



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

이학 석사 학위논문

# Authenticated Computation of Control Signal from Dynamic Controllers

(동적제어계 제어신호의 인증된 계산)

2020년 8월

서울대학교 대학원

수리과학부

이승범

# Authenticated Computation of Control Signal from Dynamic Controllers

(동적제어계 제어신호의 인증된 계산)

지도교수 천정희

이 논문을 이학 석사 학위논문으로 제출함

2020년 7월

서울대학교 대학원

수리과학부

이승범

이승범의 이학 석사 학위논문을 인준함

2020년 6월

위 원 장 심                    형                    보 (인)

부 위 원 장 천                    정                    희 (인)

위            원 현                    동                    훈 (인)

# Authenticated Computation of Control Signal from Dynamic Controllers

A dissertation  
submitted in partial fulfillment  
of the requirements for the degree of  
Master of Mathematics  
to the faculty of the Graduate School of  
Seoul National University

by

Seungbeom Lee  
Dissertation Director : Professor Jung Hee Cheon

Department of Mathematical Sciences  
Seoul National University

August 2020

© 2020 Seungbeom Lee

All rights reserved.

# Abstract

## Authenticated Computation of Control Signal from Dynamic Controllers

Seungbeom Lee

Department of Mathematical Sciences

The Graduate School

Seoul National University

Significant concerns on networked control system are security problems caused by the network or the controller, since a compromise on them can cause a devastating behavior or entire failure of the system.

In this paper, we first propose a fundamental solution to this problem by exploiting the verifiable computation to prevent malicious behavior of controller. First, we propose a new authenticated computation to check the matrix-vector multiplications—the main arithmetic of a controller—and to check the updates on the states of the controller. It enables a plant-side not only to check computations of a controller with much less computational cost than that required for the computations itself, but also to detect any compromise on the network or the controller.

In addition, the proposed authenticated computation can be applied to linear dynamic systems without any additional asymptotic computational overhead on the actuator and the controller, since the verification cost of the actuator is independent from the dimension of the states.

To further reduce the cost of the actuator, we also propose a batch verification and multi-exponentiation method. These methods dramatically reduce the constant overhead of the controller so that the performance estimation of the proposed scheme demonstrates its applicability in practice.

**Key words:** Verifiable Computation, Discrete Logarithm, Dynamic System

**Student Number:** 2017-23983

# Contents

<b>Abstract</b>	<b>i</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Problem Formulation and Preliminaries</b>	<b>5</b>
2.1 Notation . . . . .	5
2.2 Problem Formulation . . . . .	6
2.3 Conversion of Real-valued Parameters to Integers . . . . .	7
2.4 Verifiable Computation . . . . .	9
2.5 Freivalds' Algorithm: Verifying Matrix Multiplication . . . . .	10
2.6 Discrete Logarithm Assumption on Finite Group . . . . .	11
<b>3 Verification of Controller Computation</b>	<b>13</b>
3.1 Four points of proposed VC Scheme . . . . .	14
3.1.1 Randomized Verification . . . . .	14
3.1.2 Compressed Commitments . . . . .	15
3.1.3 Knowledge of Exponent . . . . .	15
3.1.4 Proof of Equality . . . . .	16
3.2 VC schemes for linear dynamic system . . . . .	17
3.3 Security of the proposed VC . . . . .	19

## CONTENTS

3.4	Efficiency of the proposed VC . . . . .	21
3.5	Improving Efficiency . . . . .	23
3.6	Performance Estimation of proposed scheme . . . . .	25
<b>4</b>	<b>Conclusions</b>	<b>27</b>
	<b>Appendix</b>	<b>32</b>
4.1	Proof of Lemma 2 . . . . .	32
4.2	Necessity of Alternative Random Vector . . . . .	33
4.3	Algorithms: Batch Verification . . . . .	34
4.4	Algorithms: Multi-exponentiation . . . . .	35
	<b>Abstract (in Korean)</b>	<b>37</b>
	<b>Acknowledgement (in Korean)</b>	<b>38</b>

# Chapter 1

## 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 [KF15, FSB17, KLS<sup>+</sup>16]. 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 [TSSJ15], it can not be a fundamental solution, since HE does not provide non-malleability, i.e., an adversary can manipulate the

## CHAPTER 1. INTRODUCTION

encrypted controller’s signal so that the signal is decrypted to an incorrect value.

To handle this problem, Homomorphic Authenticated Encryption (HAE) [GW13, JY14] has been proposed to achieve non-malleability in HE, but cost overhead of the verification process makes this scheme impractical. To this end, efficient HAE was proposed in [CHH<sup>+</sup>18] for linear dynamic system. However, it requires considerable computational cost and information as much as the controller for the plant-side to verify the validity of the encrypted control signal. Moreover, the proposed system can be applied only to finite number of states, and this is serious drawback for infinite dynamic system.

In this paper, we investigate a new approach that enables a plant-side to detect any misbehavior of the controller or any forgery on the signal efficiently in infinite time horizon. The idea is to apply the recent primitive from complexity theory and cryptography called *Verifiable Computation* (VC) 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, with the *proof* generated by the controller. The point is that an adversary can not deceive the plant-side with forged result and false proof unless he can break some cryptographic hardness assumptions. In theory, many VC schemes have shown its possibility in verifying diverse computations. However, direct application of the previous VC schemes to linear dynamic systems requires interactions between the controller and the actuator [GKR15, XZZ<sup>+</sup>19] or yields considerable verification cost for the plant-side, which is higher than repeating the computation of dynamic systems by the actuator in most cases [PHGR13, BSCTV14].

## CHAPTER 1. INTRODUCTION

With exploiting the concept of VC, we propose a novel scheme to authenticate the computation of the controller as well as the transmission of the control signals for a linear dynamic system consisting of state updates and matrix-vector multiplications. At first, we adopt a randomized verification algorithm called *Freivalds' algorithm* [Fre79] to verify these matrix-vector multiplications. In this algorithm, a plant-side prepares random verification vectors and takes their inner-product with input and output of the controller to check the correctness of the output. It can be efficient to reduce the computational cost, but a naive application of this algorithm requires 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.

To overcome this obstacle, we devise a method to perform verification algorithm working with exponent form for each step. In detail, the plant-side gives the controller a random verification vector  $\vec{r} = (r_1, \dots, r_n)$  in an exponent form  $(g^{r_1}, \dots, g^{r_n})$  for a generator  $g$  of a cyclic group  $G$ . 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 on base  $g$  is the inner-product of the state with the verification vector. Given this exponent form of state, the plant-side receive proof with lower communication cost, and the plant-side can perform a verification algorithm at reasonable computational cost. Moreover, unlike naive approach, both costs are independent of the dimension of the state, dramatically reducing total verification cost. Finally, to prevent a (malicious) controller from deceiving with an incorrect state in generation of each commitment of the state, we also devise a method to check whether each state in the commitment is consistent with each other, i.e., if the state at time

## CHAPTER 1. INTRODUCTION

$t$  is updated from the state at time  $t - 1$  and if it is used to get the output signal at time  $t$ . Putting these together, we show that the controller cannot deceive the plant-side (with proposed VC scheme) with incorrect signal in an infinite number of states, unless it breaks the hardness assumptions on the Discrete Logarithm of  $G$ .

We emphasize that the proposed VC scheme incurs no asymptotic computational overhead on the controller and the plant-side when applied to linear dynamic systems. More precisely, the additional cost of controller for the proof generation is only linear on the dimension of controller state, which is less than the cost of the original computation which is quadratic in the dimension of the state. Furthermore, the cost of plant-side for verification is only linear on the size of signal and measurement, and does not depend on the size of state.

Though the proposed gives asymptotically efficient performance, it may not be practically efficient in dynamic systems with lower dimension and small word size due to constant computational cost on the plant-side, arising from cryptographic operations. To reduce this cost, we also introduce a batch verification method which enables the plant-side to perform multiple verification processes (for each time) in batch. Together with a multi-exponentiation method, it makes the verification process cheaper than the computation even when the state dimension is not very large (e.g.,  $\approx 10$ ).

