

Encrypted Control for Networked Systems

AN ILLUSTRATIVE INTRODUCTION AND CURRENT CHALLENGES

MORITZ SCHULZE DARUP, ANDREEA B. ALEXANDRU, DANIEL E. QUEVEDO, and GEORGE J. PAPPAS

Digital Object Identifier 10.1109/MCS.2021.3062956
Date of current version: 19 May 2021

1066-033X/21©2021IEEE
Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

Cloud computing and distributed computing are becoming ubiquitous in many modern control areas such as smart grids, building automation, robot swarms, and intelligent transportation systems. Compared to "isolated" control systems, the main advantages of cloud-based and distributed control systems are resource pooling and outsourcing, rapid scalability, and high performance. However, these capabilities do not come without risks. In fact, the involved communication and processing of sensitive data via public networks and on third-party platforms promote (among other cyberthreats) eavesdropping and the manipulation of data (see "Summary"). That these threats are relevant to real-world applications is apparent from an increasing number of cyberattacks explicitly addressing industrial control systems [68]. Prominent examples are the malwares Stuxnet, Duqu, Industroyer, and Triton [14] as well as inference attacks arising from smart meters used as surveillance devices [30], [46].

Clearly, cyberattacks on control systems can be highly critical. Unlike attacks against classical IT systems, attacks on control systems may influence physical processes through digital manipulations [66]. Moreover, networked control systems are the backbone of critical infrastructure such as electric power, transportation, and water distribution networks (with further applications illustrated in "Prospective Uses of Encrypted Control in Industry"). Hence, future control schemes should counteract privacy and security threats and ensure confidentiality, integrity, and availability (see [10] and [66, Fig. S1] for details about these traditional security goals) for the involved process data. Secure control for networked systems has been intensively studied in the literature during the past decade. Comprehensive surveys can be found in [13], [17], [39], and [53] and in a special issue of *IEEE Control Systems*, "Cyberphysical Security," from 2015 (especially [66]).



Two observations are particularly important for the scope of this article. First, analogous to cyberattacks on IT systems, there exists a variety of different attacks and tailored defense mechanisms. For instance, stealth, false data injection, replay, covert, and denial-of-service (DoS) attacks can be distinguished [53]. Second, interdisciplinary solutions are required to secure control systems. Standard information-theoretic and cryptographic tools, on their own, are often not sufficient [13, Sec. 3], [71]. Most existing works focus on the integrity and availability of networked

Summary

Control systems are rapidly evolving to utilize modern computation and communication tools, such as cloud computing and geographically distributed networks, to improve performance, coverage, and scalability. However, control loops that outsource the computation of privacy-sensitive data to third-party platforms via public networks are already the subject of cyberattacks involving eavesdropping and data manipulation. Encrypted control addresses this security gap and ensures the confidentiality of the processed data in the entire control loop by encrypting the information both at the level of transmission (across the network) and at the level of computation (on corrupted computing platforms). This article presents a tutorial-style introduction to this young but emerging field in the framework of secure control for networked dynamical systems with encrypted data. Its focus is on the steps for deriving the encrypted formulations of some specific control algorithms from the standard formulations and discussing the challenges arising in this process, ranging from privacy-aware conceptualizations to changes in computation flows and quantization issues. In conclusion, a list of open problems and new directions to explore is provided to consolidate the area of encrypted control.

Prospective Uses of Encrypted Control in Industry

In recent years, "control as a service" [S1] has emerged, with companies offering personalized optimized control algorithms for better power consumption and economic efficiency. Encrypted control, which addresses privately computing control decisions while processing encrypted data, is an appealing solution to the types of attacks mentioned at the beginning of this article. Control as a service is mainly used in building automation and smart grids, and it is increasingly employed in automation systems for manufacturing and chemical production, food service, transportation systems, water supply, and sewage facility maintenance. The inertia of such systems makes encrypted control (which has overhead, compared to standard control, due to encryption) particularly feasible. Furthermore, given the criticality of some of these industrial systems, control schemes that simultaneously take into account security, safety, and constraints (such as encrypted model predictive control) are recommended. Apart from these examples, lighter-weight distributed encrypted control could be adequate in swarms of exploration and surveillance robots deployed in hazardous environments, where the capture of one robot that operates an encrypted control scheme would not jeopardize the others.

REFERENCE

[S1] O. Givehchi, J. Imtiaz, H. Trsek, and J. Jasperneite. "Control-as-a-service from the cloud: A case study for using virtualized PLCs," in *Proc. 10th IEEE Workshop Factory Commun. Syst.*, 2014, pp. 1–4. doi: 10.1109/WFCS.2014.6837587.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

That these threats are relevant to real-world applications is apparent from an increasing number of cyberattacks explicitly addressing industrial control systems.

control schemes using various defense mechanisms. For example, control-related concepts, such as detectability and the identifiability of deception attacks, are investigated in [53]. Moreover, game-theoretic approaches to address DoS attacks are, for example, considered in [31] and [43].

The emerging field of encrypted control primarily aims for the confidentiality of sensitive system states, control actions, controller parameters, and model data in the entire control loop. More generally, an encrypted controller can be defined as a networked control scheme that simultaneously ensures the control performance and privacy of the client system(s) through specialized cryptographic tools. In the framework of networked control, attacks compromising confidentiality, such as eavesdropping, might seem less critical since they do not immediately cause physical misbehavior. However, "passive" spying often precedes "active" attacks that compromise data integrity and availability [17, Sec. 3(a)]. Abstractly speaking, encrypted control is realized by modifying conventional control schemes such that they are capable of computing encrypted inputs based on encrypted states (and encrypted controller parameters) without intermediate decryptions by the controller. The basic concept is illustrated in Figure 1(b) for a cloud-based controller. In this context, it is important to note that encrypted control goes beyond secure communication channels, as in Figure 1(a), that could be realized using classical encryption, such as the Advanced Encryption Standard (AES).

Encrypted control additionally provides security against curious cloud providers and neighboring agents that, during controller evaluations, would have access to unsecured data for solely secured communications. In this context, the consideration of so-called honest-but-curious platforms is the key difference in existing secure control schemes focusing on confidentiality [42], [70], and it is part of the attack model underlying most encrypted control schemes (as specified in "Security Against What? Safety Goals and Attack Models"). Meeting these privacy demands under real-time restrictions is nontrivial and requires the codesign of controllers and suitable cryptosystems. Identifying controller formulations that can be efficiently combined with capable cryptosystems can be seen as the central task for encrypted control and will be a recurring theme throughout this article.

Encrypted control was first realized in [40] for linear state feedback. Since then, a few dozen articles have presented more efficient implementations and encrypted realizations of more complex control schemes (see "A Brief Survey of Encrypted Control" for details). The different approaches can be classified based on the controller type, system architecture, and applied cryptographic techniques. To date, linear, model predictive, and distributed controllers have been implemented in an encrypted fashion. Networked control systems can be distinguished as one cloud, multiple (noncollaborating) clouds, and multiple agents (see Figures 1–4). For cryptographic tools, solutions involving homomorphic encryption (HE), secret sharing, and hybrid methods that ensure secure multiparty computation have been proposed. Since the mentioned cryptosystems may be new to many readers, the underlying concepts are briefly summarized before giving formal descriptions.

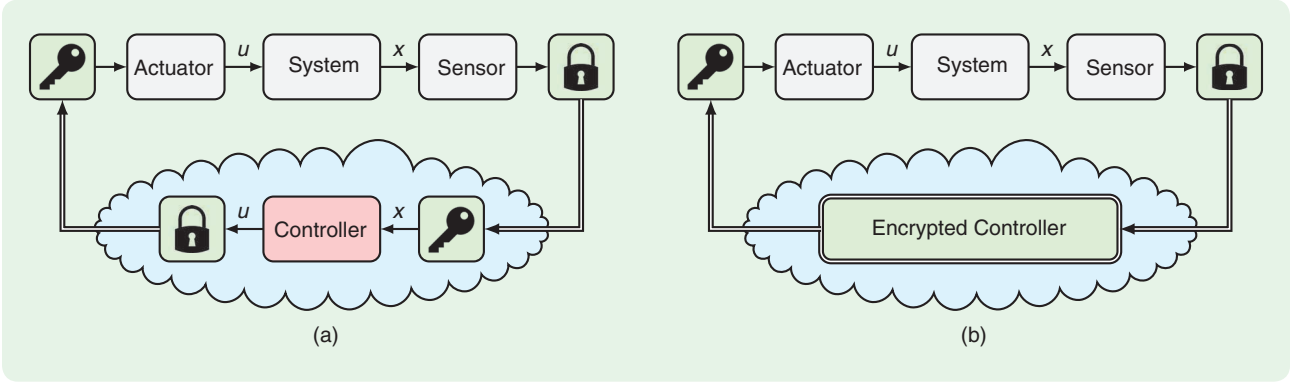


FIGURE 1 (a) A cloud-based control scheme with encrypted communications and an insecure controller evaluation and (b) a cloud-based control scheme with encrypted communications and an encrypted controller evaluation. Double arrows and double framing highlight encrypted data transmission and encrypted data processing, respectively.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

HE [12], [23], [28], [51] refers to a special family of cryptosystems that enable elementary mathematical operations (such as additions and multiplications) to be completed on encrypted data. Secret sharing [63] divides private data into multiple shares such that individual shareholders learn nothing about the information, although the material can be reconstructed by combining a certain number of shares. Secure multiparty computation [29], [54] provides protocols to perform computations on secret data (for example, based on secret shares and oblivious transfer).

This article provides an illustrative introduction to the emerging field of encrypted control for noncryptographers. It discusses and identifies the different types of existing encrypted controllers, their architectures, and the underlying cryptographic tools. Specifically, we summarize the linear controllers in [25], [40], and [59] that build on multiplicatively HE, additively HE, and secure two-party

Security Against What? Safety Goals and Attack Models

In modern cryptography, the concept of "security" is always related to a safety goal and certain attacks. Roughly speaking, an encryption scheme is considered secure against disclosure attacks if a malicious party gets no new information about a plaintext (that is, unencrypted data) from a ciphertext (that is, encrypted data), regardless of previous knowledge [36, p. 19]. The knowledge and capabilities of a bad actor are the basis for the attack model. For example, one can distinguish between ciphertext-only attacks (where a third party observes ciphertexts and attempts to obtain information about the underlying plaintexts) and known plaintext attacks (where a hacker has access to some plaintext–ciphertext tuples) [36, p. 20]. Further variants of the latter are chosen plaintext attacks (where a hacker gets access to ciphertexts for plaintexts of his or her choice) and chosen ciphertext attacks (where an outside party gets access to plaintexts for ciphertexts of its choice).

We next specify some popular security goals. In the framework of cryptography, the highest security level is perfect secrecy [85]. A cryptosystem is called *perfectly secret* if the probability distribution of the ciphertext does not depend on the plaintext (see [36, p. 29]). In other words, the probability of a message being the encrypted one is the same for all possible messages in the space. The easiest way to specify this definition is to study a simple scheme that is not perfectly secret. Consider, for example, the substitution cipher in Table S1 and assume that it is used to encrypt an English text. In this example, the probability distribution of the plaintext reflects the well-known average letter frequencies for English words [36, Fig. 1.3]. Due to the fixed substitutions, the frequencies of the ciphertexts depend on the plaintext. Hence, the scheme (a generalization of Caesar's cipher) is not perfectly secret. Perfect secrecy against ciphertext-only attacks is, however, provided by the so-called one-time pad briefly discussed in the "Basics of Homomorphic Encryption, Secret Sharing, and Multiparty Computation" section.

As apparent from its definition, perfect secrecy is an information-theoretic concept. Hence, it does not depend on the difficulty of some computational problems. In contrast, the security of most homomorphic cryptosystems is called semantic security, which roughly means that the ciphertexts are computationally indistinguishable from one another, and it builds on the hardness of certain computational problems. For example, the semantic security of ElGamal [23] requires that the discrete logarithm is difficult to compute (in the underlying group). Analogously, the semantic security of Paillier [51] builds on the decisional composite residuosity assumption, which is related to the hardness of factorization [36, p. 495]. For such schemes, it is natural to include computational properties in the security specification. In contrast to perfect secrecy, semantic security specifies that some information about the plaintext can be extracted from the ciphertext. However, the revealed information should be negligible, or extracting the information should be computationally infeasible (see [36, Def. 3.12] for a more specific definition).

Under the assumption that the underlying computational problems are hard, ElGamal and Paillier are both semantically secure against chosen plaintext attacks. Interestingly, due to their homomorphisms, they are inherently not secure against chosen ciphertext attacks. Encrypted control mainly considers attacks that focus on disclosing secret information, for example, through eavesdropping. It is assumed that attacks can occur during communication as well as in the cloud(s) and at neighboring agents. It is, however, also assumed that these parties follow prescribed protocols. In other words, the external parties honestly evaluate outsourced algorithms but are curious about the processed data. This attack model is known as honest but curious, and it is standard in cryptography [24, Ch. 2]. In addition, honest-but-curious cloud providers are realistic since it is profitable for them to learn and use the private information of users. However, it is extremely risky for their business model to deviate from the a priori established protocols and be caught.

TABLE S1 **A monoalphabetic substitution cipher.**

Plain	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Cipher	E	N	C	R	Y	P	T	G	O	L	D	F	I	S	H	M	U	Z	A	K	B	J	Q	V	W	X

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

computation, respectively. We further investigate the encrypted model predictive controllers (MPCs) in [2], [60], and [62] and the encrypted cooperative control schemes in [61] and [5]. The selected works facilitate the illustration of two core ingredients of encrypted controllers. First, fundamental cryptographic tools that enable encrypted computations are presented. Second, reformulations of conventional control schemes that support the application of those tools are discussed. In addition, we indicate current challenges in encrypted control that arise from the presented works and thus provide stimuli for future research. The outline of this article is as follows. We first provide basics of HE, secret sharing, and multiparty computation. Afterward, milestones of encrypted control based on realizations of linear, model predictive, and distributed controllers are discussed. Finally, current challenges and future trends in encrypted control are highlighted.

BASICS OF HOMOMORPHIC ENCRYPTION, SECRET SHARING, AND MULTIPARTY COMPUTATION

In what follows, *plaintexts* are data that are unencrypted. It can refer to both decrypted data as well as data that are prepared for encryption (for example, quantized numbers). In a similar manner, *ciphertexts* are encrypted data that (without secret information, such as the private key) cannot be used to retrieve plaintext that was encrypted.

Now, HE refers to a special family of cryptosystems that enable mathematical operations to be completed on encrypted data [12], [16], [23], [28], [51]. Let z_1 and z_2 be two arbitrary numbers in the message space of the cryptosystem, and denote by $\mathtt{Enc}$ and $\mathtt{Dec}$ the encryption and decryption procedure, respectively. More precisely, a cryptosystem is multiplicatively homomorphic if there exists an operation $\otimes$ that enables the evaluation of encrypted multiplications; that is, the relation

$$z_1 z_2 = \mathtt{Dec}(\mathtt{Enc}(z_1) \otimes \mathtt{Enc}(z_2)) \tag{1}$$

holds. Analogously, cryptosystems are called additively homomorphic if an operation $\oplus$ exists such that encrypted additions can be realized:

$$z_1 + z_2 = \mathtt{Dec}(\mathtt{Enc}(z_1) \oplus \mathtt{Enc}(z_2)). \tag{2}$$

A popular multiplicatively HE scheme is the ElGamal cryptosystem [23]. Additively HE is often implemented using Paillier [51] as detailed in "Paillier Encryption and Its

A Brief Survey of Encrypted Control

Encrypted linear controllers, including linear-quadratic (Gaussian) regulators, implemented in a single cloud using partially homomorphic encryption (HE), have been proposed in [15], [25], [26], [27], [40], [47], [67], and [S2]; leveled HE was introduced in [3], and fully HE was offered in [37]. An encrypted linear controller utilizing two clouds, secret sharing, and secure two-party computation was recently discussed in [59]. Homomorphically encrypted and authenticated linear control was proposed in [S3] using labeled HE. Encrypted model predictive control using a single cloud and additively HE was addressed in [57], [60], and [62]. Encrypted using two clouds and a combination of HE and secure multiparty computation was presented in [2] and [4]. The private Kalman filtering problem is considered in [S4], where a solution involving secure multiparty computation and HE is proposed. An encrypted controller for discrete-event systems was investigated in [S5]. Finally, encrypted distributed control for multiagent systems was considered in [5], [61], [S6], and [S7]. Note that encrypted optimization can be further used to design encrypted controllers (for example, model predictive control). Works that have proposed schemes for solving optimization problems on encrypted data include [S8]–[S11]. The approaches in [61], [S6], and [S7] mainly build on additively HE, whereas secure multiparty computation dominates in [5] and [S11].

