浙江大学 ZHEJIANG UNIVERSITY

USENIX

网络传感与控制研究组 Group of Networked Sensing and Control

TAT: Attesting Trajectory Integrity of Industrial Robotic Arms

Chengtao Yao *Chengcheng Zhao* *Peng Cheng* *Jiming Chen*

Zhejiang University

Chengtao Yao

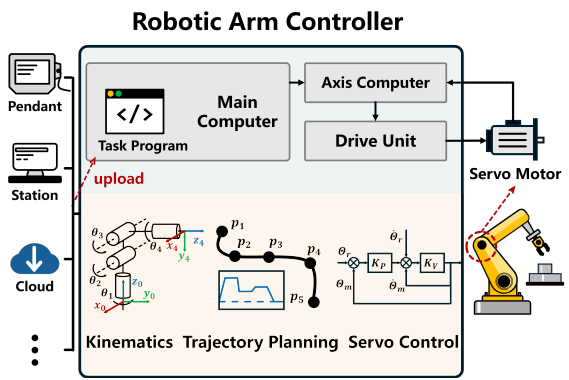
chengtao.yao@outlook.com

Aug 13, 2026

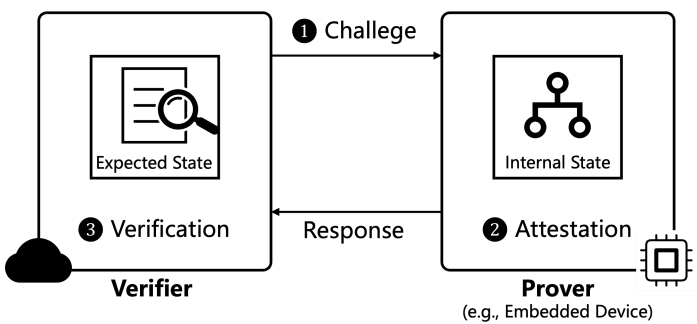
Securing Robotic Arm Motion

- **Motion** is a fundamental capability of robotic arms, enabling the complex and precise actions required for task execution.
- Robotic arm controllers translate high-level task programs into physical motion through coordinated computation and control across multiple computing units.
- Remote attestation verifies a remote device's integrity, but existing techniques mainly focus on program-level properties such as memory integrity, control-flow integrity (CFI), and data-flow integrity (DFI).

> **Can existing attestation techniques ensure the integrity of robotic arm motion?**



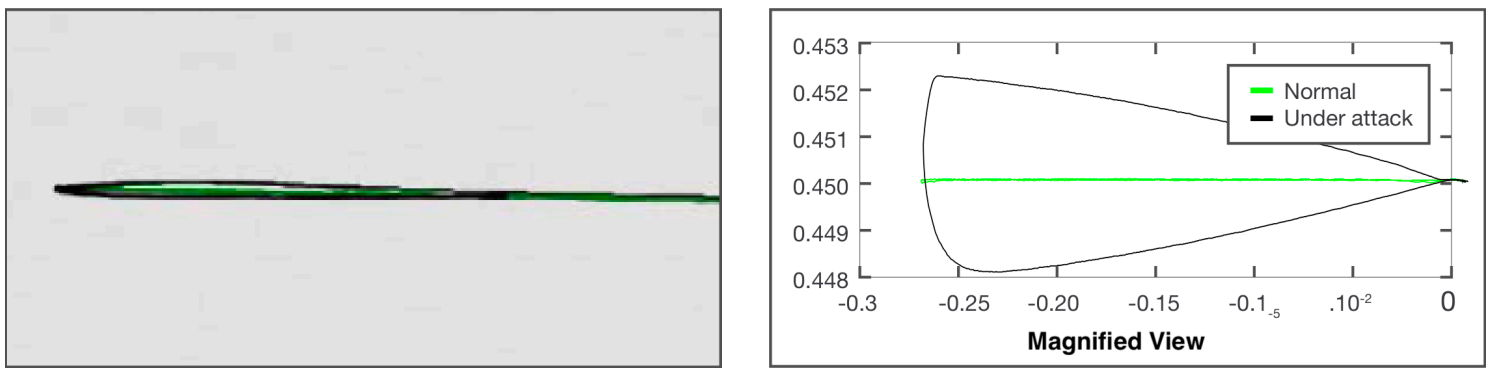
Typical architecture of industrial robotic arm controllers.



