Crowdsourcing Framework for Security Testing and Verification of Industrial Cyber-Physical Systems

Zhenyu Li, Yong Ding, Jun Li*, and Faezeh Farivar

***Abstract*—With the widespread adoption of Industrial Cyber-Physical Systems (ICPS) by global enterprises, these systems have become increasingly exposed to cybersecurity threats due to their inherent vulnerabilities. While existing protective measures can block attacks as they occur, the possibility of defense failures still exists. To proactively address potential cyber attacks on ICPS, we have designed a distributed crowdsourced testing platform specifically for ICPS. This platform not only supports security testing of ICPS deployed on-site, but also enables environment simulation and security testing of ICPS in cloud environments. Additionally, we have developed a testing and verification framework tailored to the core functions of this crowdsourced platform. This framework allows distributed multi-point testing and verification of ICPS in cloud environments. Through a series of experimental validations, we demonstrate that the proposed framework can improve testing and verification speed by approximately 2.6 times compared to Apache JMeter.**

***Index Terms*—Industrial Cyber-Physical System, Testing, Verification, Security, Crowdsourcing.**

I. Introduction

With the emergence of Industry 4.0, industrial Cyber-Physical Systems (CPS) are being increasingly deployed by major enterprises. By integrating advanced computing technologies, network communications, and physical processes, these systems significantly enhance the automation, intelligence, and efficiency of production processes [1]. Additionally, these systems can monitor and adjust production activities in real-time, reducing manual intervention while improving resource utilization and production flexibility. Such systems will form a crucial part of smart manufacturing [2]. By 2030, it is expected that the global market size for industrial cyber-physical systems to reach 177.5 billion dollars [3].

In the industrial cyber-physical systems connected to networks, which could even involve critical infrastructure, there is a high risk of cyber attacks [4], [5]. The first attack that drew experts' attention to the threats on industrial cyber-physical systems was carried out by Stuxnet. Stuxnet virus could precisely target supervisory control and data acquisition systems and cause physical damage [6].

With the continuous advancement of technology, as ordinary enterprises started connecting themselves to the internet, ransomware attacks on industrial cyber-physical systems also increased. A prominent example is the May 2021 attack on Colonial Pipeline, which resulted in significant operational disruptions [7]. Therefore, the security of ICPS has become a major issue in industrial intelligence, which underlines the necessity of conducting extensive research to ensure the security of industrial cyber-physical systems.

However, although existing security measures can alleviate current challenges to some extent, the capability for proactive security protection remains insufficient [8]. To address the security challenges of industrial cyber-physical systems, this paper conducts research on testing platforms and their core modules with the following main contributions:

- A distributed crowdsourced testing platform is designed for security testing of industrial cyber-physical systems.
- A distributed testing framework supporting variable delays is proposed to address the difficulties in post-test verification faced by crowdsourced platforms.
- A multi-threaded testing architecture supporting variable delays is designed which can improve effective verification speed by approximately 2.6 times compared to Apache JMeter.

The remaining chapters of this paper are organized as follows. Chapter II introduces the architecture of industrial cyber-physical systems along with the security challenges and solutions they have. In Chapter III, we present our distributed crowdsourced testing platform designed to address the security challenges of ICPS. We then implement the core components of the distributed crowdsourced testing platform as well as the verification framework in Chapter IV. Finally, we evaluate the performance of the testing and verification frameworks and conclude the paper.

II. Security Challenges in Industrial Cyber-Physical Systems

Due to the complex and multi-layered nature of industrial cyber-physical systems, which involve various layers such as hardware, software, and network, they face unprecedented security challenges, especially when they are intertwined with cloud services. Cloud-based cyber-physical systems, which are directly connected to the internet, deal with more security risks compared to isolated ICPS networks.

Z. Li is with School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, 541004, China, email: 22031202013@mails.guet.edu.cn.

Y. Ding is with School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, 541004, China, and also with the HKCT Institute of Higher Education, Hong Kong, 999077, China, email: stone_dingy@guet.edu.cn.

J. Li (corresponding author) is with China Industrial Control Systems Cyber Emergency Response Team, Beijing, 100040, China, email: lijun@cics-cert.org.cn.

F. Farivar is with Department of Computer and Mechatronics Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran, and also with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran 19395-5746, Iran, email: farivar@iau.ac.ir.