REFERENCES

[S2] Y. Lin, F. Farokhi, I. Shames, and D. Nešic, "Secure control of nonlinear systems using semi-homomorphic encryption," in *Proc. 57th Conf. Decis. Control*, 2018, pp. 5002–5007. doi: 10.1109/CDC.2018.8619569.

[S3] J. H. Cheon et al., "Toward a secure drone system: Flying with real-time homomorphic authenticated encryption," *IEEE Access*, vol. 6, pp. 24,325–24,339, 2018. doi: 10.1109/ACCESS.2018.2819189.

[S4] F. J. Gonzalez-Serrano, A. Amor-Martin, and J. Casamayon-Anton, "State estimation using an extended Kalman filter with privacy-protected observed inputs," in *Proc. Int. Workshop Info. Forens. Security*, 2014, pp. 54–59.

[S5] R. Fritz, M. Fauser, and P. Zhang, "Controller encryption for discrete event systems," in *Proc. 2019 Am. Control Conf.*, pp. 5633–5638. doi: 10.23919/ACC.2019.8814612.

[S6] M. Kishida, "Encrypted average consensus with quantized control law," in *Proc. 57th Conf. Decis. Control*, 2018, pp. 5850–5856. doi: 10.1109/CDC.2018.8619855.

[S7] M. Ruan, H. Gao, and Y. Wang, "Secure and privacy-preserving consensus," *IEEE Trans. Automat. Control*, vol. 64, no. 10, pp. 4035–4049, Oct. 2019. doi: 10.1109/TAC.2019.2890887.

[S8] A. B. Alexandru, K. Gatsis, and G. J Pappas, "Privacy preserving cloud-based quadratic optimization," in *Proc. 55th Annu. Allerton Conf.*, 2017, pp. 1168–1175. doi: 10.1109/ALLERTON.2017.8262869.

[S9] Y. Lu and M. Zhu, "Privacy preserving distributed optimization using homomorphic encryption," *Automatica*, vol. 96, pp. 314–325, Oct. 2018. doi: 10.1016/j.automatica.2018.07.005.

[S10] Y. Shoukry et al., "Privacy-aware quadratic optimization using partially homomorphic encryption," in *Proc. 55th Conf. Decis. Control*, 2016, pp. 5053–5058. doi: 10.1109/CDC.2016.7799042.

[S11] K. Tjell and R. Wisniewski, "Privacy preservation in distributed optimization via dual decomposition and ADMM," in *Proc. 58th Conf. Decis. Control*, 2019, pp. 7203–7208. doi: 10.1109/CDC40024.2019.9028969.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

Future control schemes should counteract privacy and security threats and ensure confidentiality, integrity, and availability.

Homomorphisms." Encryption schemes that are both additively and multiplicatively homomorphic are called fully homomorphic. In principle, fully HE schemes can be used to encrypt arbitrary functions [28]. However, full HE is computationally highly demanding and currently not a competitive option for encrypted control on power- and memory-constrained devices. Nevertheless, so-called somewhat, or leveled, fully HE schemes [12], [16], which support a limited number of encrypted multiplications and (a limited or unlimited number of) encrypted additions, have already been used for encrypted control and may become more relevant in the future.

If multiple cloud servers and agents are involved in control law evaluations, secret sharing (combined with other multiparty computation techniques) is an option to realize encrypted control. Roughly speaking, secret sharing enables dividing and recombining private data in such a way that individual shares reveal no information about the secret [63]. More formally, a secret-sharing scheme that splits data into $S \geq 2$ shares in such a way that at least $L \in \{2, \ldots, S\}$ shares are required to reconstruct the information is called an (L,S) threshold scheme. In the simplest case, $S = 2$ shareholders are involved, which implies $L = 2$ (that is, the secret data are split into two shares, and both shares are required to reconstruct the information). As an example, assume that the secret is an integer $z \in \{0, \ldots, \varphi - 1\}$ for some user-defined $\varphi \in \mathbb{N}$ with $\varphi \geq 2$. Then, a (2,2) additive secret sharing scheme can be realized as follows. An integer $r \in \{0, \ldots, \varphi - 1\}$ is chosen uniformly at random, and the ciphertext

$$c := z + r \bmod \varphi \tag{3}$$

is computed, where mod refers to the standard modulo operation. The two numbers c and r can be considered two shares of the secret z. In this context, it is easy to see that neither c nor r reveals any information about z. In fact, r is a random number, and c, as in (3), corresponds to the ciphertext of a so-called one-time pad. Since one-time pads are perfectly secure (see [64] and "Security Against What? Safety Goals and Attack Models"), c reveals no information about z apart from the fact that $z \in \{0, \ldots, \varphi - 1\}$. Nevertheless, combining the shares c and r enables one to reconstruct z according to $z = c - r \bmod \varphi$.

Secure multiparty computation consists of a set of techniques that involves the secure processing of data between multiple mutually distrusting parties [24]. A secure multiparty computation scheme can be realized by splitting private data among multiple parties by using secret sharing, garbled circuits [7], and (threshold) HE [18] and then jointly computing a mutually agreed upon function on pieces of information through hybrid versions employing the aforementioned tools. An example of a protocol essential to secure multiparty computation is

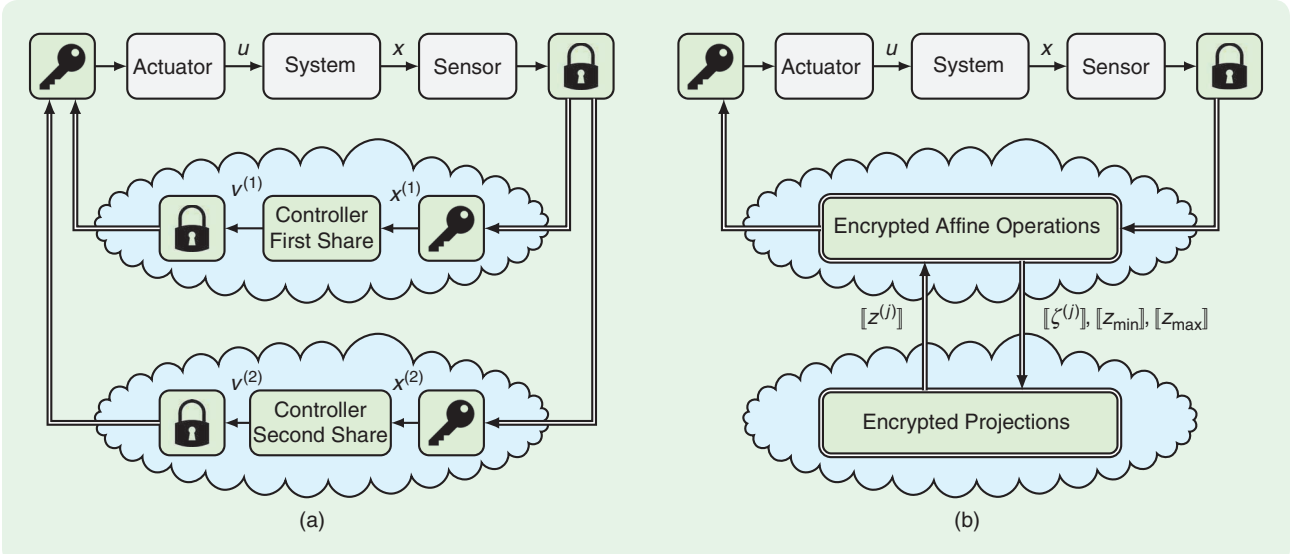


FIGURE 2 (a) An encrypted control scheme based on two-party computation without a data exchange between the clouds and (b) the encrypted implementation of a projected gradient scheme based on two noncolluding clouds with an encrypted data exchange.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

(1-out-of-2) oblivious transfer. To illustrate this protocol, consider a party that holds two numbers $z_0, z_1 \in \{0, \ldots, \varphi - 1\}$ and a second party with a bit $\tau \in \{0, 1\}$. The problem underlying oblivious transfer is to enable the second party to obtain z_τ without finding out the other number ($z_{1-\tau}$) and revealing τ to the first party. There exist various implementations of oblivious transfer [24, Ch. 3.7]. A simple realization based on additively HE is presented that will later be used to implement an encrypted MPC.

To prepare the oblivious transfer, the second party establishes a Paillier cryptosystem by generating a public key used for encryption and homomorphic operations and a secret key required for decryption (see "Paillier Encryption and Its Homomorphisms" for details). The first party then generates two random numbers (r_0 and r_1), computes $z_0 + r_0 \bmod \varphi$ and $z_1 + r_1 \bmod \varphi$ according to (3), and sends the results to the second party. Based on its bit τ, the second party selects, encrypts, and

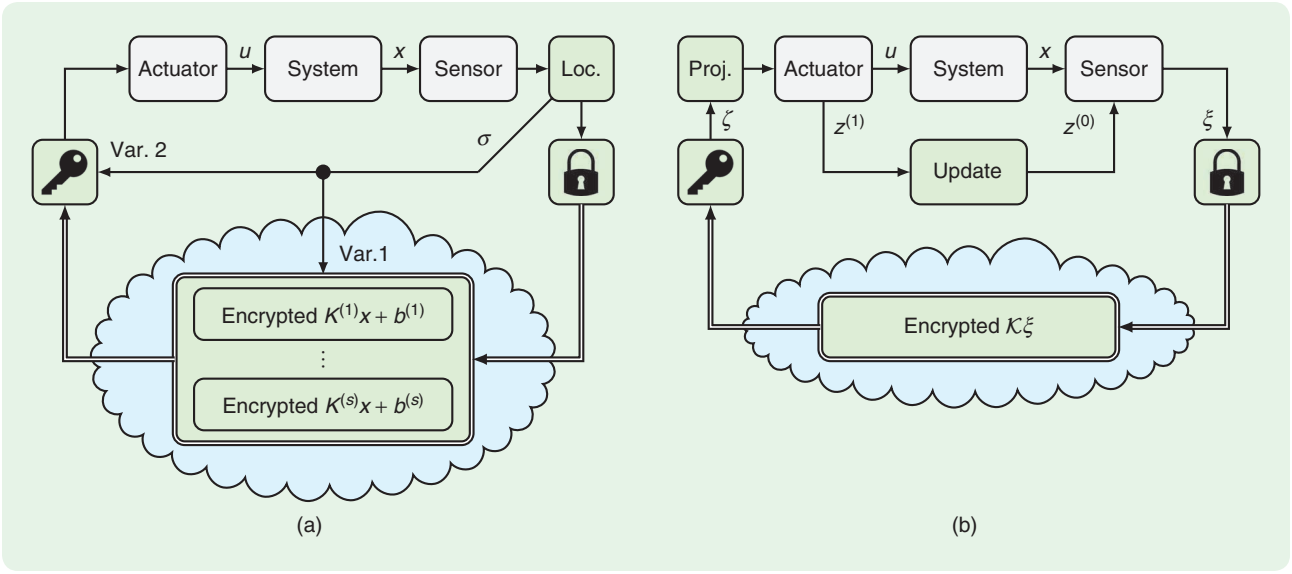


FIGURE 3 Encrypted model predictive controller implementations based on (a) the parametric offline solution of the underlying optimal control problem and (b) real-time iterations of a projected gradient scheme. In (a), two variants (Var.) regarding the communication of the current segment σ are illustrated. In both variants, the localization (Loc.) of $\mathcal{P}^{(\sigma)}$ is carried out at the sensor. In (b), the projection step (Proj.) is carried out at the actuator.

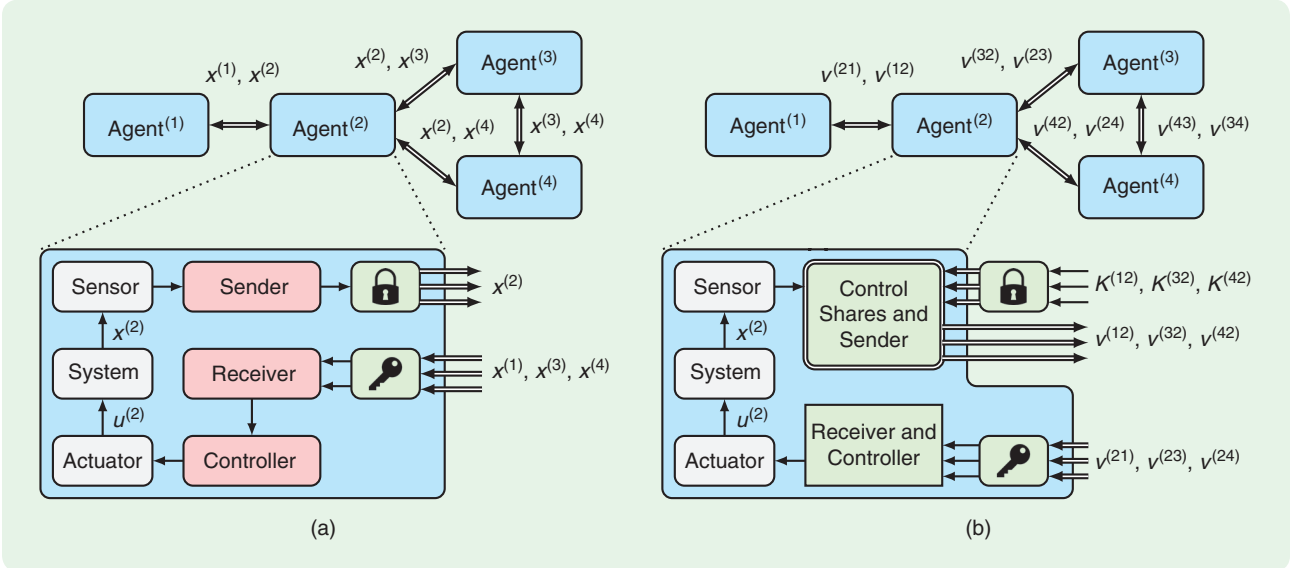


FIGURE 4 (a) A cooperative multiagent system with encrypted communications and (b) a multiagent system operated by an encrypted cooperative control scheme. In (b), control shares $\boldsymbol{v}^{(ij)} = \boldsymbol{K}^{(ij)}\boldsymbol{x}^{(j)}$ are computed, encrypted, and communicated instead of the classical exchange of neighboring states $\boldsymbol{x}^{(j)}$, as in (a).

