantee that the system energy keeps decreasing until it goes to 0. To proof the stability of the system, we need to find an appropriate Lyapunov function for the control system defined by (1).*

To make use of Lemma 2 to prove stability of the system, we define the following finite-state cost function as a discrete-time Lyapunov function, which is given by

$$\bar{J}(k) = \sum_{\tau=0}^{N-1} h(\hat{\boldsymbol{x}}(k+\tau|k), \hat{u}(k+\tau|k)) + V_N(\hat{\boldsymbol{x}}(k+N|k)), \tag{23}$$

where $V_N(\hat{\boldsymbol{x}}(k+N|k)) = \|\hat{\boldsymbol{x}}(k+N|k)\|^2$ is the terminal cost at step $k+N$.

We also need the following assumption:

ASSUMPTION 1. *Given the cost function (23), we assume that there exists a local controller $\boldsymbol{u}(k) = K_f(\boldsymbol{x}(k))$ such that*

$$V_N(\boldsymbol{x}(k+1)) - V_N(\boldsymbol{x}(k)) \leq -h(\boldsymbol{x}(k), K_f(\boldsymbol{x}(k))).$$

With Lemma 1 and 2 and Assumption 1, the proof of Theorem 3 is presented as follow:

Proof of Theorem 3: Suppose that the control system has received the solution $\boldsymbol{U}(k)$ from the cloud at time k, but has not received $\boldsymbol{U}(k+1)$ at time $k+1$. If the control system keeps using the remaining control signal $\{\hat{\boldsymbol{u}}(k+1|k), \ldots, \hat{\boldsymbol{u}}(k+N-1|k)\}$, then the difference between the real cost $\tilde{J}(k+1|k)$ using the remaining control signal and the previous cost $J(k)$ is

$$\begin{aligned}
\triangle J_1 :=\ & \tilde{J}(k+1|k) - \bar{J}(k) \\
=\ & -h(\hat{\boldsymbol{x}}(k|k), \hat{\boldsymbol{u}}(k|k)) \\
+\ & \sum_{i=1}^{N-1} \Big\{ h(\hat{\boldsymbol{x}}(k+i|k+1), \hat{\boldsymbol{u}}(k+i|k+1)) \\
-\ & h(\hat{\boldsymbol{x}}(k+i|k), \hat{\boldsymbol{u}}(k+i|k)) \Big\} \\
+\ & h(\hat{\boldsymbol{x}}(k+N|k+1), K_f(\hat{\boldsymbol{x}}(k+N|k+1))) \\
+\ & V_N(\hat{\boldsymbol{x}}(k+N+1|k+1) - V_N(\hat{\boldsymbol{x}}(k+N|k).
\end{aligned}$$

Next, by adding and subtracting $V_N(\hat{\boldsymbol{x}}(k+N|k+1)$, we can get

$$\begin{aligned}
\triangle J_1 =\ & -h(\hat{\boldsymbol{x}}(k|k), \hat{\boldsymbol{u}}(k|k)) \\
+\ & \sum_{i=1}^{N-1} \Big\{ h(\hat{\boldsymbol{x}}(k+i|k+1), \hat{\boldsymbol{u}}(k+i|k+1)) \\
-\ & h(\hat{\boldsymbol{x}}(k+i|k), \hat{\boldsymbol{u}}(k+i|k)) \Big\} \\
+\ & h(\hat{\boldsymbol{x}}(k+N|k+1), K_f(\hat{\boldsymbol{x}}(k+N|k+1))) \\
+\ & V_N(\hat{\boldsymbol{x}}(k+N+1|k+1) - V_N(\hat{\boldsymbol{x}}(k+N|k+1). \\
+\ & V_N(\hat{\boldsymbol{x}}(k+N|k+1) - V_N(\hat{\boldsymbol{x}}(k+N|k+1). \qquad (24)
\end{aligned}$$

Since the control system uses the remaining control inputs, then we have $\hat{\boldsymbol{u}}(k+i|k+1) = \hat{\boldsymbol{u}}(k+i|k), \forall i \in [1, N-1]$. According to Lemma 1, we can get

$$\begin{aligned}
& h(\hat{\boldsymbol{x}}(k+i|k+1), \hat{\boldsymbol{u}}(k+i|k+1)) \\
& -h(\hat{\boldsymbol{x}}(k+i|k), \hat{\boldsymbol{u}}(k+i|k)) \\
=\ & \|\hat{\boldsymbol{x}}(k+i|k+1)\|^2 + \eta\|\hat{\boldsymbol{u}}(k+i|k+1)\|^2 \\
& -\|\hat{\boldsymbol{x}}(k+i|k)\|^2 - \eta\|\hat{\boldsymbol{u}}(k+i|k)\|^2 \\
=\ & \|\hat{\boldsymbol{x}}(k+i|k+1)\|^2 - \|\hat{\boldsymbol{x}}(k+i|k)\|^2 \quad \text{(triangle inequality)} \\
\leq\ & \|\hat{\boldsymbol{x}}(k+i|k+1) - \hat{\boldsymbol{x}}(k+i|k)\| \leq \|\boldsymbol{A}^{i-1}\boldsymbol{D}\|^2\bar{\omega}. \qquad (25)
\end{aligned}$$

and

$$\begin{aligned}
& V_N(\hat{\boldsymbol{x}}(k+N|k+1) - V_N(\hat{\boldsymbol{x}}(k+N|k) \\
=\ & \|\hat{\boldsymbol{x}}(k+N|k+1)\|^2 - \|\hat{\boldsymbol{x}}(k+N|k)\|^2 \\
\leq\ & \|x(k+N|k+1) - x(k+N|k)\|^2 \quad \text{(triangle inequality)} \\
\leq\ & \|\boldsymbol{A}^{N-1}\boldsymbol{D}\|^2\bar{\omega}. \qquad (26)
\end{aligned}$$

