IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 23, 2021, accepted October 23, 2021, date of publication November 11, 2021,
date of current version November 24, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3127515*

# SECS/GEMsec: A Mechanism for Detection and Prevention of Cyber-Attacks on SECS/GEM Communications in Industry 4.0 Landscape

**SHAMS UL ARFEEN LAGHARI[1], SELVAKUMAR MANICKAM[1], AYMAN KHALLEL AL-ANI[2], SHAFIQ UL REHMAN[3], AND SHANKAR KARUPPAYAH[1]**

[1]National Advanced IPv6 Centre, Universiti Sains Malaysia (USM), Gelugor, Pulau Pinang 11800, Malaysia
[2]Faculty of Computing and Informatics (FCI), Universiti Malaysia Sabah, Kota Kinabalu, Sabah 88400, Malaysia
[3]Amity Institute of Information and Technology (AIIT), Amity University Rajasthan (AUR), Jaipur 303002, India

Corresponding author: Selvakumar Manickam (selva@usm.my)

This work was supported and fully funded by SanDisk Storage Malaysia Sdn. Bhd. (Company No. 955766-P) under Grant 308.PNAV.41560111.

**ABSTRACT** Industry 4.0 as a driving force is making huge strides, particularly in the manufacturing sector, where all integral components involved in the production processes are getting digitally interconnected. Fused with improved automation and robotics, machine learning, artificial intelligence, big data, cloud computing, and the Internet of Things (IoT), this open network interconnectivity makes industrial systems increasingly vulnerable to cyber-attacks. While the impacts and intentions of cyber-attacks vary, they always have a detrimental effect on manufacturers, including financial losses, supply chain disruption, loss of reputation and competitiveness, and theft of corporate secrets. Semiconductor Equipment Communication Standard/Generic Equipment Model (SECS/GEM) is a legacy Machine-to-Machine (M2M) communication protocol used profoundly in the semiconductor and other manufacturing industries. It is mainly designed to be utilized in a controlled and regulated factory environment separated from external networks. Industry 4.0 has revolutionized the manufacturing industry and has brought SECS/GEM back to the limelight as it lacks security safeguards to protect against cyber-attacks. This paper proposes a digital signature-based security mechanism that offers authentication, integrity, and protection against cyber-attacks. The proposed mechanism is compared with the industry-standard SECS/GEM implementation in terms of processing time, payload overhead, and resilience against cyber-attacks. The results indicate that SECS/GEMsec effectively prevented untrusted entities from establishing communication links with legit industrial equipment while maintaining message integrity by discarding forged messages. Additionally, it protected SECS/GEM communications against Denial-of-Service (DoS) attacks, Replay attacks, and False-Data-Injection-Attack (FDIA) attacks.

**INDEX TERMS** Cybersecurity, DoS-attack, IIoT, industry 4.0, M2M, machine-to-machine communications, SECS/GEM.

## I. INTRODUCTION

In the modern manufacturing environment, information and communication technologies have transformed into a force to be reckoned with, becoming a powerful asset in the manufacturing industry. Industries are embracing digital transformation in the way manufacturing is carried out to remain competitive in the world of advanced manufacturing [1], [2]. According to experts, the level of integration of man and machine is quite high, and this process is only in its initial stage. The result of this integration is the emergence of cyber-physical systems (CPS). CPS integrates the cybernetic principle, computer hardware, and software technologies. This includes robotics, cloud computing, 5G networks, big data analytics, machine learning, IoT, and integrated manufacturing, qualitatively integrating new mechanisms into the manufacturing environment capable of perceiving changes, self-learning, adapting,

The associate editor coordinating the review of this manuscript and approving it for publication was Jemal H. Abawajy.

and reacting to them [3]. The key idea of such systems is the integration of physical space and cyberspace. The ability to perceive the environment and adapt to it is another feature of CPS.

Although the line separation between Operational Technology (OT) and Information Technology (IT) is disappearing, the emphasis is still on protecting the OT assets [1]. The protection of IT assets is mostly still not considered important enough, especially the cybersecurity aspects. Many would think that manufacturing is a closed environment, and so it is protected from cyber-attacks. FIGURE 1 shows the number of reported cyber-incidents globally in 2020, with the manufacturing sector suffering from 67 incidents and expected to grow significantly [4]. Cybercriminals are well-informed of the vulnerable IT and networking assets that exist in the manufacturing environment. Due to the failure of manufacturers to take heed of the cybersecurity issues and address them, cybercriminals are finding it easy to infiltrate such networks.

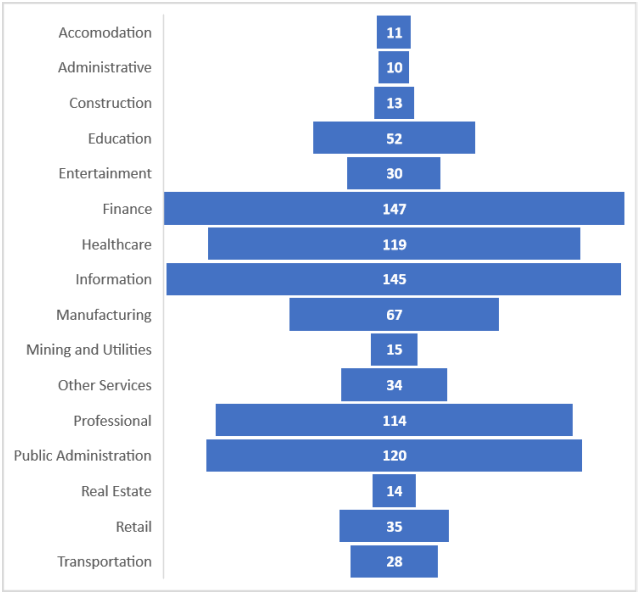


**FIGURE 1.** **Industry sectors with the most cyber-incidents globally.**

Recent cyber-incidents have reached a dangerous level in the manufacturing industry, making it a highly vulnerable and targeted sector [5], [6]. According to a recent survey by the Engineering Employers Federation (EEF), 48 percent of manufacturers have been exposed to a cyber-incident at some point, half of which caused financial loss or negatively affected the market. Similarly, according to a study performed by Cyber Security Ventures, cybercrimes would cost companies across the world $10.5 trillion per year by 2025, representing a significant rise from the $3 trillion per year estimated in 2015 [7]. While manufacturing production has picked up rapidly in recent years, the Verizon data breach investigation report 2019 described 352 cyber-incidents, of which 87 were among manufacturers. Taiwan Semiconductor Manufacturing Company (TSMC) malware attack is the biggest security breach in the history of Taiwan. It exposed the cybersecurity vulnerabilities in manufacturing environments, as this sector embraces Industry 4.0, with increased automation and networked communication [8].

SECS/GEM is an industry standard that has been widely used in virtually every semiconductor industry for several years, including surface mount technology, electronics assembly devices, photovoltaic, and solar cell manufacturing. Besides the fundamental capabilities, SECS/GEM provides various additional capabilities that enable the rapid transformation of traditional factories into smart ones via M2M communication, automation, and real-time data acquisition for monitoring, control, and analytics. It cannot be denied that the SECS/GEM interface is becoming an increasingly important requirement for newly built industrial equipment. Nevertheless, it is also a bitter truth that SECS/GEM is absolutely devoid of security features and hence absolutely vulnerable when utilized in an Industry 4.0 environment without the required safeguards and security measures. This is because once cybercriminals overcome an industrial network's frontline defenses (such as firewalls), they get easy access to SECS/GEM machines and can do anything they want. Thus, the purpose of this study is to secure SECS/GEM operations by authenticating the communicating entities, preventing alterations to the message in transit, and strong defense against cyber-attacks.