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

Identifying controller formulations that can be efficiently combined with capable cryptosystems can be seen as the central task for encrypted control.

sends back $\texttt{Enc}(z_\tau + r_\tau \bmod \varphi)$ and $\texttt{Enc}(\tau)$. Next, the first party computes

$$c := \texttt{Enc}(z_\tau + r_\tau \bmod \varphi) \oplus ((\texttt{Enc}(\tau) \oplus \texttt{Enc}(-1)) \odot r_0) \oplus (\texttt{Enc}(\tau) \odot (-r_1)) \tag{4}$$

by exploiting (2) and another (related) homomorphism of the Paillier scheme that is denoted by $\odot$. As noted in "Paillier Encryption and Its Homomorphisms," Paillier offers not only (2) but the relation (S5) that enables multiplication between one encrypted factor and one plaintext integer [that is, $\texttt{Enc}(z_1) \odot z_2$]. Afterward, the first party returns the resulting ciphertext c to the second party, which finally obtains z_τ from evaluating $\texttt{Dec}(c) \bmod \varphi$ by using its secret key. Obviously, z_τ is correctly received since (4) actually implements

$$z_\tau + r_\tau \bmod \varphi + r_0(\tau - 1) - r_1\tau = z_\tau + r_\tau \bmod \varphi - r_\tau.$$

The privacy of the oblivious transfer follows from the security of the additively homomorphic cryptosystem and the statistical security of the secret-sharing scheme. While the preceding primer presents basic functionalities of cryptosystems that enable encrypted computations, there are plenty of techniques in the literature that improve the efficiency and usability of these tools. Relevant cryptographic software libraries and advantageous hardware are surveyed in "Software for Encrypted Computations and Hardware Acceleration."

We complete this technical section with a visual analogy. Typical encryption schemes can be envisioned as safe boxes, where a plaintext is stored safely and can be retrieved only by knowing the combination code. On the other hand, a scheme that facilitates encrypted computations (such as the ones presented in this section) can be viewed as a safe glove box: one can manipulate the plaintext stored inside through the gloves (through ciphertext

Paillier Encryption and Its Homomorphisms

So far, the majority of encrypted controllers use the Paillier cryptosystem [51]. The standard realization of the scheme considers the additive group of integers modulo P (denoted as $\mathbb{Z}_P$) as the message space, where P is the product of two large primes p and q. For the purposes of this article, it is convenient to think of $\mathbb{Z}_P$ as the set $\{0, \ldots, P-1\} \subset \mathbb{N}$. During the key generation procedure, two large primes $p, q \in [2^{\ell-1}, 2^\ell]$ of the same "length" $\ell \in N$ are selected, and the public key is specified by $P = pq$. The secret key, which is known only to the entity that needs to decrypt the data, is computed as $S = (p-1)(q-1)$. The encryption of a number z from $\mathbb{Z}_P$ is realized by

$$\texttt{Enc}(z, r) := (P+1)^z r^P \bmod P^2, \tag{S1}$$

where r is randomly picked from $\mathbb{Z}^*_{P^2}$, the multiplicative group of integers modulo P^2. The resulting ciphertext c is likewise an element of $\mathbb{Z}^*_{P^2}$, and the decryption is completed by computing

$$\texttt{Dec}(c) := \frac{(c^S \bmod P^2) - 1}{P} \mu \bmod P \tag{S2}$$

using the secret key S, where μ is the multplicative inverse of S modulo P. It is easy to prove the correctness of the scheme; that is, $z = \texttt{Dec}(\texttt{Enc}(z, r))$ for every $z \in \mathbb{Z}_P$ and every $r \in \mathbb{Z}^*_{P^2}$.

The semantic security of the Pallier scheme builds on the assumption that the decisional composite residuosity problem [36, p. 495] is hard to solve, which is currently the case for a large P (thousands of bits). Regarding homomorphisms, it can be inferred from (S1) and (S2) that the Paillier cryptosystem is additively homomorphic. As a result,

$$z_1 + z_2 = \texttt{Dec}(\texttt{Enc}(z_1, r_1)\, Enc(z_2, r_2) \bmod P^2) \tag{S3}$$

for all $z_1, z_2 \in \mathbb{Z}_P$ and $r_1, r_2 \in \mathbb{Z}^*_{P^2}$. At this point, recall that the messages are integers. Hence, multiplications $z_1 z_2$ can be implemented as sums of the form

$$z_1 z_2 = \underbrace{z_1 + \cdots + z_1}_{z_2\text{-times}}. \tag{S4}$$

Repeatedly applying the homomorphism (S3) to the right-hand side of (S4) yields the relation

$$z_1 z_2 = \texttt{Dec}(\texttt{Enc}(z_1, r)^{z_2} \bmod N^2) \tag{S5}$$

that holds for all $z_1, z_2 \in \mathbb{Z}_P$ and all $r \in \mathbb{Z}^*_{P^2}$. Relation (S5) facilitates the implementation of partially encrypted multiplications with one encrypted factor (in this case, z_1). This feature is an advantage of additively homomorphic cryptosystems over multiplicatively homomorphic schemes, which do not support partially encrypted additions.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

manipulation) without the need to retrieve it outside the box by using the combination code.

ENCRYPTED LINEAR CONTROL

As briefly discussed in the introduction, the first encrypted controllers implement linear control laws of the form

$$\boldsymbol{u}(k) = \boldsymbol{K}\boldsymbol{x}(k), \tag{5}$$

where $\boldsymbol{K} \in \mathbb{R}^{m\times n}$. Such linear controllers are typically applied to linear (discrete-time) systems

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \tag{6}$$

with $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$. However, for the moment, the system dynamics are not of interest. The first provably secure implementation of (5) was realized in [40] by using the multiplicatively homomorphic ElGamal scheme [30]. Note that encrypted implementations of (5) were proposed slightly earlier in [27] (and in [40]) using Rivest–Shamir–Adleman (RSA) encryption [56]. However, these implementations build on the multiplicative homomorphism of the so-called textbook RSA that is considered insecure [36, p. 412].

The following summarizes three existing realizations of encrypted linear control that build on different cryptographic techniques. For completeness, note that not only static linear controllers, as in (5), have been considered in the framework of encrypted control. Numerous works [15], [37], [40], [47], [67] focus on the encryption of linear dynamic controllers of the form

Software for Encrypted Computations and Hardware Acceleration

One of the goals of encrypted computation is to provide wide-range use, which requires optimized code and versatility of functions. To this end, there are several available open source software packages. Regarding partially homomorphic encryption (HE), a multiplicity of implementations exists due to the applications' relatively simple functionality. Refer to `python-paillier` [S12], which implements the Paillier scheme, with the security caveat that the default encoding is a floating-point encoding with a public exponent. For fully HE, note the C++ libraries `HE Library` [S13], `Simple Encrypted Arithmetic Library` [S14], and `PALISADE` [S15]. Due to the high complexity of the implemented schemes, these libraries require expert users. However, they provide a large variety of capabilities. One important capacity, which can be decisive when judging the realizability of encrypted control, is the possibility of effectuating the encrypted computations on the native 64-b arithmetic of current processors, rather than expensive multiprecision arithmetic. Also refer to the compilers for fully homomorphic cryptosystems in [S16] and [S17] that automate the choice of the best parameters for the runtime and the best parallelization techniques.

Various software tools are also available in the realm of secure multiparty computation. They implement, for example, garbled circuits, oblivious transfer schemes, and secret-sharing protocols and further facilitate transforming high-level code into low-level secure protocols. The article [S18] surveys these tools and comments on their usability and availability. Finally, note that special hardware can achieve much better performance for some encrypted computations than commodity hardware can. By hardwiring specific computations in GPUs and reconfigurable hardware, such as field-programmable gate arrays (FPGAs), a significant speedup can be obtained. This path, which is a research area on its own, is taken by a plethora of works, such as [S19]–[S24]. In the scope of this article, a custom digital design for encrypted control on an FPGA that makes use of hardware-optimized modular multiplications is proposed in [S25].

REFERENCES

[S12] "Data61. Python paillier." Github. 2013. https://github.com/data61/python-paillier

[S13] S. Halevi and V. Shoup, "Algorithms in HElib," in *Proc. Annu. Cryptol. Conf.*, 2014, pp. 554–571. doi: 10.1007/978-3-662-44371-2_31.

[S14] "Microsoft SEAL (release 3.4)." Github. Oct. 2019. https://github.com/Microsoft/SEAL

[S15] PALISADE. Jan. 2020. https://palisade-crypto.org/

[S16] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, "nGraph-HE: A graph compiler for deep learning on homomorphically encrypted data," in *Proc. 16th ACM Intl. Conf. Comput. Front.*, 2019, pp. 3–13. doi: 10.1145/3310273.3323047.

[S17] R. Dathathri et al., "CHET: An optimizing compiler for fully-homomorphic neural-network inferencing," in *Proc. 40th ACM SIGPLAN Conf. Program. Lang. Des. Implement.*, 2019, pp. 142–156. doi: 10.1145/3314221.3314628.

[S18] M. Hastings, B. Hemenway, D. Noble, and S. Zdancewic, "SoK: General purpose compilers for secure multi-party computation," in *Proc. IEEE Symp. Security Privacy*, 2019, pp. 479–496. doi: 10.1109/SP.2019.00028.

[S19] D. B. Cousins, K. Rohloff, and D. Sumorok, "Designing an FPGA-accelerated homomorphic encryption co-processor," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 2, pp. 193–206, 2016. doi: 10.1109/TETC.2016.2619669.

[S20] Y. Doröz, E. Öztürk, and B. Sunar, "Accelerating fully homomorphic encryption in hardware," *IEEE Trans. Comput.*, vol. 64, no. 6, pp. 1509–1521, 2014. doi: 10.1109/TC.2014.2345388.

[S21] T. Pöppelmann, M. Naehrig, A. Putnam, and A. Macias, "Accelerating homomorphic evaluation on reconfigurable hardware," in *Proc. Int. Workshop Cryptogr. Hardware Embedded Syst.*, 2015, pp. 143–163. doi: 10.1007/978-3-662-48324-4_8.

[S22] M. S. Riazi, K. Laine, B. Pelton, and W. Dai, "HEAX: An architecture for computing on encrypted data," in *Proc. 25th Intl. Conf. Architect. Support Program. Lang. Oper. Syst.*, 2020, pp. 1295–1309. doi: 10.1145/3373376.3378523.

[S23] S. S. Roy, F. Turan, K. Jarvinen, F. Vercauteren, and I. Verbauwhede, "FPGA-based high-performance parallel architecture for homomorphic computing on encrypted data," in *Proc. Int. Symp. High Perform. Comput. Architect.*, 2019, pp. 387–398. doi: 10.1109/HPCA.2019.00052.

[S24] I. San, N. At, I. Yakut, and H. Polat, "Efficient Paillier cryptoprocessor for privacy-preserving data mining," *Security Commun. Net.*, vol. 9, no. 11, pp. 1535–1546, 2016. doi: 10.1002/sec.1442.

[S25] J. Tran, F. Farokhi, M. Cantoni, and I. Shames, "Implementing homomorphic encryption based secure feedback control," *Control Eng. Pract.*, vol. 97, p. 104,350, Apr. 2020. doi: 10.1016/j.conengprac.2020.104350.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

$$\boldsymbol{\xi}(k+1) = \mathcal{A}\boldsymbol{\xi}(k) + \mathcal{B}\boldsymbol{y}(k), \tag{7a}$$

$$\boldsymbol{u}(k) = \mathcal{C}\boldsymbol{\xi}(k) + \mathcal{D}\boldsymbol{y}(k), \tag{7b}$$

where $\boldsymbol{\xi}(k)$ and $\boldsymbol{y}(k) := \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k)$ refer to the controller state and the system's output, respectively. Nevertheless, encrypting (7) is significantly harder than (5) since iterating the controller state in an encrypted fashion is more demanding (as briefly explained in the "Quantization Effects, Stability Guarantees, and Continuous Operation" section). The interested reader is referred to the original works for more details.

First Implementation Using Elgamal Encryption

The encrypted implementation of (5) in [40] is based on the following two observations. First, the control action can be computed based on the matrix

$$\boldsymbol{\Phi}(k) := \begin{pmatrix} \boldsymbol{K}_{11}\boldsymbol{x}_1(k) & \dots & \boldsymbol{K}_{1n}\boldsymbol{x}_n(k) \\ \vdots & \ddots & \vdots \\ \boldsymbol{K}_{m1}\boldsymbol{x}_1(k) & \dots & \boldsymbol{K}_{mn}\boldsymbol{x}_n(k) \end{pmatrix} \in R^{m\times n} \tag{8}$$

that contains solely products of the controller parameters $\boldsymbol{K}_{ij}$ and the states $\boldsymbol{x}_j(k)$. In fact, $\boldsymbol{u}_i(k)$ [as in (5)] is obviously equivalent to

$$\boldsymbol{u}_i(k) = \sum_{j=1}^{n} \boldsymbol{\Phi}_{ij}(k) \tag{9}$$

for every $i \in \{1,\dots,m\}$. Second, an encrypted cloud-based computation of (8) can be implemented using ElGamal. To this end, the current state $\boldsymbol{x}(k)$ is measured and encrypted at the sensor. The following uses the shorthand notation $[\![\cdot]\!]$ to highlight encrypted values. We remark that the notation is slightly informal since it combines the mapping of the data onto the message space of the cryptosystem and the subsequent encryption without giving details of the actual implementation. A suitable procedure is presented in "Forming Integer Messages From Quantized Data," and the interested reader is referred to the respective original works.

The resulting ciphertexts $[\![\boldsymbol{x}(k)]\!]$ are then sent to the cloud, where the mn products

$$[\![\boldsymbol{\Phi}_{ij}(k)]\!] = [\![\boldsymbol{K}_{ij}]\!] \otimes [\![\boldsymbol{x}_j(k)]\!] \tag{10}$$

are evaluated in an encrypted fashion through a multiplicative homomorphism of the form (1). The encrypted control scheme is completed by transmitting the encrypted matrix $[\![\boldsymbol{\Phi}(k)]\!]$ to the actuator, where it is decrypted and the control action is computed according to (9). In alignment with Figure 1, the core of an encrypted controller is the ability to compute encrypted control actions based on encrypted system states. Clearly, the previously summarized scheme shows this characteristic. The scheme additionally offers an encryption of the controller parameters $\boldsymbol{K}$, as apparent from the computation of $[\![\boldsymbol{\Phi}(k)]\!]$. The encrypted computation of $[\![\boldsymbol{\Phi}(k)]\!]$ builds on the

Forming Integer Messages From Quantized Data

Most of the cryptosystems currently used for encrypted control offer integer messages spaces of finite size. Hence, to encrypt system states and securely evaluate control laws, states and controller parameters must be mapped onto these message spaces. The first step of this mapping is usually an element-wise approximation of the real-valued states and parameters with fixed-point numbers from the set $\mathbb{Q}_{\beta,\gamma,\delta} := \{-\beta^\gamma, -\beta^\gamma + \beta^{-\delta}, \dots, \beta^\gamma - 2\beta^{-\delta}, \beta^\gamma - \beta^{-\delta}\}$, where $\beta, \gamma, \delta \in \mathbb{N}$ with $\beta \geq 1$ can be understood as the basis, magnitude, and resolution of the set, respectively. For example, for $\beta = 10$ and $\gamma = \delta = 1$, obtain $\mathbb{Q}_{\beta,\gamma,\delta} = \{-10, -9.9, \dots, 9.8, 9.9\}$. The actual quantization is realized with a user-defined mapping $g : \mathbb{R} \to \mathbb{Q}_{\beta,\gamma,\delta}$ (for example, rounding off or up) that provides fixed-point approximations of the form $\hat{x} := g(x)$. We do not further specify the surjective mapping g, apart from the restriction that

$$|g(x) - x| \leq \beta^{-\delta} \quad \text{for every} \quad x \in [-\beta^\gamma, \beta^\gamma]. \tag{S6}$$

