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

Authenticated Computation of Control Signal from Dynamic Controllers

Jung Hee Cheon[1], Dongwoo Kim[1], Junsoo Kim[2], Seungbeom Lee[2], Hyungbo Shim[2]

Abstract— **Significant concerns on networked control systems are modifications on the control signals caused by a compromise on the network or the controller, since it can cause a devastating behavior or even entire failure of the system. In this paper, we present a fundamental solution to this problem by proposing a new authenticated computation that checks the matrix-vector multiplications—the main arithmetic of a controller—and verifies the updates on the states of the controller. It enables the plant-side not only to check the computation of the controller with much less computational cost than that required for the computation itself, but also to detect any modifications on the control signals.**

I. Introduction

Networked control system is getting growing attention recently along its diverse use cases, such as, connected cars exchanging information with other cars via wireless network, and drones flying under the control of a remote user. In the meantime, as the demand on networked control increases, the threat of attacks from outside of the plant-side through forgery of the signals on the network or compromise on the controller has been a rising menace.

To decrease this risk, various methods have been proposed to ensure the integrity of signals from a controller. For example, the notion of encrypted control system, which considers the use of Homomorphic Encryption (HE), has been introduced [1], [2], [3]. With HE, the controller operates directly on encrypted data, and the actuator can solely decrypt the control signal. This encryption of data prevents adversaries from getting information on the system. However, in terms of defending recent cyber-attacks as in [4], it can not be a fundamental solution, since HE does not provide non-malleability, i.e., an adversary can manipulate the encrypted controller's signal so that the signal is decrypted to an incorrect value.

To handle this problem, Homomorphic Authenticated Encryption (HAE) [5], [6] has been proposed to achieve non-malleability in HE, but cost overhead of the verification process makes this scheme impractical. To overcome this drawback, efficient HAE was proposed in [7] for linear dynamic system. However, it still requires considerable computational cost as much as the controller for the plant-side to verify the validity of the encrypted control signal. Moreover, the proposed scheme can only be applied for finite time horizon, which is a serious drawback.

In this paper, we investigate a new approach that enables the plant-side to *efficiently* detect any misbehavior of the controller or any forgery on the signal in infinite time horizon. The idea is to introduce the recent primitive from complexity theory and cryptography called *Verifiable Computation* (VC) [8] to the dynamic system. With VC, the plant-side can verify the correctness of the control signal from the controller in much less computational cost than that required to compute the signal. This somewhat counter-intuitive process is possible with the help of an additional information called *proof* provided by the controller. In principle, the proof based on fast probabilistic checking and cryptographic primitives enables the plain-side to efficiently verify the correctness of given control signal *with high probability*, unless the signal is forged by an adversary who can break a cryptographic hardness assumption.

In this paper, by exploiting the fundamental idea of VC, we propose a novel scheme to authenticate the correctness of the control signals in a linear dynamic system. Here, the signals are computed mainly by matrix-vector multiplications comprising a vector representing the state of the controller, which is also updated by matrix-vector multiplications (see (1)). At first, we adopt Freivalds' algorithm [9] to verify these matrix-vector multiplications. In this algorithm, the plant-side prepares *random verification vectors* and takes their inner-product with input and output of the controller to check correctness of the control signal. However, a naive application of this algorithm forces the plant-side to obtain the state of the controller at each sampling time (to check updates on the state), which incurs substantial communication cost and makes the computational cost on the plant-side proportional to the dimension of the state; it is not efficient enough.

To overcome this obstacle, we devise a cryptographic method to perform verification algorithm in an exponent form. In detail, let g be a generator of a cyclic group[1] G, then the plant-side gives the controller each random verification vector $\vec{r} = (r_1, \ldots, r_n) \in \mathbb{Z}_p^n$ (for the Freivald's algorithm) in an exponent form $(g^{r_1}, \cdots, g^{r_n})$, where n is the dimension of the controller's state. These exponent forms not only hide the verification vector, but also enable the controller to return a proof as a group element h of G, whose (hidden) logarithm to the base g is the inner-product of the state with the verification vector. It enables the plant-side not only to receive the proofs in lower communication cost, but also to perform verification algorithms at less

*This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) (No. NRF-2017R1A5A1015626 and No. NRF-2017R1E1A1A03070342).

[1]J. H. Cheon and D. Kim are with IMDARC, Department of Mathematical Sciences, Seoul National University, Korea. email: jhcheon@snu.ac.kr, dwkim606@snu.ac.kr

[2]J. Kim, S. Lee, and H. Shim are with ASRI, Department of Electrical and Computer Engineering, Seoul National University, Korea. email: kjs9044@snu.ac.kr, sblee90@snu.ac.kr, hshim@snu.ac.kr

[1]We refer to Definition 2 in Section II.

computational cost. In particular, both costs are *independent* of the dimension of the state, a notable improvement from the naive approach.

Above method, however, has a subtle problem that a (compromised) controller can deceive the plant-side by using wrong states when generating each proof for the state update and the signal computation. Therefore, we also devise a method to check whether each proof is generated from the correct state, i.e., if the state at time t is updated from the previous state at time $t-1$ and is also used to compute the output signal at time t. Putting these together, the proposed scheme guarantees that the controller can not deceive the plant-side with incorrect signal, unless it breaks the well-known hardness assumption[2] on the discrete logarithm problem in G.

The proposed scheme applied to linear dynamic systems enables an actuator (of the plant-side) to detect any modifications on the given controller signal caused by the forgery of signals (on the network) or the compromise on the controller. As a result, it can fundamentally secure the plant from malfunctions caused by all possible external adversaries from the plant-side.

