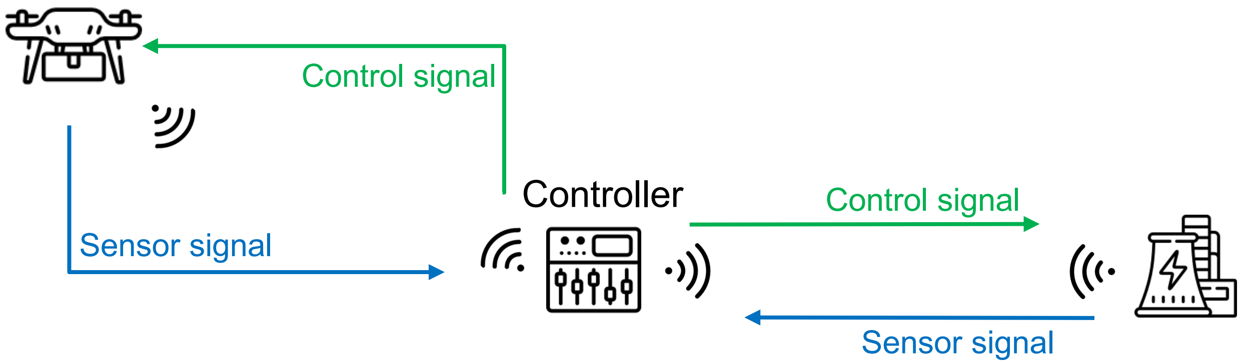
Authenticated Computation of Control Signal from Dynamic Controllers

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

[1)] IMDARC, Department of Mathematical Sciences, Seoul National University, Korea
[2)] ASRI, Department of Electrical and Computer Engineering, Seoul National University, Korea

Security Issues on Networked Control System

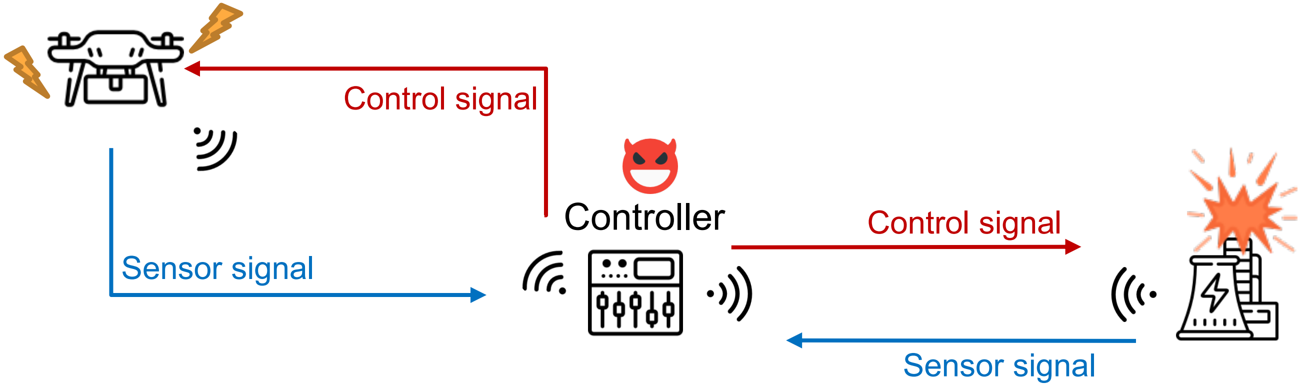
Networked Control System with Remote Controller



Problem: plant-side ( or ) can **not** always trust control signals.

Security Issues on Networked Control System

Networked Control System with Remote Controller



Problem: plant-side ( or ) can not always trust control signals.

Compromise on the signal / controller ⇒ Misbehavior / Failure of the system

Proposed Solution - Authentication of Control Signals

Proposed solution: Let the plant-side verify the control signal,

i.e., the computation of controller!

How?

Naive Sol: Re-executing the controller computation (☹ burden on the plant-side)

Proposed Solution - Authentication of Control Signals

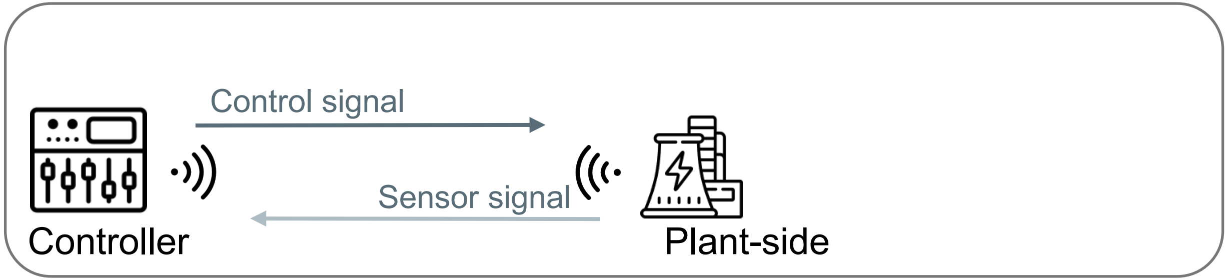
Proposed solution: Let the plant-side verify the control signal,

i.e., the computation of controller!

How?

Naive Sol: Re-executing the controller computation (☹ burden on the plant-side)

Proposed Sol: Adapt the Verifiable Computation (VC) from complexity theory and cryptography



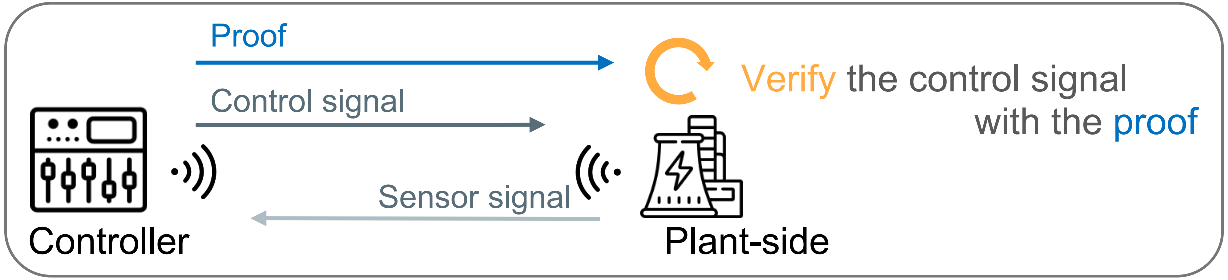
Proposed Solution - Authentication of Control Signals

Proposed solution: Let the plant-side verify the control signal,
i.e., the computation of controller!

How?

Naive Sol: Re-executing the controller computation (☹ burden on the plant-side)

Proposed Sol: Adapt the Verifiable Computation (VC) from complexity theory and cryptography



- Controller provides a proof that its computation is correct

Proposed Solution - Authentication of Control Signals

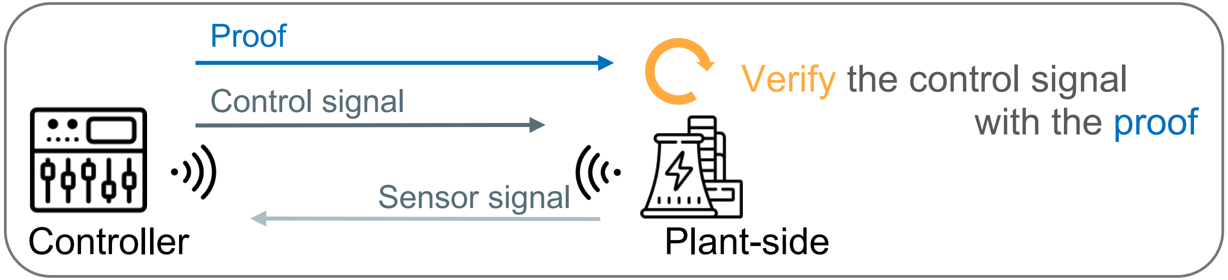
Proposed solution: Let the plant-side verify the control signal,

i.e., the computation of controller!

How?