As a result, the proposed application of VC to linear dynamic systems enables an actuator to detect all possible attacks on the network and the controller, such as forgery on signals or compromise on the controller. Besides, as it incurs only a low cost overhead to the actuator and the controller in asymptotic and actual cost, it can be a promising solution in theory and in practice.

# Chapter 2

## Problem Formulation and Preliminaries

### 2.1 Notation

In this paper, we use the following notations.  $\mathbb{Z}$  denotes the set of integers, and  $\mathbb{F} := \mathbb{Z}_p$  denotes the field of prime order  $p$ , i.e., an integer modulo  $p$ . Vectors are denoted by lower letters  $\vec{x}, \vec{y}, \vec{z}$  while matrices are denoted by upper letters, e.g.,  $A, B, C$ , and  $\vec{x} \cdot \vec{y}$  denotes inner-product of vectors.  $A\vec{x}$  or  $\vec{x}^\top A$  denotes multiplication between a matrix and a vector. Let  $G$  denote the finite group having  $g$  as an element, where multiplication is denoted by  $\cdot$  and  $g^m$  denotes  $g \cdots g$  (multiplied by  $m$ -times). For a vector  $\vec{x} := (x_1, \dots, x_n)$ , the  $g^{\vec{x}}$  denotes  $(g^{x_1}, \dots, g^{x_n})$ . We say that a function is *negligible* in  $\lambda$ , and denote it by  $negl(\lambda)$ , if it is  $o(\lambda^{-c})$  for every fixed constant  $c$ . We use standard notation from probability:  $\Pr[A|B]$  denotes the conditional probability;  $\Pr[x \leftarrow X : E]$  denotes the probability of  $E$  occurs when  $x$  is sampled from  $X$ . Finally,  $y \leftarrow M(x)$  denotes the event

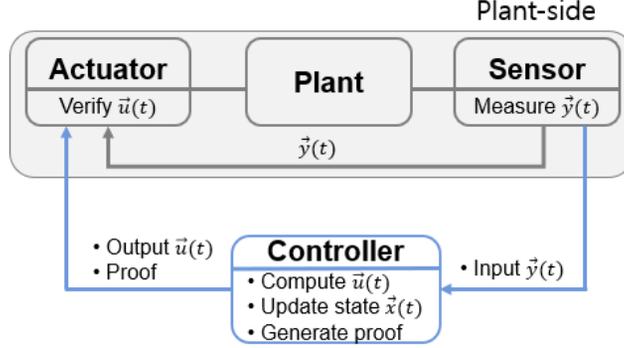


Figure 2.1: Configuration of control system with verifiable computation (VC)

that a machine (or an algorithm)  $M$  outputs  $y$  given an input  $x$ , while  $x \stackrel{\S}{\leftarrow} X$  denotes that  $x$  is sampled uniformly from the set  $X$ .

## 2.2 Problem Formulation

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

$$\begin{aligned}\vec{z}(t+1) &= \mathbf{A}\vec{z}(t) + \mathbf{B}\vec{y}(t), \\ \vec{u}(t) &= \mathbf{C}\vec{z}(t) + \mathbf{D}\vec{y}(t)\end{aligned}\tag{2.1}$$

where  $\vec{z}(t) \in \mathbb{R}^n$  is the state with initial value  $\vec{z}(0) = \vec{z}_0 \in \mathbb{R}^n$ ,  $\vec{u}(t) \in \mathbb{R}^l$  is the output, and  $\vec{y}(t) \in \mathbb{R}^m$  is the input of the controller. In the closed-loop system consisting of the controller and the plant, the controller receives the signal  $\vec{y}(t)$  (from the sensor), performs the operation of Eq. (2.1), then transmits the output signal  $\vec{u}(t)$  to the actuator, as described in Fig. 2.1.

## CHAPTER 2. PROBLEM FORMULATION AND PRELIMINARIES

Note that the VC scheme's objectives are as follows:

- *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 operations of Eq. (2.1) 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 Eq. (2.1).

Here, we must assume that the actuator has information on the matrices  $A, B, C, D$  of Eq. (2.1), sensor measurement  $\vec{y}(t)$  at each time  $t$ , and the initial state  $\vec{z}(0)$ . If not, it is impossible to distinguish the correct  $\vec{u}(t)$  from incorrect one computed from compromised input: matrices, measurements or initial state.

### 2.3 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 over a finite field while the operations(Eq.(2.1)) are over a field of real numbers. Luckily, we can convert the system(Eq.(2.1)) so that the system 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<sup>1</sup>. Then, a matrix  $H \in \mathbb{R}^{n \times l}$

---

<sup>1</sup>If the controller is not observable, it implies that the controller of Eq.(2.1) is not

## CHAPTER 2. PROBLEM FORMULATION AND PRELIMINARIES

can be found by pole-place techniques so that the eigenvalues of  $\mathbf{A} - \mathbf{H}\mathbf{C}$  are all integers. Then, with an invertible matrix  $\mathbf{T} \in \mathbb{R}^{n \times n}$ , the matrix  $\mathbf{A} - \mathbf{H}\mathbf{C}$  can be transformed into Jordan canonical form so that we obtain  $\mathbf{T}(\mathbf{A} - \mathbf{H}\mathbf{C})\mathbf{T}^{-1} \in \mathbb{Z}^{n \times n}$ . See [KSH19, Section IV] for more details. Then, the system Eq.(2.1) can be converted to the system over integers as

$$\begin{aligned} \vec{x}(t+1) &= A\vec{x}(t) + B \left[ \frac{\vec{y}(t)}{l_1} \right] + H[l_2^2 \cdot \vec{u}_{\mathbb{Z}}(t)] \in \mathbb{Z}^n \\ \vec{u}_{\mathbb{Z}}(t) &= C\vec{x}(t) + D \left[ \frac{\vec{y}(t)}{l_1} \right] \in \mathbb{Z}^l, \end{aligned} \tag{2.2}$$

with the initial value  $\vec{x}(0) = \mathbf{T}z_0/(l_1l_2)$ , where the matrices  $A := \mathbf{T}(\mathbf{A} - \mathbf{H}\mathbf{C})\mathbf{T}^{-1}$ ,  $B := [(\mathbf{T}\mathbf{B} - \mathbf{H}\mathbf{D})/l_2]$ ,  $C := [\mathbf{C}/l_2]$ ,  $D := [\mathbf{D}/l_2^2]$ , and  $H := [\mathbf{H}/l_2]$  are converted into integer component matrices, and the state  $\vec{z}$  and the output  $\vec{u}$  are converted into  $\vec{x}$  and  $\vec{u}_{\mathbb{Z}}$ , with the value of  $\mathbf{T}\vec{z}/(l_1l_2)$  and  $\vec{u}/(l_1l_2^2)$  respectively.

Finally, the following proposition suggests that the performance error of the system Eq. (2.2) over integers is negligible compared with Eq. (2.1), when the factors  $1/l_1$  and  $1/l_2$  is chosen sufficiently large.

**Proposition 1** ([KSH19]). *On Eq. (2.1) and Eq. (2.2), 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  $\|l_1l_2 \cdot \mathbf{T}^{-1} \cdot \vec{x}(t) - \vec{z}(t)\| < \epsilon$  and  $\|l_1l_2^2 \cdot \vec{u}(t) - \vec{v}(t)\| < \epsilon$  for all  $t \geq 0$ .*

As a result, in the rest of this paper, we consider the problem of verifying the operation of Eq. (2.2) over integers instead of Eq. (2.1) over real numbers.

---

of minimal realization, so can be reduced to obtain an observable pair. For the details, see [KSH19, Section IV], for example.

## 2.4 Verifiable Computation

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

**Definition 1** (Verifiable Computation [PHGR13]). *A verifiable computation scheme consists of three polynomial-time algorithms (KeyGen, Compute, Verify) defined as follows.*

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

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

- *Correctness: For any function  $F$  and any input  $\mu$ , if we run  $\text{KeyGen}(F, \lambda) \rightarrow (EK_F, VK_F)$  and  $\text{Compute}(EK_F, \mu) \rightarrow (\nu, \pi_\nu)$ , then we always get  $\text{Verify}(VK_F, \mu, \nu, \pi_\nu) \rightarrow \text{acc}$ .*

- *Security: For any function  $F$  and any probabilistic polynomial-time adversary  $\mathcal{A}$ , the following probability is  $\text{negl}(\lambda)$ .*

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

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

Usually, the verifier (in this case, the actuator) performs `KeyGen` (one-time) and `Verify` algorithms while the prover (in this case, the controller) performs `Compute` algorithm to generate the proof of its computation correctness.