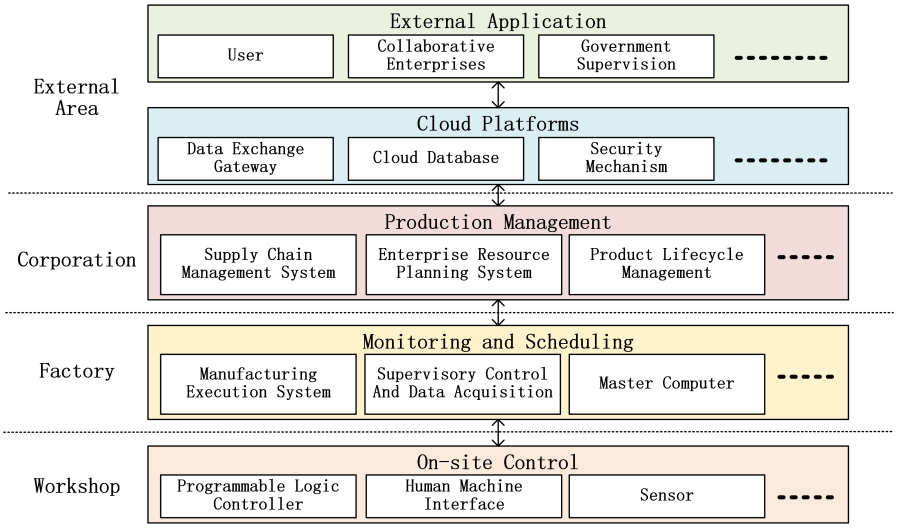

Fig. 1: Architecture of Industrial Cyber-Physical Systems.

A. Architecture of Industrial Cyber-Physical Systems

Cloud-based ICPS architectures are upgrades from traditional cyber-physical systems [9]. Different research perspectives have provided multiple classifications for these systems. To analyze security issues at the system level, this paper adopts a four-layer approach: the workshop layer, the factory layer, the corporation layer, and the external area layer, which together, form an integration of hardware, software, platforms, and cross-network systems. The architecture of cloud-based ICPS is shown in Figure 1.

In practice, the workshop layer primarily interacts with the physical environment to complete production tasks. The factory layer monitors workshop device and controls scheduling according to production demands in order to optimize production processes. At the corporation level, multiple factories are managed to optimize resource allocation and improve efficiency based on actual manufacturing needs. Additionally, the external area allows for the exchange of production information with users and collaborative enterprises through cloud platforms, as well as supporting government supervision.

B. Security Challenges

Cloud-based ICPS can significantly enhance production efficiency through real-time data processing and intelligent decision-making while reducing operational costs. However, increasing reliance on cloud computing platforms can also lead to a rising risk of cyberattacks spreading from the Internet into the enterprise. As a result, issues such as data security, privacy protection, internal network and system security are becoming increasingly prominent. For example, in terms of cyber-security mechanisms, cyber-physical systems in edge networks often use communication methods that lack message authentication to speed up communication efficiency, which makes it difficult to defend against false data injection attacks. In addition, many of the communication protocols used by cyber-physical systems lack message encryption, making it difficult to protect against eavesdropping attacks. Moreover, many cyber-physical systems in internal edge networks lack sufficient proactive security checks due to their long-term deployment in internal networks, resulting in undetected and unresolved security vulnerabilities [10].

Therefore, attackers can exploit existing security vulnerabilities in ICPS by combining different attack methods related to these vulnerabilities, potentially causing damage to the components of ICPS or sending incorrect signals to them and severely impact the entire production process and even triggering safety incidents. Hence, addressing these software-induced security issues requires not only traditional protection measures but also proactive management of system vulnerabilities.

C. Security Solutions

In industrial cyber-physical systems, identifying and addressing system vulnerabilities in advance is crucial for ensuring security. Vulnerability testing is typically divided into black-box, gray-box, and white-box [11], each with its advantages and disadvantages. They are often used in combination in practice.

1) Black-box testing: This method works from an external perspective without knowledge of the system's internal structure. It detects vulnerabilities through input-output behavior, simulating an external attacker's viewpoint. While effective for evaluating external threats, it may miss deeper vulnerabilities due to the lack of internal logic understanding.

2) Gray-box testing: Positioned between black-box and white-box testing, gray-box testing provides testers with partial system information. By combining external and internal perspectives, it offers better vulnerability detection, especially suitable for complex systems.

3) White-box testing: This approach tests the system from the inside, with testers having access to source code and internal structure. It excels in uncovering logical flaws and coding defects, particularly algorithm vulnerabilities. However, its coverage depends on the tester's experience and time constraints.