Challenge–response protocol in remote attestation.

Attacks on Robotic Arm Motion

- **Production Logic Tampering**
  - Malicious code injection
  - Control-flow hijacking and data-oriented attacks
- **Motion Parameter Tampering**
  - Tampering with low-level controller parameters through exposed APIs or protocol interfaces



Trajectory deviation caused by parameter tampering on an ABB IRB140:
the proportional gain of the first axis is reduced to 50% of its original value. [1]

[1] Quarta et al., “An Experimental Security Analysis of an Industrial Robot Controller,” IEEE S&P 2017.

CFI/DFI Limitations

Incomplete Motion Semantics

- The motion implicitly starts from the current pose.
- High-level movement commands define trajectories coarsely using a target position and speed.
- The same command may produce different trajectories due to different preceding branches.

> ***Task-level CFI/DFI cannot determine the intended trajectory.***

Physical Knowledge Dependency

- Kinematic models (e.g., D-H parameters)
- Calibration parameters (e.g., encoder resolution)
- Reference frames and transformations (e.g., the world frame)
- . . .

> ***Extending CFI/DFI across the entire control stack is impractical.***

```
// Event: Move to a branch-specific pick position
if (part_id == 1) {
  WriteCoords(pick_pos1, mode=1, speed=10);
  while (!IsInPosition(pick_pos1, offset)) {
    usleep(200);
  }
} else if (part_id == 2) {
  WriteCoords(pick_pos2, mode=1, speed=10);
  while (!IsInPosition(pick_pos2, offset)) {
    usleep(200);
  }
}
// Event: Pick up the part
SetGripper(1);
sleep(2);

// Event: Move to the shared place position
WriteCoords(place_pos, mode=1, speed=20);
while (!IsInPosition(place_pos, offset)) {
  usleep(200);
}
```

Different motion trajectories based on the pick position.

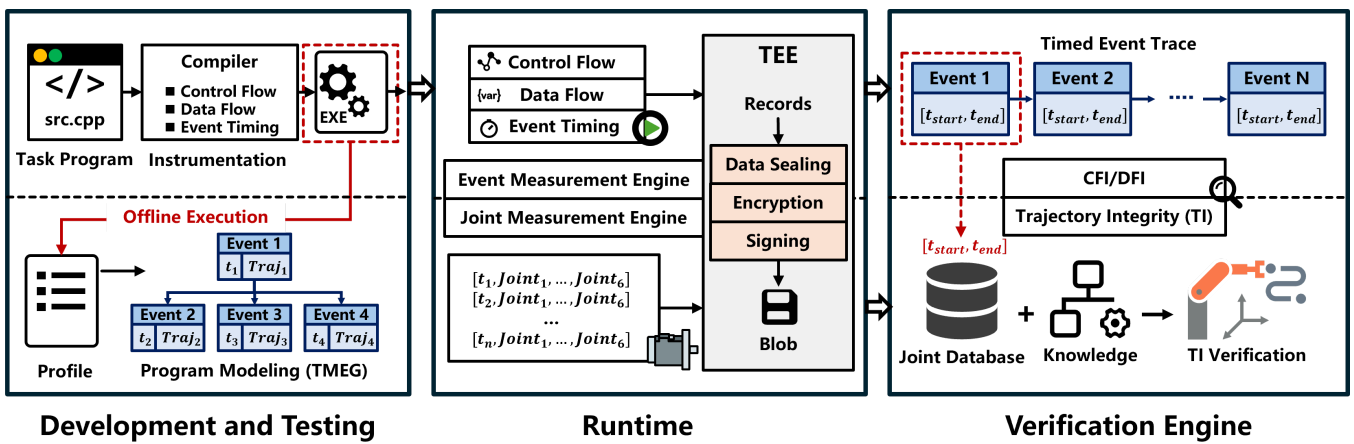
```
// Event: Place the part
SetGripper(0);
sleep(2);
```

Trajectory Integrity

From a motion perspective, we view robotic arm task execution as a sequence of bounded events:

- Movement Events: Follow a predefined trajectory within the expected duration.
- Static Events: Maintain a stable position within a reasonable time bound.