In other words, the quantization error should be limited by the resolution for real-valued data in the range of $\mathbb{Q}_{\beta,\gamma,\delta}$. It remains to address the mapping from $\mathbb{Q}_{\beta,\gamma,\delta}$ to a suitable subset of the set of integers. In this context, the central observation is that $\beta^\delta \mathbb{Q}_{\beta,\gamma,\delta} \subset \mathbb{Z}$; that is, scaling $\mathbb{Q}_{\beta,\gamma,\delta}$ with the factor β^δ results in a subset of $\mathbb{Z}$. It is often convenient to provide positive integers in a set of the form $\{0,\dots,\varphi-1\}$, where φ is a user-defined parameter. This can be realized based on the function $f : \mathbb{Z} \to \{0,\dots,\varphi-1\}$ with $f(z) := z \bmod \varphi$. The combination of the quantization via g, the scaling with β^δ, and the mapping f leads to $f(\beta^\delta g(x)) \in \{0,\dots,\varphi-1\}$. The resulting integers can then be encrypted and processed through homomorphic operations. In this context, it is important to note that additions and multiplications are structurally preserved by scaling and the mapping f [59, eq. (10)]. Hence, multiplicatively homomorphic ElGamal and additively homomorphic Paillier can, indeed, be utilized to implement (10) and (11), respectively. Moreover, the scaling and mapping can be (partially) inverted and we refer to [59, Sec. 2(c)] for details on a partial inverse of f. It is easy to see that inverting g, that is, the quantization, is generally not possible. Hence, the corresponding quantization errors must be addressed. As apparent from (S6), such errors can be reduced by increasing β, γ, and δ. However, the cardinality of $\mathbb{Q}_{\beta,\gamma,\delta}$ is simultaneously increased. Thus, when implementing an encrypted controller, the user must tune these parameters to ensure a faithful quantization at a reasonable computation cost.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

multiplicative homomorphism of ElGamal. However, since ElGamal is (originally) not additively homomorphic, summing up the various rows of $\boldsymbol{\Phi}(k)$ to obtain $\boldsymbol{u}(k)$ must be completed at the actuator. As a consequence, the full matrix $[\![\boldsymbol{\Phi}(k)]\!]$, that is, mn ciphertexts, must be transmitted from the cloud to the actuator. Since the original control scheme requires the transmission of only m control inputs, the high communication load of the scheme becomes apparent. In addition, mn decryptions are required at the actuator. Hence, this scheme also reveals more information to the actuator than the required $\boldsymbol{u}(k)$. However, it is important to note that the control gain $\boldsymbol{K}$ remains private.

Alternative Implementation Using Paillier Encryption

The high communication load and the large number of decryptions can be avoided by using the additively homomorphic Paillier cryptosystem instead of multiplicative homomorphic ElGamal encryption. This observation was first made and implemented in [25]. As before, the proposed scheme calls for encryptions of the system states at the sensor, and the resulting ciphertexts $[\![\boldsymbol{x}(k)]\!]$ are again sent to the cloud. However, by exploiting the homomorphism (2) [or more precisely, (S3) and (S5)], the scheme directly computes encrypted inputs in the cloud via

$$[\![\boldsymbol{u}_i(k)]\!] = (\boldsymbol{K}_{i1} \odot [\![\boldsymbol{x}_1(k)]\!]) \oplus \cdots \oplus (\boldsymbol{K}_{in} \odot [\![\boldsymbol{x}_n(k)]\!]) \tag{11}$$

instead of the intermediate result $[\![\boldsymbol{\Phi}(k)]\!]$, as in the preceding. The encrypted inputs $[\![\boldsymbol{u}(k)]\!]$ are then sent to the actuator, where the decryption is completed with the help of (S2).

The presented encrypted controllers based on ElGamal and Paillier encryption, respectively, offer many similarities. First, they both implement the same control law using HE. A second commonality becomes apparent when analyzing the intermediate results $[\![\boldsymbol{\Psi}_{ij}(k)]\!] := \boldsymbol{K}_{ij} \odot [\![\boldsymbol{x}_j(k)]\!]$ in (11). Decrypting the corresponding ciphertext matrix $[\![\boldsymbol{\Phi}(k)]\!]$ would result in the same plaintext matrix as the decryption of $[\![\boldsymbol{\Psi}(k)]\!]$ from (10). The two schemes differ, however, in terms of the pre- and postprocessing of intermediate results. In contrast to $[\![\boldsymbol{\Phi}(k)]\!]$, $[\![\boldsymbol{\Psi}(k)]\!]$ is computed without encrypting $\boldsymbol{K}$, which eventually enables determining $[\![\boldsymbol{u}(k)]\!]$ in the cloud. As a consequence, the communication load and the number of required decryptions at the actuator are reduced. Hence, if encrypting $\boldsymbol{K}$ is not required, encrypted linear control can be more efficiently implemented using Paillier.

Nevertheless, implementing encrypted control using any homomorphic cryptosystem is numerically demanding. This is apparent from Table 1, which lists computation times and operation counts for encryptions, encrypted operations, and decryptions using the ElGamal and Paillier cryptosystems, respectively. Ultimately, the comparison between the two implementations reveals the important tradeoff between privacy and efficiency: concealing more information, such as control gains and system states, incurs more communication cost than concealing only the states.

Efficient Implementation Using Secret Sharing

The computational effort can be significantly reduced if tailored controller architectures and cryptographic tools are considered. As an illustrative example, consider the encrypted linear controller in [59] that is based on secret sharing and two-party computation. At the sensor, the current state $\boldsymbol{x}(k)$ is measured, a random manipulation $\boldsymbol{r}(k)$ is generated, and the shares $\boldsymbol{x}^{(1)}(k) := \boldsymbol{x}(k) + \boldsymbol{r}(k)$ and $\boldsymbol{x}^{(2)}(k) := \boldsymbol{r}(k)$ are specified. These shares are then sent to two noncolluding clouds (for example, operated by two different providers). The linear control law (5) is evaluated for the respective share by computing

$$\boldsymbol{v}^{(1)}(k) := \boldsymbol{K}\boldsymbol{x}^{(1)}(k) \quad \text{and} \quad \boldsymbol{v}^{(2)}(k) := \boldsymbol{K}\boldsymbol{x}^{(2)}(k),$$

respectively. The resulting values $\boldsymbol{v}^{(1)}(k)$ and $\boldsymbol{v}^{(2)}(k)$ are independently transmitted to the actuator, where the control action is determined as $\boldsymbol{u}(k) = \boldsymbol{v}^{(1)}(k) - \boldsymbol{v}^{(2)}(k)$. Since the

TABLE 1 Computation times (in milliseconds) for single encryption, encrypted operation, and decryption using ElGamal and Paillier cryptosystems, respectively, with different (public) key lengths. The listed times reflect average runtimes for Python procedures executing on a 2.7-GHz Intel Core i7-7500U processor. ElGamal encryption was implemented according to [36, pp. 400–401]. The listed periods of use for the different keys reflect the recommendations [48] of the U.S. National Institute of Standards and Technology. The bottom row summarizes the operation counts for the encrypted linear controllers for systems with n states and m control inputs.

		ElGamal			Paillier			
Key Length	Use (Recommended)	Enc	Mul	Dec	Enc	MulByC	Add	Dec
1024	Outdated	1.065	0.023	0.963	1.523	0.017	0.021	0.412
2048	2016–2030	6.191	0.024	5.828	10.511	0.055	0.064	2.887
3072	Beyond 2030	19.633	0.031	18.132	28.731	0.082	0.118	9.54
Operation count		n	mn	mn	n	mn	$m(n-1)$	m

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

manipulation $\boldsymbol{r}(k)$ is newly generated in each time step k, the individual shares $\boldsymbol{x}^{(1)}(k)$ and $\boldsymbol{x}^{(2)}(k)$ contain no information about the actual state $\boldsymbol{x}(k)$ (if some additional technicalities are considered). Likewise, $\boldsymbol{v}^{(1)}(k)$ and $\boldsymbol{v}^{(2)}(k)$ contain no information about $\boldsymbol{u}(k)$. The actual implementation in [59] is based on one-time pads (which were summarized in the preceding) and, similar to the implementations using HE, involves the quantization of the states and controller parameters.

It is important to understand the role of the two clouds. In this context, first note that involving two clouds that are not collaborating is realistic. Nevertheless, it must be guaranteed that the first cloud has no access to the information transmitted to and from the second one and vice versa. Fortunately, this can be easily realized by establishing secure communication channels through standard symmetric encryption schemes (such as the AES [20]). The resulting scheme is illustrated in Figure 2(a). Computationally, it is significantly more efficient than the previously discussed schemes since demanding asymmetric (that is, public key) encryption and encrypted operations are not required. In fact, the needed symmetric encryption is relatively cheap. For instance, an AES encryption (and decryption) with a 128-b key (that offers a security level comparable to the 3072-b keys in Table 1) requires less than 1 ms. As in the case of the solution involving the Paillier scheme, this solution considers public control gains.

ENCRYPTED MODEL PREDICTIVE CONTROL

Encrypted linear controllers serve as an intuitive proof of concept for encrypted control. The previously discussed schemes illustrate that encrypted cloud-based control can be realized "naturally" by using existing cryptographic tools. However, cloud-based implementations of linear control laws are of limited use for real-world applications. Indeed, employing cloud services is more useful for numerically demanding control strategies, such as optimization-based control. A popular control scheme that involves optimization is model predictive control. Model predictive control for linear systems (6) typically builds on solving an optimal control problem (OCP) of the form

$$\begin{aligned} V(\boldsymbol{x}) := \min_{\substack{\tilde{\boldsymbol{x}}(0),\dots,\tilde{\boldsymbol{x}}(N)\\ \tilde{\boldsymbol{u}}(0),\dots,\tilde{\boldsymbol{u}}(N-1)}} \; & \tilde{\boldsymbol{x}}(N)^\top \boldsymbol{P}\tilde{\boldsymbol{x}}(N) + \sum_{\kappa=0}^{N-1} \tilde{\boldsymbol{x}}(\kappa)^\top \boldsymbol{Q}\tilde{\boldsymbol{x}}(\kappa) + \tilde{\boldsymbol{u}}(\kappa)^\top \boldsymbol{R}\tilde{\boldsymbol{u}}(\kappa) \\ \text{subject to } & \tilde{\boldsymbol{x}}(0) = \boldsymbol{x}, \\ & \tilde{\boldsymbol{x}}(\kappa+1) = \boldsymbol{A}\tilde{\boldsymbol{x}}(\kappa) + \boldsymbol{B}\tilde{\boldsymbol{u}}(\kappa), \quad \forall \kappa \in \{0,\dots,N-1\} \\ & \tilde{\boldsymbol{x}}(\kappa) \in \mathcal{X}, \qquad \forall \kappa \in \{0,\dots,N-1\} \\ & \tilde{\boldsymbol{u}}(\kappa) \in \mathcal{U}, \qquad \forall \kappa \in \{0,\dots,N-1\} \\ & \tilde{\boldsymbol{x}}(N) \in \mathcal{T} \end{aligned} \tag{12}$$

in every time step for the current state $\boldsymbol{x} = \boldsymbol{x}(k)$.

The OCP is characterized by the following parameters. The integer N refers to the prediction horizon. The weighting matrices $\boldsymbol{P} \in \mathbb{R}^{n\times n}, \boldsymbol{Q} \in \mathbb{R}^{n\times n}$, and $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ specify the control objective. The sets $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ describe the state and input constraints of the system, respectively. Finally, the terminal set $\mathcal{T} \subseteq \mathcal{X}$ can be used to enforce the stability of the closed-loop system and the recursive feasibility of the MPC (see [45, Sec. 3.3] for details). In this context, the closed-loop dynamics are determined by the feedback

$$\boldsymbol{u}(k) = \tilde{\boldsymbol{u}}^*(0), \tag{13}$$

where $\tilde{\boldsymbol{u}}^*(0),\dots,\tilde{\boldsymbol{u}}^*(N-1), \tilde{\boldsymbol{x}}^*(0),\dots,\tilde{\boldsymbol{x}}^*(N)$ refer to the optimizers for (12). For simplicity, assume throughout the article that $\boldsymbol{P}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ are positive definite and that $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{T}$ are convex polyhedrons. Under these assumptions, (12) is a strictly convex quadratic program (QP) with a unique optimizer. Implementing model predictive control requires solving this QP in every time step. Depending on the size of m, n, and N and the number of hyperplanes describing $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{T}$, solving (12) can be numerically demanding, and a cloud solution becomes beneficial. Hence, encrypted cloud-based model predictive control offers interesting applications.

First Implementation Using Offline Optimization

The first encrypted MPC was introduced in [62]. The proposed implementation builds on the three following observations. First, the OCP (12) can be rewritten as a QP of the form

$$\begin{aligned} \boldsymbol{z}^*(\boldsymbol{x}) = \operatorname{argmin}_{\boldsymbol{z}} \; & \frac{1}{2}\boldsymbol{z}^\top \boldsymbol{H}\boldsymbol{z} + \boldsymbol{x}^\top \boldsymbol{F}^\top \boldsymbol{z} \\ \text{subject to } & \boldsymbol{G}\boldsymbol{z} \le \boldsymbol{E}\boldsymbol{x} + \boldsymbol{h} \end{aligned} \tag{14}$$

with the decision variables

$$\boldsymbol{z} := \begin{pmatrix} \tilde{\boldsymbol{u}}(0) \\ \vdots \\ \tilde{\boldsymbol{u}}(N-1) \end{pmatrix} \in \mathbb{R}^{Nm} \tag{15}$$

and the parameters $\boldsymbol{x} = \boldsymbol{x}(k)$; refer to [8, Sec. 2.1] for details about the computation of the vector $\boldsymbol{h}$ and the matrices $\boldsymbol{E}$, $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{H}$. Second, the parameterized solution of (14) is known to offer a certain structure. The optimizer $\boldsymbol{z}^*(\boldsymbol{x})$ is a piecewise affine function of the parameter $\boldsymbol{x}$. Since

$$\boldsymbol{u}(k) = \boldsymbol{C}\boldsymbol{z}^*(\boldsymbol{x}(k)) \quad \text{with} \quad \boldsymbol{C} := (\boldsymbol{I}_m \;\; 0) \in \mathbb{R}^{m\times Nm}, \tag{16}$$

according to (13), the feedback likewise becomes a piecewise affine function of the form

$$\boldsymbol{u}(k) = \begin{cases} \boldsymbol{K}^{(1)}\boldsymbol{x}(k) + \boldsymbol{b}^{(1)} & \text{if } \boldsymbol{x}(k) \in \mathcal{P}^{(1)}, \\ \vdots & \vdots \\ \boldsymbol{K}^{(s)}\boldsymbol{x}(k) + \boldsymbol{b}^{(s)} & \text{if } \boldsymbol{x}(k) \in \mathcal{P}^{(s)}. \end{cases} \tag{17}$$

The individual affine segments are specified by the matrices $\boldsymbol{K}^{(\sigma)} \in \mathbb{R}^{m\times n}$, vectors $\boldsymbol{b}^{(\sigma)} \in \mathbb{R}^m$, and polyhedra $\mathcal{P}^{(\sigma)} \subseteq \mathcal{X}$. The number of segments s is limited by the (typically conservative) upper bound 2^ϑ (see [8, Sec. 4.4]), where

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

ϑ refers to the number of inequalities in (15). Third, for moderate sizes of m, N, and ϑ, the parameters $\boldsymbol{K}^{(\sigma)}$, $\boldsymbol{b}^{(\sigma)}$, and $\mathcal{P}^{(\sigma)}$ in (17) can actually be computed offline, that is, before the runtime of the controller. A suitable procedure, which can be understood as a multiparametric active set solver, is described in [8, Sec. 4.3]. Based on (17), one can implement an MPC without online optimization. Computing the control action $\boldsymbol{u}(k)$ is then (typically) realized by a two-stage procedure. First, the polyhedron $\mathcal{P}^{(\sigma)}$ that contains the current state $\boldsymbol{x}(k)$ is identified. Second, the corresponding affine segment is evaluated. It easy to see that the second step can be implemented in an encrypted fashion. Given the active segment in terms of the index σ, one can compute

$$[\![\boldsymbol{u}(k)]\!] = (\boldsymbol{K}^{(\sigma)} \odot [\![\boldsymbol{x}(k)]\!]) \oplus [\![\boldsymbol{b}^{(\sigma)}]\!] \tag{18}$$

analogously to (11) by using Paillier. With a slight abuse of the notation, we extend the homomorphic operations $\odot$ and $\oplus$ to matrix–vector products and vector additions, respectively. In this context, evaluating $\boldsymbol{K}^{(\sigma)} \odot [\![\boldsymbol{x}(k)]\!]$ also involves homomorphic additions (for $n > 1$), as apparent from (11).