II. Preliminaries and Problem Formulation

A. Notations

Let $\mathbb{Z}$ denotes the set of integers, and $\mathbb{F} := \mathbb{Z}_p$ denotes the field of prime order p, i.e, an integer modulo p. A column vector is denoted by using arrow like $\vec{x}$ while matrices are denoted by capital letters, e.g., A, B, C, and $\vec{x} \cdot \vec{y}$ denotes the inner-product of vectors. Let $A\vec{x}$ and $\vec{x}^\top A$ denote the multiplication between a matrix and a vector. We use standard notation from probability theory: for a set X, $x \leftarrow X$ signifies that an element x is sampled uniformly randomly from X, and $\Pr[x \leftarrow X : E]$ denotes the probability of the event E in this case. $\Pr[A|B] := \Pr[A \wedge B]/\Pr[B]$ denotes the conditional probability where $\wedge$ implies the logical 'and'. Finally, $y \leftarrow \mathcal{A}(x)$ denotes the event that an adversary $\mathcal{A}$ outputs y given an input x. We say that a function is *negligible* in λ, and denote it by $negl(\lambda)$, if it is $o(\lambda^{-c})$ for every fixed constant c; we recommend readers who are not familiar with this context to regard $negl(\lambda)$ roughly as $1/2^\lambda$ and to regard an event with *negligible* probability as does not occur in practice.

B. Problem Formulation

Consider a discrete-time controller on a linear dynamic system given as

$$\begin{aligned} \vec{z}(t+1) &= \mathsf{A}\vec{z}(t) + \mathsf{B}\vec{y}(t), \\ \vec{u}(t) &= \mathsf{C}\vec{z}(t) + \mathsf{D}\vec{y}(t) \end{aligned} \tag{1}$$

where $\vec{z}(t) \in \mathbb{R}^n$ is the state with initial value $\vec{z}(0) = \vec{z}_0$. In the closed-loop system consisting of a controller and a plant, the controller receives the signal $\vec{y}(t) \in \mathbb{R}^m$ (from the sensor), performs the computation (1), then transmits the output signal $\vec{u}(t) \in \mathbb{R}^l$ to the actuator, as described in Fig. 1.

[2]Given g and g^r, it is hard to find r. See Assumption 1.

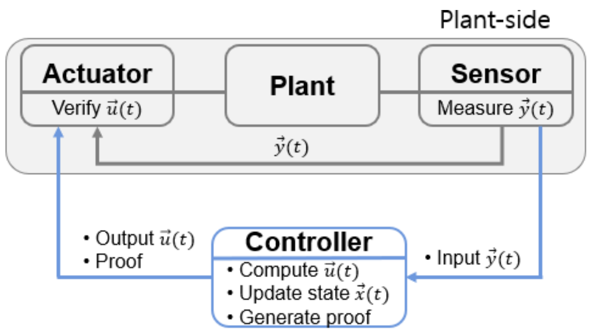

Fig. 1: Configuration of control system with verifiable computation (VC)

The objectives of the proposed scheme (in Section III) are:

- *Verification of* $\vec{u}(t)$: For each time t, the actuator can verify if the given output signal $\vec{u}(t)$ from the controller is the correct output from the computation (1) or not.
- *Efficiency*: The computational cost to verify the correctness of output signal (given from the controller) must be less than that required to compute the output $\vec{u}(t)$ from (1).

Here, we assume that the actuator has sensor measurement $\vec{y}(t)$ at each time t, the initial state $\vec{z}_0$, and the information on the matrices A, B, C, D of (1). The information on the matrices are required only to generate a key used for the verification, and the matrices are not required for the verification of output signals.

C. Conversion of Real-valued Parameters to Integers

In the following section, we will introduce the VC scheme for linear dynamic system achieving the aforementioned goal. However, the scheme is applicable only to operations of a finite field while the computation (1) is over a field of real numbers. Luckily, we can convert the given system so that it only utilizes multiplications and additions over *bounded* integers during the whole time period. Then, the operations can be naturally represented by operations over a (sufficiently large) finite field.

The conversion proceeds as follows. Let us assume, without loss of generality, that the pair (A, C) is observable[3]. Then, a matrix $\mathsf{H} \in \mathbb{R}^{n \times l}$ can be found by the pole-placement techniques so that the eigenvalues of $\mathsf{A} - \mathsf{HC}$ are all integers. Then, with an invertible matrix $\mathsf{T} \in \mathbb{R}^{n \times n}$, the matrix $\mathsf{A}-\mathsf{HC}$ can be transformed into Jordan canonical form so that we obtain $\mathsf{T}(\mathsf{A} - \mathsf{HC})\mathsf{T}^{-1} \in \mathbb{Z}^{n \times n}$. See [10, Section IV] for more details. Then, the system (1) can be converted to the system over integers as

$$\begin{aligned} \vec{x}(t+1) &= A\vec{x}(t) + B\left\lceil \frac{\vec{y}(t)}{l_1} \right\rfloor + H\lceil l_2^2 \cdot \vec{u}_{\mathbb{Z}}(t) \rfloor \ \in \mathbb{Z}^n \\ \vec{u}_{\mathbb{Z}}(t) &= C\vec{x}(t) + D\left\lceil \frac{\vec{y}(t)}{l_1} \right\rfloor \ \in \mathbb{Z}^l, \end{aligned} \tag{2}$$

[3]If the controller is not observable, it implies that the controller of (1) is not of minimal realization, so can be reduced to obtain an observable pair. For the details, see [10, Section IV], for example.

with the initial value $\vec{x}(0) = \mathsf{T}\vec{z}_0/(l_1 l_2)$, where the matrices $A := \mathsf{T}(\mathsf{A} - \mathsf{HC})\mathsf{T}^{-1}$, $B := \lceil(\mathsf{TB} - \mathsf{HD})/l_2\rfloor$, $C := \lceil\mathsf{C}/l_2\rfloor$, $D := \lceil\mathsf{D}/l_2^2\rfloor$, and $H := \lceil\mathsf{H}/l_2\rfloor$ consist of integers, and $\vec{x}$ and $\vec{u}_{\mathbb{Z}}$ have the values of $\mathsf{T}\vec{z}/(l_1 l_2)$ and $\vec{u}/(l_1 l_2^2)$, respectively. Here, l_1 and l_2 are scale factors, that are typically small positives.