Using Assumption 1, we get

$$\begin{aligned}
& h(\hat{\boldsymbol{x}}(k+N|k+1), K_f(\hat{\boldsymbol{x}}(k+N|k+1))) + \\
& V_N(\hat{\boldsymbol{x}}(k+N+1|k+1) - V_N(\hat{\boldsymbol{x}}(k+N|k+1) \leq 0. \qquad (27)
\end{aligned}$$

Substituting (25), (26) and (27) into (24) yields the upper bound for $\triangle J_1$, which is

$$\triangle J_1 \leq -h(\hat{\boldsymbol{x}}(k|k), \hat{\boldsymbol{u}}(k|k) + \sum_{i=0}^{N-1} ||\boldsymbol{A}^i\boldsymbol{D}||^2\bar{\omega},$$

In addition, if

$$\sum_{i=0}^{N-1} \|\boldsymbol{A}^i\boldsymbol{D}\|\|^2\bar{\omega} \leq \sigma h(\boldsymbol{x}(k|k), \hat{\boldsymbol{u}}(k|k), \qquad (28)$$

where $\sigma \in (0,1)$, we obtain

$$\triangle J_1 \leq (\sigma - 1)h(\hat{\boldsymbol{x}}(k|k), \hat{\boldsymbol{u}}(k)) < 0.$$

According to Lemma 2, if (28) holds, then using $\hat{\boldsymbol{u}}(k+1|k)$ can still guarantee the stability of the control system at time $k+1$.

Likewise, if the control system keeps using $\hat{\boldsymbol{u}}(k+\tau|k)$ at time $k+\tau$, we can get

$$\begin{aligned}
\triangle J_2 :=\ & \tilde{J}(k+2|k) - \tilde{J}(k+1|k) \\
=\ & -h(\hat{\boldsymbol{x}}(k+1|k), \hat{\boldsymbol{u}}(k+1|k)) \\
+\ & \sum_{i=2}^{N} \Big\{ h(\hat{\boldsymbol{x}}(k+i|k+2), \hat{\boldsymbol{u}}(k+i|k+2)) \\
-\ & h(\hat{\boldsymbol{x}}(k+i|k+1), \hat{\boldsymbol{u}}(k+i|k+1)) \Big\} \\
+\ & h(\hat{\boldsymbol{x}}(k+N+1|k+2), K_f(\hat{\boldsymbol{x}}(k+N+1|k+2))) \\
+\ & V_N(\hat{\boldsymbol{x}}(k+N+2|k+1) - V_N(\hat{\boldsymbol{x}}(k+N+1|k).
\end{aligned}$$

Following the same procedure, as the previous time step, we obtain

$$\triangle J_2 \leq -h(\boldsymbol{x}(k+1|k), \hat{\boldsymbol{u}}(k+1|k)) + \sum_{i=0}^{N-1} \|\boldsymbol{A}^i\boldsymbol{D}\|^2\bar{\omega}.$$

Therefore, using $\hat{\boldsymbol{u}}(k+1|k)$ can still guarantee the stability of the control system at time $k+2$ if

$$\sum_{i=0}^{N-1} \|\boldsymbol{A}^i\boldsymbol{D}\|^2\bar{\omega} < \sigma h(\hat{\boldsymbol{x}}(k+1), \hat{\boldsymbol{u}}(k+1|k)),$$

where $\sigma \in (0,1)$.

By using the same procedure, if the control system keeps using $\hat{\boldsymbol{u}}(k+\tau|k)$ at time $k+\tau$, the system is still stable if

$$\triangle J_\tau = \tilde{J}(k+\tau|k) - \tilde{J}(k+\tau-1|k) \leq 0.$$

Using the similar steps above, we can obtain that

$$\triangle J_\tau \leq -h(\boldsymbol{x}(k+\tau-1|k), \hat{\boldsymbol{u}}(k+\tau-1|k)) + \sum_{i=0}^{N-1} \|\boldsymbol{A}^i\boldsymbol{D}\|^2\bar{\omega}.$$

Therefore, at time $t+\tau$, if the inequality,

$$\sum_{i=0}^{N-1} \|\boldsymbol{A}^i\boldsymbol{D}\|^2\bar{\omega} \leq \sigma h(\hat{\boldsymbol{x}}(k+\tau-1), \hat{\boldsymbol{u}}(k+\tau-1|k)),$$

holds, then the stability of the control system can be guaranteed using $\hat{\boldsymbol{u}}(k+\tau|k)$. In other words, if (17) is satisfied, then the control inputs $\hat{\boldsymbol{u}}(k+\tau|k)$ cannot guarantee stability of the control system. This completes the proof of Theorem 3.