Hence, an encrypted MPC can be realized as follows. At the sensor, the current state $\boldsymbol{x}(k)$ is measured, and the corresponding $\mathcal{P}^{(\sigma)}$ is identified. Afterward, $\boldsymbol{x}(k)$ is encrypted, and $[\![\boldsymbol{x}(k)]\!]$ is sent to the cloud with the index σ. In the cloud, $[\![\boldsymbol{u}(k)]\!]$ is computed according to (18) and transmitted to the actuator. There, $[\![\boldsymbol{u}(k)]\!]$ is decrypted, and $\boldsymbol{u}(k)$ is applied. The resulting encrypted MPC is illustrated in Figure 3(a). Note that a robust MPC provides an effective way to address the quantization errors that appear during the preparation of (17) for the encrypted implementation in (18) (see "Forming Integer Messages From Quantized Data" for details). Instead of the "nominal" OCP in (12), one can consider the OCP underlying a tube-based, robust MPC and design the disturbance set such that it covers quantization errors that appear. Since the resulting OCP can again be rewritten as a QP of the form (14), an encrypted implementation can be realized analogously to the scheme presented here (see [62, Sec. 3] for details).

We conclude the presentation of the scheme with a brief discussion of the information available to the cloud. Obviously, the clouds obtain the plaintexts $\boldsymbol{K}^{(1)},\ldots,\boldsymbol{K}^{(s)}$ and the ciphertexts $[\![b^{(1)}]\!],\ldots,[\![\boldsymbol{b}^{(s)}]\!]$. During runtime, the cloud additionally gets access to the current index σ and accordingly selects the segment to be evaluated. In principle, knowing σ could enable an intruder to narrow $\boldsymbol{x}(k)$ down to $\mathcal{P}^{(\sigma)}$. However, the cloud has no direct access to the polyhedrons $\mathcal{P}^{(1)},\ldots,\mathcal{P}^{(s)}$. Moreover, under the (realistic) assumption that the cloud has no information about the parameters of the OCP (12), it seems hard to reconstruct $\mathcal{P}^{(\sigma)}$ from knowledge of $\boldsymbol{K}^{(1)},\ldots,\boldsymbol{K}^{(s)}$. Nonetheless, if knowledge of σ that is available to the cloud is considered insecure, the sensor sends σ to the actuator [see Figure 3(a) for an illustration]. Since the cloud has no information about the active segment, it has to evaluate (18) for all $\sigma \in \{1,\ldots,s\}$, and it must transmit all resulting $[\![\boldsymbol{u}(k)]\!]$ to the actuator. There, only the ciphertexts corresponding to segment σ are decrypted and applied.

Comparing both variants, the latter offers more security. However, it is also less efficient since the computation effort in the cloud and the communication load to the actuator are significantly increased. Apart from that, both variants share a common drawback. They both require $\mathcal{P}^{(\sigma)}$ to be identified at the sensor. There exist efficient algorithms for localizing $\mathcal{P}^{(\sigma)}$ that, for example, make use of binary search trees [69]. Neglecting encrypted operations for the moment, identifying $\mathcal{P}^{(\sigma)}$ is significantly more demanding than evaluating the corresponding affine controller segment. In summary, both variants of the proposed scheme show that encrypted cloud-based MPCs can be realized. However, the scheme is applicable only for moderate m, N, and ϑ. Moreover, the computational effort at the sensor is high.

Online Optimization Using Real-Time Iterations

The drawbacks of the previous scheme can be avoided by considering online optimization instead of evaluating (17) offline. Several methods for solving QPs online exist. Examples include interior point methods, active set procedures, and proximal algorithms [52]. The various solvers all iteratively approach the optimal solution. We next summarize the encrypted MPC in [60] that builds on a projected gradient scheme (PGS), that is, a structurally simple proximal algorithm.

To apply the scheme, introduce the set

$$\mathcal{Z}(\boldsymbol{x}) := \{\boldsymbol{z} \in \mathbb{R}^{mN} \,|\, \boldsymbol{G}\boldsymbol{z} \leq \boldsymbol{E}\boldsymbol{x} + \boldsymbol{h}\}$$

that reflects the state-dependent constraints in (14). Then, a (primal) projected gradient descent can be realized by recurrently evaluating the iterations

$$\boldsymbol{z}^{(j+1)} := \mathrm{proj}_{\mathcal{Z}}(\boldsymbol{z}^{(j)} - \rho(\boldsymbol{H}\boldsymbol{z}^{(j)} + \boldsymbol{F}\boldsymbol{x})), \tag{19}$$

where $\boldsymbol{H}\boldsymbol{z} + \boldsymbol{F}\boldsymbol{x}$ is the gradient of the objective in (14), ρ is the step size, and $\mathrm{proj}_{\mathcal{Z}}(\cdot)$ denotes a projection onto $\mathcal{Z}(\boldsymbol{x})$. The choice of the parameter ρ affects the convergence (or divergence) of the scheme. Convergence of the iterates $\boldsymbol{z}^{(j)}$ to the optimizer $\boldsymbol{z}^*(\boldsymbol{x})$ is guaranteed for every $\rho \in (0, 2\lambda_{\max}^{-1}(\boldsymbol{H}))$ [50], where $\lambda_{\max}(\boldsymbol{H})$ refers to the largest eigenvalue of the positive-definite Hessian $\boldsymbol{H}$. Using the iterations (19) to solve (14) is reasonable only if the projection onto $\mathcal{Z}(\boldsymbol{x})$ can be efficiently evaluated. This is, for example, the case if only box-shaped input constraints are present. More precisely, this special case is characterized by

$$\mathcal{X} = \mathcal{T} = \mathbb{R}^n \quad \text{and} \quad \mathcal{U} = \{\boldsymbol{u} \in \mathbb{R}^m \,|\, \boldsymbol{u}_{\min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max}\}, \tag{20}$$

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

where $\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max} \in \mathbb{R}^m$ reflect lower and upper bounds for the inputs, respectively. Under this assumption, $\mathcal{Z}(\boldsymbol{x}) = \{\boldsymbol{z} \in \mathbb{R}^{mN} \,|\, \boldsymbol{z}_{\min} \leq \boldsymbol{z} \leq \boldsymbol{z}_{\max}\}$ are obtained with

$$\boldsymbol{z}_{\min} := \begin{pmatrix} \boldsymbol{u}_{\min} \\ \vdots \\ \boldsymbol{u}_{\min} \end{pmatrix} \in \mathbb{R}^{Nm} \quad \text{and} \quad \boldsymbol{z}_{\max} := \begin{pmatrix} \boldsymbol{u}_{\max} \\ \vdots \\ \boldsymbol{u}_{\max} \end{pmatrix} \in \mathbb{R}^{Nm}.$$

The projection then yields (21), which can (at least in plaintext) be efficiently evaluated:

$$\operatorname{proj}_{\mathcal{Z}}(\boldsymbol{\zeta}) = \min\{\max\{\boldsymbol{z}_{\min}, \boldsymbol{\zeta}\}, \boldsymbol{z}_{\max}\}. \tag{21}$$

Regarding the encrypted implementation, even the simplified projection remains demanding. In contrast, the computation of the projection's argument $\boldsymbol{\zeta} := (\boldsymbol{I}_{Nm} - \rho\boldsymbol{H})\boldsymbol{z}^{(j)} - \rho\boldsymbol{F}\boldsymbol{x}$ in (19) can be easily encrypted analogously to the Paillier-based linear scheme. This observation motivates the encrypted control scheme proposed in [60]. In this approach, only a single solver iteration is evaluated per time step. More precisely, at step k, the cloud evaluates

$$[\![\boldsymbol{\zeta}(k)]\!] = ((\boldsymbol{I}_{Nm} - \rho\boldsymbol{H}) \odot [\![\boldsymbol{z}^{(0)}(k)]\!]) \oplus ((-\rho\boldsymbol{F}) \odot [\![\boldsymbol{x}(k)]\!]) \tag{22}$$

based on encrypted states $[\![\boldsymbol{x}(k)]\!]$ and an encrypted initial guess $[\![\boldsymbol{z}^{(0)}(k)]\!]$ using the Paillier homomorphisms. The cloud then sends $[\![\boldsymbol{\zeta}(k)]\!]$ to the actuator, where it is decrypted and the projection (21) is evaluated to obtain $\boldsymbol{z}^{(1)}(k) = \operatorname{proj}_{\mathcal{Z}}(\boldsymbol{\zeta}(k))$. Finally, the input $\boldsymbol{u}(k) = \boldsymbol{C}\boldsymbol{z}^{(1)}(k)$ is applied analogously to (16).

It remains to comment on the initial guess $\boldsymbol{z}^{(0)}(k)$. The scheme in [72, Sec. 4(a)] suggests reusing $\boldsymbol{z}^{(1)}(k)$ to initialize $\boldsymbol{z}^{(0)}(k+1)$. More precisely, $\boldsymbol{z}^{(1)}(k)$ is forwarded from the actuator to the sensor, buffered, and then used to warm-start

$$\boldsymbol{z}^{(0)}(k+1) = \boldsymbol{D}\boldsymbol{z}^{(1)}(k) \tag{23}$$

at the following sampling instance, where $\boldsymbol{D} \in \mathbb{R}^{Nm \times Nm}$ is a user-defined update matrix. At step k, the sensor sends the encrypted initial guess $[\![\boldsymbol{z}^{(0)}(k)]\!]$ to the cloud along with $[\![\boldsymbol{x}(k)]\!]$. An illustration of the resulting encrypted controller is given in Figure 3(b). From a computational point of view, the previously described scheme is viable. However, from an optimal control perspective, the scheme seems questionable. Surprisingly, the coupling of solver iterations and sampling instances can perform well. The resulting real-time iterations work particularly well for MPCs, since the underlying OCPs for consecutive time steps are often very similar. The good performance and certain stability certificates have even been proven analytically in [72, Sec. 4]. Clearly, the idea of encrypted real-time iterations can also be extended to solvers that can handle more complicated constraints than (20), including state constraints. For instance, encrypted real-time iterations of the alternating direction method of multipliers (ADMM) are considered in [57, Ch. 11.5]. Interestingly, the encrypted ADMM scheme involves encrypted updates of dual variables (or Lagrange multipliers) in the cloud, which causes the same issues as the encrypted realization of the dynamical controller in (7). This relation is, however, not surprising since proximal algorithms, such as the PGS and the ADMM, can often be interpreted as dynamical controllers.

Online Optimization Using Multiparty Computation

Focusing on the computations (22) completed in the cloud, the encrypted real-time iterations can be interpreted as encrypted linear control for an augmented system state (consisting of $\boldsymbol{x}$ and $\boldsymbol{z}^{(0)}$). Hence, the utility of the cloud is again questionable. Moreover, real-time iterations may be stabilizing. However, they are often suboptimal. Both drawbacks can be offset with the encrypted MPC proposed in [2]. In fact, [2] presents a scheme based on two noncolluding clouds [detailed in Figure 2(b)] that evaluates multiple projected gradient iterations per time step in an encrypted fashion. Thus, the computational load in the cloud(s) is increased, while the sensor and the actuator handle only the necessary encryption and decryption of data, respectively.

In addition, multiple solver iterations generally increase the accuracy of the approximations. Convergence to the optimum $\boldsymbol{z}^*(x)$ is further improved in [2] by implementing a fast gradient method. However, for simplicity, the central ingredients of the scheme for the PGS considered previously are presented. Following the concept in [2], the encryption of the PGS iterations (19) can be realized as follows. The first cloud (or server) is concerned with the encrypted evaluation of the affine argument of the projection. Hence, it computes

$$[\![\boldsymbol{\zeta}^{(j+1)}(k)]\!]^{(2)} = ((\boldsymbol{I} - \rho\boldsymbol{H}) \odot [\![\boldsymbol{z}^{(j)}(k)]\!]^{(2)}) \oplus ((-\rho\boldsymbol{F}) \odot [\![\boldsymbol{x}(k)]\!]^{(2)})$$

analogously to (22), where it makes use of an additively homomorphic cryptosystem that has been established by the second cloud, as indicated by the superscript "(2)." Hence, the second cloud provides the public key that enables encryption and homomorphic operations and holds the secret key required for decryption. To complete an iteration j, one must perform the projection $\boldsymbol{z}^{(j+1)}(k) = \operatorname{proj}_{\mathcal{Z}}(\boldsymbol{\zeta}^{(j+1)}(k))$ in an encrypted manner. According to (21), the projection onto the box constraints $\mathcal{Z}$ can be separated into the two steps

$$\bar{\boldsymbol{\zeta}}^{(j+1)}(k) := \max\{\boldsymbol{z}_{\min}, \boldsymbol{\zeta}^{(j+1)}(k)\}, \quad \text{and} \tag{24a}$$

$$\boldsymbol{z}^{(j+1)}(k) = \min\{\bar{\boldsymbol{\zeta}}^{(j+1)}(k), \boldsymbol{z}_{\max}\}. \tag{24b}$$

We explain the encrypted evaluation of (24a) and omit the analogous handling of (24b). The procedure makes use of a protocol for private comparison (see [21] for details) that is briefly summarized in the following. In this context, consider a party that holds two ciphertexts $[\![z_0]\!]$ and $[\![z_1]\!]$ encrypted with an additively homomorphic cryptosystem

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

established by a second party. As detailed in [4, Sec. 9.4.5], private comparison then enables the second party to learn a bit τ satisfying $z_\tau = \max\{z_0, z_1\}$ without revealing τ to the first party or the plaintexts z_0 and z_1 to either of the two entities. Using this protocol, the encrypted evaluation of (24a) can be realized within five steps, as follows:

1) The first cloud encrypts $z_{\min}$ using the public key from the second cloud (or receives $[\![z_{\min}]\!]^{(2)}$ from the sensor) and randomly assigns the elements $[\![\zeta_i^{(j+1)}(k)]\!]^{(2)}$ and $[\![z_{\min,i}]\!]^{(2)}$ to $[\![a_i]\!]^{(2)}$ and $[\![b_i]\!]^{(2)}$ to mask their order.
2) The two clouds run a private comparison protocol on $[\![a]\!]^{(2)}$ and $[\![b]\!]^{(2)}$ that provides a vector $\tau \in \{0,1\}^{Nm}$ (such that $\tau_i = 1$ whenever $a_i \le b_i$) to the second cloud.
3) The first cloud generates random vectors r and s and masks $[\![a]\!]^{(2)}$ and $[\![b]\!]^{(2)}$ by computing $[\![a]\!]^{(2)} \oplus [\![r]\!]^{(2)}$ and $[\![b]\!]^{(2)} \oplus [\![s]\!]^{(2)}$. The resulting ciphertexts $[\![a+r]\!]^{(2)}$ and $[\![b+s]\!]^{(2)}$ are sent to the second cloud.
4) The second cloud defines $[\![c_i]\!]^{(2)}$ as $[\![a_i + r_i]\!]^{(2)} \oplus [\![0]\!]^{(2)}$ whenever $\tau_i = 0$ and as $[\![b_i + s_i]\!]^{(2)} \oplus [\![0]\!]^{(2)}$; otherwise, it does so where the addition of $[\![0]\!]^{(2)}$ implies a refreshment of the encryption. The ciphertexts $[\![c]\!]^{(2)}$ are then returned to the first cloud together with $[\![\tau]\!]^{(2)}$.
5) The first cloud computes, for every $i \in \{1, \ldots, Nm\}$,

$$[\![\tilde{\zeta}_i^{(j+1)}(k)]\!]^{(2)} = [\![c_i]\!]^{(2)} \oplus (r_i \odot ([\![\tau_i]\!]^{(2)} \oplus [\![-1]\!]^{(2)})) \oplus ((-s_i) \odot [\![\tau_i]\!]^{(2)}). \tag{25}$$