Naive Sol: Re-executing the controller computation (🙁 burden on the plant-side)

Proposed Sol: Adapt the Verifiable Computation (VC) from complexity theory and cryptography



- Controller provides a proof that its computation is correct
- 😁: Faster verification than re-execution
- 🙁: Overhead on the controller (for generation of the proof)

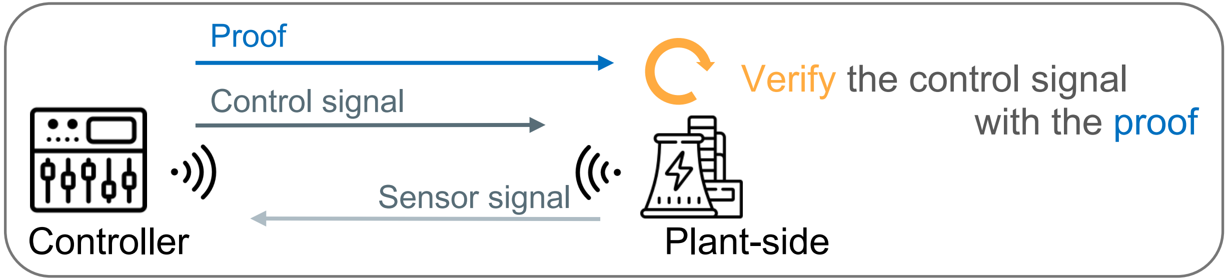
Proposed Solution - Authentication of Control Signals

Proposed solution: Let the plant-side verify the control signal,
i.e., the computation of controller!

How?

Naive Sol: Re-executing the controller computation (☹ burden on the plant-side)

Proposed Sol: Adapt the Verifiable Computation (VC) from complexity theory and cryptography



- Controller provides a proof that its computation is correct
- 😁: Faster verification than re-execution
- ☹: Overhead on the controller (for generation of the proof)

Goal: optimized VC for Controller Computation

The Target - Linear Dynamic System (with Integers)

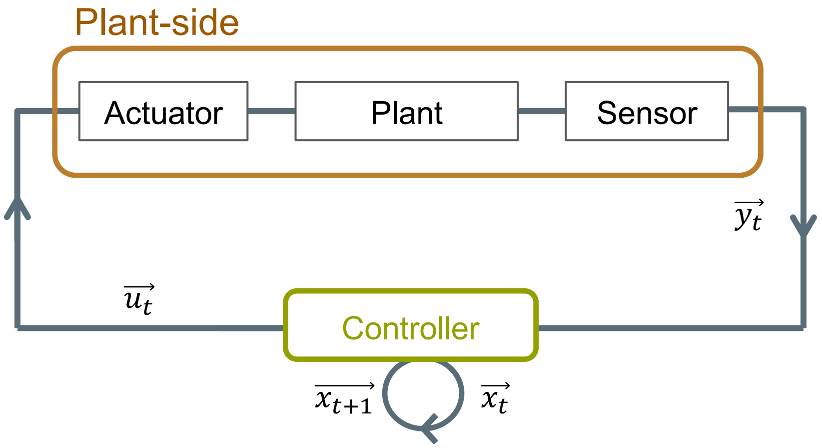
Consider a Linear Dynamic System with Discrete-time Controller

- Controller's computation

$$\begin{pmatrix} \overrightarrow{x_{t+1}} \\ \overrightarrow{u_t} \end{pmatrix} = \begin{pmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{pmatrix} \begin{pmatrix} \overrightarrow{x_t} \\ \overrightarrow{y_t} \end{pmatrix}$$

$\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}$: matrices over $\mathbb{R}$,
$\vec{x}, \vec{u}, \vec{y}$: vectors over $\mathbb{R}$,

$\overrightarrow{x_t}$: state of the controller at time t,
$\overrightarrow{u_t}$: controller signal,
$\overrightarrow{y_t}$: sensor signal.



The Target - Linear Dynamic System (with Integers)

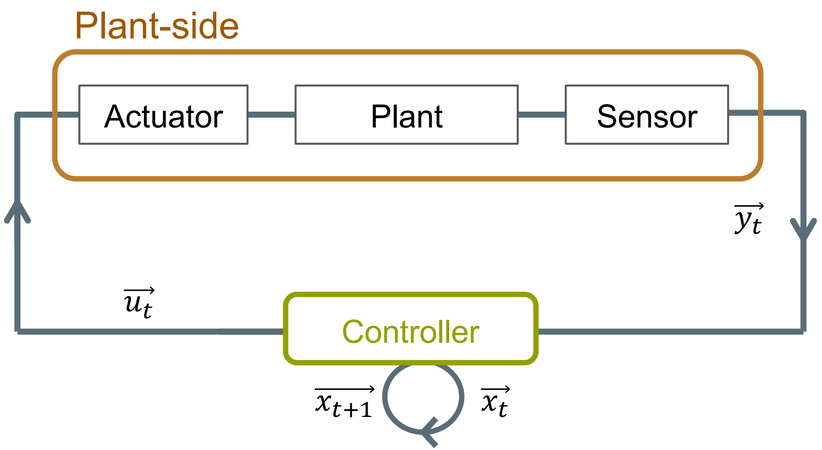
Consider a Linear Dynamic System with Discrete-time Controller

- Controller's computation

$$\begin{pmatrix} \overrightarrow{x_{t+1}} \\ \overrightarrow{u_t} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} \overrightarrow{x_t} \\ \overrightarrow{y_t} \end{pmatrix}$$

A, B, C, D: matrices over $\mathbb{R}$,
$\vec{x}, \vec{u}, \vec{y}$: vectors over $\mathbb{R}$,

$\overrightarrow{x_t}$: state of the controller at time t,
$\overrightarrow{u_t}$: controller signal,
$\overrightarrow{y_t}$: sensor signal.



* We can assume that all values are over $\mathbb{Z}$ with certain bound, i.e., over $\mathbb{Z}_p \coloneqq \mathbb{Z}/p\mathbb{Z}$
(Integers modulo a prime p)

Conversion of B, C, and D into integer matrices: done by scaling & truncation.
Conversion of A into integer matrix w/o scaling & truncation is needed and is possible.
Details are presented in the session (FrA09.3).

Preliminaries

Verifiable Computation & Cryptography

What is Verifiable Computation (VC)?

Goal: Verify the result of delegated computation (F).

Algorithms (λ: security parameter):

- **KeyGen** $(F, \lambda) \rightarrow EK_F$ & VK_F ;

  generate Evaluation Key & Verification Key for F
- **Compute & Proof Gen** $(EK_F, x) \rightarrow (y, \pi_y)$;

  compute $y = F(x)$ and a proof π_y
- **Verify** $(VK_F, x, y, \pi_y) \rightarrow \{accept, reject\}$

Requirements:

(*Soundness*) With $y \neq F(x)$, an adversary can not forge a proof π_y s.t. **Verify** $(VK_F, x, y, \pi_y) = accept$

(*Efficiency*) The function **Verify** should be faster than computing $y = F(x)$

What is Verifiable Computation (VC)?

Goal: Verify the result of delegated computation (F).

Algorithms (λ: security parameter):

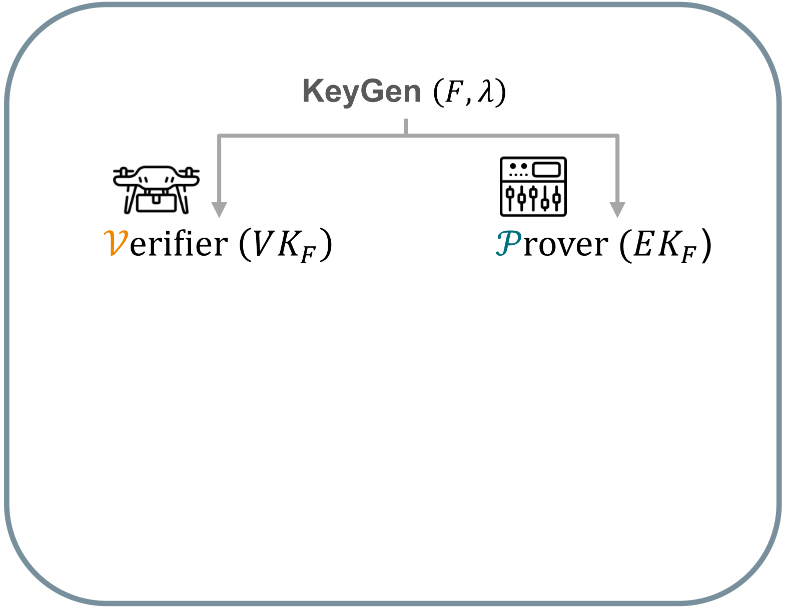
- **KeyGen** $(F, \lambda) \rightarrow EK_F$ & VK_F ;

  generate Evaluation Key & Verification Key for F

- **Compute & Proof Gen** $(EK_F, x) \rightarrow (y, \pi_y)$;

  compute $y = F(x)$ and a proof π_y

- **Verify** $(VK_F, x, y, \pi_y) \rightarrow \{accept, reject\}$



Requirements:

(*Soundness*) With $y \neq F(x)$, an adversary can not forge a proof π_y s.t. **Verify** $(VK_F, x, y, \pi_y) = accept$

(*Efficiency*) The function **Verify** should be faster than computing $y = F(x)$

What is Verifiable Computation (VC)?