This work addresses the security issues posed to SECS/GEM communications and proposes SECS/GEMsec mechanism that offers a modest level of security to protect against cyber-attacks. The contributions of this work are highlighted as under:

- The first contribution is a proof of concept in which cyber-attacks are conducted against typical SECS/GEM communications, demonstrating the vulnerabilities of SECS/GEM processes against cyber-attacks.
- Second, the authentication of SECS/GEM entities is achieved by employing RSA and hashed signatures.
- The integrity of the message is preserved throughout transmission, thus avoiding various cyber-attacks on SECS/GEM communications, which is our third contribution.
- The fourth contribution is that the proposed mechanism effectively protects SECS/GEM communications against DoS, replay, and FDIA attacks.

The remainder of the paper is structured as follows: Section II discusses the literature review and associated studies in detail. Section III covers the fundamental principles and characteristics of SECS/GEM procedures. Section-IV presents proof-of-concept of cyberattacks and discusses techniques for attacking SECS/GEM devices in an industrial network. Section V discusses the proposed mechanism in detail. Section VI analyzes and elaborates on the findings and experiments. Finally, Section VII summarizes the results and discusses future work.

## II. LITERATURE REVIEW

Within the prospects of Industry 4.0, cybersecurity is essential in protecting businesses from losing their competitive edge. However, the emphasis of cybersecurity in the manufacturing sector was, until recently, to protect enterprise perimeters, i.e., prevent unauthorized access to the production network. A number of industrial and IIoT communication protocols are available such as SECS/GEM, Modbus, Message Queuing Telemetry Transport (MQTT), Open Platform Communications – Unified Architecture (OPC UA), Constrained Application Protocol (CoAP), Data Distribution Service (DDS), and many more. Cybersecurity was not a primary consideration throughout the development of these protocols since they were designed mainly for use in closed, air-gapped and trusted industrial networks. Industry 4.0 as a driving force requires interconnectivity with industrial networks to access real-time equipment/machine data whenever and wherever needed. The security of these protocols against cyber-attacks must, therefore, be assured.

Because Industry 4.0 has become a driving force for the manufacturing industry in recent years, the protocols mentioned above have been brought to public attention. Researchers are developing solutions that will make these protocols secure for communications in the Industry 4.0 environment. Due to the fact that these protocols were designed with a single objective in mind, performance above all else, the majority of these protocols lack security measures and are therefore vulnerable to cyberattacks. The underlined sections provide a succinct overview of the security features offered by these protocols.

### A. MESSAGE QUEUING TELEMETRY TRANSPORT

MQTT has been a de facto IoT standard protocol for M2M communication and offers an open-source, lightweight, publish/subscribe model. It thrives in low bandwidth and high latency network conditions. It is suitable for devices with limited computational power, memory, storage, and battery backup. It has fascinated academia and the industry to carry out analysis on security issues as well as the research on defensive solutions [9].

MQTT is highly vulnerable to cyber-attacks because its modus operandi is primarily responsible for security issues [10]. Its design is made for resource-constrained devices and is meant to be lightweight; therefore, it communicates data as plaintext instead of performing any encryption on the header or payload. Accordingly, if encryption is to be done on the header or payload, such as by Transport Layer Security (TLS), it must bring an increased computational overhead on an already feeble device. Moreover, Link Control Message is used by various MQTT brokers to provide authentication. Furthermore, copious security solutions have been proposed to resolve security issues faced by the MQTT [9], [11], [12].

### B. OPEN PLATFORM COMMUNICATIONS – UNIFIED ARCHITECTURE

The OPC Foundation is an industry consortium that develops and maintains standards for open connectivity of industrial tools and systems [13]. The OPC-UA protocol standard, developed by the OPC Foundation, is the most widely used M2M communication protocol specification for the automation industry [2], [14]. In control and automation applications, OPC offers a technology that supports interoperability and heterogeneity. It is mostly used in the production of electronic components for industrial applications. OPC UA is designed with security in mind, and as a result, it has a broad range of fundamental security features, such as authentication, integrity, confidentiality, and authorization. In addition to various security modes (i.e., integrity, both confidentiality as well as integrity, etc.) that define encryption and digital signature processes in order to establish a secure channel between the two communicating entities, OPC UA offers a number of other features such as redundancy, heartbeat, buffering, binary transport, etc., [14]. Depending on the message protection mode chosen by the client, the client negotiates a security protocol to employ in order to secure the messages transmitted during the initial handshake. Among seven security policies offered by OPC, only one offers adequate security protection, and two have been abandoned due to the usage of cryptographic primitives that have been proven to be vulnerable to cyber-attacks.

Although OPC UA offers robust security features, strict security configurations must be implemented to function correctly; otherwise, attackers may be able to get access to sensitive information via port stealing, eavesdropping wireless communications, etc. [14], [15].

### C. CONSTRAINED APPLICATION PROTOCOL (CoAP)

Another application protocol specifically designed for resource-constrained networks and devices is CoAP [16], specified in RFC 7252 [17]. It runs over User Datagram Protocol (UDP) by obeying REST-Architecture and acts alike to HTTP. To secure communication, CoAP does not use TLS since it runs over UDP, which is an insecure transport protocol, so Datagram Traffic Layer Security (DTLS) is used by CoAP instead of using TLS as a security solution.

As UDP connections are bare and unreliable, some modifications were made upon CoAP based upon TLS protocol to create a security-enabled variant termed as CoAPS. The disadvantages like missed or out-of-order packets and link terminations have been redressed through a few enhancements in TLS. The handshake here is alike in TLS. Though, there is a likelihood of retransmission of handshake messages. There is a strong possibility that the server sends a verification query to confirm that the client machine sent its 'hello' message from an authentic source address. Nevertheless, Denial of Service (DoS) attacks are prevented through this additional feature.

### D. DATA DISTRIBUTION SERVICE (DDS)

DDS [18] is an M2M protocol that is, like MQTT, based on the publish/subscribe model and is designed to provide efficient data transmission capabilities to real-time systems. DDS is implemented to function on both Transmission Control Protocol (TCP) and UDP, offering two security algorithm options for DDS, i.e., TCP with TLS or UDP with DTLS. These algorithms are not meant for resource-constrained devices as they are computationally heavy; To address this issue, the Object Management Group (OMG) proposed a DDS security specification that describes a robust security framework that is considered optimal for IoT devices.

### E. REALIZATION OF SECURITY FEATURES IN SECS/GEM

Unlike all the protocols described earlier, SECS/GEM does not have built-in security capabilities [3]. High-Speed SECS Message Services (HSMS) is SECS/GEM's messaging protocol and is implemented over TCP, which defines no security mechanism. It does not provide verification of the connecting entity; No certification or credentials are required. The payload is transmitted in plaintext and does not involve any form of end-to-end encryption. Messages are obfuscated by simply encapsulating the data using a binary encoding process to make it cryptic when viewed by a human. However, anyone with a basic knowledge of binary encoding and some understanding of SECS/GEM can easily decipher the message and extract the data.