The three final steps represent a variant of the oblivious transfer introduced in the "Basics of Homomorphic Encryption, Secret Sharing, and Multiparty Computation" section. Specifically, (25) is a variant of (4). The computation of $[\![z^{(j+1)}(k)]\!]^{(2)}$ based on $[\![\tilde{\zeta}^{(j+1)}(k)]\!]^{(2)}$ and $[\![z_{\max}]\!]^{(2)}$ can be realized analogously. After evaluating $J \ge 1$ iterations, the first cloud holds $[\![z^{(J)}(k)]\!]^{(2)}$, that is, an encrypted approximation of $z^*(x(k))$. To provide $u(k) = Cz^{(J)}(k)$ to the actuator, the encryption must be changed from $[\![\cdot]\!]^{(2)}$ to $[\![\cdot]\!]^{(\mathrm{act})}$, where the superscript "(act)" represents a cryptosystem created by the actuator (or the system operator). The standard way for the first cloud to obtain $[\![u(k)]\!]^{(\mathrm{act})}$ from $[\![u(k)]\!]^{(2)}$ is to send to the second cloud a masked version $[\![u(k)+t]\!]^{(2)}$. The second cloud uses its secret key to first decrypt, and then the public key of the actuator to encrypt, $[\![u(k)+t]\!]^{(\mathrm{act})}$. The first cloud then subtracts the mask to obtain the desired value: $[\![u(k)]\!]^{(\mathrm{act})} = [\![u(k)+t]\!]^{(\mathrm{act})} \oplus [\![-t]\!]^{(\mathrm{act})}$.

It remains to comment on the performance of the scheme. First of all, due to the multiple solver iterations, the scheme usually performs better than the previously presented controller based on real-time iterations. However, multiple iterations may also increase the effect of quantization errors, as specified in [2, Sec. 6]. Still, robustness can be guaranteed for a suitable design of the quantization. Regarding the computational effort of the scheme, we note that the communication load is substantial, specifically for the comparison protocol in step 2 in the preceding. In fact, let l be the number of bits required to represent $[\![a]\!]^{(2)}$ (or $[\![b]\!]^{(2)}$). The necessary communication rounds between the two clouds are then of order l per solver iteration.

ENCRYPTED DISTRIBUTED CONTROL

So far, the article has focused on centralized cloud-based controllers. For many large-scale applications, centralized control strategies are not tractable. For such systems, distributed control schemes are often the method of choice. Distributed control is particularly useful if the system of interest consists of numerous subsystems with local actuating and sensing elements. In the simplest case, the $M \in \mathbb{N}$ subsystems show linear dynamics of the form

$$x^{(i)}(k+1) = A^{(ii)}x^{(i)}(k) + B^{(i)}u^{(i)}(k) + \sum_{j=1, i\neq j}^{M} A^{(ij)}x^{(j)}(k),$$

where $A^{(ii)} \in \mathbb{R}^{n_i \times n_i}$ and $B^{(i)} \in \mathbb{R}^{n_i \times m_i}$ express local dynamics and the various $A^{(ij)} \in \mathbb{R}^{n_i \times n_j}$ (for $i \neq j$) represent physical couplings between the subsystems. Importantly, distributed control usually aims for a common goal, for example, constant power frequency in a smart grid. Realizing such a goal through local control actions $u^{(i)}$ requires cooperation (and hence communication between the agents). In this context, an agent represents one subsystem and the corresponding local sensor, control, and actuator (see Figure 4).

In general, communication will not take place between every pair of agents. The graph describing the communication network is usually incomplete. Assume that all communication links are bidirectional and that every pair of agents can, in principle, interchange information through a direct link and other agents. In addition, we assume that no parallel links exist. In other words, consider a communication graph that is undirected, connected, and simple. More formally, the communication graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is characterized as follows. The vertex set $\mathcal{V} := \{1, \ldots, M\}$ and the edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ represent the M agents and the communication links between them, respectively. If $(i, j) \in \mathcal{E}$, then the agents i and j (with $i \neq j$) are communicating with each other (and called neighbors). The set $\mathcal{N}_i := \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$ collects all neighbors of agent i. The bidirectionality of the communication then implies $i \in \mathcal{N}_j$ whenever $j \in \mathcal{N}_i$ [for all $(i, j) \in \mathcal{E}$].

Cooperative Control Using Sparse Feedback

Any cooperative control scheme must comply with the communication network. In this context, a structurally simple approach is a sparse linear control scheme of the form

$$\begin{pmatrix} u^{(1)}(k) \\ \vdots \\ u^{(M)}(k) \end{pmatrix} = \begin{pmatrix} K^{(11)} & \ldots & K^{(1M)} \\ \vdots & \ddots & \vdots \\ K^{(M1)} & \ldots & K^{(MM)} \end{pmatrix} \begin{pmatrix} x^{(1)}(k) \\ \vdots \\ x^{(M)}(k) \end{pmatrix}, \tag{26}$$

where the off-diagonal block matrices $K^{(ij)} \in \mathbb{R}^{m_i \times n_j}$ are zero matrices whenever $j \notin \mathcal{N}_i$, that is, whenever no communication link exists. The local controllers can then be written as

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

$$\boldsymbol{u}^{(i)}(k) = \boldsymbol{K}^{(ii)}\boldsymbol{x}^{(i)}(k) + \sum_{j\in\mathcal{N}_i} \boldsymbol{K}^{(ij)}\boldsymbol{x}^{(j)}(k). \tag{27}$$

While structurally simple, designing such distributed controllers is not trivial; refer to [44] for details of tailored optimization-based design procedures. As requested, evaluating $\boldsymbol{u}^{(i)}(k)$ according to (27) requires only information about local state $\boldsymbol{x}^{(i)}(k)$ and neighboring states $\boldsymbol{x}^{(j)}(k)$. Typically, information regarding neighboring agents is provided by directly communicating the states $\boldsymbol{x}^{(j)}(k)$ between the agents. This can be a security threat if the local state information is sensitive and if neighboring agents are curious about the data.

Encrypted Cooperative Control

Encrypted control facilitates overcoming this threat and ensuring the confidentiality of the local states. The following briefly summarizes the implementation proposed in [61]. We note that encrypted cooperative control was previously realized in [26, Sec. 4] for a randomized control scheme. The first step of the procedure in [61] does not involve any encryption. It simply proposes to compute the "control shares"

$$\boldsymbol{v}^{(ij)}(k) \coloneqq \boldsymbol{K}^{(ij)}\boldsymbol{x}^{(j)}(k) \tag{28}$$

at agent j and to communicate $\boldsymbol{v}^{(ij)}(k)$ instead of $\boldsymbol{x}^{(j)}(k)$. At agent i, the control action is then computed via

$$\boldsymbol{u}^{(i)}(k) = \boldsymbol{K}^{(ii)}\boldsymbol{x}^{(i)}(k) + \sum_{j\in\mathcal{N}_i} \boldsymbol{v}^{(ij)}(k) \tag{29}$$

instead of (27). This small modification already hides the local states $\boldsymbol{x}^{(j)}(k)$ from agent i. Without any information about $\boldsymbol{K}^{(ij)}$ and $\boldsymbol{x}^{(j)}(k)$ (including the dimension nj), it is hard to reconstruct $\boldsymbol{x}^{(j)}(k)$ from $\boldsymbol{v}^{(ij)}(k)$. Moreover, even if information concerning $\boldsymbol{K}^{(ij)}$ is available to, or can be reconstructed by, agent i, retrieving $\boldsymbol{x}^{(j)}(k)$ leads to an observability problem that requires further information about $\boldsymbol{A}^{(jj)}$, $\boldsymbol{B}^{(j)}$, and $\boldsymbol{u}^{(j)}(k)$ (see [5, Sec. 6] for details).

The downside of the proposed modification is that agent j would get access to the matrix $\boldsymbol{K}^{(ij)}$ and the control shares $\boldsymbol{v}^{(ij)}(k)$. Both quantities could disclose information about the control strategy of the neighboring agent i, which could likewise be a security threat. This threat can, however, be eliminated using HE. To see this, we first note that the computation of the control shares (28) is very similar to the evaluation of the linear controller (5). Hence, an encrypted computation of $\boldsymbol{v}^{(ij)}(k)$ can be realized by evaluating

$$[\![\boldsymbol{v}^{(ij)}(k)]\!]^{(i)} = [\![K^{(ij)}]\!]^{(i)} \odot x^{(j)}(k) \tag{30}$$

using additively HE similar to (11), where the superscript "(i)" indicates that the public key of agent i is used for encryption. As a consequence, multiple cryptosystems are simultaneously applied; their interactions are specified shortly. For the moment, note that $[\![\boldsymbol{K}^{(ij)}]\!]^{(i)}$ will be provided to agent j before runtime by some trustworthy entity (and not by agent i).

Comparing (11) and (30), note that the two implementations differ in terms of the partially encrypted multiplications of the controller parameters and the system states. More precisely, in (30), the controller parameters are encrypted, and the states are plaintexts, whereas the roles are swapped in (11). Nevertheless, both variants can be realized using a homomorphism of the form (S5). The differences between (11) and (30) reflect slightly different setups. Equation (11) is evaluated in the cloud, which should not obtain any information about the system states. In contrast, (30) is computed at agent j, which should not get access to $\boldsymbol{K}^{(ij)}$ (and which necessarily measures $\boldsymbol{x}^{(j)}$ in plaintext). However, both setups coincide in that neither the cloud nor the agent should get information about the computed control actions and shares, respectively.

As briefly mentioned, $[\![\cdot]\!]^{(i)}$ stands for encryption with the public key of agent i. As a consequence, only agent i is capable of decrypting $[\![\boldsymbol{v}^{(ij)}(k)]\!]^{(i)}$ and, in principle, $[\![\boldsymbol{K}^{(ij)}]\!]^{(i)}$ by using its secret key. Since the multiagent system consists of M agents, M cryptosystems are in simultaneous use. They are generated before runtime by the individual agents along with the corresponding secret keys. While the latter are kept private, the public keys are broadcast between the agents. Another item that must be generated before runtime is the (sparse) control law (26). The corresponding procedure in [44] requires a centralized design. The resulting block matrices $\boldsymbol{K}^{(ij)}$ contain sensitive information. Agent j, knowing $\boldsymbol{K}^{(ij)}$, would render the encrypted computation (30) useless since computing $\boldsymbol{v}^{(ij)}(k)$ in plaintext would then be trivial for agent j.

To overcome this issue, assume that the centralized controller design is completed by some trustworthy entity, for example, a system operator. After computing the block matrices $\boldsymbol{K}^{(ij)}$, this entity repeats the following two steps for every agent $i \in \{1, \ldots, M\}$. First, it sends $\boldsymbol{K}^{(ii)}$ to agent i via a secure channel. Second, for every neighbor $j \in \mathcal{N}_i$, it encrypts $\boldsymbol{K}^{(ij)}$ through the public key of agent i and sends the result $[\![\boldsymbol{K}^{(ij)}]\!]^{(i)}$ to agent j. During runtime, every agent executes data dissemination and assimilation phases in every time step k. During the data dissemination phase, every agent j initially measures its own state $\boldsymbol{x}^{(j)}$. It then computes the encrypted control shares $[\![\boldsymbol{v}^{(ij)}(k)]\!]^{(i)}$ according to (30) for every $i \in \mathcal{N}_j$ and sends the result to the neighbor i. During the assimilation phase, every agent i first receives $[\![\boldsymbol{v}^{(ij)}(k)]\!]^{(i)}$ from all the neighbors $j \in \mathcal{N}_i$. It then decrypts the shares by using its secret key and computes the local control action according to (29). The resulting encrypted cooperative control scheme is presented in Figure 4(b).

Improving Security

The key challenge in encrypted cooperative control is to ensure the privacy of the local state $\boldsymbol{x}^{(j)}(k)$ while being required to compute (27), that is, the control action of the neighboring agent i. Providing $\boldsymbol{v}^{(ij)}(k)$ instead of $\boldsymbol{x}^{(j)}(k)$ to agent i without revealing $\boldsymbol{v}^{(ij)}(k)$ and $\boldsymbol{K}^{(ij)}$ to agent j is a

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

significant step toward a solution to this challenge. However, depending on the available information about the networked control system, it might be possible for agent i to reconstruct $\boldsymbol{x}^{(j)}(k)$ from $\boldsymbol{v}^{(ij)}(k)$ and data from previous time steps. In principle, there is an obvious way to render the reconstruction problem more difficult. Providing the aggregated share

$$\boldsymbol{w}^{(i)}(k) := \sum_{j \in \mathcal{N}_i} \boldsymbol{v}^{(ij)}(k) \tag{31}$$

to agent i instead of the individuals shares $\boldsymbol{v}^{(ij)}(k)$ makes it significantly harder to reconstruct the individual $\boldsymbol{x}^{(j)}(k)$ presupposed $|\mathcal{N}_i| \geq 2$.

Consider, for example, the case of dynamically independent subsystems without physical couplings, that is, $\boldsymbol{A}^{(ij)} = \boldsymbol{0}$ whenever $i \neq j$, and let $\mathcal{N}_i = \{j_1, \ldots, j_{|\mathcal{N}_i|}\}$. It is easy to show that the pair

$$\left(\begin{pmatrix} \boldsymbol{A}^{(i,j_1)} & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{A}^{(i,j_{|\mathcal{N}_i|})} \end{pmatrix}, \begin{pmatrix} \boldsymbol{K}^{(i,j_1)} & \ldots & \boldsymbol{K}^{(i,j_{|\mathcal{N}_i|})} \end{pmatrix} \right)$$

can be unobservable (which is desirable here) even if all individual pairs $(\boldsymbol{A}^{(ij)}, \boldsymbol{K}^{(ij)})$ are observable (see [5, Sec. 7] for details). Designing an efficient protocol that provides $\boldsymbol{w}^{(i)}(k)$ to agent i without revealing $\boldsymbol{v}^{(ij)}(k)$ is not straightforward. A related problem (called private sum aggregation) has a long history of dedicated research [9], [11], [35], [65]. Conceptually, one strategy implemented in [5] is as follows. At every time step k, the neighbors of agent i agree to some random shares $\boldsymbol{r}^{(ij)}(k)$ that sum up to the zero vector in $\mathbb{R}^{m_i}$; that is,

$$\sum_{j \in \mathcal{N}_i} \boldsymbol{r}^{(ij)}(k) = \boldsymbol{0}. \tag{32}$$

More precisely, these shares should be as described in the "Efficient Implementation Using Secret Sharing" section. All neighboring agents then send $\boldsymbol{v}^{(ij)}(k) + \boldsymbol{r}^{(ij)}(k)$ to agent i instead of $\boldsymbol{v}^{(ij)}(k)$. Now, although agent i receives $|\mathcal{N}_i|$ individual shares, it is impossible to reconstruct the individual $\boldsymbol{v}^{(ij)}(k)$ since the random shares $\boldsymbol{r}^{(ij)}(k)$ are unknown to agent i. However, by exploiting relation (32), summing the received shares immediately leads to (31) and thus facilitates the evaluation of (29). Regarding the secure implementation, agent j computes

$$(\llbracket \boldsymbol{K}^{(ij)} \rrbracket^{(i)} \odot \boldsymbol{x}^{(j)}(k)) \oplus \llbracket \boldsymbol{r}^{(ij)}(k) \rrbracket^{(i)}$$

instead of (30) and submits the resulting ciphertexts to agent i. As a result, an encrypted control scheme is obtained that still hides neighboring control strategies in terms of $\boldsymbol{K}_{ij}$ by using HE. In addition, masking by random values is used to keep the control shares $\boldsymbol{v}^{(ij)}(k)$ of neighboring agents secret.