## 2.5 Freivalds' Algorithm: Verifying Matrix Multiplication

Freivalds' algorithm [Fre79] is a probabilistic algorithm for verification of matrix multiplication, and here we introduce the version for matrix-vector multiplication which is the main concern of the scheme. In the following, let  $\mathbb{F}$  be a finite field and  $F$  be an  $n \times m$  matrix over  $\mathbb{F}$ .

- *Algorithm:* A verifier randomly samples  $\vec{r} \in \mathbb{F}^n \setminus \{\vec{0}\}$  and precomputes  $\vec{f} := \vec{r}^\top F \in \mathbb{F}^m$ . For an input  $\vec{\mu} \in \mathbb{F}^m$  and a claimed output  $\vec{v} \in \mathbb{F}^n$  (by a prover), the verifier accepts the output only if  $\vec{r} \cdot \vec{v} = \vec{f} \cdot \vec{\mu}$ .
- *Security:*  $\Pr[\text{verifier accepts } \vec{v} \mid \vec{v} \neq F\vec{\mu}] \leq \frac{n}{|\mathbb{F}|}$  by the following lemma 1; if  $\vec{v} \neq F\vec{\mu}$  is an incorrect output that verifier accepts,  $\vec{v}$  is a solution of the nonzero polynomial  $r : \mathbb{F}^n \rightarrow \mathbb{F}$  defined by  $r(\vec{x}) := \vec{r} \cdot \vec{x} - \vec{f} \cdot \vec{\mu}$ .

- *Efficiency*: The vector  $\vec{r}$  and  $\vec{f}$  can be used over many verifications. Therefore, given these vectors, the computational cost (which is measured by number of multiplications required) of verification (checking if  $\vec{r} \cdot \vec{v} = \vec{f} \cdot \vec{\mu}$ ) is  $n + m$  which is much less than the cost  $nm$  of evaluating  $F\vec{\mu}$ .

**Lemma 1** (Schwartz-Zippel [Sch80]). *Let  $\mathbb{F}$  be a finite field, and  $f : \mathbb{F}^\ell \rightarrow \mathbb{F}$  be an  $\ell$ -variate nonzero polynomial of total degree (the sum of degrees of each variable)  $\delta$ . Then,*

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

## 2.6 Discrete Logarithm Assumption on Finite Group

Our VC scheme (Section 2.4) proposed in this paper relies on the following Discrete Logarithm assumption. For given security parameter  $\lambda$ , let  $G_\lambda := \langle g \rangle$  be a finite group of order a prime  $p$  of size  $\lambda$  bits where  $g$  is a generator of  $G_\lambda$ .

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

$$\Pr[x \xrightarrow{\$} \mathbb{Z}_p : x \leftarrow \mathcal{A}(G_\lambda, g, g^x)] = \text{negl}(\lambda).$$

Our scheme also exploits following well-known assumption on which the security of many VC schemes [GGPR13, PHGR13, BSCTV14, XZZ<sup>+</sup>19] are based.

## CHAPTER 2. PROBLEM FORMULATION AND PRELIMINARIES

**Assumption 2** (*n*-PKE [Gro10]). *The n-PKE (n-power knowledge of exponent) assumption holds for a finite field  $G_\lambda := \langle g \rangle$ , if for all probabilistic polynomial-time adversary  $\mathcal{A}$ , there exists a probabilistic polynomial time extractor  $\chi_{\mathcal{A}}$  such that the following probability is negligible.*

$$\Pr \left[ \begin{array}{l} \alpha, s_1, \dots, s_n \xleftarrow{\$} \mathbb{Z}_p \setminus \{0\} \\ \sigma := (G_\lambda, g, g^{s_1}, \dots, g^{s_n}, \\ \quad \quad \quad g^{\alpha s_1}, \dots, g^{\alpha s_n}) : \quad \quad \quad (c^* = c^\alpha) \\ (c, c^*) \leftarrow \mathcal{A}(\sigma) \quad \quad \quad \wedge (c \neq \prod_{i=1}^n g^{a_i s_i}) \\ (a_1, \dots, a_n) \leftarrow \chi_{\mathcal{A}}(\sigma) \end{array} \right]$$

where  $\chi_{\mathcal{A}}$  is given  $\mathcal{A}$ 's random tape.

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

# Chapter 3

## Verification of Controller Computation

In this chapter, we propose a verifiable computation scheme for checking computations of a controller, and analyze its security and efficiency.

With the integer conversion (Section 2.3), we can assume that following discrete-time controller is given over a sufficiently large finite field  $\mathbb{F}$ .<sup>1</sup> For simplicity of description and without loss of generality, we can substitute  $H[l_2^2 \cdot \vec{u}(t)]$  in Eq. 2.2 by  $H\vec{u}(t)$ , since an actuator given  $\vec{u}(t)$  can compute  $[l_2^2 \cdot \vec{u}(t)]$  by itself. Therefore, we will adopt the VC scheme on following discrete time system,

$$\begin{aligned}\vec{x}(t+1) &= A\vec{x}(t) + B\vec{y}(t) + H\vec{u}(t) \in \mathbb{F}^n, \\ \vec{u}(t) &= C\vec{x}(t) + D\vec{y}(t) \in \mathbb{F}^l.\end{aligned}\tag{3.1}$$

---

<sup>1</sup>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 2.3) is composed of bounded integers.

### 3.1 Four points of proposed VC Scheme

We first describe the design rationale of the VC scheme for linear dynamic system proposed in this paper.

#### 3.1.1 Randomized Verification

The starting point is to use Freivalds' algorithm (Section 2.5) for the actuator to check the computation Eq. (3.1) of 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*  $\vec{a} := \vec{r}^\top A$ ,  $\vec{b} := \vec{r}^\top B$ ,  $\vec{h} := \vec{r}^\top H$ ,  $\vec{c} := \vec{s}^\top C$ ,  $\vec{d} := \vec{s}^\top D$ . Then, on each time  $t$ , if the actuator is given the states  $\vec{x}(t+1)$ ,  $\vec{x}(t)$ , the signal  $\vec{u}(t)$  from the controller, and the measurement  $\vec{y}(t)$  from the sensor, the actuator checks if the following equations hold.

$$\begin{aligned} \vec{r} \cdot \vec{x}(t+1) &= \vec{a} \cdot \vec{x}(t) + \vec{b} \cdot \vec{y}(t) + \vec{h} \cdot \vec{u}(t), \\ \vec{s} \cdot \vec{u}(t) &= \vec{c} \cdot \vec{x}(t) + \vec{d} \cdot \vec{y}(t). \end{aligned} \tag{3.2}$$

Due to Freivalds' algorithm (Section 2.5), given states and signal is correct with high probability if these equations hold. Note that the actuator can check the equations only with the inner-product values, which we call *compressed states*, of  $\vec{x}(t)$  or  $\vec{x}(t+1)$  with the random verification vectors, in less computational cost than the whole computation (Eq. (3.1)). For this to work, 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.