The criticality of cybersecurity issues in manufacturing has been recognized recently, and several studies have been carried out to recommend appropriate security mechanisms for Industry 4.0 and IIoT [19]. Relying on devices participating in the IIoT network is crucial to the smooth functioning of the network. A single hacked node may become malevolent, bringing the whole system to a halt or causing catastrophes. Therefore, it is vital that the equipment and machinery interacting in the IIoT environment establish a reliable relationship and communicate only with trusted and authorized devices. Various studies address cybersecurity issues in the industry and propose authentication mechanisms as a potential solution.

Esfahani *et al.* [20] have proposed a protocol for a lightweight authentication mechanism for M2M communication based on Hash and XOR operations. The proposed mechanism achieves authentication in two stages (a) registration phase (b) authentication phase. The registration phase registers sensors with the Authentication Server (AS), and the routers are provided with secure pre-shared keys generated by the AS. In the authentication phase, both the routers and sensors mutually authenticate each other. The proposed mechanism [20] requires an authentication server to authenticate entities, whereas SECS/GEM is a point-to-point protocol, meaning that the equipment configured in passive mode can communicate with only one host at a time. Therefore, the proposed mechanism cannot be employed to authenticate SECS/GEM equipment.

Karati *et al.* [21] proposed a certificate-less signature (CLS) scheme based on the bilinear pairing to provide authentication of information in IIoT systems. In this scheme, the signer requires two exponentiations during the signature generation process. The verifier, on the other hand, requires two exponentiations with one pairing computation to verify a signature. The authors Y. Zhang *et al.* [22] demonstrated that by presenting four types of falsified signature attacks, the CLS scheme [21] does not achieve the stated security features as claimed. Therefore, the research [22] recommends enhancements to [21] by introducing a Robust Certificateless Signature (RCLS) scheme based on elliptic curve cryptography. In addition to robustness, RCLS [22] also offers protection against four types of signature forgery attacks that are not discussed in [21]. Moreover, W. Yang *et al.* [23] have claimed RCLS [22] to be insecure by showing that an attacker having the capabilities of replacing a public key can easily impersonate other legitimate users to upload false messages. They showed that this is possible by forging the valid signatures of the victim, so the validity of the data cannot be maintained, as claimed by [22].

K. Mahmood *et al.* [24] proposed a lightweight authentication mechanism based on a hybrid Diffie-Hellman approach that employs AES and RSA for generating session keys. The scheme offers mutual authentication, preventing replay and Man-in-the-Middle (MITM) attacks while achieving message integrity. To guarantee the message is cryptographically safe, the benefit of a cryptographic hash-based message authentication code is used. However, the use of a public-key encryption scheme and dependence on Certificate Authority (CA) increases the overall communication and computation overheads.

Mumtaz et al [25] have devised an authentication mechanism based on RSA public-key cryptography for the IoT ecosystem using cutting-edge industry standards. In conjunction with a proxy-based security service provider, the proposed mechanism offers security services such as X.509 certificate, RSA-based Public Key Infrastructure (PKI), challenge/response protocols, among other related services. The method exhibits a novel system model, protocol design, architecture, and threat evaluation against known adversaries. The proposed mechanism was chosen to be developed as an add-on service for a range of additional critical applications such as smart cities, cyber-physical systems, and so on that require X.509 certificates based on hard tokens. The add-on service model enables the proposed mechanism to be used in conjunction with other security services such as privacy, integrity, confidentiality, non-repudiation, and anonymity of the identities.

T. Shah *et al.* [26] have presented a multi-key-based mutual authentication mechanism. In this approach, the shared secret between the IoT server and the IoT device is called a secure vault, which is a collection of equal-sized keys. Initial contents of the secure vault are shared between the server and the IoT device, and contents of the secure vault change after every successful communication session.

S. F. Aghili *et al.* [27] address the security vulnerabilities of contemporary M2M authentication protocols proposed for IIoT networks in order to protect against numerous cyberattacks, including DoS attacks, router impersonation attacks, and smart-sensor traceability attacks. The research conducted demonstrates that a compromised smart device may acquire the secret key of the router and the session key, which another smart device is using to establish a secure channel with the router.

E. Lara *et al.* [28] addressed issues of resource-constrained IoT devices and proposed an authentication protocol for IIoT networks. The proposed mechanism is believed to be lightweight and makes use of basic operations such as XOR, addition/subtraction, and the hash function to accomplish its intended design objectives. In order to authenticate the communicating network entities, the proposed mechanism requires just four messages to be exchanged between the principals. Using the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool and Burrows–Abadi–Needham (BAN) logic, it was successfully evaluated for security, and an informal study of its resistance to known attacks was also conducted and found to be successful.

KK. Kolluru *et al.* [29] address the authentication and message integrity issues with heterogeneous devices in the manufacturing industry. The authors discuss the Service-Oriented Architecture-Based (SOA) Arrowhead Framework, which was previously proposed using the concept of local clouds. The local clouds provide a set of mandatory and support core systems to enable industrial automation applications. One of these mandatory core systems is an Authentication, Authorization, and Accounting (AAA) system used to authenticate and provide access control to the devices in a local cloud. In an industrial context, with multiple stakeholders, the AAA must support fine-grained access control. The proposed mechanism is Next Generation Access Control (NGAC)-based AAA solution to achieve fine-grained service-level access control between IoT devices and machines in an industrial network.

The authentication mechanisms that use a digital signature-based algorithm have dependencies on Certification Authorities, multiple-key exchange mechanisms; thus, they increase the complexity of authentication mechanisms mentioned above. Moreover, as the connection is point-to-point and can persist for weeks, a Certificate Authority is rendered as unnecessary [30], [31]. The more complex the mechanisms are, the more resources, i.e., bandwidth and processing time, will be required to complete the process or operations. The studies [23] and [27] showed that the mechanisms discussed above introduce vulnerabilities, and their defense mechanism is broken. In other words, the adaptation of the security mechanism presented above will enable attackers to target the weaknesses of existing mechanisms and launch attacks such as DoS attacks, impersonation attacks, and replay attacks on the SECS/GEM communications. This would result in disrupted network connectivity, theft of confidential data, and damage to reputation.

## III. SECS/GEM PROCESSES & FEATURES

Semiconductor Equipment and Material International (SEMI) is an association that has members of more than 2000 organizations globally [32]. It deals with products, equipment, and services needed by manufacturing industries. SEMI has released various specifications that manage communication between host and factory equipment, such as the SEMI E4, E5, E30, and E37 standards. These standards are collectively known as SECS/GEM. The SECS/GEM protocol is an industry standard that is widely in operation across many manufacturing industries worldwide [3], [33], [34]. It acts as a backbone of the semiconductor industry and is heavily used in the world's leading enterprises, including Intel, Samsung, TSMC, IBM, Qualcomm, Broadcom, UMC, SK Hynix, Micron, TXN, Toshiba, NXP, proving as a de facto communication protocol and control system since decades [3].