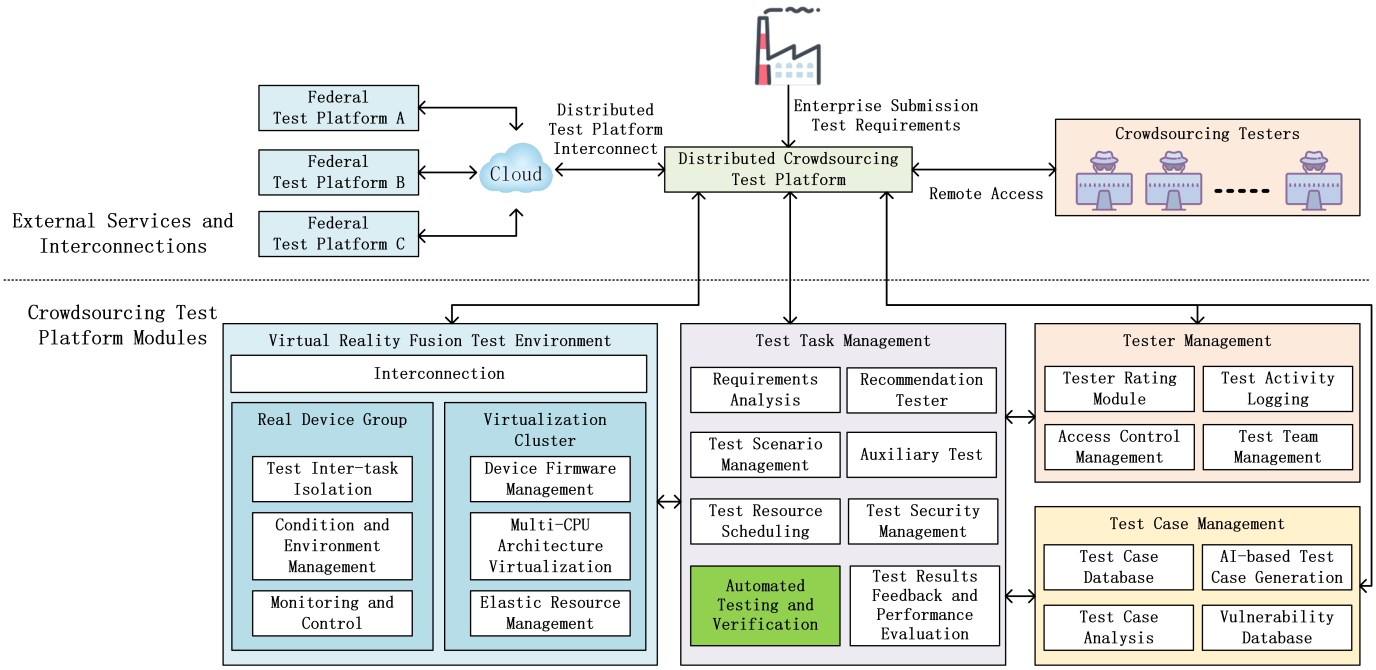

Fig. 2: Design of Distributed Crowdsourcing Test Platform

Directly testing a running deployed system may disrupt production. To avoid affecting the normal operation of ICPS, testing methods like the aforementioned can be carried out using testbeds for safe testing. For example, Shi et al. [12] proposed a cybersecurity testbed for vehicular networks which provides a test environment for CAN Bus security. Additionally, Riquelme-Dominguez et al. [13] developed a virtual reality testing platform that enables communication between simulated and physical devices for testing purposes. Furthermore, Batarseh et al. [14] proposed a cyber-physical water test platform that leverages AI and cybersecurity practices to enhance water resource management capabilities.

Nevertheless, single testbeds may not effectively cover diverse testing scenarios and may struggle to support multiple testing teams in addressing security vulnerabilities. Therefore, to better ensure system security, a combination of various testing methods is usually necessary to expand the testing scope and continuously optimize strategies during security assessments [15].

III. Distributed Crowdsourcing Test Platform Design

To comprehensively anticipate and address security threats faced by ICPS, we design a distributed crowdsourcing testing platform, which is shown in Figure 2. It includes 1) a virtual-reality fusion testing environment, 2) tester management, 3) test case management, and 4) test task management. This platform enables ICPS security testing without interfering with enterprise's production environment. Additionally, the platform can interconnect with other federated testing platforms, forming a distributed and interconnected ICPS software testing alliance that provides rich testing scenarios.

A. Virtual Reality Funsion Test Environment

The virtual-reality fusion testing environment consists of three main components: Virtualization clusters, physical device groups, and network interconnection, providing the foundational operating environment for the entire testing platform. The details of these components are discussed below:

1) Virtualization Cluster: To build a comprehensive testing environment, the virtualization cluster needs to support various CPU architectures to simulate basic testing environments and ICPS firmware. It also requires elastic resource management capabilities to optimize resource usage and support more tests given the limited resources.

2) Physical Device Group: During the security testing process, there are cases where it is difficult or impossible to obtain the device firmware, making it impossible to simulate the device directly. Therefore, real physical devices are required to provide a complete testing environment for testers. However, if the state of a single physical device changes during the testing process, it can affect the results of other tests using the same device. Hence, physical devices must be managed in isolated environments for different testing tasks and instances. For each physical device, different isolation strategies should be implemented according to the task requirements. Moreover, physical devices may malfunction during security testing by receiving incorrect test commands, making subsequent testing tasks difficult. As a result, physical device groups must support the management and restoration of initial states and testing environments, as security tests can render devices inoperable. Monitoring and control of physical device states are also necessary to effectively validate test cases.

3) Network Interconnection: To interconnect different structures of devices and provide a transparent testing environment, the network interconnection module needs to uti-

lize technologies like Software-Defined Networking (SDN) to connect devices across different architectures and network layers. Moreover, it should enable interconnection with other federated testing platforms to support ICPS security testing across a broader range of scenarios.