### 3.1.2 Compressed Commitments

To resolve this problem, we may try to replace the state  $\vec{x}(t)$  by its *homomorphic commitment*, e.g.,  $g^{\vec{a}\cdot\vec{x}(t)}$  or  $g^{\vec{c}\cdot\vec{x}(t)}$ , which can be used to check the Eq. (3.2) without revealing the state itself. More precisely, at the setup stage, the actuator sends the verification vectors  $\vec{r}$ ,  $\vec{a}$ , and  $\vec{c}$  to the controller in their commitment forms<sup>2</sup>, i.e.,  $g^{\vec{r}}$ ,  $g^{\vec{a}}$ ,  $g^{\vec{c}}$ , and receives the commitments  $g^{\vec{r}\cdot\vec{x}(t+1)}$ ,  $g^{\vec{a}\cdot\vec{x}(t)}$ ,  $g^{\vec{c}\cdot\vec{x}(t)}$  of the compressed states computed by the controller. Then, the actuator checks Eq. (3.2) with the received commitments,  $\vec{u}(t)$  and  $\vec{y}(t)$ . Note that the commitment must be linearly homomorphic for the controller to derive the commitments of the compressed states from those of the verification vectors, and we use the commitment function ( $m \mapsto g^m$  and  $\vec{m} \mapsto g^{\vec{m}}$ ) that satisfies this property and is secure under the Discrete Logarithm assumption (Assumption 1).

### 3.1.3 Knowledge of Exponent

One may worry, however, that malicious controller deceives the actuator by sending a wrong signal  $\vec{u}'$  along with rigged commitments, i.e., the commitment which satisfies the equation Eq. (??) in the compressed commitment form, but is not generated by using  $\vec{u}$  and  $\vec{x}$ . To prevent this behavior, we adopt the  $n$ -PKE assumption (Assumption 2) to force the controller to generate the commitments of compressed states from the commitments of the verification vectors. In that case, even if the controller provides an output in their exponent form (i.e., committed form) and the actuator checks Eq. (3.2) in their exponent form, the assumption guarantees that the controller must *know* one solution of Eq. (3.2) corresponding to the

---

<sup>2</sup>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 be accepted by the actuator.

commitment. If the controller computed the solution from Eq. (3.1), it is the correct one. Otherwise, the controller generated a solution of Eq. (3.2) that does not satisfy Eq. (3.1). However, the security provided by randomized verification in Freivalds' algorithm implies that the probability to find such a solution is upper-bounded by negligible value  $n/|\mathbb{F}|$ , and also the probability that the adversary generates a solution of Eq. (3.2) without satisfying Eq. (3.1). To adopt this in our scheme, the actuator needs to roughly duplicate all the random vectors in commitment form to the controller by raising a randomly chosen fixed power. Refer to the below VC scheme for more detail.

### 3.1.4 Proof of Equality

By linearity check, actuator can verify that given proof is generated with using evaluation key and some vector which controller knows ( $\vec{x}(t)$  if controller is valid). However, we still need to confirm that the same state  $\vec{x}(t)$  at time  $t$  is used through compressed commitments in generation of both the state  $\vec{x}(t+1)$  and the signal  $\vec{u}(t)$ . In fact, randomized verification based on compressed commitments enables the actuator to check whether the signal  $\vec{u}(t)$  outputted by the controller satisfies the Eq. (3.2) upon receiving compressed commitments of  $\vec{x}(t)$ , but does not check if the value  $\vec{x}(t)$  in the first equation is same as  $\vec{x}(t)$  in the second equation in the Eq. (3.2). Also,  $\vec{x}(t)$  in its compressed commitment at time  $t$  can be different from  $\vec{x}(t)$  in its compressed commitment at time  $(t+1)$ . It can be achieved by letting the controller provide a proof of equality of  $\vec{x}(t)$  and  $\vec{x}'(t)$  when sending their compressed commitments  $g^{\vec{r}\cdot\vec{x}(t)}$  and  $g^{\vec{a}\cdot\vec{x}'(t)}$ . To this end, we let the controller generate and provide to the actuator  $g^{\vec{r}\cdot\vec{x}(t)}$ ,  $g^{(\vec{a}-\vec{r})\cdot\vec{x}(t)}$ , and their powers by random verification vectors  $\rho$  and  $\alpha$ , respectively, upon re-

## CHAPTER 3. VERIFICATION OF CONTROLLER COMPUTATION

ceiving  $g^{\vec{r}}$  and  $g^{\vec{a}-\vec{r}}$  from the actuator. The proof also relies on the  $n$ -PKE assumption.

We remark that the verification vector  $\vec{r}(t)$  and  $\vec{r}(t+1)$  at time  $t$  and  $(t+1)$  must be random and independent for the security proof, i.e. if verification vectors are dependent, an adversary can generate false proof without knowledge of randomized vectors. Since the proof involves only two consecutive rounds, however, the VC scheme will use different  $\vec{r}_0$  and  $\vec{r}_1$  alternatively on each time  $t$  of even and odd.

### 3.2 VC schemes for linear dynamic system

Now, the formal description of the VC scheme which is composed of three algorithms (KeyGen, Compute, Verify) is as follows.

#### [The VC Scheme for Linear Dynamic Systems]

- **KeyGen**( $\lambda, F := \{A, B, C, D, H\}$ )  $\rightarrow (EK_F, VK_F)$ : Let  $\lambda$  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 the Eq.(3.2). Take a finite field  $\mathbb{F} := \mathbb{Z}/p\mathbb{Z}$  of prime order  $p$  greater than  $2^\lambda$ , 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:

$$\left( g^{\vec{r}_0}, g^{\vec{a}_1 - \vec{r}_0}, g^{\vec{c} - \vec{r}_0}, g^{\rho_0 \vec{r}_0}, g^{\alpha_0(\vec{a}_1 - \vec{r}_0)}, g^{\gamma_0(\vec{c} - \vec{r}_0)}, \right. \\ \left. g^{\vec{r}_1}, g^{\vec{a}_0 - \vec{r}_1}, g^{\vec{c} - \vec{r}_1}, g^{\rho_1 \vec{r}_1}, g^{\alpha_1(\vec{a}_0 - \vec{r}_1)}, g^{\gamma_1(\vec{c} - \vec{r}_1)} \right),$$

## CHAPTER 3. VERIFICATION OF CONTROLLER COMPUTATION

and the verification key  $VK_F$  as:

$$(\alpha_0, \alpha_1, \gamma_0, \gamma_1, \rho_0, \rho_1, \vec{s}, \vec{b}_0, \vec{b}_1, \vec{h}_0, \vec{h}_1, \vec{d}).$$

- **Compute**( $EK_F, t, \vec{x}(t), \vec{y}(t)$ )  $\rightarrow (t, \vec{u}(t), \pi_t)$ : For time  $t$ , derive the updated state  $\vec{x}(t+1)$  and  $\vec{u}(t)$  from  $\vec{x}(t), \vec{y}(t)$  with the Eq.(3.1), and output the proof  $\pi_t$  defined as follows where we abbreviate  $\vec{x}(t+1)$  to  $\vec{x}_1$  and  $\vec{x}(t)$  to  $\vec{x}$ :

– when  $t$  is even,

$$(g^{\vec{r}_0 \cdot \vec{x}_1}, g^{(\vec{a}_0 - \vec{r}_1) \cdot \vec{x}}, g^{(\vec{c} - \vec{r}_1) \cdot \vec{x}}, \\ g^{\rho_0 \vec{r}_0 \cdot \vec{x}_1}, g^{\alpha_1 (\vec{a}_0 - \vec{r}_1) \cdot \vec{x}}, g^{\gamma_1 (\vec{c} - \vec{r}_1) \cdot \vec{x}}).$$

– when  $t$  is odd,

$$(g^{\vec{r}_1 \cdot \vec{x}_1}, g^{(\vec{a}_1 - \vec{r}_0) \cdot \vec{x}}, g^{(\vec{c} - \vec{r}_0) \cdot \vec{x}}, \\ g^{\rho_1 \vec{r}_1 \cdot \vec{x}_1}, g^{\alpha_0 (\vec{a}_1 - \vec{r}_0) \cdot \vec{x}}, g^{\gamma_0 (\vec{c} - \vec{r}_0) \cdot \vec{x}}).$$

- **Verify**( $VK_F, t, \vec{u}(t), \vec{y}(t), \pi_{t-1}, \pi_t$ )  $\rightarrow acc$  or  $rej$ : Parse the proof  $\pi_{t-1}$  and  $\pi_t$  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.<sup>3</sup>

– Equation Check (when  $t$  is even)

$$G_1 = G_2 \cdot g_1 \cdot g^{\vec{b}_0 \cdot \vec{y} + \vec{h}_0 \cdot \vec{u}}, \quad g^{\vec{s} \cdot \vec{u} - \vec{d} \cdot \vec{y}} = G_3 \cdot g_1 \quad (3.3)$$

---

<sup>3</sup>When  $t = 0$ , verifier uses the initial state  $\vec{x}_0$  instead of  $\pi_{-1}$  to get  $g_1 = g^{\vec{r}_1 \cdot \vec{x}_0}$  required for the Equation Check.