HSMS serves as a transport protocol for SECS/GEM communications within industrial semiconductor networks [35]. HSMS is a rudimental derivation from Transmission Control Protocol / Internet Protocol (TCP/IP) by fundamentally applying the same methods of creating a link as specified in RFC-793 with minor modifications [36]. However, it discerns between active and passive connection modes as against the guidelines prescribed in RFC 793; wherein, either party could initiate communication. The server is configured in a passive mode and opens a port for listening to incoming connections, whereas the actively configured devices are responsible for initiating a connection. On establishing the successful connection between host and equipment, the HSMS protocol transmits binary encoded SECS-II messages. The connection is kept alive for transmitting data until desired by either side or purposely rendered offline (i.e., firmware or software updates, add/remove machines in the production line, maintenance, and so on). The HSMS's message format is shown in FIGURE 2. The length of the message is indicated by the Length Bytes, which embraces the header and payload. SessionID is a 16-bit unsigned integer that uniquely identifies a session between specific session entities (i.e., usually a host and equipment). The SECS/GEM messages are of numerous categories associated with a distinct set of activities; for example, streams 1, 3, and 7 correspond to equipment status, material status, and recipe management, respectively, whereas the functions are specific messages within a particular stream category [3]. The most significant bit (MSB) in the header field Stream, denoted by W, is used to specify whether or not the response to the request message is needed (i.e., MSB = 1 indicates that a reply is required). The PType

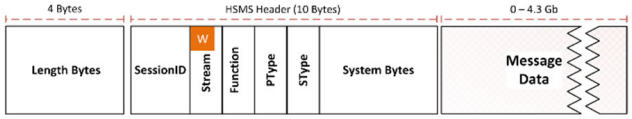


**FIGURE 2.** HSMS message structure.

(presentation type) field is used to provide the encoding method for control or data messages, whereas the SType field specifies whether the message is a control or a data message (i.e., SType will always be zero for data messages). The SystemBytes header field is a monotonically increasing integer number that is used to associate request and response messages; each pair of request and response messages will have the same value of SystemBytes.

The SECS/GEM messages are binary encoded and transmitted as a stream of bytes; wherein, the first four bytes represent the length of the message. The minimum and maximum sizes of a message allowed on the HSMS protocol are 10 bytes and 4.3 gigabytes, respectively.

### A. SECS/GEM MESSAGE TYPES

SECS/GEM messages may be classified into two different types, namely, control messages and data messages. As the name implies, the control messages are used for establishing and maintaining a communication link between a host and the equipment. On the other hand, data messages are application-specific and are used to control operations and offer real-time insights into industrial equipment. The header field SType is used to differentiate between the control messages and data messages; for example, if SType has a non-zero value, the message is considered a data message; otherwise, it is regarded as a control message. Based on the SType values, the specifics and purpose of control and data messages are given in TABLE 1.

**TABLE 1. The purpose and values of the SType header field.**

| Command | SType Value | Description |
|---|---|---|
| Data | 0 | Data (SECS-II) messages |
| Select.req | 1 | Request to establish a connection |
| Select.res | 2 | Response to link establishment request |
| Deselect.req | 3 | Request to end communication |
| Deselect.res | 4 | Response to end communication |
| Linktest.req | 5 | Periodic heartbeat to verify the link |
| Linktest.res | 6 | Response to ascertain the link status |
| Reject.req | 7 | Response to not supported valid message |
| - | 8 | Not used |
| Separate.req | 9 | End communication unilaterally |
| - | 10 | Not used |
| - | 11-127 | Reserved for subsidiary standards |
| - | 128-255 | Reserved, not used |

### B. SECS/GEM CONNECTION STATES

It is paramount to understand the SECS/GEM connection status to establish a connection with the equipment. At the basic level, there are two states, CONNECTED and NOT-CONNECTED. As the name implies, NOT-CONNECTED state refers to an entity that listens on the TCP port and waits for connection requests or all existing connections being terminated. This means if an entity is in a NOT-CONNECTED state, it does not initiate any connection. Once a successful TCP connection is established with the host endpoint, the status changes from NOT-CONNECTED to CONNECTED. The two NOT-SELECTED and SELECTED sub-states are in a merged state. Once the entities enter the CONNECTED state, they will now wait for HSMS link establishment requests.

When the entity gets a connection setup request through an HSMS control message, the entity's status switches to the SELECTED state. SECS-II and related messages can now be transmitted between the entity and the host endpoints when this state change happens. FIGURE 3 shows the HSMS state model.

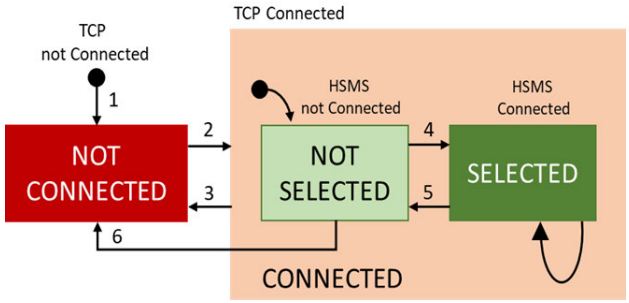


**FIGURE 3. HSMS connection states [1].**

### C. HSMS COMMUNICATION PROCESSES

It is essential to control and monitor SECS/GEM messages exchanged between the host and the equipment. Requests for data can happen on either endpoint of the connection in a typical communication mode. Relevant data is generated depending on the request and sent by the target entity to the requesting entity.

In FIGURE 4, various communication processes in SECS/GEM are shown. The entity configured in active mode (i.e., typically the host) must send a TCP request to the

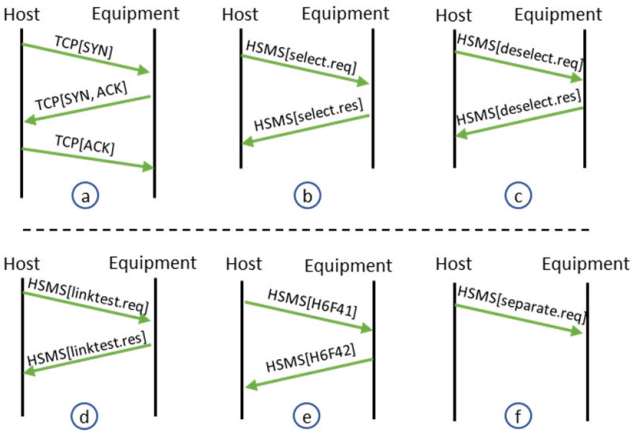


**FIGURE 4. SECS/GEM's connection establishment, control and data message processes.**

equipment entity that is configured in passive mode to initiate communication. Upon receiving a response from the equipment, the host will acknowledge by establishing a TCP connection between the two entities (i.e., FIGURE 4a). The equipments state will change to a CONNECTED state, and it will wait for the initiation of an HSMS connection. Then, the host will request to initiate an HSMS connection, and the equipment will respond to that message and change its state from NOT-SELECTED to SELECTED. Now, the link is complete and ready for SECS-II message exchange (FIGURE 4e). TCP and HSMS connections creation process is illustrated in FIGURE 4a and FIGURE 4b.

The connections within an industrial network are kept alive for several days or perhaps weeks. It is prevalent in industrial networks that either side of the communication has nothing to transmit; thus, testing the communication link before sending a message is necessary. FIGURE 4d shows a situation where idle time is identified when there is no data exchanged between the two entities. In order to keep the HSMS connection alive, the control messages (i.e., linktest.req and linktest.res) are exchanged at regular intervals to ascertain that the connection is not broken.

When it is no longer required to maintain the connection and communication must be ended, HSMS does this via the use of either the deselect.req or the separate.req control messages, depending on the situation. The difference between these two control messages is that deselect.req must wait for a response message from the communicating entity before terminating the connection. In contrast, separate.req can terminate the connection unilaterally without waiting for a response message. FIGURE 4c and FIGURE 4f demonstrate the steps required to teardown a communication link between a host and equipment.