Goal: Verify the result of delegated computation (F).

Algorithms (λ: security parameter):

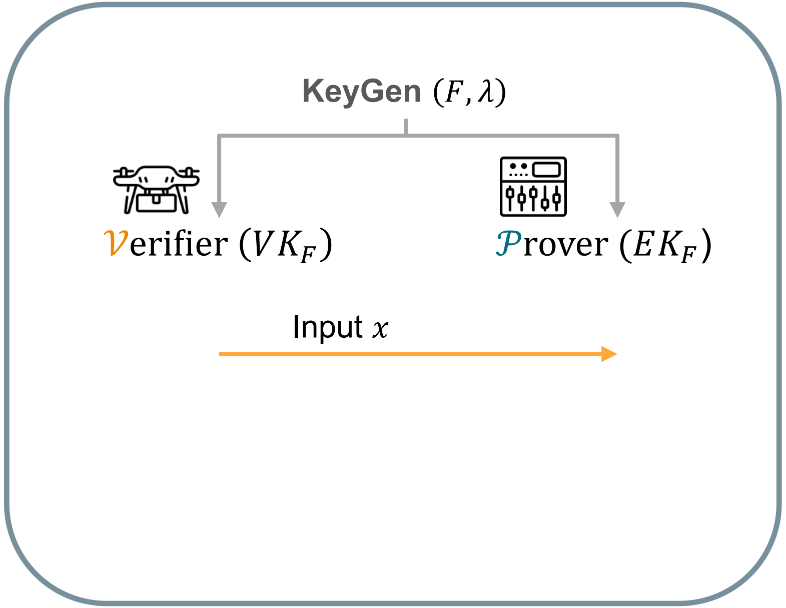
- **KeyGen** $(F, \lambda) \rightarrow EK_F \, \& \, VK_F$;

  generate Evaluation Key & Verification Key for F

- **Compute & Proof Gen** $(EK_F, x) \rightarrow (y, \pi_y)$;

  compute $y = F(x)$ and a proof π_y

- **Verify** $(VK_F, x, y, \pi_y) \rightarrow \{accept, reject\}$



Requirements:

(*Soundness*) With $y \neq F(x)$, an adversary can not forge a proof π_y s.t. **Verify** $\left(VK_F, x, y, \pi_y\right) = accept$

(*Efficiency*) The function **Verify** should be faster than computing $y = F(x)$

What is Verifiable Computation (VC)?

Goal: Verify the result of delegated computation (F).

Algorithms (λ: security parameter):

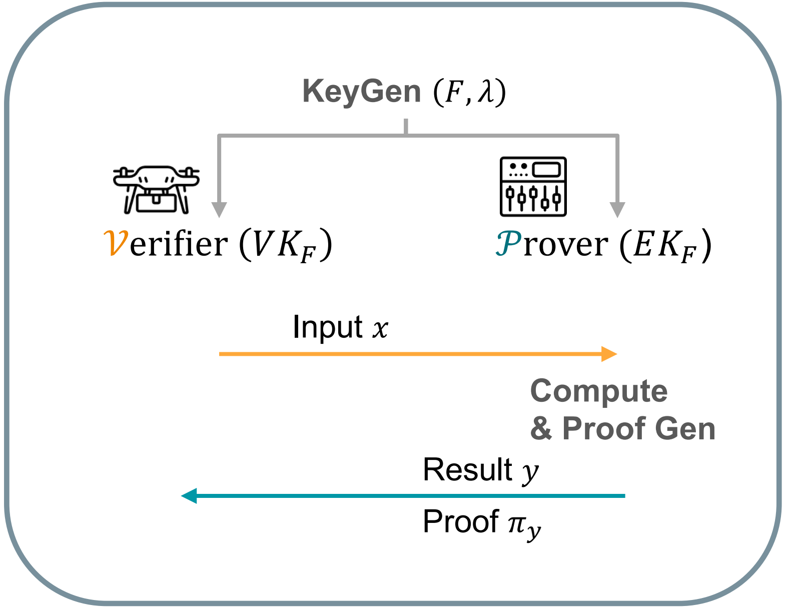
- **KeyGen** $(F, \lambda) \rightarrow EK_F \,\&\, VK_F$;

  generate Evaluation Key & Verification Key for F

- **Compute & Proof Gen** $(EK_F, x) \rightarrow (y, \pi_y)$;

  compute $y = F(x)$ and a proof π_y

- **Verify** $(VK_F, x, y, \pi_y) \rightarrow \{accept, reject\}$



Requirements:

(*Soundness*) With $y \neq F(x)$, an adversary can not forge a proof π_y s.t. **Verify** $(VK_F, x, y, \pi_y) = accept$

(*Efficiency*) The function **Verify** should be faster than computing $y = F(x)$

What is Verifiable Computation (VC)?

Goal: Verify the result of delegated computation (F).

Algorithms (λ: security parameter):

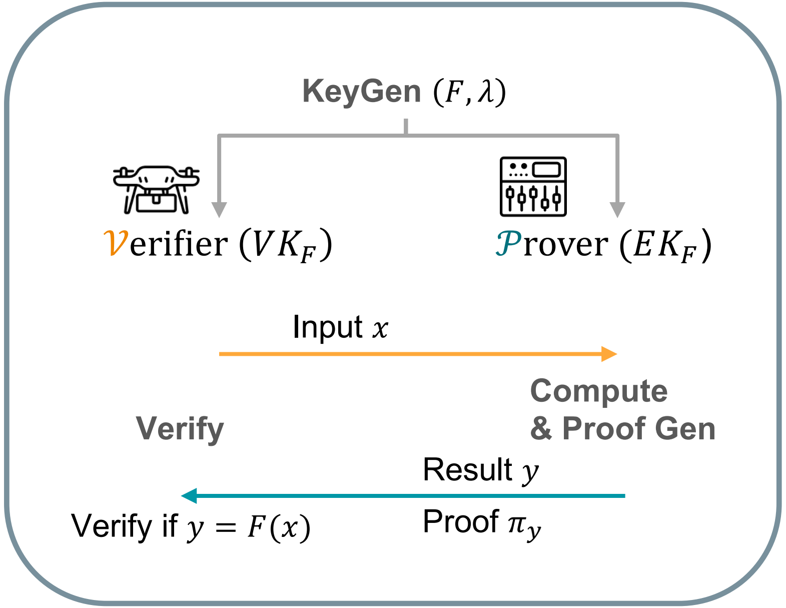
- **KeyGen** $(F, \lambda) \rightarrow EK_F \,\&\, VK_F$;

  generate Evaluation Key & Verification Key for F

- **Compute & Proof Gen** $(EK_F, x) \rightarrow (y, \pi_y)$;

  compute $y = F(x)$ and a proof π_y

- **Verify** $(VK_F, x, y, \pi_y) \rightarrow \{accept, reject\}$



Requirements:

(*Soundness*) With $y \neq F(x)$, an adversary can not forge a proof π_y s.t. **Verify** $(VK_F, x, y, \pi_y) = accept$

(*Efficiency*) The function **Verify** should be faster than computing $y = F(x)$

Freivalds' algorithm: a VC for Matrix Multiplication

Goal: Verify a matrix-vector multiplication ($\vec{x} \in \mathbb{Z}_p^m \to F\vec{x}$) for a matrix $F \in \mathbb{Z}_p^{n \times m}$