## CHAPTER 3. VERIFICATION OF CONTROLLER COMPUTATION

– Linearity Check (when  $t$  is even)

$$G_1^{\rho_0} = G'_1, \quad G_2^{\alpha_0} = G'_2, \quad G_3^{\gamma_0} = G'_3 \quad (3.4)$$

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

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

*Proof.* The correctness follows from that of Freivalds' algorithm (Section 2.5) and the fact that discrete group  $G$  is additive, i.e.,  $g^{m_1} \cdot g^{m_2} = g^{m_1+m_2}$ . The security and the efficiency will be described in the following subsection.  $\square$

### 3.3 Security of the proposed VC

Given that  $n$ -PKE assumption (Assumption 2) is true, we show that no probabilistic polynomial-time adversary can generate neither a false proof with an incorrect output  $\vec{u}'(t)$  ( $\neq \vec{u}(t)$ ), nor false proof whose first component,  $g^{\vec{r}_i \cdot \vec{x}'_1}$ , is generated with a manipulated state  $\vec{x}'_1 \neq \vec{x}_1$  on correct output  $\vec{u}(t)$ , which makes the **Verify** algorithm output *acc* with non-negligible probability for any time  $t \in \mathbb{Z}_{\geq 0}$ , without falsifying the Discrete Logarithm assumption (Assumption 1).

Without loss of generality, assume  $t$  is even, and by induction hypothesis on time  $t$ ,  $g_1$  is from the correct proof, i.e., we can assume that  $g_1$  from  $\pi'_{t-1}$  is  $g^{\vec{r}_1 \cdot \vec{x}}$ . Let  $\pi'_t = (G_1, G_2, G_3, G'_1, G'_2, G'_3)$  be the false proof (from the adversary) accepted by **Verify** algorithm, which consists of knowledge

### CHAPTER 3. VERIFICATION OF CONTROLLER COMPUTATION

of exponent(3.1.3) and proof of equality(3.1.4) Then, passing the linearity check, whose security is ensured by  $n$ -PKE assumption, implies that  $G_1 = g^{\vec{r}_0 \cdot \vec{x}'_1}$ ,  $G_2 = g^{(\vec{a}_0 - \vec{r}_1) \cdot \vec{x}'}$ , and  $G_3 = g^{(\vec{c} - \vec{r}_1) \cdot \vec{x}''}$  for some  $\vec{x}'_1$ ,  $\vec{x}'$ , and  $\vec{x}''$  all of which the adversary *knows*.<sup>4</sup> Moreover, the proof  $\pi'_t$  and the output  $\vec{u}'$  from the adversary pass the equation check, and it implies that

$$\begin{aligned}\vec{r}_0 \cdot \vec{x}'_1 &= (\vec{a}_0 - \vec{r}_1) \cdot \vec{x}' + \vec{r}_1 \cdot \vec{x} + \vec{b}_0 \cdot \vec{y} + \vec{h}_0 \cdot \vec{u}', \\ \vec{s} \cdot \vec{u}' &= (\vec{c} - \vec{r}_1) \cdot \vec{x}'' + \vec{r}_1 \cdot \vec{x} + \vec{d} \cdot \vec{y}.\end{aligned}\tag{3.5}$$

With honest computation, the adversary can get the correct values  $\vec{x}'_1$ ,  $\vec{x}$ , and  $\vec{u}$  which satisfy that

$$\begin{aligned}\vec{r}_0 \cdot \vec{x}'_1 &= (\vec{a}_0 - \vec{r}_1) \cdot \vec{x} + \vec{r}_1 \cdot \vec{x} + \vec{b}_0 \cdot \vec{y} + \vec{h}_0 \cdot \vec{u}, \\ \vec{s} \cdot \vec{u} &= (\vec{c} - \vec{r}_1) \cdot \vec{x} + \vec{r}_1 \cdot \vec{x} + \vec{d} \cdot \vec{y}.\end{aligned}\tag{3.6}$$

Subtracting Eq.(3.6) from Eq.(3.5) and rearranging, we get

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

This equation implies that if the adversary had generated a false proof on incorrect output, i.e.,  $\vec{u}' \neq \vec{u}$ , then  $(\vec{u}' - \vec{u} - C(\vec{x}'' - \vec{x}) || \vec{x}'' - \vec{x})$  is a nonzero vector which is also *known* to the adversary and is perpendicular to the random vector  $(\vec{s} || \vec{r}_1)$  hidden by  $g^{\vec{s}}$  and  $g^{\vec{r}_1}$ . Otherwise, the adversary had generated the false proof with  $\vec{x}'_1 \neq \vec{x}'_1$  and  $\vec{u}' = \vec{u}$  (if not, the previous case occurs), then  $(\vec{x}'_1 - \vec{x}'_1 - A(\vec{x}' - \vec{x}) || \vec{x}' - \vec{x})$  is a nonzero vector which is also *known* to the adversary and is perpendicular to the random vector

---

<sup>4</sup>In formal proof, we exploit the extractor  $\chi_{\mathcal{A}}$  to get these vectors  $\vec{x}'_1$ ,  $\vec{x}'$ , and  $\vec{x}''$  from  $\mathcal{A}$ .

## CHAPTER 3. VERIFICATION OF CONTROLLER COMPUTATION

$(\vec{r}_0 || \vec{r}_1)$  hidden by discrete logarithm problem. ( $g^{\vec{r}_0}$  and  $g^{\vec{r}_1}$ )

Now, following lemma implies that such adversary falsifies 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  $\lambda$  bits. If there exists a probabilistic polynomial-time adversary  $\mathcal{A}$  such that*

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

*is non-negligible, the Discrete Logarithm assumption (Assumption 1) does not hold for  $G_\lambda$ .*

*Proof.* See Appendix 4.1. □

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  $g^{\vec{r} \cdot \vec{x}'_1}$  and  $g^{\vec{r} \cdot \vec{x}'}$  in the proof with  $(\vec{x}'_1 || \vec{x}') \neq (\vec{x}_1 || \vec{x})$  such that  $(\vec{x}'_1 - \vec{x}_1) - (I - A)(\vec{x}' - \vec{x}) = 0$  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 also forge the signal after this time (without captured by **Verify** algorithm) using the forged state accepted at this time.

### 3.4 Efficiency of the proposed VC

Now we analyze the efficiency of the proposed VC scheme based on the required computational cost. To consider the cost, we count the number of multiplications over a finite field  $\mathbb{F}$  and the number of operations over

## CHAPTER 3. VERIFICATION OF CONTROLLER COMPUTATION

a finite group  $G$ . For simplicity, we do not count the number of additions over  $\mathbb{F}$  whose contribution to the total cost is dominated by that of multiplications.

The required cost of each algorithm in the proposed VC scheme is in TABLE 3.1 where  $\times_{\mathbb{F}}$ ,  $\times_G$ , and  $\text{Exp}_G$  denote the multiplications over  $\mathbb{F}$ , the multiplications  $((g_1, g_2) \mapsto g_1 \cdot g_2)$  over the group  $G$ , and the exponentiation operations  $((g, x) \mapsto g^x)$  in  $G$ , respectively. And  $n, m, l$  denote the dimension of  $\vec{x}, \vec{y}, \vec{u}$  in the control system of Eq.(3.1), respectively. Note that the cost of controller and actuator in applying the proposed VC scheme is *optimal* with respect to asymptotic cost analysis. The cost of controller to generate a proof on each time  $t$  is proportional to  $n$  which will be dominated by the cost of computing  $\vec{x}$  and  $\vec{u}$  as  $n$  increases. On the other hand, given that the actuator had performed **KeyGen** (one-time), it performs **Verify** with *constant* (3 for Knowledge of Exponent, others for Proof of Equality) number of operations on  $G$  in addition to  $2m + 2l$  multiplication cost in  $\mathbb{F}$ . The actuator's cost is asymptotically optimal in a sense that any VC scheme should require  $m + l$  cost to *read* the given signal  $\vec{u}$  and  $\vec{y}$  for verification. Note that the cost of **Verify** will be dominated by the cost of **Compute**( $\vec{x}, \vec{u}$ ) as  $n$  increases.