## IV. ATTACKS ON SECS/GEM

FIGURE 5 shows the structure of the production network in an industrial setting. The attacker must be informed of the most recent *SystemBytes* value in order to conduct a successful attack. This is critical because we know that the

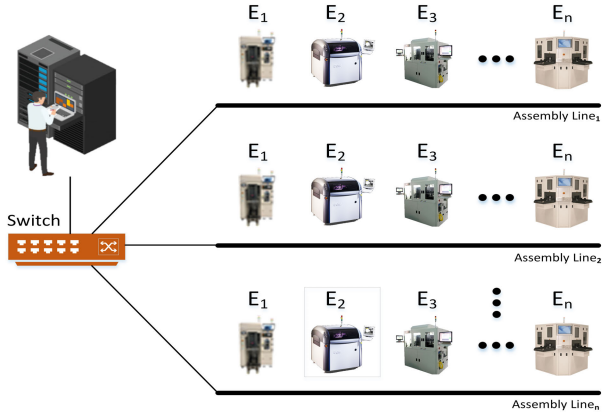


**FIGURE 5.** A simplified and generalized industrial shop-floor network.

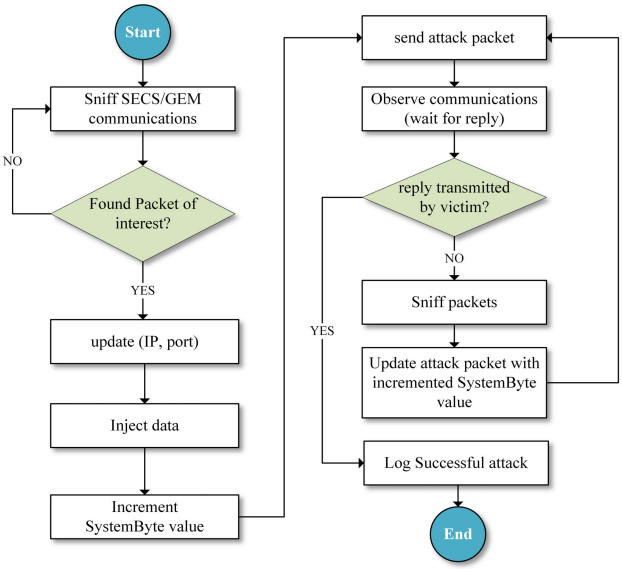


**FIGURE 6.** Flowchart of steps for cyberattacks on SECS/GEM communications.

*SystemBytes* value is increasing monotonically, and if the attacker is aware of this, he may increment it by one and inject his own malicious content into the established connection between a host and equipment.

The flowchart in FIGURE 7 illustrates the procedures necessary to conduct a successful attack. In order to begin, the perpetrator must be capable of sniffing the transmission. Once the attacker has access to the communications, he may inspect packets and choose the most appropriate message for the attack. The attack may be carried out in a variety of ways, but the most effective method is to wait for a control message from the host. Upon receiving the intended control message, the attacker needs to increment the SystemBytes

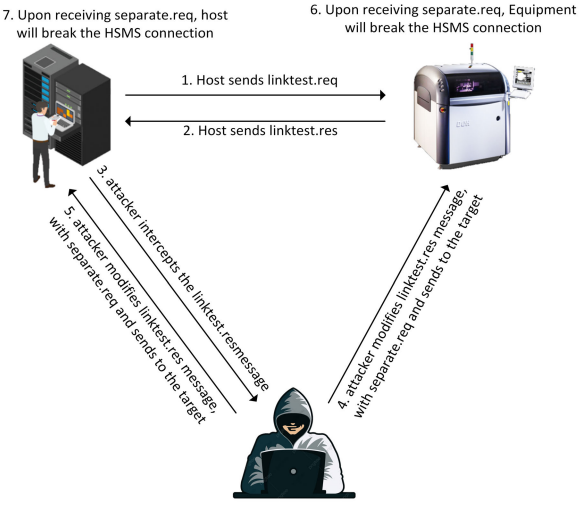


**FIGURE 7.** Scenario of capturing, intercepting & attacking SECS/GEM communications.

value and change the SType value in the intercepted message to transform it into a new control request message to disrupt the communication. Remember that SECS/GEM communications are paired as request/response, with an odd function number indicating a request and an even function number indicating a response message. It is advantageous to wait for a reply message to ensure a successful attack since the reply message is usually returned instantaneously. For example, a request message needs just a single increase in the *SystemBytes* value; yet, the likelihood is that the equipment will respond before the attacker message is transmitted. By contrast, reply messages guarantee that the conversation has concluded and that an attack may be initiated prior to any party sending another message. As a result, an attack based on a response message has a higher probability of success.

### A. DoS ATTACK

DoS is a type of cyberattack that, as its name suggests, aims to deny a particular online service to the intended users by flooding the server with requests, making the service slow or even unreachable. DoS attacks can also be easily carried out on the industrial network, especially those communicating using SECS/GEM [1]. There are many ways to accomplish a DoS attack on SECS/GEM communications. For example, hackers can capture message.req (i.e., any valid control message) from an actual host/equipment, use it to forge a malicious message, and send the forged message.req to the recipient endpoint. After using the port capture method [37], attackers can carry out MITM attack followed through with a DoS attack, causing the host to be unaware of connection termination indefinitely [37].

We used HSMS control messages to execute the DoS attack. We leveraged message.req in this research to perform a successful DoS attack on the HSMS-based interactions. As mentioned in the preceding section, an HSMS message with SType $= 9$ is used to terminate the HSMS connection immediately. Except for the SType value, the separate.req control message is the same as the deselect.req message. Its purpose is to terminate the HSMS connection at once without notifying the host. No reply or acknowledgment message is required.

FIGURE 7 details the modus operandi of the DoS attack on SECS/GEM communications. In most cases, the perpetrator tests the network and attempts to identify nodes listening to port 5000. This is often performed passively on HSMS communication. The attacker monitors the network and captures the most recent response message sent by the victim machine. The probability of an attack in reply messages is higher than in request messages as it typically takes longer to request data successionally.

The most damaging type of attack that may be conducted against SECS/GEM communications is a DoS attack. In this attack, the attacker monitors the network traffic for a message of interest, such as the linktest.req/linktest.res message pair. As soon as the attacker captures a message of this kind, the attacker manipulates it by making slight modifications, such as altering the SType value from 6 to 9 (i.e., changing it from linktest.res to separate.req), and sends it to the target, which causes the targeted entity to terminate the SECS/GEM connection abruptly without waiting for an acknowledgment message. The target entity would treat the received message as legit because there is no security mechanism in standard SECS/GEM implementations that prevent such attacks. As industrial machines are connected sequentially in the production line, the attacker can disrupt communication by targeting different equipment each time in order to avoid suspicion. The result of this type of attack would be devastating as production will stop, and it will be challenging to pinpoint the exact cause of the failure. Once the connection between the host and the equipment is terminated, the attacker immediately sends a connection establishment request to the host in order to retain the connection. Because the equipment can only accept one connection at a time, the equipment will ignore subsequent requests from the legitimate host until the attacker retains the connection.