0. $\mathcal{V}$erifier prepares $\vec{r} \in \mathbb{Z}_p^n$ and $\vec{f} \coloneqq \vec{r}^{\mathrm{T}} F$

$\mathcal{V}$erifier $(\vec{r}, \vec{f})$

$\mathcal{P}$rover (F)

1. Input $\vec{x}$

2. Compute $\vec{y} = F\vec{x}$

2. Result $\vec{y}'$

3. Check if $\vec{r} \cdot \vec{y}' = \vec{f} \cdot \vec{x}$

Freivalds' algorithm: a VC for Matrix Multiplication

Goal: Verify a matrix-vector multiplication ($\vec{x} \in \mathbb{Z}_p^m \to F\vec{x}$) for a matrix $F \in \mathbb{Z}_p^{n \times m}$

0. $\mathcal{V}$erifier prepares $\vec{r} \in \mathbb{Z}_p^n$ and $\vec{f} \coloneqq \vec{r}^{\mathrm{T}} F$

$\mathcal{V}$erifier $(\vec{r}, \vec{f})$ | $\mathcal{P}$rover (F)

1. Input $\vec{x}$ →

2. Compute $\vec{y} = F\vec{x}$

← 2. Result $\vec{y}'$

3. Check if $\vec{r} \cdot \vec{y}' = \vec{f} \cdot \vec{x}$

* $\vec{r}$ and $\vec{f}$ must be hidden from $\mathcal{P}$!

Freivalds' algorithm satisfies:

(*Soundness*) If $\vec{y}' \neq F\vec{x}$, the check gives that $\vec{r} \cdot \vec{y}' \neq \vec{f} \cdot \vec{x}$ *with high probability* $\left(1 - \frac{n}{p}\right)$

$\because$ for nonzero $\vec{v} \in \mathbb{Z}_p^n$, $\vec{r} \cdot \vec{v} = 0$ *with probability* $\frac{n}{p}$

(*Efficiency*) Checking if $\vec{r} \cdot \vec{y}' = \vec{f} \cdot \vec{x}$ takes $2n$ mults.

$\ll$

Computing $\vec{y} = F\vec{x}$ takes nm mults.

A quick introduction to

Finite Group and Cryptographic Assumptions

Assume, with λ (security parameter), computation resource of an adversary $\mathcal{A}$ is limited by 2^{λ} operations.

- Consider a Finite Group $G = (\{g^i\}_{i \in \mathbb{Z}_p}, \cdot)$ of order $p \geq 2^{\lambda}$, then
  - for $x, y \in \mathbb{Z}_p$, $g^x \cdot g^y = g^{(x+y \bmod p)}$
  - e.g.) $G = \{2^i\}_{i \in \mathbb{Z}_3} \subset \mathbb{Z}_7$ with $\cdot$ as the multiplication in $\mathbb{Z}_7$
- Discrete Logarithm assumption (DL)

A quick introduction to

Finite Group and Cryptographic Assumptions

Assume, with λ (security parameter), computation resource of an adversary $\mathcal{A}$ is limited by 2^{λ} operations.

- Consider a Finite Group $G = (\{g^i\}_{i \in \mathbb{Z}_p}, \cdot\ )$ of order $p \geq 2^{\lambda}$, then
  - for $x, y \in \mathbb{Z}_p$, $g^x \cdot g^y = g^{(x+y \bmod p)}$
  - e.g.) $G = \{2^i\}_{i \in \mathbb{Z}_3} \subset \mathbb{Z}_7$ with $\cdot$ as the multiplication in $\mathbb{Z}_7$

- Discrete Logarithm assumption (DL)
  - Given g and g^x $(x \leftarrow \mathbb{Z}_p)$, $\mathcal{A}$ can not retrieve x
  - variant: Given g and $g^{\vec{x}} \coloneqq (g^{x_1}, g^{x_2}, \ldots, g^{x_n})$ $(\vec{x} \leftarrow \mathbb{Z}_p^n)$, $\mathcal{A}$ can not retrieve $\vec{y} \neq \vec{0}$ s.t. $\vec{x} \cdot \vec{y} = 0$

Proposed VC for Controller Computation

Design Rationale

VC for Controller Computation – First idea

Goal: Verify the Controller's Computation, i.e., $\begin{pmatrix} \overrightarrow{x_{t+1}} \\ \overrightarrow{u_t} \end{pmatrix} = \begin{pmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{pmatrix} \begin{pmatrix} \overrightarrow{x_t} \\ \overrightarrow{y_t} \end{pmatrix}$

I. Apply Freivalds' Algorithm (Matrix-vector mults → Inner-product of vectors)

VC for Controller Computation – First idea

Goal: Verify the Controller's Computation, i.e., $\begin{pmatrix} \overrightarrow{x_{t+1}} \\ \overrightarrow{u_t} \end{pmatrix} = \begin{pmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{pmatrix} \begin{pmatrix} \overrightarrow{x_t} \\ \overrightarrow{y_t} \end{pmatrix}$

I. Apply Freivalds' Algorithm (Matrix-vector mults → Inner-product of vectors)

- $\mathcal{V}$erifier (plant-side) prepares $\vec{r}, \vec{s}$ and $\vec{a} \coloneqq \vec{r}^T A,\ \vec{b} \coloneqq \vec{r}^T B,$
  $\vec{c} \coloneqq \vec{s}^T C,\ \vec{d} \coloneqq \vec{s}^T D,$
- $\mathcal{P}$rover (controller) computes and sends $\overrightarrow{x_{t+1}}'$ and $\overrightarrow{u_t}'$ to $\mathcal{V}$
- $\mathcal{V}$ checks if

$$\vec{r} \cdot \overrightarrow{x_{t+1}}' = \vec{a} \cdot \overrightarrow{x_t} + \vec{b} \cdot \overrightarrow{y_t}$$
$$\vec{s} \cdot \overrightarrow{u_t}' = \vec{c} \cdot \overrightarrow{x_t} + \vec{d} \cdot \overrightarrow{y_t}$$

(Freivalds' Algorithm) ⇒ If signals are forged, i.e., $\overrightarrow{x_{t+1}}' \neq \overrightarrow{x_{t+1}}$ or $\overrightarrow{u_t}' \neq \overrightarrow{u_t}$,

then (with high probability),
two equations do not hold!

* $\vec{r}, \vec{s}, \vec{a}, \vec{b}, \vec{c}, \vec{d}$ must be hidden from $\mathcal{P}$ (or $\mathcal{A}$)

VC for Controller Computation – First idea

Goal: Verify the Controller's Computation, i.e., $\begin{pmatrix} \overrightarrow{x_{t+1}} \\ \overrightarrow{u_t} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} \overrightarrow{x_t} \\ \overrightarrow{y_t} \end{pmatrix}$

I. Apply Freivalds' Algorithm (Matrix-vector mults → Inner-product of vectors)

- $\mathcal{V}$erifier (plant-side) prepares $\vec{r}, \vec{s}$ and $\vec{a} \coloneqq \vec{r}^T A,\ \vec{b} \coloneqq \vec{r}^T B,$
  $\vec{c} \coloneqq \vec{s}^T C,\ \vec{d} \coloneqq \vec{s}^T D,$
- $\mathcal{P}$rover (controller) computes and sends $\overrightarrow{x_{t+1}}'$ and $\overrightarrow{u_t}'$ to $\mathcal{V}$
- $\mathcal{V}$ checks if

$$\vec{r} \cdot \overrightarrow{x_{t+1}}' = \vec{a} \cdot \overrightarrow{x_t} + \vec{b} \cdot \overrightarrow{y_t}$$
$$\vec{s} \cdot \overrightarrow{u_t}' = \vec{c} \cdot \overrightarrow{x_t} + \vec{d} \cdot \overrightarrow{y_t}$$

(Freivalds' Algorithm)

If signals are forged,
i.e., $\overrightarrow{x_{t+1}}' \neq \overrightarrow{x_{t+1}}$ or $\overrightarrow{u_t}' \neq \overrightarrow{u_t}$
,
then (with high probability),
two equations do not hold!