Finally, the following proposition suggests that the performance error of the system (2) over integers is negligible compared with (1), when the factors $1/l_1$ and $1/l_2$ are chosen sufficiently large.

Proposition 1 ([10]) *On* (1) *and* (2)*, given $\epsilon > 0$, there exist $l_1' > 0$ and $l_2' > 0$ such that for every $l_1 < l_1'$ and $l_2 < l_2'$, it guarantees that $\|l_1 l_2 \cdot \mathsf{T}^{-1} \cdot \vec{x}(t) - \vec{z}(t)\| < \epsilon$ and $\|l_1 l_2^2 \cdot \vec{u}(t) - \vec{u}_{\mathbb{Z}}(t)\| < \epsilon$ for all $t \geq 0$.* $\Diamond$

Therefore, in the rest of this paper, we consider the problem of verifying the computation (2) over integers instead of (1) over real numbers.

D. Verifiable Computation

Assume a situation where a verifier outsources to a prover an evaluation of a function F on an input μ. In our case, the plant-side is a verifier, the controller is a prover, and the computation (2) corresponds to the function F. A verifiable computation (VC) scheme enables the verifier to verify if the result ν output by the prover is correct (i.e., $F(\mu) = \nu$), with less computational cost than that required to evaluate F. The definition of VC is as follows.

Definition 1 (Verifiable Computation [11]) *A verifiable computation scheme consists of three algorithms (*$\mathsf{KeyGen}, \mathsf{Compute}, \mathsf{Verify}$*) defined as follows.*

- $\mathsf{KeyGen}(F, \lambda) \to (EK_F, VK_F)$*: The key generation algorithm takes the function F to be outsourced and a security parameter*[4] *λ as input; it outputs an evaluation key EK_F, and a verification key VF_F.*
- $\mathsf{Compute}(EK_F, \mu) \to (\nu, \pi_\nu)$*: Given the evaluation key EK_F and an input μ, the computation algorithm outputs $\nu = F(\mu)$ and a proof π_ν of ν's correctness.*
- $\mathsf{Verify}(VK_F, \mu, \nu, \pi_\nu) \to acc$ *or rej: With the verification key VK_F, an input μ, and a claimed output ν with a proof π_ν, the verification algorithm outputs acc if $F(\mu) = \nu$, and outputs rej otherwise.*

It satisfies the correctness, security, and efficiency whose definitions are:

- Correctness*: For any function F and any input μ, if we run* $\mathsf{KeyGen}(F, \lambda) \to (EK_F, VK_F)$ *and* $\mathsf{Compute}(EK_F, \mu) \to (\nu, \pi_\nu)$*, then we always get* $\mathsf{Verify}(VK_F, \mu, \nu, \pi_\nu) \to acc$*.*
- Security*: For any function F and any probabilistic polynomial-time adversary*[5] *$\mathcal{A}$, the following probability is $negl(\lambda)$.*

$$\Pr\left[\begin{array}{l} (F(\mu^*) \neq \nu^*) \wedge \\ (\mathsf{Verify}(VK_F, \mu^*, \nu^*, \pi^*) \to acc) \end{array} \middle| \begin{array}{l} (\mu^*, \nu^*, \pi^*) \\ \leftarrow \mathcal{A}(EK_F) \end{array}\right]$$

- Efficiency*: In computational cost, the algorithm* Verify *is cheaper than the evaluation of F.* $\Diamond$

The security condition, in other words, says that a computationally bounded (e.g., can perform only $< 2^\lambda$ operations) adversary can *not* generate an incorrect output ν^* (even though he/she can choose an input μ^*) and a forged proof π^* that will be accepted (acc) by the Verify algorithm with non-negligible probability, i.e., it will be rejected (rej) with high probability (e.g., more than $1 - 1/2^\lambda$ probability). Usually, the verifier (in our case, the actuator) performs KeyGen (one-time) and Verify algorithms while the prover (in our case, the controller) performs $\mathsf{Compute}$ algorithm to generate the proof of its computation correctness.

E. Freivalds' Algorithm: Verifying Matrix Multiplication

We introduce Freivalds' algorithm [9] which is a VC specialized for matrix-vector multiplications. Let $\mathbb{F}$ be a finite field and F be an $n \times m$ matrix over $\mathbb{F}$. A verifier outsources (then verifies) a computation of $F\vec{\mu}$ on an input $\vec{\mu}$ of its choice. In this algorithm, the prover does not need to generate a proof.

- *Algorithm*: A verifier randomly samples $\vec{r} \in \mathbb{F}^n \setminus \{\vec{0}\}$ and precomputes $\vec{f} := \vec{r}^\top F \in \mathbb{F}^m$ (both $\vec{r}$ and $\vec{f}$ correspond to VK_F in VC). For an input $\vec{\mu} \in \mathbb{F}^m$ and a claimed output $\vec{\nu} \in \mathbb{F}^n$ (given by a prover), the verifier accepts the output only if $\vec{r} \cdot \vec{\nu} = \vec{f} \cdot \vec{\mu}$.
- *Security*: $\Pr[\text{verifier accepts } \vec{\nu} \,|\, \vec{\nu} \neq F\vec{\mu}] \leq n/|\mathbb{F}|$ by the following Lemma 1; if $\vec{\nu} \neq F\vec{\mu}$ is an incorrect output that verifier accepts, $\vec{\nu}$ is a solution of the nonzero polynomial $r : \mathbb{F}^n \to \mathbb{F}$ defined by $r(\vec{x}) := \vec{r} \cdot \vec{x} - \vec{f} \cdot \vec{\mu}$.
- *Efficiency*: The pair $\vec{r}$ and $\vec{f}$ can be used over many verifications. Thus, given these vectors, the computational cost (measured by the number of multiplications required) of verification (checking if $\vec{r} \cdot \vec{\nu} = \vec{f} \cdot \vec{\mu}$) is $n+m$ which is less than the cost nm of evaluating $F\vec{\mu}$.