Knowing that the HSMS protocol is based on the TCP protocol, attackers may leverage vulnerabilities discovered in the TCP connection management process and exploit TCP's RST flag to cause the TCP connection established between a host and the equipment to be terminated. When the equipment receives a TCP RST message, it immediately disconnects the connection and sets the status of the device to NOT-CONNECTED (FIGURE 3). At this stage, the attacker has the chance to initiate a connection setup request to the TCP port address 5000 in order to launch a DoS attack (usually, HSMS is configured on this port). Any request received on this port will be processed by the equipment, which will then establish a TCP connection with the attacker via the three-way handshake protocol. On the other hand, since the host is totally ignorant of the reason for the communication breakdown, it will try to reconnect with the equipment; however, these attempts will be futile because the attacker has already taken control of the network connection. The attacker may keep the connection open for the duration as long as needed, due to which communication will be suspended for an undefined period of time. FIGURE 8 depicts the effective

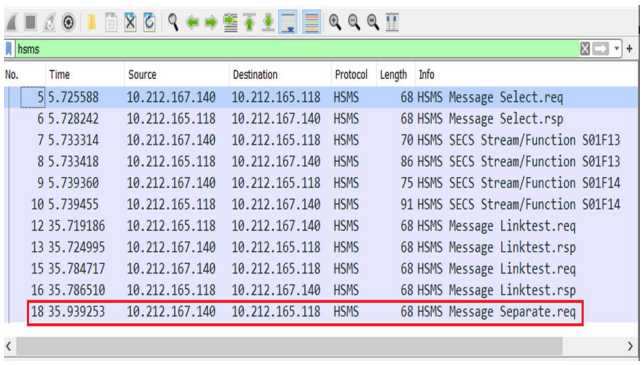


**FIGURE 8.** Wireshark capture of a successful injection of Separate.req attack message.

injection of the Separate.req message that happened during the HSMS connection establishment between the host and the equipment and how the equipment responded by terminating the connection.

Data messages may also be used to conduct a DoS attack against SECS/GEM communications. This condition holds because the HSMS standard restricts the processing of messages to 1Hz (i.e., one message per second). The attacker has the ability to send any large data packets to the equipment in order to limit its processing capability.

### B. REPLAY ATTACK

A replay attack occurs when an attacker monitors ongoing communication between a host and equipment, intercepts it, and then at a later stage, resends (replays) the captured message to the victim for fraudulent purposes. In order to avoid replay attacks, many methods are available, the most frequent of which is to preserve the freshness of the message via the use of timestamp or nonce. Fortunately, the SystemBytes header field is available in SECS/GEM to pair the request and response messages. Every request message must include a unique SystemBytes value, and the corresponding response message must contain the same SystemBytes value as the request message. Although the SystemBytes value may be used to avoid replay attacks, it cannot be utilized to do so for two reasons: (1) SystemBytes is predictable since it is monotonically increasing, and (2) the message is transmitted in plaintext, making the values easy to intercept. SECS/GEM is vulnerable to replay attacks [1].

### C. FALSE-DATA-INJECTION-ATTACK (FDIA)

The concept of an FDIA attack was introduced for the first time in the context of smart-grid networks. Within industrial networks, the attacker takes advantage of the measurements from the equipment so that undiscovered inaccuracies are injected into the equipment's variables, which will ultimately have an adverse effect on the equipment's functioning and the overall manufacturing process. SECS/GEM communications are completely vulnerable to FDIA attacks, and there is no mechanism to guard against these attacks [1].

## V. PROPOSED MECHANISM

The SECS/GEM's specifications do not define any mechanism to detect cyberattacks on SECS/GEM communications. The HSMS is SECS/GEM's transport protocol and describes equipment states and enables the host to interact with the equipment in order to control and monitor operations in real-time. The control part of SECS/GEM enables operators to configure and manage equipment. This feature is extremely useful to interact with the equipment effectively; however, if the communication is compromised, threat-actors may disrupt the communication and launch cyber-attacks, including a DoS attack. Hence, the goals for our research are as follows:

1. To propose a security mechanism for authenticating entities communicating over SECS/GEM.
2. To devise a security mechanism that ascertains message integrity.
3. To design a security mechanism that detects and prevents cyber-attacks carried against SECS/GEM communications.
4. To ascertain that the proposed mechanism is simplistic and must not modify the existing message structure or packet format.
5. To evaluate the performance of the proposed mechanism in terms of processing time, control message overhead, and resilience against cyber-attacks.

In order to achieve the first objective, the proposed mechanism uses the digital signature approach to encrypt the message hash on the sender side and decrypt it on the receiver side. Digital signature algorithms use asymmetric-key cryptography, which means a public key algorithm is employed. The RSA algorithm with key size 2048 is used to encrypt the hash generated on all SECS/GEM messages. The 2048-bit key size is chosen to reduce the control overhead associated with each message transmitted between a host and equipment. A key size of 4096 bits significantly improves security over 2048-bit but at the cost of double the control overhead. SECS/GEM is a point-to-point system; thus, keys are directly installed on the two ends, and there is no need to use a certificate authority for key distribution and verification. The receiver is only allowed to decrypt the received message using the sender's public key; this authenticates the message and ensures that it originated from the correct sender.

In order to achieve the second objective, the proposed mechanism uses SHA-256 to maintain message integrity [38]. SHA-256 was chosen because it has not been broken yet, provides a reasonable level of security, and is fast on 32-bit machines. In our proposed mechanism, we have employed SHA-256 to compute the hash for all messages required to be transmitted. The hash value for the given message is computed using SHA-256, and then this hash is encrypted using the RSA algorithm to generate the message signature. The SECS/GEM header has a four-byte field named *SystemBytes*, which is incremented monotonically with each request message. This field serves two critical functions. First, it maintains message freshness since each new SECS/GEM request message will always contain a new *SystemBytes* value. Second, the *SystemBytes* value in the reply message will always be identical to that in the corresponding request message. Even if an attacker crafts a new message using an incremented *SystemBytes* value, the message will be rejected as the message signature cannot be forged without the private key. Thus, the proposed mechanism accomplishes the third research objective when it detects and discards messages with duplicate, stale, or forged *SystemBytes* values.

In order to achieve the fourth objective, the proposed mechanism takes advantage of TCP flow and appends signature at the end of the message without modifying any field in the message structure. In this manner, the message signature is included in the same TCP payload as the message, eliminating the need for a separate control message to transmit

the hash. In order to append the signature at the end of the HSMS message, we require a message length value that can be extracted from the first 4 bytes of any HSMS message. This simplifies the process and requires no change in the message structure. FIGURE 9 illustrates the pseudocode for the encrypting hash on the sender side.

| Algorithm: Encrypt hash | |
|---|---|
| 1. | Receive a message from HSMS |
| 2. | hash1 = compute hash using SHA-256 (key, message) |
| 3. | signature = encrypt hash1 using sender's private key |
| 4. | Length = Determine message length |
| 5. | message = append signature (message + signature) |
| 6. | Send message |
| 7. | Exit |

**FIGURE 9. Algorithm to hash and encrypt the hash of the message on the sender side.**

Finally, the fifth objective may be accomplished by comparing the processing time and control message overhead of both standard SECS/GEM and SECS/GEMsec implementations. Given the fact that standard SECS/GEM lacks security features, the proposed mechanism's resilience to cyber-attacks is calculated independently.

The functionality of the proposed SECS/GEMsec mechanism is implemented within SECS/GEM itself; therefore, when a message is needed to be sent to the destination entity, the encryption process takes place immediately at that stage. Upon receiving a message from the upper-layer (i.e., SECS-II), the HSMS protocol structures the contents appropriately and is ready to send the message. The transmitting entity must first identify the message length by analyzing the first four bytes of the message received, as shown in FIGURE 9. Following that, the message digest is calculated using SHA-256. The message digest is then sent to the Digital Signature Algorithm, which uses the sender's private key to encrypt the generated hash value. Once the message's hash has been encrypted to generate the signature, the signature is appended to the message and sent to the destination entity.