We also present the storage requirement of each component in the proposed VC scheme such as evaluation key  $EK_F$ , verification key  $VK_F$ , and the proof  $\pi_t$  in TABLE 3.2 where  $|\mathbb{F}|$  and  $|G|$  denotes the bitsize of each element of  $\mathbb{F}$  and  $G$ , respectively. For example, the evaluation key  $EK_F$  is composed of 12 number of elements of  $G$ . Note that the size of  $VK_F$  and  $EK_F$  is less than the storage requirement for the matrices  $A, B, C, D, H$  asymptotically. Also, after the actuator generates  $VK_F$  (whose size does not depend on the size of controller state) with **KeyGen** process, it does

Table 3.1: Computational cost of the VC scheme

	$\times_{\mathbb{F}}$	$\times_G$	$\text{Exp}_G$
KeyGen	$n^2 + nm + 2nl + ml$		$12n$
Compute $(\vec{x}, \vec{u})$	$n^2 + nm + 2nl + ml$		
Compute $(\pi_t)$		$6n$	$6n$
Verify	$2m + 2l$	$5$	$3 + 2$
Verify <sub>(Batch, Multi-Exp)</sub>		$8\kappa + 9\delta + 6$	$\frac{1}{B}$

Table 3.2: Storage requirement for the VC scheme

$VK_F$	$EK_F$	$\pi_t$	$A, B, C, D, H$
$(4n + 2l + 6)  \mathbb{F} $	$12n  G $	$6  G $	$(n^2 + nm + 2nl + ml)  \mathbb{F} $

not need the matrices  $A, B, C, D, H$  for **Verify** process. Note that the size of proof  $\pi_t$  is constant and is independent of the dimension of controller state or that of signals such as  $\vec{u}$  or  $\vec{y}$ .

### 3.5 Improving Efficiency

While the proposed VC shows optimal cost overhead in asymptotics, the actual cost of **Verify** can be more costly than that of **Compute** $(\vec{x}, \vec{u})$  when  $n$  is not sufficiently large (see TABLE 3.1). It is due to the cost of  $\text{Exp}_G$  (an exponentiation operation), each of which requires  $\log |\mathbb{F}|$  number of multiplications over  $G$  (Algorithm 2 in Appendix 4.4) where  $\log |\mathbb{F}| \approx 2\lambda = 256$ .<sup>5</sup> In this section, we propose methods to reduce the cost of **Verify**

<sup>5</sup>Here we assume 256-bit BN-curve [BN05] with 128 bits of security.

## CHAPTER 3. VERIFICATION OF CONTROLLER COMPUTATION

algorithm.

At first, we reduce the cost for Linearity Check(3  $\text{Exp}_G$ 's) with a batch verification strategy. Note that the Linearity Check is checking, for given  $\alpha$ , if  $g_i^\alpha = g'_i$  for given many pairs of  $(g_i, g'_i)$ 's. Our idea is to gather many pairs  $(G_i, G'_i)$  from each time  $t$ , then to perform the checks at once using the batch verification algorithm [BGR98]. We refer Algorithm 1 in Appendix 4.3 for the detailed description. As a result, when we use batch verification with  $B$ -time series, the *amortized* cost of Linearity Check in `Verify` is reduced from 3  $\text{Exp}_G$  to  $9\delta + 6$  multiplications over  $G$  and  $\frac{1}{B} \text{Exp}_G$ .

Secondly, we propose a method to mitigate the cost for Equation Check ( $(2m + 2l)$  multiplications over  $\mathbb{F}$  and 2  $\text{Exp}_G$ ) in `Verify` using multi-exponentiation. In abstraction, the task of Equation Check is to compute  $g^{\vec{s}\cdot\vec{u}-\vec{d}\cdot\vec{y}}$ , given  $\vec{s}$  and  $\vec{u}$  of dimension  $l$  each, and  $\vec{d}$  and  $\vec{y}$  of dimension  $m$  each. The naive method is to compute  $\vec{s}\cdot\vec{u} - \vec{d}\cdot\vec{y}$  first, then exponentiation ( $\vec{s}\cdot\vec{u} - \vec{d}\cdot\vec{y} \mapsto g^{\vec{s}\cdot\vec{u}-\vec{d}\cdot\vec{y}}$ ) resulting in  $l + m$  number of multiplications over  $\mathbb{F}$  and 1  $\text{Exp}_G$ . The Multi-exponent method exploits the fact that  $\vec{s} := (s_1, s_2, \dots, s_l)$  and  $\vec{d} := (d_1, d_2, \dots, d_m)$  is known in advance, and stores values precomputed from  $\vec{s}, \vec{d}$ . Then, with this values, one can compute  $g^{\vec{s}\cdot\vec{u}-\vec{d}\cdot\vec{y}}$  for given  $\vec{s}$  and  $\vec{d}$  in much less computational cost. We refer Algorithm 3 in Appendix 4.4 for detailed description. Concretely, the cost of Equation Check is reduced to  $8\kappa$  multiplications over  $G$  where  $\kappa$  is the bitsize (usually, 16–32) of each component of  $\vec{u}$  and  $\vec{y}$ , using  $2(2^m + 2^l)|G|$  storage for precomputed values, or  $4\kappa$  multiplications over  $G$ , using  $2^{m+l+1}|G|$  storage for precomputed values.

In TABLE 3.1, we recorded the resulting cost at `Verify`<sub>Batch, Multi-Exp</sub> which was amortized by  $B$ -time series with  $2^{-\delta}$  probability of false acceptance where  $\kappa$  is the bitsize of each component of  $\vec{u}$  and  $\vec{y}$ .

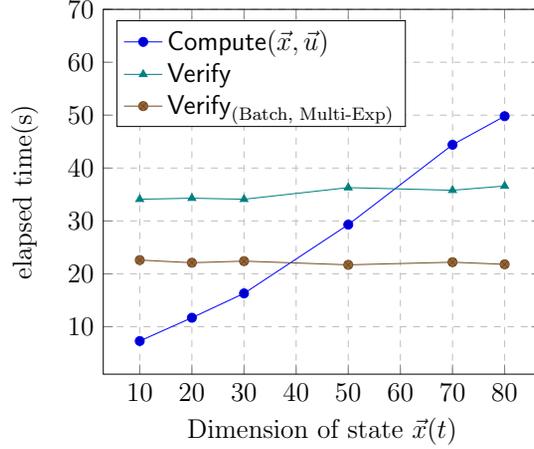


Figure 3.1: Cost estimation on various dimension of state

### 3.6 Performance Estimation of proposed scheme

Dim of state	Comp( $\vec{x}, \vec{u}$ )	Comp( $\pi_t$ )	Verify	Verify <sub>batch</sub>
10	7.3s	303.8s	34.1s	22.6s
20	11.7s	362.4s	34.3s	21.7s
30	16.3s	602.8s	34.1s	22.0s
50	29.3s	1088.6s	36.3s	22.4s
70	44.4s	1644.7s	35.8s	21.9s
80	49.8s	2021.4s	36.6s	22.2s

Table 3.3: Elapsed time of the VC scheme

To quantify the efficiency of proposed VC scheme, we tested each algorithm with example parameters<sup>6</sup> with control system having transfer function coefficients. In this experiment, we didn't use multi-exponentiation method, since dimension of output and input is only 1. The cost is es-

<sup>6</sup> $n \in \{10, 20, 30, 50, 70, 80\}, m = l = 1, \kappa = 16, \delta = 10, B = 16.$

## CHAPTER 3. VERIFICATION OF CONTROLLER COMPUTATION

timated by the number of  $\times_F$  or  $\times_G$  while  $\text{Exp}_G$  is converted<sup>7</sup> to  $\frac{3|G|}{2}$  number (See Algorithm 2) of  $\times_G$ 's. In Table 3.3, we can compare the elapsed time of each algorithm on various dimension of state  $\vec{x}(t)$ . The proof generation ( $\text{Compute}(\pi_t)$ ) is costly than the computation of signals ( $\text{Compute}(\vec{x}, \vec{u})$ ), but not more than  $\times 45$ . Note that the cost of verification ( $\text{Verify}$  or  $\text{Verify}_{\text{Batch}}$ ) is *constant* regardless of the dimension of state. We also remark that  $\text{Verify}_{\text{Batch}}$  with efficiency improvement (Section 3.5) is much cheaper than  $\text{Verify}$ , which makes it cheaper than  $\text{Compute}(\vec{x}, \vec{u})$  even when the dimension of state is about 35.