B. Tester Management

Tester management focuses on categorizing and rating testers to support the management of tester-task assignments. The modules involved are as below:

1) Tester Rating: This module realizes the categorization and rating of testers by evaluating the technical level and ability of crowdsourcing testers, which can provide data support for the allocation of different testers according to the difficulty of different testing tasks in the test task management.

2) Test Activity Logging: During security testing, there is a risk that testers may maliciously damage the test platform or test environment. Therefore, the tester management module needs to log all the operations performed by testers to provide basic data for blocking and controlling abnormal behaviors. In addition, the test activity logging module can also provide data support for tester rating.

3) Access Control Management: The entire testing process requires access control management based on the status of the test task and the operation of the testers, which can be realized by restricting the access of the testers to non-essential equipment and areas accordingly. Thus, the access control management module can ensure the security control of the whole security testing life cycle. In addition, the access control management module not only enhances the security of the test platform, but also allows the test platform administrator to execute security control down to the level of each tester.

4) Test Team Management: Since individual testers might not have comprehensive knowledge of all aspects, team management is introduced to facilitate collaboration in security testing through building a holistic test environment.

C. Test Case Management

Test case management aims to provide technical support for testers by enabling them to write usable test codes with minimal specialized knowledge, which in turn reduces testing difficulty. The components involved are discussed below:

1) Vulnerability Database: By collecting and analyzing publicly available security vulnerabilities and the corresponding verification payloads, this database provides foundational support for assisted generation of test cases.

2) Test Case Database: During ICPS security testing, numerous test cases are generated. Verified and effective test cases are stored in the test case database for future reference by crowdsourced testers.

3) Test Case Analysis Module: To effectively analyze security vulnerabilities, this module leverages the vulnerability and test case databases to identify common characteristics in new test cases.

4) AI-Based Test Case Generation Module: To assist testers in ICPS security testing, this module utilizes AI technology to conduct deeper training or fine-tuning on existing database data, generating test case codes to expedite security testing.

D. Test Task Management

Test task management is the core function of the entire distributed testing platform. It provides full lifecycle management for crowdsourced security testing of ICPS. The following subsections discuss the key functions of the testing platform.

1) Requirement Analysis: By analyzing the testing requirements submitted by enterprises and combining them with the platform's existing resources, testing tasks can be quickly constructed to initiate the process.

2) Tester Recommendation: By analyzing testers' historical data, the system can quickly match testing tasks with the skills of those individuals who are familiar with ICPS security testing and are the best fit so that high-quality outputs are guaranteed.

3) Test Scenario Management: Through template-based management of commonly used test scenarios, the system can help to quickly build test environments and consequently, reduce the workload of platform administrators.

4) Test Resource Scheduling: Before testing begins, the resources needed for instantiating the test environment must be allocated according to the test scenarios designed for the tasks. This will ensure that resource shortages do not occur during testing.

5) Test Security Management: For the security management of crowdsourced testing, the test platform needs to build a security baseline on the basis of the access control management of testers, which can protect the security of the test platform and the security of the test environment that does not involve testers. For example, when the test task is in progress, the test security management module needs to ensure that the background traffic generation program of the test environment required for the task is secure and controllable. In addition, the test security management module also needs to timely check whether the hardware and software of the test platform providing basic services have installed the latest security updates.

6) Auxiliary Testing: The system provides secure access management for test cases, offering testers a module that meets the needs of rapid testing while restricting access to non-essential test cases.

7) Automated Testing and Verification: This is a core component of the distributed crowdsourced testing platform. It assists testers in quickly conducting tests and verifications on test subjects. The framework supports the platform in quickly verifying test results submitted by testers, ensuring that the submitted test cases are effective in triggering security vulnerabilities. This module is the main focus of this paper, which is described in detail in the next section.

8) Test Result Feedback and Performance Evaluation: After a test is completed, the test reports must be fed back to the enterprises that provided the requirements so that they can implement security upgrades and patches accordingly. Additionally, the performance of the testers needs to be evaluated, and rewards shall be allocated based on the submitted test results.