FIGURE 10 illustrates the process for validating the received message on the destination entity. The first step is to extract the message's signature, which involves inspecting the message's initial four bytes and calculating its length. Following that, the message's signature size is extracted in order to retrieve the signature. It is possible that the transmission was intercepted, and the message was modified or changed during transit. As a result, it is not guaranteed that the message arrived with the precise contents that the sender included; thus, the receiver is required to decrypt the encrypted hash value using the sender's public key in order to ascertain that message has indeed arrived from the legitimate sender.

| Algorithm: Decrypt hash | |
|---|---|
| 1. | Receive message |
| 2. | Determine Message length |
| 3. | signature = extract signature from message |
| 4. | decryptedHash = decrypt signature using sender's public key |
| 5. | hash = compute hash using SHA-256 (message) |
| 6. | ***if*** decryptedHash == hash, ***then*** |
| 7. | accept the message and exit |
| 8. | ***else*** |
| 9. | discard the message and exit |

**FIGURE 10. Algorithm to decrypt and verify hash on the receiver side.**

After the hash value is decrypted, the receiver calculates the hash value for the message itself. The receiver will compare the two hash values and accept the message only if the hash value produced on the receiver side matches with the decrypted hash value transmitted with the message; otherwise, the message will be rejected. The functioning of both the sender and receiver is shown in FIGURE 11.

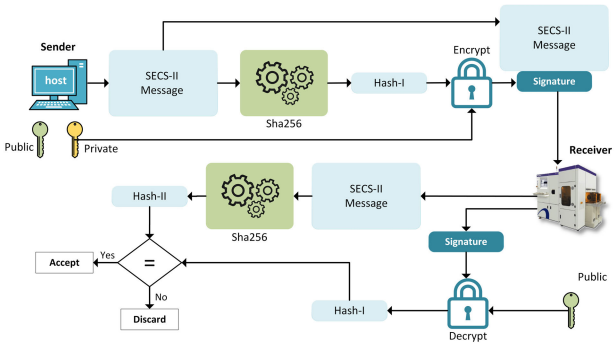


**FIGURE 11. The proposed mechanism to hash the message, encrypt the hash on the sender; on the receiver, decrypt and verify the message.**

## VI. EVALUATION AND DISCUSSION

This study proposes a security mechanism that authenticates communicating entities, ensures message integrity, and successfully prevents various cyber-attacks against SECS/GEM communications in an Industry 4.0 ecosystem, including DoS attacks, replay attacks, and FDIA attacks. The proposed mechanism's processing time, control message overhead, and resilience against cyber-attacks was evaluated, and the findings were compared to those of standard SECS/GEM implementations.

### A. SETTING UP THE TESTBED

SECS/GEMsec is developed in accordance with SECS/GEM standard specifications. The scenarios used to assess the functioning and resilience against cyber-attacks are evaluated in an industrial information technology security laboratory. This is accomplished by installing Python-based SECS/GEM implementations [39] on two separate workstations that act as the host and equipment, respectively. For the experiment scenario, the perpetrators are assumed to have circumvented network security and penetrated the firewall and able to eavesdrop on the conversation and sniff packets exchanged between the host and equipment. FIGURE 12 depicts a typical industrial network architecture with attackers present.

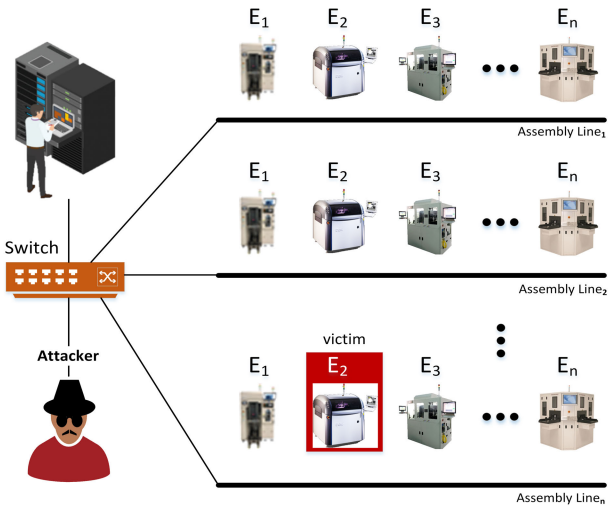


**FIGURE 12.** Testbed environment - an attack scenario.

As previously stated, SECS/GEM is a point-to-point communication protocol that requires at least one of the communicating entities (i.e., the host or the equipment) to be configured in Active-Mode to initiate the connection establishment request. While either of the two entities may be configured in active mode, we have set up the host in active mode. The equipment is usually configured in passive mode (i.e., the entity configured in passive mode will open up a port and listen for the incoming connection requests). The attacker is set up to perform a DoS attack on SECS/GEM communications. The host and equipment specifications are mirrored from the machines recently purchased in 2020 in the semiconductor industry. The specifications of hardware and software used for conducting the experiments are presented in TABLE 2.

**TABLE 2.** Hardware and software requirements for the testbed environment.

| Device Role | Specifications | Operating System |
|---|---|---|
| Host | **CPU:** Intel® i7-9750H @ 2.60GHz x 6<br>**RAM:** 16 GB | Win-10 |
| Equipment | **CPU:** Intel® i5-6500 @ 3.20 GHz x 4<br>**RAM:** 2 GB | Win-10 |
| Attacker | **CPU:** Intel® Ci3-330M @ 2.13GHz x 2<br>**RAM:** 8 GB | Ubuntu 2020.3 |
| Switch | Cisco Catalyst 2960 Fast Ethernet | - |

### B. PROCESSING TIME

This experiment aimed to determine the overall processing time for both the standard SECS/GEM implementations and the proposed mechanism. Due to the fact that the standard SECS/GEM implementations lack security features, its processing time is clearly the shortest. On the other hand, the proposed mechanism requires control information to be piggybacked onto each message exchanged between the two entities, incurring additional control bytes; nevertheless, the benefit is moderately secure communication and protection against cyber-attacks. The total processing time for generating and verifying the request and reply messages have been calculated on both sides. For instance, the host computes the processing time of the request message, and the equipment computes the processing time of the same message upon receipt. The equipment will then calculate the processing time for the reply message, and the host will compute the processing time upon receipt of the reply message. In this manner, the processing time of the request/response message pair is calculated in four different stages of a message lifecycle: once for the host send, once for the equipment receive, once for the equipment send, and once for the host receive.

The total processing time (PT) between a host and equipment is obtained by subtracting the starting time (St) from the ending time (Et) of the generating process (Gp) at sending entity and the verification process (Vp) running on the receiving entity. The SECS/GEM messages to obtain the summation of the Gp and Vp for the four stages of the message lifecycle are shown in Equation-1 [40].

$$PT = Time_{end(Gp|Vp)} - Time_{start(Gp|Vp)} \tag{1}$$

Moreover, there are possibilities that the processing time may be affected by other operating system operations. Thus to ensure the reliability of the results, the experiments were repeated 20 times to get the average processing time.

Based on the experiment results, the processing time of the proposed mechanism is higher than the standard SECS/GEM implementations. This is because computing the message's hash and encrypting the resulting hash into a signature on the transmitting end and computing message hash, decrypting the signature, and verifying them on receiving end has increased the computational burden required to achieve the desired degree of security.