Lemma 1 (Schwartz-Zippel [12]) *Let $\mathbb{F}$ be a finite field, and $f : \mathbb{F}^\ell \to \mathbb{F}$ be an ℓ-variate nonzero polynomial of total degree (the sum of degrees of each variable) δ. Then,*

$$\Pr[\vec{x} \leftarrow \mathbb{F}^\ell : f(\vec{x}) = 0] \leq \frac{\delta}{|\mathbb{F}|}.$$

$\Diamond$

F. Hardness Assumptions on Finite Groups

We first recall the definition of finite groups.

Definition 2 (Finite Group) *A finite group G of order prime p having $g \in G$ as a generator is:*

- *It has p elements including the identity 1.*
- *Every element is of the form g^i for some $i \in \{0, 1, 2, \ldots, p-1\}$, and the identity $1 = g^0$.*
- *An operation called multiplication (denoted by $\cdot$) is defined as $g^a \cdot g^b := g^{a+b \bmod p}$.*

[4]It determines the degree of security of given scheme, see the definition of security that follows.

[5]Roughly, it captures all probabilistic algorithms that work in feasible time in practice.

We let $G_\lambda := \langle g \rangle$ denote a finite group G_λ of order a prime $p > 2^\lambda$ having $g \in G_\lambda$ as a generator. For a vector $\vec{x} := (x_1, \ldots, x_n)$, $g^{\vec{x}}$ denotes $(g^{x_1}, \ldots, g^{x_n})$. $\Diamond$

As an example, for a prime p, the nonzero elements of $\mathbb{Z}_{p+1}$ form a finite group of order p for given multiplication over $\mathbb{Z}_{p+1}$ (one can set any nonzero element as a generator).

The security of proposed VC scheme (Section III) in this paper relies on the following cryptographic hardness assumptions on a finite group.

Assumption 1 (Discrete Logarithm) *The discrete logarithm assumption holds for $G_\lambda := \langle g \rangle$; that is, for all probabilistic polynomial-time adversary $\mathcal{A}$, the probability*

$$\Pr[x \leftarrow \mathbb{Z}_p \setminus \{0\} : x \leftarrow \mathcal{A}(G_\lambda, g, g^x)]$$

is $negl(\lambda)$. $\Diamond$

In other words, the adversary can not find which x is in the exponent of g given g^x, i.e., it can not compute a logarithm of g^x to the base g.

The proposed scheme also exploits the following well-known assumption on which the security of many VC schemes [13], [11], [14], [15] are based.

Assumption 2 (n-PKE [16]) *The* n-PKE *(n-power knowledge of exponent) assumption holds for $G_\lambda := \langle g \rangle$; that is, for all probabilistic polynomial-time adversary $\mathcal{A}$, the probability*

$$\Pr\left[\begin{array}{lcl} \alpha, s_1, \ldots, s_n \leftarrow \mathbb{Z}_p \setminus \{0\}, & & \\ \sigma := (G_\lambda, g, g^{s_1}, \ldots, g^{s_n}, & & (c^* = c^\alpha) \\ \qquad g^{\alpha s_1}, \ldots, g^{\alpha s_n}), & : & \wedge(c \neq \prod_{i=1}^n g^{a_i s_i}) \\ (c, c^*) \leftarrow \mathcal{A}(\sigma), & & \end{array}\right],$$

where $\mathcal{A}$ knows[6] *$(a_1, \ldots, a_n)$, is $negl(\lambda)$.* $\Diamond$

It implies that if an adversary $\mathcal{A}$ given $g^{\vec{s}}$ and $g^{\alpha\vec{s}}$ can output g_1 and g_2 such that $g_1^\alpha = g_2$ with non-negligible probability, the only way is that he/she generated g_1 (and g_2, respectively) via the linear combination (in the exponent) of given $g^{\vec{s}}$ (and $g^{\alpha\vec{s}}$, respectively) with the coefficients (e.g., $\{a_i\}_{i=1}^n$) he/she *knows*.

III. Verification of Controller Computation

In this section, we propose a verifiable computation (VC) scheme for checking computation of a controller. With the integer conversion (Section II-C), we can assume that we are given the following discrete-time controller over a (sufficiently large) finite field[7] $\mathbb{F}$:

$$\begin{aligned} \vec{x}(t+1) &= A\vec{x}(t) + B\vec{y}(t) + H\vec{u}(t) \ \in \mathbb{F}^n, \\ \vec{u}(t) &= C\vec{x}(t) + D\vec{y}(t) \ \in \mathbb{F}^l \end{aligned} \tag{3}$$

in which, for simplicity of description and without loss of generality, we first replaced the notation $\vec{u}_{\mathbb{Z}}$ with $\vec{u}$, and substituted $H\lceil l_2^2 \cdot \vec{u}_{\mathbb{Z}}(t)\rfloor$ in (2) by $H\vec{u}(t)$, since an actuator given $\vec{u}_{\mathbb{Z}}(t)$ can compute $\lceil l_2^2 \cdot \vec{u}_{\mathbb{Z}}(t)\rfloor$ by itself.

[6]It is quite complicated to formally define one "knows" something. We refer to [16] for a formal definition.

[7]The size of the finite field $\mathbb{F} = \mathbb{Z}/p\mathbb{Z}$ should be large so that the modular reduction (by p) does not occur during the computation. It is possible since the integer system (Section II-C) is composed of bounded integers.