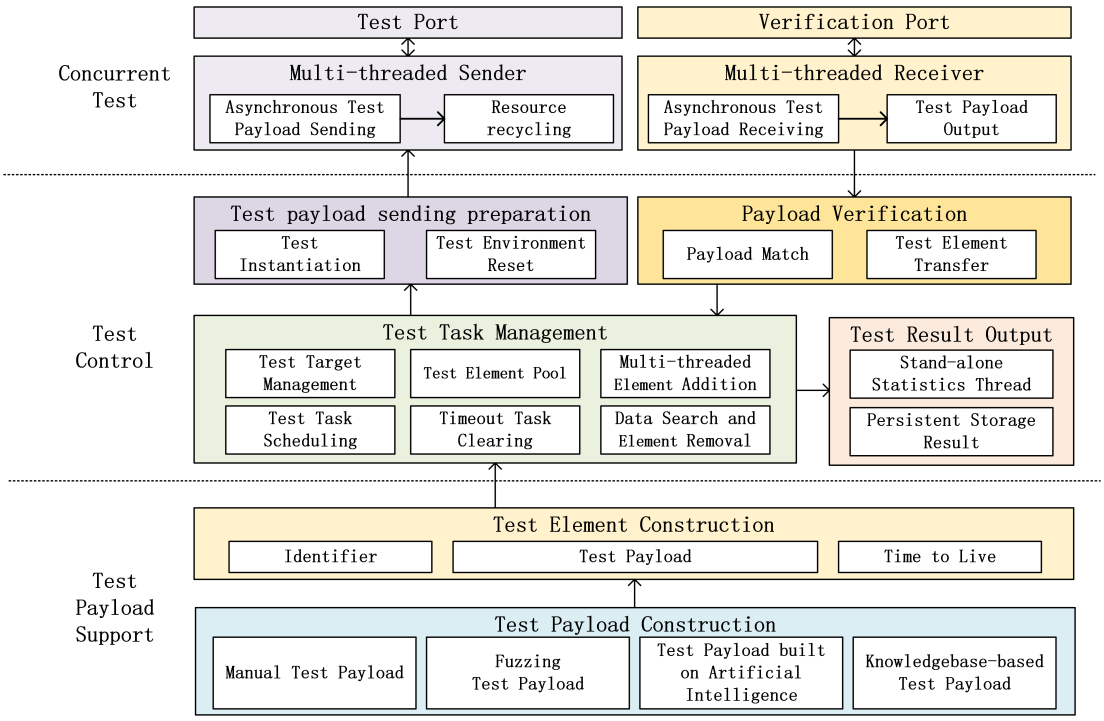

Fig. 3: Architecture of Testing and Verification Framework

IV. Testing & Verification Framework of Distributed Crowdsourcing Test Platform

To address the requirements of distributed crowdsourced testing, this paper proposes a framework for automated testing and verification which is investigated as a core part of the testing platform to verify the presence of vulnerabilities in target ICPS. The special design of this framework allows it to support multi-threaded testing with variable delays. Industrial cyber-physical systems data can come from multiple locations and the results are not necessarily returned to the same places. Therefore, the automated testing and verification framework needs to be designed for distributed scenarios. The following sections discuss the proposed framework.

A. Architecture of Testing and Verification Framework with Delay Support

Considering the importance of the impact of network latency, a distributed testing and verification framework that supports different network delays is proposed, as shown in Figure 3. The framework consists of three main parts: test payload support, test control, and high-concurrency testing. All parts utilize the proposed test element model, which allows for rapid testing and verification in the presence of varying delays, thereby achieving efficient test confirmation.

B. Test Payload Support

The main objective of the test payload unit is to build test models that support verification under various delays, based on existing test payloads. This part can be divided into two main modules: test payload construction and test element construction.

In the payload construction module, the generation methods of test payloads are diversified to meet different testing requirements. Specifically, test payloads can be manually constructed, generated through fuzz testing, created by using AI techniques, or extracted from a knowledge base. The choice of generation method should be based on test objectives and acceptable database types to ensure that the generated test payloads are targeted and effective.

Once the test payloads are generated, the process moves to the test element construction module. In this module, each test element needs to be assigned a unique identifier for precise matching and lifecycle management purposes during the test task management. Additionally, a lifespan should be specified for each test element, allowing the test control component to identify and manage timed-out test tasks. This will ensure controllability and reliability of the testing process. Such an approach enables the testing process to better address the diverse requirements of complex scenarios. It enhances the accuracy and comprehensiveness of testing and verification.

C. Test Control

After constructing the test elements, the testing and verification work can begin. The main goal of the test control part is to send test payloads to designated targets based on test task requirements, retrieve the associated test results from the specified locations, and then verify the results against the test payloads. Therefore, the test control part includes four main modules: test task management, test payload sending preparation, payload verification, and test result output.

First, the test task management module configures the test targets according to the requirements of the test task. Subsequently, the test task management module adds the test

elements to be sent to the test element pool, which ensures that the test elements can be sent as scheduled. After that, the test task scheduling component timely transfers the elements in the test element pool to the test payload sending preparation module according to the requirements of the test task. In addition, the test task management module needs to use the timeout task clearing component to periodically clean up the sent timeout tasks and timeout test elements during the whole testing process, which can prevent the test element pool from accumulating too many unprocessed elements, thus realizing the enhancement of the system resource management and the improvement of operation efficiency.

Subsequently, in the module for test payload sending preparation, the system begins to instantiate test elements and transmit them to the multi-threaded sender for execution. Once the transmission is complete, the environment is cleared to prepare for the next batch of test data.

Meanwhile, in the payload verification module, the task manager continuously receives payloads that need to be verified from the multi-threaded receiver and performs matching operation. This process ensures the accuracy of test data and verifies that every test element's payload is adequately verified.