Trajectory Integrity (TI) requires robotic arm motion to conform to its intended spatial path and temporal profile.



Overview of TAT workflow

Program Modeling

TMEG Definition

We propose the **Timed Motion Event Graph (TMEG)** to model task programs with motion semantics. A TMEG is a directed graph $G = (\Sigma, E, \alpha, \beta, \gamma)$, where:

- Σ is the set of vertices that represent motion events, and E is the set of directed edges indicating the execution order.
- α maps each event to its corresponding command sequence in the task program.
- β associates events with reference profiles $\pi = (t, \tau)$, specifying duration t and trajectory τ.
- γ captures profile dependencies between predecessor and successor events.

TMEG Construction

- **Static Program Analysis:**
  - Identifies event boundaries based on control flow and blocking patterns.
  - Constructs the initial event graph and instruments the event boundaries.
- **Dynamic Reference Profile Generation:**
  - Collects timestamped event traces and real-time positions by running the instrumented program in controlled tests.
  - Aligns the collected data to generate reference profiles for each event.
  - Records profile dependencies between predecessor and successor events across task branches.

Measurement Engines

TAT records only task execution behavior and raw joint readings, minimizing runtime overhead without bringing the controller stack into the TCB.

- **Event Measurement Engine**
  - Monitors the control flow and data flow of the task program.
  - Records the entry and exit timestamps of each event.
  - Uploads measurements through the challenge–response protocol.
- **Joint Measurement Engine**
  - Continuously samples raw motor encoder byte streams through a bypass module.
  - Periodically uploads measurements to the verifier.
- **Trusted Protection:** The TEE securely stores, encrypts, and signs all measurements before transmission.

Verification Engine

TAT verifies both task program execution (**CFI/DFI**) and physical motion (**TI**).

Trajectory Reconstruction

- Aligns the recorded events with the corresponding raw joint readings using timestamps.
- Reconstructs the executed trajectory using calibration and kinematic knowledge.

TI Verification

- Duration: Checks the event duration against its reference temporal bound.
- Trajectory: Compares the measured and reference trajectories using Absolute Trajectory Error (ATE) for global deviations and Relative Pose Error (RPE) for local deviations.

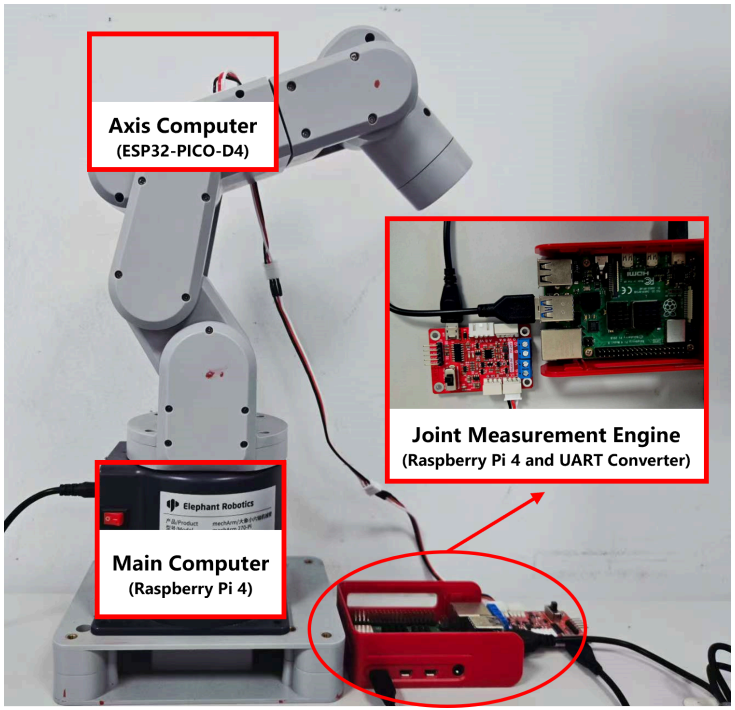
$$\mathrm{ATE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\mathrm{trans}(T_{m,i}^{-1} T_{r,i})\|^2}, \quad \mathrm{RPE} = \sqrt{\frac{1}{N-\Delta}\sum_{i=1}^{N-\Delta} \|\mathrm{trans}((T_{m,i}^{-1} T_{m,i+\Delta})^{-1} (T_{r,i}^{-1} T_{r,i+\Delta}))\|^2}$$