Not surprisingly, the improved security comes at a price. Preserving privacy requires different shares $\boldsymbol{r}^{(ij)}(k)$ at every time step k. Recurrently generating these shares of zero is computationally demanding. One solution is to contract a trusted entity (for example, a system operator) that generates the secret shares of zero for one (or more) time step(s) in advance and distributes them to the agents in the network. However, depending on the size of the network, the random number generation can be very costly (both in terms of computation time and storage space). Alternatively, a decentralized method to generate the secret shares of zero can be enforced. As discussed in [5], the neighboring agents of an agent i then require two communication rounds to agree on suitable shares $\boldsymbol{r}^{(ij)}(k)$ for an upcoming time step k.

QUANTIZATION EFFECTS, STABILITY GUARANTEES, AND CONTINUOUS OPERATION

For the simplicity of exposition, many technical details have been omitted during the presentation of the various encrypted control schemes. This section notes important details for the correct functioning of encrypted control. First, the implementation of all presented encrypted control schemes requires quantization, for example, the system states $\boldsymbol{x}_j(k)$ and controller parameters (such as $\boldsymbol{K}_{ij}$ for the linear schemes), to enable encryption and secret sharing. As discussed in "Forming Integer Messages From Quantized Data," the quantization is irreversible and consequently leads to errors. Carefully investigating such errors is essential since they may destabilize a control loop [22], [33]. Hence, we illustrate quantization effects for linear controllers and note that an analogous analysis can be carried out for the other schemes. With quantization,

$$\hat{\boldsymbol{u}}(k) := \hat{\boldsymbol{K}}\hat{\boldsymbol{x}}(k), \tag{33}$$

is applied instead of (5), where $\hat{\boldsymbol{K}}$ and $\hat{\boldsymbol{x}}(k)$ refer to quantized parameters and states, respectively, and $\hat{\boldsymbol{u}}(k)$ denotes the resulting control action.

A classical approach is to interpret the quantization-based deviations $\hat{\boldsymbol{u}}(k) - \boldsymbol{u}(k)$ as disturbances [22, Sec. 3]. Following this approach, which has been successfully applied to the encrypted control schemes in [2] and [60]–[62], the closed-loop dynamics for a linear plant (6) controlled by (33) can be written as

$$\boldsymbol{x}(k+1) = (\boldsymbol{A} + \boldsymbol{BK})\boldsymbol{x}(k) + \boldsymbol{Bd}(k) \tag{34}$$

with the state-dependent quantization errors

$$\boldsymbol{d}(k) := \hat{\boldsymbol{K}}\hat{\boldsymbol{x}}(k) - \boldsymbol{K}\boldsymbol{x}(k) = \hat{\boldsymbol{K}}(\hat{\boldsymbol{x}}(k) - \boldsymbol{x}(k)) + (\hat{\boldsymbol{K}} - \boldsymbol{K})\boldsymbol{x}(k) \tag{35}$$

that are regarded as disturbances hereafter. The dynamics (34) can now be investigated using standard techniques from robust control. For example, under the assumption that the original control law (5) has been designed such that $\boldsymbol{A} + \boldsymbol{BK}$ is (Schur) stable, it is straightforward to certify the input-to-state stability with respect to $\boldsymbol{d}$ [34, Example 3.4]. Moreover, additionally assuming bounded disturbances

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

Guaranteeing privacy usually increases both the computation effort and the communication load associated with networked control.

$\boldsymbol{d}(k)$, robust positively invariant (RPI) sets (see, for example, [41]) can be considered. Interestingly,

$$\|\hat{\boldsymbol{x}}(k)-\boldsymbol{x}(k)\|_\infty \leq \beta^{-\delta} \quad \text{and} \quad |\hat{\boldsymbol{K}}_{ij}-\boldsymbol{K}_{ij}| \leq \beta^{-\delta}$$

whenever $\|\boldsymbol{x}(k)\|_\infty \leq \beta^{\gamma}$ and $|\boldsymbol{K}_{ij}| \leq \beta^{\gamma}$, according to (S6) in "Forming Integer Messages From Quantized Data." Hence, bounds for $\boldsymbol{d}(k)$, as in (35), can be guaranteed and even kept arbitrarily small by suitably designing the quantization in terms of the parameters $\beta, \gamma, \delta \in \mathbb{N}$ and by restricting the attention to states in the set $\mathcal{X}_{\text{res}} := \{x \in \mathbb{R}^n \,|\, \|x(k)\|_\infty \leq \beta^{\gamma}\}$. The disturbances $\boldsymbol{d}(k)$ are then contained in the bounded set

$$\mathcal{D} := \{\boldsymbol{d} = \hat{\boldsymbol{K}}\Delta\boldsymbol{x} + (\hat{\boldsymbol{K}}-\boldsymbol{K})\boldsymbol{x} \in \mathbb{R}^m \,|\, \|\Delta x\|_\infty \leq \beta^{-\delta}, \boldsymbol{x} \in \mathcal{X}_{\text{res}}\}.$$

A set $\mathcal{R} \subset \mathbb{R}^n$ is RPI for (34), subject to the constraints $\boldsymbol{x}(k) \in \mathcal{X}_{\text{res}}$ and $\boldsymbol{d}(k) \in \mathcal{D}$, if

$$(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K})\mathcal{R} + B\mathcal{D} \subseteq \mathcal{R} \quad \text{and} \quad \mathcal{R} \subseteq \mathcal{X}_{\text{res}}, \tag{36}$$

where the set-valued addition refers to a Minkowski sum. Classical results on RPI sets (as, for example, collected in [41]) can now be used to investigate the closed-loop behavior of an encrypted control scheme. For instance, it is easy to see that every state trajectory of (34) that starts in the maximal RPI set, that is, the largest set $\mathcal{R}$ satisfying (36), converges to the minimal RPI set [41, Remark 4.1] without leaving $\mathcal{X}_{\text{res}}$.

While the conservative interpretation of quantization errors as disturbances works sufficiently well for many encrypted control schemes, quantized control is an established field of research (see, for example, [19], [33], and [49] for an overview) and more sophisticated approaches exist. Applying these techniques could be an interesting direction for future research. However, quantization effects are typically not the bottleneck for most existing encrypted controllers since quantization is a design parameter. In other words, if the original control law is stabilizing, then there usually exists a suitable quantization that preserves (robust) stability.

Depending on the nature of the cryptosystem underlying an encrypted controller, errors other than those related to quantization can occur. For example, HE schemes based on the learning-with-errors hardness problem [55] require some noise to ensure confidentiality. Similar to the preceding discussion, such noise can be interpreted as a disturbance, and its effects can be accordingly bounded. For a single encrypted operation, neither the quantization nor the noise will significantly affect the computation. However, if the errors due to quantization and noise accumulate, undesired effects may result. Such an accumulation appears, for example, for encrypted implementations of dynamical controllers, as in (7). Iterating the encrypted controller states in the cloud usually implies an accumulation of quantization errors and noise, which might impede continuous and seamless operation for an unbounded runtime. A continuous operation can be easily realized by regular "refreshments" of the encrypted controller state through the client as, for example, considered in [3], [47], and [58]. Since a "refreshment" usually implies decryption and re-encryption, such solutions can sometimes be costly and inconvenient.

More sophisticated schemes that ensure continuous and seamless operation have been proposed in [15], [37], [38], and [67]. In [37], the bootstrapping technique from fully HE is used to enable the cloud to complete necessary refreshments on its own. However, since bootstrapping is still a very expensive operation, [37] employs three controllers operating at the same time for the same system. In contrast, special transformations are considered in [15] and [38] to ensure that the accumulated errors stay within certain limits. Finally, [67] designed a stabilizing quantizer for a dynamic controller encrypted with the multiplicatively homomorphic ElGamal scheme, and [47] discussed the reset of the dynamic controller's state using the additively homomorphic Paillier scheme.

CURRENT CHALLENGES AND FUTURE RESEARCH DIRECTIONS

We presented various encrypted control schemes to illustrate different controller types, suitable architectures, and useful cryptographic tools. Although encrypted control is a young field of research, it is quickly emerging, and the number of proposed schemes in the literature is already too large to be comprehensively covered in such an introductory article. Nevertheless, the presented selection illustrates two important characteristics of encrypted control. First, realizing encrypted controllers requires in-depth knowledge of both cryptography and control theory. Encrypted control typically builds on suitable controller reformulations in combination with tailored cryptographic methods. Second, guaranteeing privacy usually increases both the computation effort and the communication load associated with networked control. This is, for example, apparent from the fact that homomorphically encrypted ciphertexts are typically significantly longer (measured in bits) than the corresponding plaintexts. For instance, ciphertexts resulting from the Paillier cryptosystem for a public key with 2048 b have 4096 b (see Table 1 and "Paillier

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

Encryption and Its Homomorphisms"). Regarding multiparty computation, the communication load is increased since the number of communicating parties typically grows compared to nonshared realizations.

The presented encrypted controllers also share some limitations that are common for existing schemes. However, that will be overcome by future realizations. First, partial HE (such as Paillier and ElGamal) and relatively simple multiparty computations dominate the current generation of encrypted controllers, and modern cryptosystems have not been frequently employed. Two exceptions to this observation are the controllers proposed in [37] and [3] that apply (only recently published) fully respectively leveled HE schemes. Second, existing encrypted controllers often rely on simplified and unrealistic setups. These simplifications include neglected latency, costly computations at sensors and actuators, and underused clouds. Third, most existing controllers require an "unencrypted" offline design of the scheme, and privacy guarantees are exclusively given for the online computations.

In summary, the main bottleneck of encrypted control is the high additional computational effort and communication load. Hence, methods reducing this overload are of great interest. In this context, more realistic setups that include, for example, communication via lossy networks with latency, would further promote real-world applications, such as those in "Prospective Uses of Encrypted Control in Industry." Moreover, from a control perspective, the encryption of more powerful controllers and robust stability certificates despite quantization errors and cryptograhic noise are needed. Finally, combining encrypted control schemes with other techniques ensuring the *security* [66] and *safety* [6] of control systems would be desirable. The previous challenges indicate some interesting directions for future research into encrypted control. Researchers are invited to contribute to this challenging and emerging field, which includes the following concrete problems:

1) proposing encrypted implementations of more complex and nonlinear control schemes and realizing the encrypted design of controllers and their parameters
2) reducing the computational effort and communication load associated with encrypted control, exploiting, for example, hardware acceleration and compilers for HE
3) considering quantization schemes that are not based on fixed-point numbers and further investigating the effect of quantization errors while providing robust stability certificates for encrypted control systems
4) implementing encrypted control in distributed applications, such as smart grids and vehicular networks, and verifying its effectiveness while investigating the effect of, and counteracting, lossy communications and latency
5) considering "active" attacks (that is, malicious parties) and coupling encrypted control with complementary schemes focusing on the integrity and availability of process data
6) ensuring the safety of control systems without reducing security, which is challenging since most known techniques for ensuring safe control are highly nonlinear and require access to plenty of process data
7) using techniques from encrypted control to secure industrial applications, such as control as a service, smart grids, and transportation systems.

Regarding the latter challenge, note that some encrypted computations are already deployed in industry. For instance, distributed systems, such as blockchains, make use of cryptographic tools for coordination. Specifically, privacy-preserving frameworks, including Zcash [32], utilize heavy cryptography for performing transactions without leaking information. Furthermore, secure federated learning by Google [1], [11] exploits private aggregation to train a model from a multitude of different agents while reducing the amount of information collected. Such implementations focus on scalability, speed, and coordination and could thus serve as guidance for deploying encrypted control in practice.

ACKNOWLEDGMENTS

M. Schulze Darup and A.B. Alexandru contributed equally to this work and share the first authorship. Moritz Schulze Darup gratefully acknowledges the support of the German Research Foundation, under grant SCHU 2940/4-1.

AUTHOR INFORMATION

Moritz Schulze Darup (moritz.schulzedarup@tu-dortmund.de) received the diploma degree (M.Sc. degree) in mechanical engineering, the B.Sc. degree in physics, and the Ph.D. degree in control engineering from Ruhr University Bochum, Germany, in 2008, 2010, and 2014, respectively. From 2014 to 2016, he was a postdoctoral researcher at Oxford University, U.K. From 2017 to 2020, he was affiliated with Paderborn University, Germany, where he first served as a lecturer in the Automatic Control Group. Subsequently, he became an assistant professor of encrypted control and led an independent research group funded by the German Research Foundation. Since October 2020, he has been a professor of control and cyberphysical systems at TU Dortmund University, Dortmund, 44227, Germany. His research interests include secure, optimal, and predictive control for networked systems.

Andreea B. Alexandru received the B.Eng. degree from the Department of Automatic Control and Systems Engineering, "Politehnica" University of Bucharest, Romania, in 2015. In 2021, she obtained the Ph.D. degree from the Department of Electrical and Systems Engineering, University of Pennsylvania, advised by George J. Pappas. She is currently a postdoctoral associate at the University of Maryland, College Park, 20742, USA, in the Department of Computer Science at the College of Computer, Mathematical and Natural Sciences. At the University of Pennsylvania, she was affiliated with the Seclab: Computer Security Research

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

at Penn, the General Robotics, Automation, Sensing, and Perception (GRASP) Laboratory, and the Penn Research in Embedded Computing and Integrated Systems Engineering Center. Her research interests include applied cryptography, as well as the privacy and security of control systems and machine learning. She holds three nominations for student best article awards at the International Conference on Cyberphysical Systems, American Control Conference, and International Symposium on Advanced Topics in Electrical Engineering. In 2019, she received a National Science Foundation Impact: Redefining Electrical and Computer Engineering Faculty professional development award and was selected as a rising star in electrical engineering and computer sciences.

Daniel E. Quevedo received the Ingeniero Civil Electrónico and M.Sc. degrees from Universidad Técnica Federico Santa María, Valparaíso, Chile, in 2000, and in 2005, he received the Ph.D. degree from the University of Newcastle, Australia. He is a professor of cyberphysical systems in the School of Electrical Engineering and Robotics, Queensland University of Technology (QUT), Brisbane, Queensland, 4000, Australia. Before joining QUT, he established and led the chair of Automatic Control, Paderborn University, Germany. He was supported by a full scholarship from the alumni association during his time at the Universidad Técnica Federico Santa María and received several university prizes upon graduating. He received the IEEE Conference on Decision and Control Best Student Article Award in 2003 and was also a finalist in 2002. In 2009, he was awarded a five-year research fellowship from the Australian Research Council. He is a corecipient of the 2018 IEEE Transactions on Automatic Control George S. Axelby Outstanding Article Award. His research interests are in networked control systems, control of power converters, and cyberphysical systems security.