Once matching is completed, the process moves to reporting the test result output. At this point, the test elements are removed from the element pool in the task manager and transferred to the test result module. In the result module, an independent thread analyzes the test results and generates parsed outputs. Finally, the results are persistently stored for subsequent analysis and reference. Throughout the above steps, the test control part ensures smooth completion of test tasks and provides detailed test results to assist in comprehensive verification.

D. Concurrent Testing

In the concurrent testing part, since the testing input and verification output locations are inconsistent in a distributed environment, the testing and verification process shall be split into two independent components: a multi-threaded sender and a multi-threaded receiver.

The multi-threaded sender is responsible for sending the test payload to the interface where the test target receives the payload. To ensure efficient test execution, the sender needs to work asynchronously, which guarantees rapid transmission of test payloads and avoids blockades in sending tasks. Moreover, after each transmission is completed, the sender should promptly release the system resources it occupied to maintain system stability and operational efficiency.

On the other hand, the multi-threaded receiver is responsible for continuously querying the verification interface and receiving the test payload to be verified. The receiver needs to work in a non-blocking mode, meaning that during the querying process of the receiver, even if the verification interface is unable to respond in time (e.g. due to being busy because of processing the data or too much access requests), the data receiver must be able to obtain other test payloads that are supposed to be verified. As a result, the receiver, using this continuous query and reception mechanism, is able to acquire and process test data in the distributed environment to achieve quick verification of the test results.

V. Evaluation

In this section, the proposed testing and verification framework is evaluated from three perspectives: data exchange speed during testing and verification, successful verification count, and valid testing verification rate.

A. System Configuration

During the experiment, multiple virtual machines with 8 Intel(R) Xeon(R) Gold 6128 @ 3.40GHz vCPUs and 16GB of memory were set up on a virtualization platform. Additionally, to simulate cross-network connections, random delays were added to the data transmission between the testing and verification interfaces. Therefore, the experiment included tests with no additional delay and tests with random delays ranging from 1ms to 10ms. Different performance tests were also conducted with thread counts of 1, 2, and 4. However, the method proposed in this paper requires that both the sender and receiver are working for testing and verification, where each sender and receiver is running in a separate thread in order to work properly. As a result, the total number of threads during execution is always even, making single-threaded testing impossible. Therefore, the proposed method does not include single-thread tests. Additionally, three other software tools, i.e. Apache JMeter, Locust, and a baseline implementation of the method proposed in this paper using the same programming language to imitate JMeter (defined as Baseline in the experimental data), were selected for comparison.In this case, the experimental control group works by performing send followed by receive, and they can use a single thread to complete their work and thus the number of test threads can be set to one. To effectively evaluate sequential execution devices, the test target was set to return only one test payload that needs verification for each access.

B. Evaluation of Valid Testing and Verification Rate

The main contribution of this paper is proposing a testing and verification framework that supports distributed unpredictable delays. To effectively assess the success rate of testing under distributed unpredictable delays, an evaluation of the valid testing verification rate was done whose results are

TABLE I: Valid Verification Rate

Method	Thread Count	Verification Rate for Different Delay	
		0 ms	1 ∼10 ms
Baseline	1	**100%**	N/A
	2	73.91%	N/A
	4	54.05%	N/A
Apache JMeter	1	**100%**	N/A
	2	87.72%	N/A
	4	63.19%	N/A
Locust	1	**100%**	66.67%
	2	99.94%	66.67%
	4	98.86%	66.85%
Our Method	2	**100%**	**100%**
	4	**100%**	**100%**

reported in Table I. The results show that under conditions of multi-threaded contention and varying delays, the proposed method, by utilizing the test element pool, is able to search for and verify previously sent test payloads, even in the case of unordered responses due to unpredictable delays, which ensures that the proposed method achieves full verification of previously sent payload. Meanwhile, with one thread and no unpredictable delays, other methods also managed to achieve 100% verification due to sequential execution. However, with thread counts greater than one, random competition and querying resulted in valid verification rates as low as 54% (e.g. when four threads were tested simultaneously without unpredictable delays). Additionally, under conditions of unpredictable delay, the valid verification rates of both Apache JMeter and the baseline method (which was implemented following JMeter principles) do not apply because their test task scheduling mechanisms are designed to verify the test payload as soon as it is sent, resulting in JMeter and the baseline approach almost to have each thread verify a test payload sent by another thread earlier in the test. Locust, due to its sending mechanism, achieved a verification rate of approximately 66.67%. As it can be seen, the proposed method outperforms the other methods.