A. VC Scheme for Linear Dynamic System

We first describe the design rationale of the proposed VC scheme.

1) Probabilistic Verification: The starting point is to use Freivalds' algorithm (Section II-E) for the actuator to check the computation (3) of a controller, that is mainly composed of matrix-vector multiplications. More precisely, assume that the actuator sampled random vectors $\vec{r} \in \mathbb{F}^n, \vec{s} \in \mathbb{F}^l$ and precomputed the *verification vectors* $\vec{a} := \vec{r}^\top A$, $\vec{b} := \vec{r}^\top B$, $\vec{h} := \vec{r}^\top H$, $\vec{c} := \vec{s}^\top C$, and $\vec{d} := \vec{s}^\top D$. Then, on each time t, the actuator, given states $\vec{x}(t+1)$, $\vec{x}(t)$ and signal $\vec{u}(t)$ from the controller and measurement $\vec{y}(t)$ from the sensor, checks if the following equations hold:

$$\begin{aligned} \vec{r} \cdot \vec{x}(t+1) &= \vec{a} \cdot \vec{x}(t) + \vec{b} \cdot \vec{y}(t) + \vec{h} \cdot \vec{u}(t), \\ \vec{s} \cdot \vec{u}(t) &= \vec{c} \cdot \vec{x}(t) + \vec{d} \cdot \vec{y}(t). \end{aligned} \tag{4}$$

Due to Freivalds' algorithm (Section II-E), given states and signal are correct with high probability if these equations hold. For this procedure, however, the state $\vec{x}(t)$ at each time t must be delivered to the actuator, which results in substantial communication cost between the actuator and the controller (it also incurs the verification cost proportional to the size of states). Our observation for the next step is that the actuator can check the above equations *only with* the inner-product values ($\vec{r} \cdot \vec{x}(t+1)$, $\vec{a} \cdot \vec{x}(t)$, $\vec{c} \cdot \vec{x}(t)$) between the states ($\vec{x}(t)$, $\vec{x}(t+1)$) and the random verification vectors ($\vec{r}, \vec{a}, \vec{c}$), instead of the states. We call these inner-product values as *compressed states.*

2) Compressed Commitments: Let G be a finite group of order p (note that $\mathbb{F} = \mathbb{Z}_p$) having g as a generator. From the above observation, we let the controller send, instead of the states $\vec{x}(t)$ and $\vec{x}(t+1)$, the values $g^{\vec{r} \cdot \vec{x}(t+1)}, g^{\vec{a} \cdot \vec{x}(t)}$, and $g^{\vec{c} \cdot \vec{x}(t)}$ which we call *compressed commitments*, each having a compressed state as the exponent value. Then, the actuator can check (4) with these values. For this, the actuator gives the vectors $g^{\vec{r}}, g^{\vec{a}}$, and $g^{\vec{c}}$ to the controller. Note that these vectors hide the verification vectors $\vec{r}, \vec{a}, \vec{c}$ under the Discrete Logarithm assumption (Assumption 1).[8] On the other hand, with these vectors, the controller can compute the required compressed commitments for verification.

3) Knowledge Assumption: Note, however, that a malicious adversary can deceive the actuator by sending a wrong signal $\vec{u}'(t)$ along with rigged commitments, e.g., it can send $\vec{u}'(t) = D\vec{y}(t)$ and g^0 instead of $g^{\vec{c} \cdot \vec{x}(t)}$, and pass the check (4). To prevent this behavior, we adopt the n-PKE assumption (Assumption 2) to force the controller (or an adversary) to combine *only* the coefficients of given vector $g^{\vec{r}}$ (or $g^{\vec{a}}$, $g^{\vec{c}}$) *linearly* in generating the corresponding compressed commitment. For example, if the controller is also given $g^{\rho\vec{r}}$ for a randomly chosen scalar ρ, and is demanded to generate two compressed commitments g_1 and g_2 satisfying $g_1^\rho = g_2$, its only choice is to set $g_1 = g^{\vec{r} \cdot \vec{x}}$ for a coefficient vector $\vec{x}$ it knows, by the n-PKE assumption. The assumption also guarantees that if an adversary can generate compressed

[8]If $\vec{r}$, $\vec{a}$, and $\vec{c}$ are disclosed to the controller, it can generate wrong state $\vec{x}'(t)$ and wrong signal $\vec{u}'(t)$ which will pass the check (4).

commitments and a wrong signal $\vec{u}'(t) \neq \vec{u}(t)$ passing the check (4), he/she must know the solution $(\vec{z}_1, \vec{z}_2, \vec{z}_3)$ of (4) that is different from $(\vec{x}(t+1), \vec{x}(t), \vec{x}(t))$, which develops to the security proof of the proposed scheme.

4) Proof of Equality: Finally, since the check (4) continues over time t, the actuator should be confirmed that the state $\vec{x}(t+1)$ generated (and verified) in time t is also used in time $t+1$ to generate the state $\vec{x}(t+2)$. For example, a (malicious) prover can provide, independent of the compressed commitment $g^{\vec{r}\cdot\vec{x}(t+1)}$ at time t, other compressed commitment $g^{\vec{a}\cdot\vec{z}(t+1)}$ at time $t+1$ using a different state $\vec{z}(t+1)$. To prevent this, the actuator sends $g^{\vec{a}-\vec{r}}$ to the controller, then demands a proof $g^{(\vec{a}-\vec{r})\cdot\vec{w}}$ which satisfies that $g^{\vec{a}\cdot\vec{z}(t+1)} = g^{(\vec{a}-\vec{r})\cdot\vec{w}} \cdot g^{\vec{r}\cdot\vec{x}(t+1)}$. Combining this strategy with n-PKE assumption as in the previous case (see the scheme below), it can be guaranteed that the controller must set $\vec{z}(t+1) = \vec{w} = \vec{x}(t+1)$ to pass the check. In the proposed scheme, we implicitly perform this check by giving $g^{\vec{a}-\vec{r}}$ instead of $g^{\vec{a}}$ to the controller and letting the actuator compute $g^{\vec{a}\cdot\vec{x}} = g^{(\vec{a}-\vec{r})\cdot\vec{x}} \cdot g^{\vec{r}\cdot\vec{x}}$.