---

<sup>7</sup>We assume 256-bit BN-curve [BN05] with 128 bits of security.

# Chapter 4

## Conclusions

We proposed a Verifiable Computation scheme tailored to linear dynamic system which enables an actuator to verify controller's computation in order to detect any adversarial behaviors on the network or the controller such as forged signal or compromised controller. The proposed scheme is practical in a sense that the cost of verification is much cheaper than the cost of controller computation and the overhead of controller is not significant, so it is applicable for securing various control systems.

Extending this work to the control system with user's input, or reducing the computational cost of the proposed VC scheme will be an interesting future work as well as their efficient implementations.

# Bibliography

- [BGR98] Mihir Bellare, Juan A Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 236–250. Springer, 1998.
- [BN05] Paulo SLM Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *International Workshop on Selected Areas in Cryptography*, pages 319–331. Springer, 2005.
- [BSCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 781–796, 2014.
- [CHH<sup>+</sup>18] 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*, 6:24325–24339, 2018.
- [Fre79] Rūsiņš Freivalds. Fast probabilistic algorithms. In *International Symposium on Mathematical Foundations of Computer Science*, pages 57–69. Springer, 1979.

## BIBLIOGRAPHY

- [FSB17] Farhad Farokhi, Iman Shames, and Nathan Batterham. Secure and private control using semi-homomorphic encryption. *Control Engineering Practice*, 67:13–20, 2017.
- [GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 626–645. Springer, 2013.
- [GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM (JACM)*, 62(4):1–64, 2015.
- [Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 321–340. Springer, 2010.
- [GW13] Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 301–320. Springer, 2013.
- [JY14] Chihong Joo and Aaram Yun. Homomorphic authenticated encryption secure against chosen-ciphertext attack. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 173–192. Springer, 2014.

## BIBLIOGRAPHY

- [KF15] Kiminao Kogiso and Takahiro Fujita. Cyber-security enhancement of networked control systems using homomorphic encryption. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 6836–6843. IEEE, 2015.
- [KLS<sup>+</sup>16] Junsoo Kim, Chanhwa Lee, Hyungbo Shim, Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Encrypting controller using fully homomorphic encryption for security of cyber-physical systems. *IFAC-PapersOnLine*, 49(22):175–180, 2016.
- [KSH19] Junsoo Kim, Hyungbo Shim, and Kyoohyung Han. Dynamic controller that operates over homomorphically encrypted data for infinite time horizon. *arXiv preprint arXiv:1912.07362*, 2019.
- [PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE, 2013.
- [Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.
- [TSSJ15] André Teixeira, Iman Shames, Henrik Sandberg, and Karl Henrik Johansson. A secure control framework for resource-limited adversaries. *Automatica*, 51:135–148, 2015.
- [XZZ<sup>+</sup>19] Tiacheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-

## BIBLIOGRAPHY

knowledge proofs with optimal prover computation. In *Annual International Cryptology Conference*, pages 733–764. Springer, 2019.

# Appendix

## 4.1 Proof of Lemma 2

*Proof.* Let  $\mathcal{A}_1$  be the probabilistic polynomial-time adversary of this lemma, i.e., the following probability (which will be denoted by  $\Pr[A_1]$ ) is non-negligible.

$$\Pr \left[ \begin{array}{l} \vec{x} \xleftarrow{\$} \mathbb{Z}_p^n : \\ \wedge (\vec{y} \neq \vec{0}) \wedge (\vec{x} \cdot \vec{y} = 0) \end{array} \quad \begin{array}{l} (\vec{y} \leftarrow \mathcal{A}_1(G_\lambda, g, g^{\vec{x}})) \\ \end{array} \right] =: \Pr[A_1]$$

Now, we construct an adversary  $\mathcal{A}$  who can break the Discrete Logarithm assumption (Assumption 1) using this adversary  $\mathcal{A}_1$ . For given  $(G_\lambda, g, g^x)$  where  $x \xleftarrow{\$} \mathbb{Z}_p$ , the adversary  $\mathcal{A}$  generates  $g^{\vec{x}} := (g^x, g^{x_2}, \dots, g^{x_n})$  where  $x_2, \dots, x_n$  are sampled randomly from  $\mathbb{Z}_p$ . Then he sends  $(G_\lambda, g, g^{\vec{x}})$  to  $\mathcal{A}_1$  who, in response, will output nonzero  $\vec{y} := (y_1, \dots, y_n)$  such that  $\vec{x} \cdot \vec{y} = 0$  with non-negligible probability. Now, since  $\mathcal{A}$  knows  $\{x_i\}_{i=2}^n$  and  $\vec{y}$ , he 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  $x, x_2, \dots, x_n$  are all sampled randomly from  $\mathbb{Z}_p$ . Therefore,

$$\begin{aligned} \Pr[x \xleftarrow{\$} \mathbb{Z}_p : x \leftarrow \mathcal{A}(G_\lambda, g, g^x)] &= \Pr[A_1] \cdot \Pr[y_1 \neq 0 | \vec{y} \neq \vec{0}] \\ &\geq \Pr[A_1]/n \end{aligned}$$

is also non-negligible when  $n = \text{poly}(\lambda)$ , i.e., the Discrete Logarithm assumption does not hold for  $G_\lambda$ .

□

## 4.2 Necessity of Alternative Random Vector

**Theorem 2.** *For (3.3), if  $\vec{r}(t) := \vec{r}_1(t) = \vec{r}_2(t)$ , then prover can forge status without having any knowledge about  $\vec{r}(t)$*

*Proof.* Prover can forge status if prover can send  $(\vec{z}_1(t), \vec{z}_2(t), \vec{z}_3(t))$  instead of  $(\vec{x}(t), \vec{x}(t+1), \vec{x}(t))$  where  $\vec{z}_1(t) \neq \vec{x}(t)$ . The equation check would be written as follow.

$$\begin{cases} g^{\vec{r} \cdot \vec{z}_2(t)} = g^{(\vec{a} - \vec{r}) \cdot \vec{z}_1(t)} * g^{\vec{r} \cdot \vec{x}(t)} * g^{\vec{b} \cdot \vec{y}(t)} * g^{\vec{h}(t) \cdot \vec{u}(t)} \\ g^{\vec{s} \cdot \vec{u}(t)} = g^{(\vec{c} - \vec{r}) \cdot \vec{z}_3(t)} * g^{\vec{r} \cdot \vec{x}(t)} * g^{\vec{d} \cdot \vec{y}(t)} \end{cases} \quad (4.1)$$

Although  $\vec{r}(t)$  is hidden to the controller, rest information,  $(\vec{x}(t), \vec{y}(t), \vec{u}(t))$ , is known to it, and  $\vec{z}_1(t), \vec{z}_2(t)$  are vectors that controller can handle. Therefore, for any  $\vec{z}_1(t) \neq \vec{x}(t)$ , we can define  $\vec{z}_2(t)$  which suffices following equation.

$$\vec{z}_2(t) := (A' - I)\vec{z}_1(t) + \vec{x}(t) + B'\vec{y}(t) \quad (4.2)$$

This  $(\vec{z}_1(t), \vec{z}_2(t), \vec{x}(t))$  satisfies former equation of Eq.4.1, so actuator will verify it as correct data, though it is compromised. And this pair does not require any knowledge about actuator's secret key  $\vec{r}(t)$ .