C. Evaluation of Successful Verification Count

The experimental results for the successful verification count are shown in Figure 4. The results show that the proposed method outperforms all the competitors in tests with different delay ranges. Specifically, with no additional unpredictable delays, the proposed method achieved a 0.87xincrease in verification count compared to the baseline group when four threads are used and the data length is 8000 bytes. It achieved a 2.6 fold increase compared to Apache JMeter and a 12.24 fold increase compared to Locust. Under conditions of unpredictable delay, both JMeter and the baseline method had a verification rate of 0, resulting in zero successful verification. Compared to Locust, which had a non-zero verification

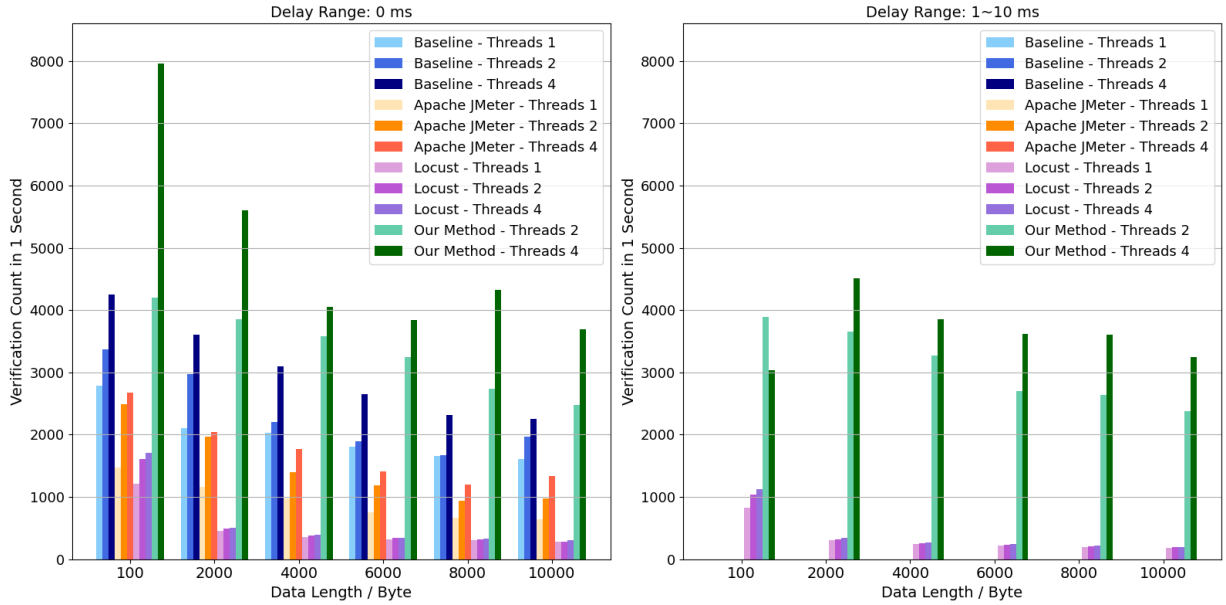

Fig. 4: Comparison of Successful Verification Count (Left: delay = 0ms, Right: delay = 1 $\sim$ 10ms).

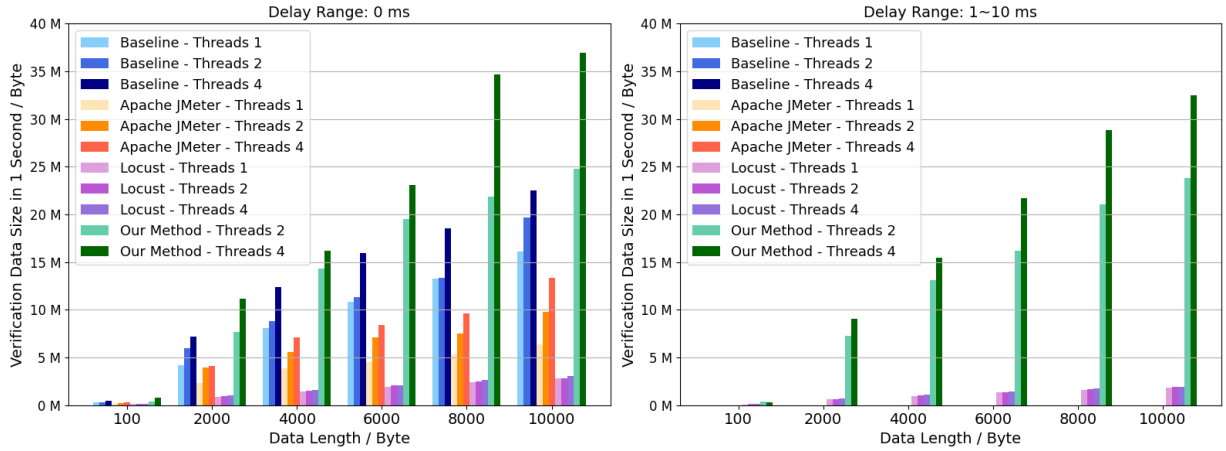

Fig. 5: Comparison of Data Verification Speed

rate, the proposed method achieved a 15.63 fold increase in verification count at the test payload size of 10000 bytes.