where $\tau_m = \{T_{m,i}\}_{i=1}^{N}$ and $\tau_r = \{T_{r,i}\}_{i=1}^{N}$ denote the measured and reference trajectories, respectively. $T_{m,i}$ and $T_{r,i}$ are time-aligned measured and reference poses; N is the number of pose pairs, Δ is the sample-index interval, and $\mathrm{trans}(\cdot)$ extracts the translation component.

Evaluation

Implementation

- **Host Platform**
  - Six-axis mechArm 270-Pi robotic arm
  - Raspberry Pi main computer and ESP32 axis computer
- **Compiler**
  - LLVM pass for TMEG construction and instrumentation
  - Ported OAT for CFI/DFI instrumentation
- **Measurement Engines**
  - Event engine: runs on the main computer
  - Joint engine: runs on a separate Raspberry Pi and directly collects motor readings via UART
  - Both engines run in the OP-TEE trusted environment
- **Verification Engine**
  - Runs on a trusted remote backend



Evaluation

Runtime Overhead

- *Task Programs:* 3 open-source examples (OP.T1-OP.T3) and 7 real-world industrial tasks (PL1.T1-PL2.T2).
- *Compilation Settings:* Original programs; with OAT instrumentation (CFI/DFI only); with TAT instrumentation.
- *Executions:* 1,000 executions per task program and compilation setting.

Task	Events			Memory Size (MB)			Execution Time (s)		
	m.	s.	t.	orig.	w/ OAT(orig.↑)	w/ TAT(orig.↑, w/ OAT↑)	orig.	w/ OAT(orig.↑)	w/ TAT(orig.↑, w/ OAT↑)
OP.T1	10	6	16	6.961	7.050 (1.28%)	7.060 (1.43%, 0.14%)	26.454	26.482 (0.11%)	26.486 (0.12%, 0.02%)
OP.T2	8	4	12	6.963	7.050 (1.25%)	7.051 (1.27%, 0.02%)	13.451	13.465 (0.10%)	13.467 (0.12%, 0.01%)
OP.T3	11	8	19	6.967	7.058 (1.31%)	7.063 (1.39%, 0.08%)	36.457	36.486 (0.08%)	36.489 (0.09%, 0.01%)
PL1.T1	14	12	26	6.966	7.047 (1.15%)	7.071 (1.50%, 0.34%)	46.457	46.481 (0.05%)	46.485 (0.06%, 0.01%)
PL1.T2	33	65	98	6.967	7.087 (1.73%)	7.127 (2.30%, 0.56%)	193.488	193.659 (0.09%)	193.721 (0.12%, 0.03%)
PL1.T3	11	16	27	6.958	7.053 (1.37%)	7.068 (1.59%, 0.22%)	34.953	34.965 (0.03%)	34.973 (0.06%, 0.02%)
PL1.T4	10	18	28	6.961	7.058 (1.40%)	7.074 (1.63%, 0.22%)	33.66	33.678 (0.05%)	33.687 (0.08%, 0.03%)
PL1.T5	13	18	31	6.966	7.065 (1.43%)	7.077 (1.59%, 0.16%)	63.461	63.531 (0.11%)	63.551 (0.14%, 0.03%)
PL2.T1	7	6	13	6.946	7.036 (1.29%)	7.048 (1.46%, 0.17%)	15.651	15.658 (0.04%)	15.661 (0.06%, 0.02%)
PL2.T2	23	7	30	6.952	7.049 (1.40%)	7.067 (1.66%, 0.25%)	188.684	188.807 (0.07%)	188.827 (0.08%, 0.01%)

- **Memory:** **1.27–2.30%** over original programs; $\leq$ **0.56%** extra over OAT.
- **Execution Time:** **0.06–0.14%** over original programs; $\leq$ **0.03%** extra over OAT.