Therefore there will be a way to compromise data without knowing hidden information, if actuator uses only one secret vector. □

### 4.3 Algorithms: Batch Verification

Let  $G$  be a finite group of order prime  $p$ , and  $\alpha \in \mathbb{F}(= \mathbb{Z}_p)$ . Assume we are given  $B$ -pairs  $\{(g_i, g'_i)\}_{i=1}^B$  of elements of  $G$ , and want to check if all pairs satisfy that  $(g_i)^\alpha = g'_i$  or not. Following algorithm provides an efficient method for this task.

---

**Algorithm 1** Batch Verification – Small exponent test [BGR98]

---

**Input:**  $\alpha \in \mathbb{F}$ ,  $\delta \in \mathbb{Z}_{>0}$ ,  $B$ -pairs  $(g_1, g'_1), \dots, (g_B, g'_B)$  of elements of  $G$  whose order is  $|\mathbb{F}|$ .

**Output:** Verify if  $g'_i = g_i^\alpha, \forall i \in \{1, \dots, B\}$

$s_i \stackrel{\$}{\leftarrow} \delta$ -bit integer,  $i \in \{1, \dots, B\}$

$g \leftarrow \prod_{i=1}^B g_i^{s_i}$

$g' \leftarrow \prod_{i=1}^B g'_i^{s_i}$

**if**  $g' = g^\alpha$  **then return** *acc*

**elsereturn** *rej*

**end if**

---

The Algorithm 1 requires  $3\delta B + 2B$  multiplications over  $G$  on average (see Algorithm 2 in Section 4.4) and 1 exponentiation operation  $((\alpha, g) \mapsto g^\alpha)$  over  $G$ . The cost is much less than a naive check requiring  $B$  exponentiation operation over  $G$ .

We also remark that if there exists at least a single instance  $(g_i, g'_i) \in \{(g_i, g'_i)\}_{i=1}^B$  such that  $(g_i)^\alpha \neq g'_i$ , then the probability that algorithm outputs *acc* is at most  $2^{-\delta}$  (see [BGR98] for the proof).

## 4.4 Algorithms: Multi-exponentiation

Let  $G$  be a finite group. For given  $g \in G$  (or  $\{g_i\}_{i \in I}$ ) and  $e \in \mathbb{Z}_{\geq 0}$  (or  $\{e_i\}_{i \in I}$ ), the following algorithm provide a method to efficiently compute  $g^e$  (or  $\prod_{i \in I} g_i^{e_i}$ , respectively). For  $\kappa$ -bit integer  $n$ , we denote the binary representation of  $n$  as  $(n_{\kappa-1} \dots n_0)_2$  where  $n_{\kappa-1}$  is the most significant bit.

---

**Algorithm 2** Square and multiply algorithm

---

**Input:**  $g \in G$ ,  $\kappa$ -bit exponent  $e$  which is  $(e_{\kappa-1} \dots e_0)_2$  in binary representation.

**Output:**  $g^e$

$R \leftarrow 1$  (an identity element of  $G$ )

$i \leftarrow \kappa - 1$

**while**  $i \geq 0$  **do**

$R \leftarrow R^2$

**if**  $e_i = 1$  **then**

$R = R \cdot g$

**end if**

$i \leftarrow i - 1$

**end while**

**return**  $R$

---

With Algorithm 2, one can compute  $g^e$  for  $\kappa$ -bit exponent  $e$  in  $3\kappa/2$  multiplications on average over  $G$ .

---

**Algorithm 3** Shamir's Multi-exponentiation algorithm

---

**Input:**  $g_0, \dots, g_{h-1} \in G$ ,  $\kappa$ -bit exponents  $e_0, \dots, e_{h-1}$  where  $e_j = (e_{j,\kappa-1} \dots e_{j,0})_2$  in binary representation.

Precomputed values  $G_I := \prod_{j=0}^{h-1} g_j^{I_j}$   
 for each  $I = (I_{h-1} \dots I_0)_2 = \sum_{j=0}^{h-1} 2^j I_j \in [0, 2^h - 1]$

**Output:**  $g_1^{e_1} \dots g_h^{e_h}$

$R \leftarrow 1$  (an identity element of  $G$ )

$i \leftarrow \kappa - 1$

**while**  $i \geq 0$  **do**

$R \leftarrow R^2$

$I \leftarrow (e_{h-1,i} \dots e_{0,i})_2$

$R \leftarrow R \cdot G_I$

$i \leftarrow i - 1$

**end while**

**return**  $R$

---

In Algorithm 3, it requires only  $2\kappa$  multiplications over  $G$  to compute  $g_1^{e_1} \dots g_h^{e_h}$  from  $\kappa$ -bit exponents  $e_i$ 's with  $2^h$  number of precomputed values.

Therefore, it requires

# 국문초록

제어기나 네트워크에 대한 위조는 위험한 상태나 체계의 정지를 야기할 수 있기에, 제어기 및 네트워크에 대한 보안 문제는 네트워크화된 제어체계가 가진 큰 우려라 할 수 있다.

본 학위 논문에서는 이런 악의적인 제어기 문제를 근본적으로 해결하기 위해 검증 가능한 계산을 처음으로 도입한다. 우선, 제어기의 주 연산인 행렬-벡터간 곱셈을 확인하고 제어기의 상태 갱신을 확인하기 위해 인증된 계산을 새로이 제시한다. 이는 플랜트 측으로 하여금 제어기의 계산을 제어기가 행하는 계산량보다 더 적은 계산량으로 확인하게 할 뿐만 아니라, 제어기나 네트워크에 위조가 있는지도 확인 가능하게 해 준다.

또한 제어기 상태의 차원과 액츄에이터의 검증 시 계산량은 독립적이기에, 인증된 계산은 액츄에이터나 제어기에 점근적으로 계산량을 추가하지 않은 채 선형동적 제어계에 도입할 수 있다.

이와 더불어 액츄에이터의 계산량을 줄이기 위해 본 논문에서는 묶음 검증과 다중 곱연산을 도입하였다. 이런 개념들은 제어기의 일정한 계산량을 크게 줄여주어서, 이 체계가 실제에 반영할 수 있을 정도로 성능 예측이 가능해지도록 하였다.

**주요어휘:** 검증가능계산, 이산로그, 동적제어계

**학번:** 2017-23983

## 감사의 글

2009년 3월 학부에 입학한 이후로 10년 동안 학부와 대학원 생활에 도움을 준 모든 분들께 감사를 표합니다. 대학원 입학 전에는 저를 암호학의 길로 이끌어 주시고, 대학원에서는 부족한 저를 진심으로 지도해주신 천정희 교수님께 진심으로 감사드립니다. 바쁘신 와중에도 논문 심사를 위해 선뜻 시간을 내 주신 심형보 교수님과 현동훈 교수님께도 감사드립니다.

또한 늦게 연구실에 들어왔음에도 연구실 생활에 적응할 수 있게 도와준 지승이부터 동기로 들어온 수민이를 비롯해 막내인 형민이와 재현이까지 모든 연구실 구성원 여러분들께 감사드립니다. 랩장을 맡고 있던 지승이와 동형암호에 대해 자세히 알려준 두형이와 용하, 언제나 쾌활하게 격려해주던 동우, 짧게나마 같이 일했던 재윤이, 선배로서 여러 조언을 아끼지 않았던 승완이와 원희, 대학원 동기임에도 먼저 들어와 적응하는 데 도움을 준 기우, 같이 많이 배운 수민이, 새로 들어온 재현이와 형민이까지 모두에게 감사드립니다. 본 논문을 쓸 계기와 방향을 제시한 천정희 교수님을 비롯하여, 논문작성에 도움을 크게 준 동우와 제어 쪽 관련 조언을 해 준 준수와 심형보 교수님께도 무한한 감사의 말을 전합니다. 이분들이 없었다면 이 논문은 빛을 보지 못했을 것입니다.

마지막으로 오랜 시간 동안 저를 믿어주시고 뒤에서 응원하시는 부모님께 감사의 인사를 전합니다. 부모님의 무한한 믿음 덕분에 여기까지 올 수 있었습니다. 감사합니다.