D. Evaluation of Data Verification Speed

The experimental results of data exchange speed are shown in Figure 5. As it can be seen, the proposed method yielded higher data verification speeds with two threads compared to the other methods which even enjoyed using four threads, regardless of the delay range. Since sending speed and data packet size are proportionally related, performance improvement rates are consistent with successful verification counts. With four threads and a 10000-byte payload, the proposed method achieved an additional 23.53MB/s in data verification speed compared to JMeter without unpredictable delays, and an additional 30.53MB/s in data verification speed compared to Locust with unpredictable delays.

VI. Conclusion

This paper introduces a testing and verification framework under a distributed crowdsourced platform for ICPS security testing. The proposed testing and verification framework supports complete test verifications even in distributed environments with unpredictable delays induced by data processing or network instability between test and verification ports. In future work, we will further explore the test and verification framework to support dynamic test templates and dynamic construction of test payloads based on the verification results, which could enhance the framework capabilities, e.g., to support fuzzy testing of disordered responses in industrial cyber-physical systems.

Acknowledgment

This article is supported by the National Key R&D Program of China under project 2023YFB3107301.

References

[1] D. G. Pivoto, L. F. de Almeida, R. da Rosa Righi, J. J. Rodrigues, A. B. Lugli, and A. M. Alberti, "Cyber-physical systems architectures for industrial internet of things applications in Industry 4.0: A literature review," *Journal of Manufacturing Systems*, vol. 58, no. PA, pp. 176–192, 2021.

[2] K. Wu, J. Xu, and M. Zheng, "Industry 4.0: review and proposal for implementing a smart factory," *International Journal of Advanced Manufacturing Technology*, vol. 133, no. 3-4, pp. 1331–1347, 2024.

[3] Zion Market Research, "Cyber-Physical Systems (CPS) Market Size, Share, Trends, and Research 2030," 2024. [Online]. Available: https://www.zionmarketresearch.com/report/cyber-physical-systems-market

[4] X. Feng, X. Zhu, Q. Han, W. Zhou, S. Wen, and Y. Xiang, "Detecting vulnerability on iot device firmware: A survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, pp. 25–1, 2023.

[5] M. Sayad Haghighi, F. Farivar, A. Jolfaei, A. B. Asl, and W. Zhou, "Cyber attacks via consumer electronics: Studying the threat of covert malware in smart and autonomous vehicles," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 825–832, 2023.

[6] Y. Jiang, S. Wu, R. Ma, M. Liu, H. Luo, and O. Kaynak, "Monitoring and Defense of Industrial Cyber-Physical Systems Under Typical Attacks: From a Systems and Control Perspective," *IEEE Transactions on Industrial Cyber-Physical Systems*, vol. 1, pp. 192–207, 2023.

[7] M. Benmalek, "Ransomware on cyber-physical systems: Taxonomies, case studies, security gaps, and open challenges," *Internet of Things and Cyber-Physical Systems*, vol. 4, no. September 2023, pp. 186–202, 2024.

[8] G. Lin, S. Wen, Q. L. Han, J. Zhang, and Y. Xiang, "Software vulnerability detection using deep neural networks: A survey," *Proceedings of the IEEE*, vol. 108, pp. 1825–1848, 10 2020.

[9] K. Zhang, Y. Shi, S. Karnouskos, T. Sauter, H. Fang, and A. W. Colombo, "Advancements in Industrial Cyber-Physical Systems: An Overview and Perspectives," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 716–729, 2023.

[10] J. Zhang, L. Pan, Q. Han, C. Chen, S. Wen, and Y. Xiang, "Deep learning based attack detection for cyber-physical system cybersecurity: A survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, pp. 377–391, 2022.

[11] X. Zhu, S. Wen, S. Camtepe, and Y. Xiang, "Fuzzing: A survey for roadmap," *ACM Computing Surveys*, vol. 54, pp. 1–36, 9 2022.

[12] D. Shi, L. Kou, C. Huo, and T. Wu, "A CAN Bus Security Testbed Framework for Automotive Cyber-Physical Systems," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[13] J. M. Riquelme-Dominguez, F. Gonzalez-Longatt, A. F. S. Melo, J. L. Rueda, and P. Palensky, "Cyber-Physical Testbed Co-Simulation Real-Time: Normal and Abnormal System Frequency Response," *IEEE Transactions on Industry Applications*, vol. 60, no. 2, pp. 2643–2652, 2023.

[14] F. A. Batarseh, A. Kulkarni, C. Sreng, J. Lin, and S. Maksud, "ACWA: an AI-driven cyber-physical testbed for intelligent water systems," *Water Practice and Technology*, vol. 18, no. 12, pp. 3399–3418, 2023.

[15] R. J. Somers, J. A. Douthwaite, D. J. Wagg, N. Walkinshaw, and R. M. Hierons, "Digital-twin-based testing for cyber–physical systems: A systematic literature review," *Information and Software Technology*, vol. 156, no. December 2022, p. 107145, 2023.