Putting these altogether, we get the proposed scheme whose formal description follows. We remark that the verification vector $\vec{r}(t)$ and $\vec{r}(t+1)$ at time t and $t+1$ must be independent to each other for the security proof. Since the proof involves only two consecutive rounds, the proposed scheme will use different $\vec{r}_0$ and $\vec{r}_1$ alternatively on each time t.

The VC Scheme for Linear Dynamic Systems

- $\mathsf{KeyGen}(\lambda, F := \{A, B, C, D, H\}) \to (EK_F, VK_F)$: Let λ be the security parameter and n, m, l be the dimensions of vectors $\vec{x}, \vec{y}, \vec{u}$, respectively, and let A, B, C, D, H be the matrices of (3). Take a finite field $\mathbb{F} := \mathbb{Z}/p\mathbb{Z}$ of prime order p greater than 2^λ, and let G be a cyclic group of order p where g is a generator. Choose $\vec{s} \in \mathbb{F}^l$, $\vec{r}_0, \vec{r}_1 \in \mathbb{F}^n$, $\alpha_0, \alpha_1, \gamma_0, \gamma_1, \rho_0, \rho_1 \in \mathbb{F}$ uniformly at random, and compute $\vec{a}_i := \vec{r}_i^\top A$, $\vec{b}_i := \vec{r}_i^\top B$, $\vec{h}_i := \vec{r}_i^\top H$ for $i \in \{0,1\}$, $\vec{c} := \vec{s}^\top C$, $\vec{d} := \vec{s}^\top D$. Set the evaluation key EK_F as:
$$(g^{\vec{r}_0}, g^{\vec{a}_1-\vec{r}_0}, g^{\vec{c}-\vec{r}_0}, g^{\rho_0\vec{r}_0}, g^{\alpha_0(\vec{a}_1-\vec{r}_0)}, g^{\gamma_0(\vec{c}-\vec{r}_0)},$$
$$g^{\vec{r}_1}, g^{\vec{a}_0-\vec{r}_1}, g^{\vec{c}-\vec{r}_1}, g^{\rho_1\vec{r}_1}, g^{\alpha_1(\vec{a}_0-\vec{r}_1)}, g^{\gamma_1(\vec{c}-\vec{r}_1)}),$$
and the verification key VK_F as:
$$(\alpha_0, \alpha_1, \gamma_0, \gamma_1, \rho_0, \rho_1, \vec{s}, \vec{b}_0, \vec{b}_1, \vec{h}_0, \vec{h}_1, \vec{d}).$$
- $\mathsf{Compute}(EK_F, t, \vec{x}(t), \vec{y}(t)) \to (t, \vec{u}(t), \pi_t)$: For time t, derive the updated state $\vec{x}(t+1)$ and $\vec{u}(t)$ from $\vec{x}(t)$, $\vec{y}(t)$ with (3), and output the proof π_t defined as follows where we abbreviate $\vec{x}(t+1)$ to $\vec{x}_+$ and $\vec{x}(t)$ to $\vec{x}$:
  - when t is even,
  $$(g^{\vec{r}_0\cdot\vec{x}_+}, g^{(\vec{a}_0-\vec{r}_1)\cdot\vec{x}}, g^{(\vec{c}-\vec{r}_1)\cdot\vec{x}},$$
  $$g^{\rho_0\vec{r}_0\cdot\vec{x}_+}, g^{\alpha_1(\vec{a}_0-\vec{r}_1)\cdot\vec{x}}, g^{\gamma_1(\vec{c}-\vec{r}_1)\cdot\vec{x}}),$$
  - when t is odd,
  $$(g^{\vec{r}_1\cdot\vec{x}_+}, g^{(\vec{a}_1-\vec{r}_0)\cdot\vec{x}}, g^{(\vec{c}-\vec{r}_0)\cdot\vec{x}},$$
  $$g^{\rho_1\vec{r}_1\cdot\vec{x}_+}, g^{\alpha_0(\vec{a}_1-\vec{r}_0)\cdot\vec{x}}, g^{\gamma_0(\vec{c}-\vec{r}_0)\cdot\vec{x}}).$$
- $\mathsf{Verify}(VK_F, t, \vec{u}(t), \vec{y}(t), \pi_{t-1}, \pi_t) \quad \to \quad acc$ or rej: Parse the proof π_{t-1} and π_t as $(g_1, g_2, g_3, g_1', g_2', g_3')$ and $(G_1, G_2, G_3, G_1', G_2', G_3')$, respectively, and perform the following checks[9]:
  - Linearity Check (when t is even)
  $$G_1^{\rho_0} = G_1', \quad G_2^{\alpha_1} = G_2', \quad G_3^{\gamma_1} = G_3' \tag{5}$$
  - Equation Check (when t is even)
  $$G_1 = G_2 \cdot g_1 \cdot g^{\vec{b}_0\cdot\vec{y}} \cdot g^{\vec{h}_0\cdot\vec{u}}, \quad g^{\vec{s}\cdot\vec{u}} = G_3 \cdot g_1 \cdot g^{\vec{d}\cdot\vec{y}} \tag{6}$$

  Then, outputs acc if all checks pass, and rej otherwise. When t is odd, $\rho_0, \alpha_1, \gamma_1, \vec{b}_0, \vec{h}_0$ are substituted by $\rho_1, \alpha_0, \gamma_0, \vec{b}_1, \vec{h}_1$ in each check.