* $\vec{r}, \vec{s}, \vec{a}, \vec{b}, \vec{c}, \vec{d}$ must be hidden from $\mathcal{P}$ (or $\mathcal{A}$)

Problem! The state $\overrightarrow{x_t}$ must be transferred from the controller to the plant-side at each time t.

VC for Controller Computation – Second idea

Goal: Verify the Computation $\vec{r} \cdot \overrightarrow{x_{t+1}} = \vec{a} \cdot \overrightarrow{x_t} + \vec{b} \cdot \overrightarrow{y_t}$ without $\overrightarrow{x_{t+1}}$ nor $\overrightarrow{x_t}$

$\vec{s} \cdot \overrightarrow{u_t} = \vec{c} \cdot \overrightarrow{x_t} + \vec{d} \cdot \overrightarrow{y_t}$

II. Let the controller compute the inner-product (but controller must not know $\vec{r}, \vec{a}, \vec{c}$)

by applying Group-based Cryptography ($\mathbb{G} = \langle g \rangle$ of order $\boldsymbol{p}$, $x \rightarrow g^x$)

VC for Controller Computation – Second idea

Goal: Verify the Computation $\vec{r} \cdot \overrightarrow{x_{t+1}} = \vec{a} \cdot \overrightarrow{x_t} + \vec{b} \cdot \overrightarrow{y_t}$ without $\overrightarrow{x_{t+1}}$ nor $\overrightarrow{x_t}$

$\vec{s} \cdot \overrightarrow{u_t} = \vec{c} \cdot \overrightarrow{x_t} + \vec{d} \cdot \overrightarrow{y_t}$

II. Let the controller compute the inner-product (but controller must not know $\vec{r}, \vec{a}, \vec{c}$)

by applying Group-based Cryptography ($\mathbb{G} = \langle g \rangle$ of order $\boldsymbol{p}$, $x \to g^x$)

- $\mathcal{V}$erifier (plant-side) prepares $g^{\vec{r}}, g^{\vec{a}}$, and $g^{\vec{c}}$, sends them to $\mathcal{P}$rover (controller).

* $g^{\vec{r}}, g^{\vec{a}}, g^{\vec{c}}$ hide $\vec{r}, \vec{a}, \vec{c}$ from $\mathcal{P}$ (or $\mathcal{A}$) by DL-assumption.

- $\mathcal{P}$ computes and sends $g_1 := g^{\vec{r} \cdot \overrightarrow{x_{t+1}}}$, $g_2 := g^{\vec{a} \cdot \overrightarrow{x_t}}$, $g_3 := g^{\vec{c} \cdot \overrightarrow{x_t}}$, and $\overrightarrow{u_t}'$ to $\mathcal{V}$.

- $\mathcal{V}$ checks if

$$g_1 = g_2 \cdot g^{\vec{b} \cdot \overrightarrow{y_t}}$$
$$g^{\vec{s} \cdot \overrightarrow{u_t}'} = g_3 \cdot g^{\vec{d} \cdot \overrightarrow{y_t}}$$

VC for Controller Computation – Second idea

Goal: Verify the Computation $\boxed{\vec{r} \cdot \overrightarrow{x_{t+1}} = \vec{a} \cdot \overrightarrow{x_t}} + \vec{b} \cdot \overrightarrow{y_t}$ without $\overrightarrow{x_{t+1}}$ nor $\overrightarrow{x_t}$

$$\vec{s} \cdot \overrightarrow{u_t} = \boxed{\vec{c} \cdot \overrightarrow{x_t}} + \vec{d} \cdot \overrightarrow{y_t}$$

II. Let the controller compute the inner-product (but controller must not know $\vec{r}, \vec{a}, \vec{c}$)

by applying Group-based Cryptography ($\mathbb{G} = \langle g \rangle$ of order $\boldsymbol{p}$, $x \to g^x$)

- $\mathcal{V}$erifier (plant-side) prepares $g^{\vec{r}}, g^{\vec{a}},$ and $g^{\vec{c}}$, sends them to $\mathcal{P}$rover (controller).

* $g^{\vec{r}}, g^{\vec{a}}, g^{\vec{c}}$ hide $\vec{r}, \vec{a}, \vec{c}$ from $\mathcal{P}$ (or $\mathcal{A}$) by DL-assumption.

- $\mathcal{P}$ computes and sends $g_1 := g^{\vec{r} \cdot \overrightarrow{x_{t+1}}}$, $g_2 := g^{\vec{a} \cdot \overrightarrow{x_t}}$, $g_3 := g^{\vec{c} \cdot \overrightarrow{x_t}}$, and $\overrightarrow{u_t}'$ to $\mathcal{V}$.

- $\mathcal{V}$ checks if

$$g_1 = g_2 \cdot g^{\vec{b} \cdot \overrightarrow{y_t}}$$
$$g^{\vec{s} \cdot \overrightarrow{u_t}'} = g_3 \cdot g^{\vec{d} \cdot \overrightarrow{y_t}}$$

$(g^x \cdot g^y = g^{x+y \bmod \boldsymbol{p}})$ $\Longrightarrow$ It is equivalent to the computation given that g_1, g_2, g_3 are as above

VC for Controller Computation – Second idea

Goal: Verify the Computation $\boxed{\vec{r} \cdot \overrightarrow{x_{t+1}} = \vec{a} \cdot \overrightarrow{x_t}} + \vec{b} \cdot \overrightarrow{y_t}$ without $\overrightarrow{x_{t+1}}$ nor $\overrightarrow{x_t}$

$\vec{s} \cdot \overrightarrow{u_t} = \boxed{\vec{c} \cdot \overrightarrow{x_t}} + \vec{d} \cdot \overrightarrow{y_t}$

II. Let the controller compute the inner-product (but controller must not know $\vec{r}, \vec{a}, \vec{c}$)

by applying Group-based Cryptography ($\mathbb{G} = \langle g \rangle$ of order $\boldsymbol{p}$, $x \to g^x$)

- $\mathcal{V}$erifier (plant-side) prepares $g^{\vec{r}}, g^{\vec{a}}$, and $g^{\vec{c}}$, sends them to $\mathcal{P}$rover (controller).

* $g^{\vec{r}}, g^{\vec{a}}, g^{\vec{c}}$ hide $\vec{r}, \vec{a}, \vec{c}$ from $\mathcal{P}$ (or $\mathcal{A}$) by DL-assumption.

- $\mathcal{P}$ computes and sends $g_1 := g^{\vec{r} \cdot \overrightarrow{x_{t+1}}}$, $g_2 := g^{\vec{a} \cdot \overrightarrow{x_t}}$, $g_3 := g^{\vec{c} \cdot \overrightarrow{x_t}}$, and $\overrightarrow{u_t}'$ to $\mathcal{V}$.

- $\mathcal{V}$ checks if

$$g_1 = g_2 \cdot g^{\vec{b} \cdot \overrightarrow{y_t}}$$
$$g^{\vec{s} \cdot \overrightarrow{u_t}'} = g_3 \cdot g^{\vec{d} \cdot \overrightarrow{y_t}}$$

$(g^x \cdot g^y = g^{x+y \bmod \boldsymbol{p}})$ $\Longrightarrow$ It is equivalent to the computation given that g_1, g_2, g_3 are as above

Problem! $\mathcal{P}$ (or $\mathcal{A}$) can pass the check (by $\mathcal{V}$) with $\overrightarrow{u_t}' \neq \overrightarrow{u_t}$ by sending different g_1, g_2, g_3 from what was asked.

VC for Controller Computation – Third idea

Goal: Enforce $\mathcal{P}$ (or $\mathcal{A}$) to send $g_1 := g^{\vec{r} \cdot \overrightarrow{x_{t+1}}}$, $g_2 := g^{\vec{a} \cdot \overrightarrow{x_t}}$, and $g_3 := g^{\vec{c} \cdot \overrightarrow{x_t}}$

III. Use the Cryptographic Assumption (n-PKE assumption)

- n-Power Knowledge of Exponent assumption (n-PKE)
  - Given $g, g^{\vec{s}}$, and $g^{\alpha \vec{s}}$, if $\mathcal{A}$ outputs g_1 and g_2 s.t. $g_1^{\alpha} = g_2$,

    the only way is to generate $g_1 = g^{\vec{s} \cdot \vec{z}}$ for $\vec{z}$ it knows.