George J. Pappas received the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, in 1998. He is currently the UPS Foundation Professor and chair of the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, Pennsylvania, 19104, USA, where he also holds a secondary appointment in the Department of Computer and Information Sciences and the Department of Mechanical Engineering and Applied Mechanics. He is a member of the General Robotics, Automation, Sensing, and Perception Lab, University of Pennsylvania, and the Penn Research in Embedded Computing and Integrated Systems Engineering Center, University of Pennsylvania. He previously served as the deputy dean for research in the School of Engineering and Applied Science. His research interests include control theory, specifically, hybrid systems, embedded systems, cyberphysical systems, and hierarchical and distributed control systems (with applications to unmanned aerial vehicles, distributed robotics, green buildings, and biomolecular networks). He has received various awards, such as the IEEE Control Systems Society Antonio Ruberti Young Researcher Prize, IEEE Transactions on Automatic Control George S. Axelby Outstanding Article Award, O. Hugo Schuck Best Article Award, the George H. Heilmeier Faculty Award for Excellence in Research, the National Science Foundation Presidential Early Career Award for Scientists and Engineers, and numerous best student article awards at the American Control Conference, IEEE Conference on Decision and Control, and International Conference on Cyberphysical Systems.

REFERENCES

[1] Google AI. "Federated learning." https://federated.withgoogle.com/\#learn (accessed May 10, 2020).

[2] A. B. Alexandru, M. Morari, and G. J. Pappas, "Cloud-based MPC with encrypted data," in *Proc. 57th Conf. Decis. Control*, 2018, pp. 5014–5019. doi: 10.1109/CDC.2018.8619835.

[3] A. B. Alexandru and G. J. Pappas, "Encrypted LQG using labeled homomorphic encryption," in *Proc. 10th ACM/IEEE Intl. Conf. Cyber-Phys. Syst.*, 2019, pp. 129–140. doi: 10.1145/3302509.3311049.

[4] A. B. Alexandru and G. J. Pappas, "Secure multi-party computation for cloud-based control," in *Privacy in Dynamical Systems*, F. Farokhi, Ed. Berlin: Springer-Verlag, 2020, pp. 179–207.

[5] A. B. Alexandru, M. Schulze Darup, and G. J. Pappas, "Encrypted cooperative control revisited," in *Proc. 58th Conf. Decis. Control*, 2019, pp. 7196–7202. doi: 10.1109/CDC40024.2019.9030124.

[6] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *Trans. Automat. Control*, vol. 62, no. 8, pp. 3861–3876, 2016. doi: 10.1109/TAC.2016.2638961.

[7] M. Bellare, V.T. Hoang, and P. Rogaway, "Foundations of garbled circuits," in *Proc. Conf. Comput. Commun. Security*, 2012, pp. 784–796. doi: 10.1145/2382196.2382279.

[8] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002. doi: 10.1016/S0005-1098(01)00174-1.

[9] F. Benhamouda, M. Joye, and B. Libert, "A new framework for privacy-preserving aggregation of time-series data," *ACM Trans. Info. Syst. Security*, vol. 18, no. 3, p. 10, 2016. doi: 10.1145/2873069.

[10] M. Bishop, *Computer Security: Art and Science*. New York: Macmillan, 2002.

[11] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 1175–1191. doi: 10.1145/3133956.3133982.

[12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Computat. Theory*, vol. 6, no. 3, pp. 13:1–13:36, 2014. doi: 10.1145/2633600.

[13] A. A. Cardenas, S. Amin, and S. Sastry. "Secure control: Towards survivable cyber-physical systems," in *Proc. 28th Intl. Conf. Distrib. Comput. Syst. Workshops*, 2008, pp. 495–500.

[14] T. M. Chen and S. Abu-Nimeh, "Lessons from Stuxnet," *Computer*, vol. 44, no. 4, pp. 91–93, 2011. doi: 10.1109/MC.2011.115.

[15] J. H. Cheon, K. Han, H. Kim, J. Kim, and H. Shim. "Need for controllers having integer coefficients in homomorphically encrypted dynamic system," in *Proc. 57th Conf. Decis. Control*, 2018, pp. 5020–5025.

[16] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. Int. Conf. Theory Appl. Cryptol Info. Security*, 2017, pp. 409–437. doi: 10.1007/978-3-319-70694-8_15.

[17] M. S. Chong, H. Sandberg, and A. M. H. Teixeira, "A tutorial introduction to security and privacy for cyber-physical systems," in *Proc. 2019 Euro. Control Conf.*, pp. 968–978. doi: 10.23919/ECC.2019.8795652.

[18] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2001, pp. 280–300. doi: 10.1007/3-540-44987-6_18.

[19] R. E. Curry, *Estimation and Control with Quantized Measurements*. Cambridge, MA: MIT Press, 1970.

[20] J. Daemen and V. Rijmen, "The Rijndael block cipher: AES proposal," in *Proc. First Candidate Conf., (AeS1)*, 1999, pp, 343–348

[21] I. Damgård and M. Geisler, and "Mikkel Krøigaard. Efficient and secure comparison for on-line auctions," in *Proc. Australasian Conf. Info. Security Privacy*, 2007, pp. 416–430. doi: 10.1007/978-3-540-73458-1_30.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.

[22] D. F. Delchamps, "Stabilizing a linear system with quantized state feedback," *IEEE Trans. Autom. Control*, vol. 35, no. 8, pp. 916–924, 1990. doi: 10.1109/9.58500.
[23] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, 1985. doi: 10.1109/TIT.1985.1057074.
[24] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Found. Trends[textregistered] Privacy Security*, vol. 2, nos. 2–3, pp. 70–246, 2018. doi: 10.1561/3300000019.
[25] F. Farokhi, I. Shames, and N. Batterham, "Secure and private cloud-based control using semi-homomorphic encryption," in *Proc. 6th IFAC Workshop Distrib. Estimation Control Netw. Syst.*, 2016, pp. 163–168.
[26] F. Farokhi, I. Shames, and N. Batterham, "Secure and private control using semi-homomorphic encryption," *Control Eng. Pract.*, vol. 67, pp. 13–20, Oct. 2017. doi: 10.1016/j.conengprac.2017.07.004.
[27] T. Fujita, K. Kogiso, K. Sawada, and S. Shin, "Security enhancements of networked control systems using RSA public-key cryptosystem," in *Proc. 10th Asian Control Conf.*, 2015, pp. 1–6. doi: 10.1109/ASCC.2015.7244402.
[28] C. Gentry, "Computing arbitrary functions of encrypted data," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 2010. doi: 10.1145/1666420.1666444.
[29] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci.*, 1982, pp. 160–164. doi: 10.1109/SFCS.1982.38.
[30] U. Greveler, P. Glösekötterz, B. Justusy, and D. Loehr, "Multimedia content identification through smart meter power usage profiles," in *Proc. Int. Conf. Info. Knowl. Eng.*, 2012, pp. 383–390.
[31] A. Gupta, C. Langbort, and T. Basar, "Optimal control in the presence of an intelligent jammer with limited actions," in *Proc. 49th Conf. Decis. Control*, 2010, pp. 1096–1101. doi: 10.1109/CDC.2010.5717544.
[32] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox. "Zcash protocol specification." Zcash. 2016. https://z.cash/
[33] R. Istepanian and J. F. Whidborne, *Digital Controller Implementation and Fragility: A Modern Perspective*. Berlin: Springer-Verlag, 2012.
[34] Z. P. Jiang and Y. Wang, "Input-to-state stability for discrete-time nonlinear systems," *Automatica*, vol. 37, no. 6, pp. 857–869, 2001. doi: 10.1016/S0005-1098(01)00028-0.
[35] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *Proc. Intl. Conf. Financial Cryptogr. Data Security*, 2013, pp. 111–125.
[36] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 2nd ed. Boca Raton, FL: CRC Press, 2014.
[37] J. Kim et al., "Encrypting controller using fully homomorphic encryption for security of cyber-physical systems," in *Proc. 6th IFAC Workshop Distrib. Estimat Control Netw. Syst.*, 2016, pp. 175–180. doi: 10.1016/j.ifacol.2016.10.392.
[38] J. Kim and H. Shim, "Encrypted state estimation in networked control systems," in *Proc. 58th Conf. Decis. Control*, 2019, pp. 7190–7195. doi: 10.1109/CDC40024.2019.9029752.
[39] W. Knowles, D. Prince, D. Hutchiso, J. F. P. Disso, and K. Jones, "A survey of cyber security management in industrial control systems," *Intl. J. Crit. Infrastruct. Protect.*, vol. 9, pp. 52–80, June 2015. doi: 10.1016/j.ijcip.2015.02.002.
[40] K. Kogiso and T. Fujita, "Cyber-security enhancement of networked control systems using homomorphic encryption," in *Proc. 54th Conf. Decis. Control*, 2015, pp. 6836–6843. doi: 10.1109/CDC.2015.7403296.
[41] I. Kolmanovsky and E. G. Gilbert, "Theory and computation of disturbance invariance sets for discrete-time linear systems," *Math. Problems Eng.*, vol. 4, no. 4, pp. 317–367, 1998. doi: 10.1155/S1024123X98000866.
[42] A. S. Leong, D. E. Quevedo, D. Dolz, and S. Dey, "On remote state estimation in the presence of an eavesdropper," in *Proc. 20th IFAC World Congr.*, 2017, pp. 7339–7344. doi: 10.1016/j.ifacol.2017.08.1482.
[43] Y. Li, L. Shi, P. Cheng, J. Chen, and D. E. Quevedo, "Jamming attacks on remote state estimation in cyber-physical systems: A game-theoretic approach," *IEEE Trans. Autom. Control*, vol. 60, no. 10, pp. 2831–2836, 2015. doi: 10.1109/TAC.2015.2461851.
[44] F. Lin, M. Fardad, and M. R. Jovanovic', "Augmented Lagrangian approach to design of structured optimal state feedback gains," *IEEE Trans. Autom. Control*, vol. 56, no. 12, pp. 2923–2929, 2011. doi: 10.1109/TAC.2011.2160022.
[45] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000. doi: 10.1016/S0005-1098(99)00214-9.
[46] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin, "Private memoirs of a smart meter," in *Proc. 2nd ACM Workshop Embedded Sens. Syst. Energy-Efficiency Build.*, 2010, pp. 61–66. doi: 10.1145/1878431.1878446.
[47] C. Murguia, F. Farokhi, and I. Shames, "Secure and private implementation of dynamic controllers using semi-homomorphic encryption," *IEEE Trans. Autom. Control*, vol. 65, no. 9, pp. 3950–3957, 2020. doi: 10.1109/TAC.2020.2992445.
[48] E. Barker, "Recommendation for key management," National Institute of Standards and Technology, Gaithersburg, MD, NIST Special Publication 800-57 Part 1, 2016.
[49] D. Nesic and D. Liberzon, "A unified framework for design and analysis of networked and quantized control systems," *IEEE Trans. Autom. Control*, vol. 54, no. 4, pp. 732–747, 2009. doi: 10.1109/TAC.2009.2014930.
[50] Y. Nesterov, "Gradient methods for minimizing composite functions," *Math. Program.*, vol. 140, no. 1, pp. 125–161, 2013. doi: 10.1007/s10107-012-0629-5.
[51] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - Eurocrypt '99* (Lecture Notes in Computer Science). Berlin: Springer-Verlag, 1999, pp. 223–238.
[52] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 123–231, 2014.
[53] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE Trans. Autom. Control*, vol. 58, no. 11, pp. 2715–2729, 2013. doi: 10.1109/TAC.2013.2266831.
[54] B. Pinkas, T. Schneider, N. Smart, and S. Williams, "Secure two-party computation is practical," in *Advances in Cryptology - Asiacrypt 2009* (Lecture Notes in Computer Science). Berlin: Springer-Verlag, 2009, pp. 250–267.
[55] O. Regev, "The learning with errors problem (invited survey)," in *Proc. 25th Annu. Conf. Computat. Complex.*, 2010, pp. 191–204. doi: 10.1109/CCC.2010.26.
[56] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978. doi: 10.1145/359340.359342.
[57] M. Schulze Darup, "Encrypted model predictive control in the cloud," in *Privacy in Dynamical Systems*, F. Farokhi, Ed. Berlin: Springer-Verlag, 2020, pp. 231–265.
[58] M. Schulze Darup and G. Book, "On closed-loop dynamics of ADMM-based MPC," in *Model Predictive Control*, T. Faulwasser, M. A. Müller, and K. Worthmann, Eds. Berlin: Springer-Verlag, 2019.
[59] M. Schulze Darup and T. Jager, "Encrypted cloud-based control using secret sharing with one-time pads," in *Proc. 58th Conf. Decis. Control*, 2019, pp. 7215–7221. doi: 10.1109/CDC40024.2019.9029342.
[60] M. Schulze Darup, A. Redder, and D. E. Quevedo, "Encrypted cloud-based MPC for linear systems with input constraints," in *Proc. 6th IFAC Nonlinear Model Predictive Control Conf.*, 2018, pp. 635–642.
[61] M. Schulze Darup, A. Redder, and D. E. Quevedo, "Encrypted cooperative control based on structured feedback," *IEEE Control Syst. Lett.*, vol. 3, no. 1, pp. 37–42, 2019. doi: 10.1109/LCSYS.2018.2851152.
[62] M. Schulze Darup, A. Redder, I. Shames, F. Farokhi, and D. Quevedo, "Towards encrypted MPC for linear constrained systems," *IEEE Control Syst. Lett.*, vol. 2, no. 2, pp. 195–200, 2018. doi: 10.1109/LCSYS.2017.2779473.
[63] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 53, no. 3, pp. 97–105, 1979. doi: 10.1145/359168.359176.
[64] C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, vol. 28, no. 4, pp. 656–715, 1949. doi: 10.1002/j.1538-7305.1949.tb00928.x.
[65] E. Shi, H. T. H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *Proc. Netw. Distrib. Syst. Security Symp.*, vol. 2, 2011, pp. 1–17.
[66] A. Teixeira, K. C. Sou, H. Sandberg, and K. H. Johansson, "Secure control systems: A quantitative risk management approach," *IEEE Control Syst.*, vol. 35, no. 1, pp. 24–45, 2015. doi: 10.1109/MCS.2014.2364709.
[67] K. Teranishi, N. Shimada, and K. Kogiso, "Stability analysis and dynamic quantizer for controller encryption," in *Proc. 58th Conf. Decis. Control*, 2019, pp. 7184–7189. doi: 10.1109/CDC40024.2019.9029356.
[68] "2019 Cyber attacks statistics." Hackmageddon. 2020. https://www.hackmageddon.com/2020/01/23/2019-cyber-attacks-statistics/
[69] P. Tøndel, T. A. Johansen, and A. Bemporad, "Computation and approximation of piecewise affine control laws via binary search trees," in *Proc. 41st Conf. Decis. Control*, 2002, pp. 3144–3149. doi: 10.1109/CDC.2002.1184353.
[70] A. Tsiamis, K. Gatsis, and G. J. Pappas, "State estimation codes for perfect secrecy," in *Proc. 56th Conf. Decis. Control*, 2017, pp. 176–181. doi: 10.1109/CDC.2017.8263662.
[71] M. van Dijk and A. Juels, "On the impossibility of cryptography alone for privacy-preserving cloud computing," in *Proc. 5th USENIX Conf. Hot Topics Security*, 2010, pp. 1–8. doi: 10.5555/1924931.1924934.
[72] R. van Parys and G. Pipeleers, "Real-time proximal gradient method for linear MPC," in *Proc. Euro. Control Conf.*, 2018, pp. 1142–1147. doi: 10.23919/ECC.2018.8550118.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:08:42 UTC from IEEE Xplore. Restrictions apply.