Theorem 1 *Our VC scheme satisfies the correctness and efficiency (Definition 1). It also satisfies the security under the Discrete Logarithm assumption (Assumption 1) and n-PKE assumption (Assumption 2) where n is the dimension of $\vec{x}(t)$.* $\diamondsuit$

Proof: The correctness follows from that of Freivalds' algorithm (Section II-E) and the fact that discrete group G is additive, i.e., $g^{m_1} \cdot g^{m_2} = g^{m_1+m_2}$. The efficiency follows since the Verify algorithm performs linear (in $|\vec{y}|$ and $|\vec{u}|$) number of operations over $\mathbb{F}$ and constant number of operations over G, while the computation of (3) requires quadratic (in $|\vec{x}|$) number of operations over $\mathbb{F}$. The security is described in the following subsection. ■

B. Security of the proposed VC

We show that, given n-PKE assumption (Assumption 2), if an adversary can generate a proof on an incorrect output $\vec{u}'(t)$ $(\neq \vec{u}(t))$ or a proof whose first component is $g^{\vec{r}_i\cdot\vec{x}_+'}$ where $\vec{x}_+' \neq \vec{x}_+$, on a correct output $\vec{u}(t)$ which makes the Verify algorithm output acc with non-negligible probability for some time $t \in \mathbb{Z}_{\geq 0}$, the adversary can break the Discrete Logarithm assumption (Assumption 1).

Without loss of generality, assume t is even, and let $\pi_t' = (G_1, G_2, G_3, G_1', G_2', G_3')$ be the first forged proof generated by the adversary, which is accepted by Verify algorithm. Therefore, g_1 from the previous proof π_{t-1} is $g^{\vec{r}_1\cdot\vec{x}}$, since the proof is correct one. Then, by n-PKE assumption, $G_1 = g^{\vec{r}_0\cdot\vec{x}_+'}$, $G_2 = g^{(\vec{a}_0-\vec{r}_1)\cdot\vec{x}'}$, and $G_3 = g^{(\vec{c}-\vec{r}_1)\cdot\vec{x}''}$ for some $\vec{x}_+'$, $\vec{x}'$, and $\vec{x}''$ all of which the adversary *knows*. Since the proof π_t' and the output $\vec{u}'$ from the adversary pass the Equation Check of Verify algorithm, it satisfies that
$$\begin{aligned} \vec{r}_0 \cdot \vec{x}_+' &= (\vec{a}_0 - \vec{r}_1) \cdot \vec{x}' + \vec{r}_1 \cdot \vec{x} + \vec{b}_0 \cdot \vec{y} + \vec{h}_0 \cdot \vec{u}', \\ \vec{s} \cdot \vec{u}' &= (\vec{c} - \vec{r}_1) \cdot \vec{x}'' + \vec{r}_1 \cdot \vec{x} + \vec{d} \cdot \vec{y}. \end{aligned} \tag{7}$$
Note that the adversary can get the correct values $\vec{x}_+$, $\vec{x}$, and $\vec{u}$ (with honest computation) which satisfy that
$$\begin{aligned} \vec{r}_0 \cdot \vec{x}_+ &= (\vec{a}_0 - \vec{r}_1) \cdot \vec{x} + \vec{r}_1 \cdot \vec{x} + \vec{b}_0 \cdot \vec{y} + \vec{h}_0 \cdot \vec{u}, \\ \vec{s} \cdot \vec{u} &= (\vec{c} - \vec{r}_1) \cdot \vec{x} + \vec{r}_1 \cdot \vec{x} + \vec{d} \cdot \vec{y}. \end{aligned} \tag{8}$$

[9]When $t = 0$, verifier uses the initial state $\vec{x}_0$ instead of π_{-1} to get $g_1 = g^{\vec{r}_1\cdot\vec{x}_0}$ required for the Equation Check.

Subtracting (8) from (7) and rearranging, we get

$$\vec{r}_0 \cdot [(\vec{x}'_+ - \vec{x}_+) - H(\vec{u}' - \vec{u}) - A(\vec{x}' - \vec{x})] + \vec{r}_1 \cdot (\vec{x}' - \vec{x}) = 0,$$
$$\vec{s} \cdot [(\vec{u}' - \vec{u}) - C(\vec{x}'' - \vec{x})] + \vec{r}_1 \cdot (\vec{x}'' - \vec{x}) = 0.$$

With this equation, we can conclude as follows. If the forged proof π'_t from the adversary is on incorrect output, i.e., $\vec{u}' \neq \vec{u}$, then[10] $(\vec{u}' - \vec{u} - C(\vec{x}'' - \vec{x}))^\top || (\vec{x}'' - \vec{x})^\top$ is a nonzero vector which is also *known* to the adversary and is perpendicular to the random vector $\vec{s}^\top || \vec{r}_1^\top$ hidden by $g^{\vec{s}}$ and $g^{\vec{r}_1}$. On the other hand, if the forged proof π'_t satisfies that $\vec{x}'_+ \neq \vec{x}_+$ and $\vec{u}' = \vec{u}$ (if not, the previous case occurs), then $(\vec{x}'_+ - \vec{x}_+ - A(\vec{x}' - \vec{x}))^\top || (\vec{x}' - \vec{x})^\top$ is a nonzero vector which is also *known* to the adversary and is perpendicular to the random vector $\vec{r}_0^\top || \vec{r}_1^\top$ hidden by $g^{\vec{r}_0}$ and $g^{\vec{r}_1}$.

Now, the following lemma implies that the adversary (who gets to know a nonzero vector perpendicular to the random vector hidden as the exponent of a group element) can break the Discrete Logarithm assumption (Assumption 1) for the group G exploited in the proposed VC scheme.