VC for Controller Computation – Third idea

Goal: Enforce $\mathcal{P}$ (or $\mathcal{A}$) to send $g_1 := g^{\vec{r}\cdot\overrightarrow{x_{t+1}}}$, $g_2 := g^{\vec{a}\cdot\overrightarrow{x_t}}$, and $g_3 := g^{\vec{c}\cdot\overrightarrow{x_t}}$

III. Use the Cryptographic Assumption (n-PKE assumption)

- $\mathcal{V}$erifier (plant-side) sends $g^{\vec{r}}, g^{\vec{a}}, g^{\vec{c}}$, and $g^{\rho\vec{r}}, g^{\alpha\vec{a}}, g^{\gamma\vec{c}}$, and $g^{\vec{a}-\vec{c}}, g^{\delta(\vec{a}-\vec{c})}$ to $\mathcal{P}$rover (controller)
- $\mathcal{P}$ computes and sends

$$
\begin{aligned}
&g_1 := g^{\vec{r}\cdot\overrightarrow{x_{t+1}}}, \quad && g_1' := g^{\rho\vec{r}\cdot\overrightarrow{x_{t+1}}} \\
&g_2 := g^{\vec{a}\cdot\overrightarrow{x_t}}, \quad && g_2' := g^{\alpha\vec{a}\cdot\overrightarrow{x_t}} \\
&g_3 := g^{\vec{c}\cdot\overrightarrow{x_t}}, \quad && g_3' := g^{\gamma\vec{c}\cdot\overrightarrow{x_t}}, \quad g_{2-3} := g^{(\vec{a}-\vec{c})\cdot\overrightarrow{x_t}}, \quad g_{2-3}' := g^{\delta(\vec{a}-\vec{c})\cdot\overrightarrow{x_t}}
\end{aligned}
$$

- $\mathcal{V}$ checks if

$$g_1^{\rho} = g_1',\ g_2^{\alpha} = g_2',\ g_3^{\gamma} = g_3',\ g_{2-3}^{\delta} = g_{2-3}',\ \text{and}\ g_2 = g_3 \cdot g_{2-3}$$

(n-PKE assumption) guarantees:

$g_1 = g^{\vec{r}\cdot\overrightarrow{z_1}},\ g_2 = g^{\vec{a}\cdot\overrightarrow{z_2}},\ g_3 = g^{\vec{c}\cdot\overrightarrow{z_3}},\ g_{2-3}^{(\vec{a}-\vec{c})\cdot\overrightarrow{z_4}}$ for some $\overrightarrow{z_1}, \overrightarrow{z_2}, \overrightarrow{z_3}, \overrightarrow{z_4}$.

($g_2 = g_3 \cdot g_{2-3}$) guarantees: $\overrightarrow{z_2} = \overrightarrow{z_3} = \overrightarrow{z_4}$

VC for Controller Computation – Third idea

Goal: Enforce $\mathcal{P}$ (or $\mathcal{A}$) to send $g_1 := g^{\vec{r}\cdot\overrightarrow{x_{t+1}}}$, $g_2 := g^{\vec{a}\cdot\overrightarrow{x_t}}$, and $g_3 := g^{\vec{c}\cdot\overrightarrow{x_t}}$

III. Use the Cryptographic Assumption (n-PKE assumption)

- $\mathcal{V}$erifier (plant-side) sends $g^{\vec{r}}, g^{\vec{a}}, g^{\vec{c}}$, and $g^{\rho\vec{r}}, g^{\alpha\vec{a}}, g^{\gamma\vec{c}}$, and $g^{\vec{a}-\vec{c}}, g^{\delta(\vec{a}-\vec{c})}$ to $\mathcal{P}$rover (controller)

- $\mathcal{P}$ computes and sends

$$g_1 := g^{\vec{r}\cdot\overrightarrow{x_{t+1}}}, \quad g_1' := g^{\rho\vec{r}\cdot\overrightarrow{x_{t+1}}}$$
$$g_2 := g^{\vec{a}\cdot\overrightarrow{x_t}}, \quad g_2' := g^{\alpha\vec{a}\cdot\overrightarrow{x_t}}$$
$$g_3 := g^{\vec{c}\cdot\overrightarrow{x_t}}, \quad g_3' := g^{\gamma\vec{c}\cdot\overrightarrow{x_t}}, \quad g_{2-3} := g^{(\vec{a}-\vec{c})\cdot\overrightarrow{x_t}}, \quad g_{2-3}' := g^{\delta(\vec{a}-\vec{c})\cdot\overrightarrow{x_t}}$$

- $\mathcal{V}$ checks if

$$g_1^{\rho} = g_1', \; g_2^{\alpha} = g_2', \; g_3^{\gamma} = g_3', \; g_{2-3}^{\delta} = g_{2-3}', \text{ and } g_2 = g_3 \cdot g_{2-3}$$

(n-PKE assumption) guarantees:

$$g_1 = g^{\vec{r}\cdot\overrightarrow{z_1}}, \; g_2 = g^{\vec{a}\cdot\overrightarrow{z_2}}, \; g_3 = g^{\vec{c}\cdot\overrightarrow{z_3}}, \; g_{2-3}^{(\vec{a}-\vec{c})\cdot\overrightarrow{z_4}} \text{ for some } \overrightarrow{z_1}, \overrightarrow{z_2}, \overrightarrow{z_3}, \overrightarrow{z_4}.$$

($g_2 = g_3 \cdot g_{2-3}$) guarantees: $\overrightarrow{z_2} = \overrightarrow{z_3} = \overrightarrow{z_4}$

Problem…?

$\overrightarrow{z_1} = \overrightarrow{x_{t+1}}$? and $\overrightarrow{z_2} = \overrightarrow{x_t}$?

They should obey system dynamics.

VC for Controller Computation – Fourth idea

Claim: Enforcing $\mathcal{P}$ (or $\mathcal{A}$) to send $g_1 = g^{\vec{r} \cdot \overrightarrow{z_1}},\ g_2 = g^{\vec{a} \cdot \overrightarrow{z_2}}, g_3 = g^{\vec{c} \cdot \overrightarrow{z_2}}$ is sufficient for our purpose!

VC for Controller Computation – Fourth idea

Claim: Enforcing $\mathcal{P}$ (or $\mathcal{A}$) to send $g_1 = g^{\vec{r}\cdot\overrightarrow{z_1}},\ g_2 = g^{\vec{a}\cdot\overrightarrow{z_2}}, g_3 = g^{\vec{c}\cdot\overrightarrow{z_2}}$ is sufficient for our purpose!

If $\mathcal{A}$ can find $\overrightarrow{z_1}$, $\overrightarrow{z_2}$ and $\overrightarrow{u_t}'$ such that

$$\vec{r}\cdot\overrightarrow{z_1} = \vec{a}\cdot\overrightarrow{z_2} + \vec{b}\cdot\overrightarrow{y_t}$$
$$\vec{s}\cdot\overrightarrow{u_t}' = \vec{c}\cdot\overrightarrow{z_2} + \vec{d}\cdot\overrightarrow{y_t}$$

note: $\mathcal{A}$ knows that

$$\vec{r}\cdot\overrightarrow{x_{t+1}} = \vec{a}\cdot\overrightarrow{x_t} + \vec{b}\cdot\overrightarrow{y_t}$$
$$\vec{s}\cdot\overrightarrow{u_t} = \vec{c}\cdot\overrightarrow{x_t} + \vec{d}\cdot\overrightarrow{y_t}$$

Then, by subtracting above equations (noting that $\vec{a} \coloneqq \vec{r}^T A,\ \ \vec{c} \coloneqq \vec{s}^T C$),
$\mathcal{A}$ gets

$$\vec{r}\cdot\left((\overrightarrow{z_1} - \overrightarrow{x_{t+1}}) - A(\overrightarrow{z_2} - \overrightarrow{x_t})\right) = 0$$
$$\vec{s}\cdot\left((\overrightarrow{u_t}' - \overrightarrow{u_t}) - C(\overrightarrow{z_2} - \overrightarrow{x_t})\right) = 0$$

VC for Controller Computation – Fourth idea

Claim: Enforcing $\mathcal{P}$ (or $\mathcal{A}$) to send $g_1 = g^{\vec{r}\cdot\overrightarrow{z_1}},\ g_2 = g^{\vec{a}\cdot\overrightarrow{z_2}}, g_3 = g^{\vec{c}\cdot\overrightarrow{z_2}}$ is sufficient for our purpose!