Evaluation

Attestation Capability

- *Parameter Modification Types:*
  - Motion Dynamics: adjust the servo controller's proportional gain.
  - Spatial Positioning: apply translational and rotational offsets to the reference frame.
- *Executions:* 100 executions per motion type and parameter setting.
- *Metrics:* ATE-based, RPE-based, and overall detection rates using 4σ thresholds.

		GT	Proportional Gain (P) Adjustment			Translational Offset		Rotational Offset	
			80% P	120% P	150% P	1mm	2mm	1°	2°
J1	A.	0.48 ± 0.24	1.29 ± 0.42	0.89 ± 0.32	1.66 ± 0.30	2.56 ± 0.22	2.10 ± 0.25	4.60 ± 1.11	7.29 ± 4.38
	R.	0.41 ± 0.10	0.90 ± 0.07	0.56 ± 0.11	0.99 ± 0.13	0.65 ± 0.15	0.79 ± 0.09	0.79 ± 0.13	0.82 ± 0.08
	D.	0.0/0.0/0.0	**0.08/0.96/0.96**	**0.07/0.01/0.07**	**0.72/0.92/0.95**	1.0/0.19/1.0	1.0/0.44/1.0	1.0/0.41/1.0	1.0/0.67/1.0
J2	A.	0.47 ± 0.16	2.74 ± 0.94	2.31 ± 0.31	1.90 ± 0.24	2.20 ± 0.45	2.67 ± 0.70	4.17 ± 0.86	7.11 ± 1.28
	R.	0.39 ± 0.08	1.22 ± 0.13	0.81 ± 0.13	1.42 ± 0.12	0.51 ± 0.17	0.72 ± 0.26	0.99 ± 0.13	1.03 ± 0.16
	D.	0.0/0.0/0.0	1.0/1.0/1.0	1.0/0.81/1.0	1.0/1.0/1.0	1.0/0.04/1.0	1.0/0.45/1.0	1.0/0.99/1.0	1.0/1.0/1.0
L1	A.	0.17 ± 0.06	1.78 ± 0.20	1.39 ± 0.24	1.62 ± 0.28	1.63 ± 0.42	2.57 ± 0.48	4.69 ± 1.68	9.63 ± 3.26
	R.	0.13 ± 0.03	0.47 ± 0.06	0.28 ± 0.11	0.33 ± 0.07	0.35 ± 0.06	0.44 ± 0.11	0.49 ± 0.08	0.52 ± 0.14
	D.	0.0/0.0/0.0	1.0/1.0/1.0	1.0/0.56/1.0	1.0/0.99/1.0	1.0/1.0/1.0	1.0/1.0/1.0	1.0/1.0/1.0	1.0/1.0/1.0
L2	A.	0.26 ± 0.10	2.15 ± 0.21	1.12 ± 0.14	1.28 ± 0.14	1.26 ± 0.25	2.08 ± 0.47	3.76 ± 0.99	7.33 ± 1.22
	R.	0.22 ± 0.04	0.44 ± 0.09	0.77 ± 0.05	0.81 ± 0.09	0.47 ± 0.13	0.50 ± 0.15	0.77 ± 0.05	0.51 ± 0.14
	D.	0.0/0.0/0.0	1.0/0.79/1.0	1.0/1.0/1.0	1.0/1.0/1.0	1.0/0.71/1.0	1.0/0.77/1.0	1.0/1.0/1.0	1.0/0.72/1.0

ATE detects global deviations; **RPE** improves detection of local jitter.

浙江大学 ZHEJIANG UNIVERSITY

USENIX

NeSC 网络传感与控制研究组 Group of Networked Sensing and Control

Thank you for listening

Artifacts:

Zenodo: doi.org/10.5281/zenodo.17958178

Email: chengtao.yao@outlook.com