Lemma 2 *Let $G_\lambda := \langle g \rangle$ be a cyclic group of order a prime p of size λ bits. If there exists a probabilistic polynomial-time adversary $\mathcal{A}$ such that*

$$\Pr\left[\vec{x} \leftarrow \mathbb{Z}_p^n \ : \ \begin{array}{c} (\vec{y} \leftarrow \mathcal{A}(G_\lambda, g, g^{\vec{x}})) \\ \wedge (\vec{y} \neq \vec{0}) \wedge (\vec{x} \cdot \vec{y} = 0) \end{array} \right]$$

is non-negligible, then the Discrete Logarithm assumption (Assumption 1) does not hold for G_λ. $\diamondsuit$

Proof: Let $\mathcal{A}_1$ be the probabilistic polynomial-time adversary of the lemma, i.e., the probability ($\Pr[\mathcal{A}_1]$) described in the lemma is non-negligible. Using this adversary $\mathcal{A}_1$, we can construct an adversary $\mathcal{A}$ who can break the Discrete Logarithm assumption (Assumption 1). For given (G_λ, g, g^x), the adversary $\mathcal{A}$ generates $g^{\vec{x}} := (g^x, g^{x_2}, \ldots, g^{x_n})$ where $x_2, \ldots, x_n$ are sampled randomly from $\mathbb{Z}_p$. Then it sends $(G_\lambda, g, g^{\vec{x}})$ to $\mathcal{A}_1$ who, in response, will output nonzero $\vec{y} := (y_1, ..., y_n)$ such that $\vec{x} \cdot \vec{y} = 0$ with non-negligible probability. Now, $\mathcal{A}$ can retrieve x as $-(\sum_{i=2}^n x_i y_i)/y_1$ unless $y_1 = 0$. Note that $\Pr[y_1 \neq 0 | \vec{y} \neq \vec{0}] \geq \frac{1}{n}$, since $x, x_2, \ldots, x_n$ are all sampled randomly from $\mathbb{Z}_p$. Therefore,

$$\begin{aligned}\Pr[x \leftarrow \mathbb{Z}_p : x \leftarrow \mathcal{A}(G_\lambda, g, g^x)] &= \Pr[\mathcal{A}_1] \cdot \Pr[y_1 \neq 0 | \vec{y} \neq \vec{0}] \\ &\geq \Pr[\mathcal{A}_1]/n\end{aligned}$$

is also non-negligible (given that $n = O(\lambda^c)$ for some constant c), and the Discrete Logarithm assumption does not hold for G_λ. ■

Note that if we took $\vec{r}_0 = \vec{r}_1 = \vec{r}$ in the VC scheme, an adversary can forge the state by sending $g^{\vec{r} \cdot \vec{x}'_+}$ and $g^{\vec{r} \cdot \vec{x}'}$ in the proof with $\vec{x}'^\top_+ || \vec{x}'^\top \neq \vec{x}^\top_+ || \vec{x}^\top$ such that $(\vec{x}'_+ - \vec{x}_+) - (I - A)(\vec{x}' - \vec{x}) = 0$ without being rejected by Verify algorithm; at this time, the adversary should send the correct signal $\vec{u}' = \vec{u}$. Note, however, that the adversary can forge the signal after this time (without being rejected by Verify algorithm) using the forged state accepted at this time.

[10]Here, || denotes the concatenation of (row) vectors.

IV. Conclusion

We proposed a verifiable computation scheme tailored to linear dynamic system, which enables an actuator to verify controller's computation in order to detect all adversarial behaviors outside the plant-side such as forged signal or compromised controller. The proposed scheme is promising in a sense that it is efficient and is applicable for securing various control systems. Extending this work to the control systems with user's input, and further improving the efficiency of the proposed scheme will be an interesting future work as well as their implementations.

References

[1] K. Kogiso and T. Fujita, "Cyber-security enhancement of networked control systems using homomorphic encryption," in *54th IEEE Conference on Decision and Control*, 2015, pp. 6836–6843.

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

[3] J. Kim, C. Lee, H. Shim, J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Encrypting controller using fully homomorphic encryption for security of cyber-physical systems," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 175–180, 2016.

[4] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, "A secure control framework for resource-limited adversaries," *Automatica*, vol. 51, pp. 135–148, 2015.

[5] R. Gennaro and D. Wichs, "Fully homomorphic message authenticators," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2013, pp. 301–320.

[6] C. Joo and A. Yun, "Homomorphic authenticated encryption secure against chosen-ciphertext attack," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2014, pp. 173–192.

[7] J. H. Cheon, K. Han, S. Hong, H. J. Kim, J. Kim, S. Kim, H. Seo, H. Shim, and Y. Song, "Toward a secure drone system: Flying with real-time homomorphic authenticated encryption," *IEEE Access*, vol. 6, pp. 24 325–24 339, 2018.

[8] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Annual Cryptology Conference*, 2010, pp. 465–482.

[9] R. Freivalds, "Fast probabilistic algorithms," in *International Symposium on Mathematical Foundations of Computer Science*, 1979, pp. 57–69.

[10] J. Kim, H. Shim, and K. Han, "Dynamic controller that operates over homomorphically encrypted data for infinite time horizon," *arXiv:1912.07362*, 2019.

[11] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *IEEE Symposium on Security and Privacy*, 2013, pp. 238–252.

[12] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *Journal of ACM*, vol. 27, no. 4, pp. 701–717, 1980.

[13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct NIZKs without PCPs," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2013, pp. 626–645.

[14] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture," in *23rd USENIX Security Symposium*, 2014, pp. 781–796.

[15] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song, "Libra: Succinct zero-knowledge proofs with optimal prover computation," in *Annual International Cryptology Conference*, 2019, pp. 733–764.

[16] J. Groth, "Short pairing-based non-interactive zero-knowledge arguments," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2010, pp. 321–340.