If $\mathcal{A}$ can find $\overrightarrow{z_1}$, $\overrightarrow{z_2}$ and $\overrightarrow{u_t}'$ such that

$$\vec{r}\cdot\overrightarrow{z_1} = \vec{a}\cdot\overrightarrow{z_2} + \vec{b}\cdot\overrightarrow{y_t}$$
$$\vec{s}\cdot\overrightarrow{u_t}' = \vec{c}\cdot\overrightarrow{z_2} + \vec{d}\cdot\overrightarrow{y_t}$$

note: $\mathcal{A}$ knows that

$$\vec{r}\cdot\overrightarrow{x_{t+1}} = \vec{a}\cdot\overrightarrow{x_t} + \vec{b}\cdot\overrightarrow{y_t}$$
$$\vec{s}\cdot\overrightarrow{u_t} = \vec{c}\cdot\overrightarrow{x_t} + \vec{d}\cdot\overrightarrow{y_t}$$

Then, by subtracting above equations (noting that $\vec{a} \coloneqq \vec{r}^T A,\ \ \vec{c} \coloneqq \vec{s}^T C$),
$\mathcal{A}$ gets

$$\vec{r}\cdot\left((\overrightarrow{z_1} - \overrightarrow{x_{t+1}}) - A(\overrightarrow{z_2} - \overrightarrow{x_t})\right) = 0$$
$$\vec{s}\cdot\left((\overrightarrow{u_t}' - \overrightarrow{u_t}) - C(\overrightarrow{z_2} - \overrightarrow{x_t})\right) = 0$$

Since $\vec{r}, \vec{s}$ are hidden (as $g^{\vec{r}}, g^{\vec{s}}$) and random,

$$(\overrightarrow{z_1} - \overrightarrow{x_{t+1}}) - A(\overrightarrow{z_2} - \overrightarrow{x_t}) = 0\ \ \&\ \ (\overrightarrow{u_t}' - \overrightarrow{u_t}) - C(\overrightarrow{z_2} - \overrightarrow{x_t}) = 0.$$

If not, $\mathcal{A}$ breaks the DL assumption (variant):

Given g and $g^{\vec{x}} \coloneqq (g^{x_1}, g^{x_2}, \dots, g^{x_n})$ $(\vec{x} \leftarrow \mathbb{Z}_p^n)$, $\mathcal{A}$ can not retrieve $\vec{y} \neq \vec{0}$ s.t. $\vec{x}\cdot\vec{y} = 0$

VC for Controller Computation – Fourth idea

Claim: Enforcing $\mathcal{P}$ (or $\mathcal{A}$) to send $g_1 = g^{\vec{r}\cdot\overrightarrow{z_1}},\ g_2 = g^{\vec{a}\cdot\overrightarrow{z_2}}, g_3 = g^{\vec{c}\cdot\overrightarrow{z_2}}$ is sufficient for our purpose!

If $\mathcal{A}$ can find $\overrightarrow{z_1}$, $\overrightarrow{z_2}$ and $\overrightarrow{u_t}'$ such that

$$\vec{r}\cdot\overrightarrow{z_1} = \vec{a}\cdot\overrightarrow{z_2} + \vec{b}\cdot\overrightarrow{y_t}$$
$$\vec{s}\cdot\overrightarrow{u_t}' = \vec{c}\cdot\overrightarrow{z_2} + \vec{d}\cdot\overrightarrow{y_t}$$

note: $\mathcal{A}$ knows that

$$\vec{r}\cdot\overrightarrow{x_{t+1}} = \vec{a}\cdot\overrightarrow{x_t} + \vec{b}\cdot\overrightarrow{y_t}$$
$$\vec{s}\cdot\overrightarrow{u_t} = \vec{c}\cdot\overrightarrow{x_t} + \vec{d}\cdot\overrightarrow{y_t}$$

Then, by subtracting above equations (noting that $\vec{a} \coloneqq \vec{r}^T A,\ \ \vec{c} \coloneqq \vec{s}^T C$), $\mathcal{A}$ gets

$$\vec{r}\cdot\left((\overrightarrow{z_1} - \overrightarrow{x_{t+1}}) - A(\overrightarrow{z_2} - \overrightarrow{x_t})\right) = 0$$
$$\vec{s}\cdot\left((\overrightarrow{u_t}' - \overrightarrow{u_t}) - C(\overrightarrow{z_2} - \overrightarrow{x_t})\right) = 0$$

Since $\vec{r}, \vec{s}$ are hidden (as $g^{\vec{r}}, g^{\vec{s}}$) and random,

$$(\overrightarrow{z_1} - \overrightarrow{x_{t+1}}) - A(\overrightarrow{z_2} - \overrightarrow{x_t}) = 0\ \ \&\ \ (\overrightarrow{u_t}' - \overrightarrow{u_t}) - C(\overrightarrow{z_2} - \overrightarrow{x_t}) = 0.$$

If $\overrightarrow{z_2} = \overrightarrow{x_t}$, we get $\overrightarrow{z_1} = \overrightarrow{x_{t+1}}$, $\overrightarrow{u_t}' = \overrightarrow{u_t}$ as desired

If not, $\mathcal{A}$ breaks the DL assumption (variant).

Given g and $g^{\vec{x}} \coloneqq (g^{x_1}, g^{x_2}, \dots, g^{x_n})$ $(\vec{x} \leftarrow \mathbb{Z}_p^n)$, $\mathcal{A}$ can not retrieve $\vec{y} \neq \vec{0}$ s.t. $\vec{x}\cdot\vec{y} = 0$

VC for Controller Computation – Fourth idea

Claim: Enforcing $\mathcal{P}$ (or $\mathcal{A}$) to send $g_1 = g^{\vec{r}\cdot\vec{z_1}},\ g_2 = g^{\vec{a}\cdot\vec{z_2}}, g_3 = g^{\vec{c}\cdot\vec{z_2}}$ is sufficient for our purpose!

If $\mathcal{A}$ can find $\vec{z_1}$, $\vec{z_2}$ and $\vec{u_t}'$ such that

$$\vec{r}\cdot\vec{z_1} = \vec{a}\cdot\vec{z_2} + \vec{b}\cdot\vec{y_t}$$
$$\vec{s}\cdot\vec{u_t}' = \vec{c}\cdot\vec{z_2} + \vec{d}\cdot\vec{y_t}$$

note: $\mathcal{A}$ knows that

$$\vec{r}\cdot\overrightarrow{x_{t+1}} = \vec{a}\cdot\vec{x_t} + \vec{b}\cdot\vec{y_t}$$
$$\vec{s}\cdot\vec{u_t} = \vec{c}\cdot\vec{x_t} + \vec{d}\cdot\vec{y_t}$$

Then, by subtracting above equations (noting that $\vec{a} \coloneqq \vec{r}^T A,\ \ \vec{c} \coloneqq \vec{s}^T C$), $\mathcal{A}$ gets

$$\vec{r}\cdot\left((\vec{z_1} - \overrightarrow{x_{t+1}}) - A(\vec{z_2} - \vec{x_t})\right) = 0$$
$$\vec{s}\cdot\left((\vec{u_t}' - \vec{u_t}) - C(\vec{z_2} - \vec{x_t})\right) = 0$$

Since $\vec{r}$, $\vec{s}$ are hidden (as $g^{\vec{r}}, g^{\vec{s}}$) and random,

$$(\vec{z_1} - \overrightarrow{x_{t+1}}) - A(\vec{z_2} - \vec{x_t}) = 0\ \ \&\ \ (\vec{u_t}' - \vec{u_t}) - C(\vec{z_2} - \vec{x_t}) = 0.$$

If $\vec{z_2} = \vec{x_t}$, we get $\vec{z_1} = \overrightarrow{x_{t+1}}$, $\vec{u_t}' = \vec{u_t}$ as desired

Holds by induction ($\mathcal{V}$ already verified $\vec{x_t}$)

Details & Optimization - refer to the paper

If not, $\mathcal{A}$ breaks the DL assumption (variant).

Given g and $g^{\vec{x}} \coloneqq (g^{x_1}, g^{x_2}, \ldots, g^{x_n})$ $(\vec{x} \leftarrow \mathbb{Z}_p^n)$, $\mathcal{A}$ can not retrieve $\vec{y} \neq \vec{0}$ s.t. $\vec{x}\cdot\vec{y} = 0$

Proposed VC for Controller Computation – Summary

Goal: Verify the Controller's Computation, i.e., $\begin{pmatrix} \overrightarrow{x_{t+1}} \\ \overrightarrow{u_t} \end{pmatrix} = \begin{pmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{pmatrix} \begin{pmatrix} \overrightarrow{x_t} \\ \overrightarrow{y_t} \end{pmatrix}$.

- It suffices to check that $\begin{pmatrix} \vec{r} \cdot \overrightarrow{x_{t+1}}' \\ \vec{s} \cdot \overrightarrow{u_t}' \end{pmatrix} = \begin{pmatrix} \vec{a} \cdot \overrightarrow{x_t} + \vec{b} \cdot \overrightarrow{y_t} \\ \vec{c} \cdot \overrightarrow{x_t} + \vec{d} \cdot \overrightarrow{y_t} \end{pmatrix}$ from Freivalds' algorithm.

- In fact, it suffices to check that $\begin{pmatrix} g_1 \\ g^{\vec{s} \cdot \overrightarrow{u_t}'} \end{pmatrix} = \begin{pmatrix} g_2 \cdot g^{\vec{b} \cdot \overrightarrow{y_t}} \\ g_3 \cdot g^{\vec{d} \cdot \overrightarrow{y_t}} \end{pmatrix}$

and that g_1, g_2, g_3 are well-generated from $EK \coloneqq (g^{\vec{r}}, g^{\vec{a}}, g^{\vec{c}}, g^{\rho \vec{r}}, g^{\alpha \vec{a}}, g^{\gamma \vec{c}}, g^{\vec{a} - \vec{c}}, g^{\delta(\vec{a} - \vec{c})})$.

Proposed VC for Controller Computation – Summary

Goal: Verify the Controller's Computation, i.e., $\begin{pmatrix} \overrightarrow{x_{t+1}} \\ \overrightarrow{u_t} \end{pmatrix} = \begin{pmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{pmatrix} \begin{pmatrix} \overrightarrow{x_t} \\ \overrightarrow{y_t} \end{pmatrix}$.

- It suffices to check that $\begin{pmatrix} \vec{r} \cdot \overrightarrow{x_{t+1}}' \\ \vec{s} \cdot \overrightarrow{u_t}' \end{pmatrix} = \begin{pmatrix} \vec{a} \cdot \overrightarrow{x_t} + \vec{b} \cdot \overrightarrow{y_t} \\ \vec{c} \cdot \overrightarrow{x_t} + \vec{d} \cdot \overrightarrow{y_t} \end{pmatrix}$ from Freivalds' algorithm.

- In fact, it suffices to check that $\begin{pmatrix} g_1 \\ g^{\vec{s} \cdot \overrightarrow{u_t}'} \end{pmatrix} = \begin{pmatrix} g_2 \cdot g^{\vec{b} \cdot \overrightarrow{y_t}} \\ g_3 \cdot g^{\vec{d} \cdot \overrightarrow{y_t}} \end{pmatrix}$

and that g_1, g_2, g_3 are well-generated from $EK \coloneqq (g^{\vec{r}}, g^{\vec{a}}, g^{\vec{c}}, g^{\rho\vec{r}}, g^{\alpha\vec{a}}, g^{\gamma\vec{c}}, g^{\vec{a}-\vec{c}}, g^{\delta(\vec{a}-\vec{c})})$.

- *Soundness*: If $\mathcal{A}$ can deceive the $\mathcal{V}$erifier with incorrect signal $\overrightarrow{u_t}' \neq \overrightarrow{u_t}$,
  then $\mathcal{A}$ breaks one of the cryptographic assumptions (DL or n-PKE)!

Proposed VC for Controller Computation – Summary

Goal: Verify the Controller's Computation, i.e., $\begin{pmatrix} \overrightarrow{x_{t+1}} \\ \overrightarrow{u_t} \end{pmatrix} = \begin{pmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{pmatrix} \begin{pmatrix} \overrightarrow{x_t} \\ \overrightarrow{y_t} \end{pmatrix}$.

- It suffices to check that $\begin{pmatrix} \vec{r} \cdot \overrightarrow{x_{t+1}}' \\ \vec{s} \cdot \overrightarrow{u_t}' \end{pmatrix} = \begin{pmatrix} \vec{a} \cdot \overrightarrow{x_t} + \vec{b} \cdot \overrightarrow{y_t} \\ \vec{c} \cdot \overrightarrow{x_t} + \vec{d} \cdot \overrightarrow{y_t} \end{pmatrix}$ from Freivalds' algorithm.

- In fact, it suffices to check that $\begin{pmatrix} g_1 \\ g^{\vec{s} \cdot \overrightarrow{u_t}'} \end{pmatrix} = \begin{pmatrix} g_2 \cdot g^{\vec{b} \cdot \overrightarrow{y_t}} \\ g_3 \cdot g^{\vec{d} \cdot \overrightarrow{y_t}} \end{pmatrix}$

and that g_1, g_2, g_3 are well-generated from $EK := (g^{\vec{r}}, g^{\vec{a}}, g^{\vec{c}}, g^{\rho \vec{r}}, g^{\alpha \vec{a}}, g^{\gamma \vec{c}}, g^{\vec{a}-\vec{c}}, g^{\delta(\vec{a}-\vec{c})})$.

- *Soundness*: If $\mathcal{A}$ can deceive the $\mathcal{V}$erifier with incorrect signal $\overrightarrow{u_t}' \neq \overrightarrow{u_t}$,
  then $\mathcal{A}$ breaks one of the cryptographic assumptions (DL or n-PKE)!

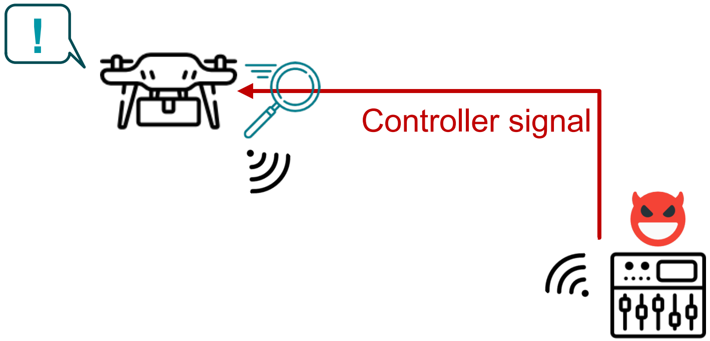
- *Efficiency (# of ops)*: $\underbrace{\textbf{Verify}}_{\propto |\overrightarrow{u_t}|, |\overrightarrow{y_t}| + \text{const for checking } g_i's}$ $(\overrightarrow{u_t}) \ll \underbrace{\text{Computation}}_{\propto |\overrightarrow{x_t}| \cdot (|\overrightarrow{x_t}| + |\overrightarrow{y_t}|)}$ of $\overrightarrow{u_t}$ and $\overrightarrow{x_{t+1}}$

$\propto |\overrightarrow{u_t}|, |\overrightarrow{y_t}|$
$+ \text{const for checking } g_i's$

$\propto |\overrightarrow{x_t}| \cdot (|\overrightarrow{x_t}| + |\overrightarrow{y_t}|)$

Conclusion

- Proposed Verifiable Computation enables a plant-side to detect all possible modifications on the control signal of linear dynamic feedback controllers!
  => Secure the system from most adversarial attacks outside the plant-side.



Conclusion

- Proposed Verifiable Computation enables a plant-side to detect all possible modifications on the control signal of linear dynamic feedback controllers!
  => Secure the system from most adversarial attacks outside the plant-side.

- On-going / Further Work

  - Implementation

  - With other Cryptographic Assumptions: DL → Post-Quantum (Lattices, Hash)

  - More Functionalities:

    1) Hiding Controller's Information (e.g., A,B,C,D) via zero-knowledge proof

    2) Handling other Dynamic System (w/ additional input from the controller)