FIGURE 13 and FIGURE 14 depict the processing time for standard SECS/GEM and the proposed SECS/GEMsec mechanism, respectively. Since the standard SECS/GEM implementation lacks security, it is apparent that the standard SECS/GEM algorithm has a low processing time (i.e., less than one millisecond). The time required to process a message is calculated for both control and data messages. Control messages are often composed of just the header; therefore, the message length is typically 10 bytes; however, data messages vary based on the payload and information to be retrieved. The processing time was determined using S01F3/S01F04 and S06F11/S06F12 messages. The processing time of each message is determined for both the sender and receiver. This is because the sender is computing the message hash and encrypting the hash in order for it to be transmitted securely to the destination, whereas it is necessary to compute the processing of the same message on the receiving side in order to evaluate the processing difference between the sending and receiving algorithms. The results indicate that sending messages takes longer to process than receiving messages because the sender is required to do encoding and packaging.

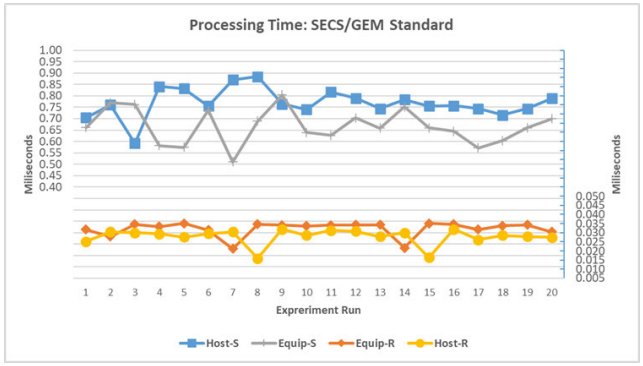


**FIGURE 13.** Standard SECS/GEM: total processing time.

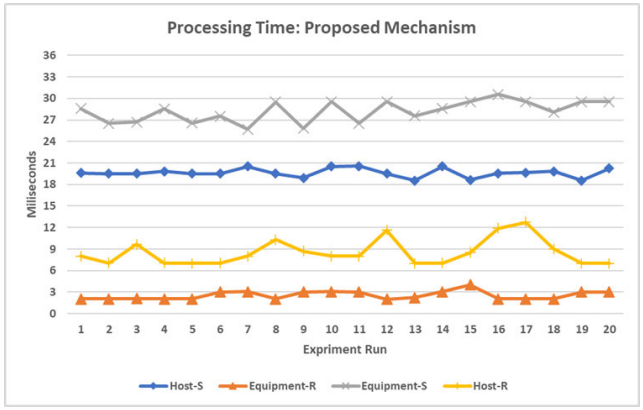


**FIGURE 14.** Proposed mechanism: processing time.

In comparison to request messages, reply messages are often bigger in size, which results in a longer processing time for hashing and encrypting them. The reply messages themselves may be as little as a single header or as big as 4.3GB. It's worth noting that a single data item may be up to 16MB in size and that SECS/GEM messages often include several data items for a requested dataset.

The processing time for the proposed mechanism is shown in FIGURE 14. As anticipated, the processing time for both the request and the response messages is longer than the standard SECS/GEM processing time. This is because the proposed mechanism has the added functionality to calculate a hash value for the given message using the SHA-256 algorithm and then encrypt the hash using the RSA algorithm [41] with a key size of 2048 bits. The message is subsequently added with the encrypted hash and is sent to the intended destination.

TABLE 3 and TABLE 4 illustrate the different statistical values calculated for the standard SECS/GEM and the proposed SECS/GEMsec mechanisms, respectively. The maximum processing time for any type of standard SECS/GEM message is less than half a millisecond. It is obvious it does not incur any overhead because it does not carry data other than the payload related to the SECS/GEM. On the contrary, for the same messages, the proposed mechanism has, on average, 19.1, 2.5, 27.7, and 8.5 milliseconds overhead for the messages Host-Send, Equipment-Receive, Equipment-Send, and Host-Receive (abbreviated as Host-S, Eqp-R, Eqp-S, Host-R), respectively, owing to the security features included to prevent cyber-attacks.

**TABLE 3.** Standard SECS/GEM: processing time (in milliseconds).

| | *Host-S* | *Eqp-R* | *Eqp-S* | *Host-R* |
|---|---|---|---|---|
| **Min** | 0.5412 | 0.0427 | 0.5844 | 0.0405 |
| **Max** | 0.5634 | 0.0439 | 0.6077 | 0.0412 |
| **Mean** | 0.5535 | 0.0432 | 0.5943 | 0.0409 |
| **St.Dev.** | 0.0073 | 0.0004 | 0.0079 | 0.0003 |
| **Overhead** | - | - | - | - |
| **Overhead (%)** | - | - | - | - |

**TABLE 4.** Proposed mechanism: processing time, with overhead (in milliseconds).

| | *Host-S* | *Eqp-R* | *Eqp-S* | *Host-R* |
|---|---|---|---|---|
| **Min** | 18.6 | 2.0 | 25.7 | 7.0 |
| **Max** | 20.5 | 4.0 | 30.6 | 12.7 |
| **Mean** | 19.6 | 2.6 | 28.2 | 8.5 |
| **St.Dev.** | 0.6 | 0.6 | 1.5 | 1.8 |
| **Overhead** | 19.1 | 2.5 | 27.6 | 8.5 |
| **Overhead (%)** | 97.2% | 98.3% | 97.9% | 99.5% |

### C. CONTROL MESSAGE OVERHEAD

This section discusses the traffic overhead incurred by the proposed mechanism for added features of preventing cyber-attacks on SECS/GEM communications. As stated previously, SECS/GEM packages data with high density and incurs little control overhead. The traffic overhead was calculated by measuring the total message size and the size of the message's signature. The signature's size depends on the key size; for example, the signature computed with key size 2048 bits would incur 256 bytes with every message exchanged between the two nodes. Usually, the request messages have small message sizes, often comprised of the only header, in which case the message size is only 10 bytes. Every 10 bytes header-only message incurring 256 bytes of message signature looks overwhelming; however, the payoff is obvious, which is secure communication.

TABLE 5 and TABLE 6 show the message size and the traffic overhead of the proposed mechanism with key sizes 2048 and 4096 bits. The traffic overhead of SECS/GEM was calculated on both data as well as control messages. Obviously, there is no overhead for the standard SECS/GEM as it does not provide any security feature; thus, the data packaging density is highest. However, the overhead is incurred with the proposed mechanism because, along with each SECS/GEM message, the message signature is appended to prevent modifications and attacks. The overhead is observed to be diminishing as the message size increases and therefore has a negligible impact on the performance.

**TABLE 5.** Control overhead with RSA key size = 2048 bits.

| | *Message Size (bytes)* | *Key size (bytes)* | *Total* | *Control Overhead (%)* |
|---|---|---|---|---|
| **Control msg header-only** | 10 | 256 | 266 | 96.241% |
| **Data msg (1KB)** | 1024 | 256 | 1280 | 20.000% |
| **Data msg (1MB)** | 1048576 | 256 | 1048576 | 0.024% |
| **Data msg (10MB)** | 10485760 | 256 | 10485760 | 0.002% |

**TABLE 6.** Control overhead with RSA key size = 4096 bits.

| | *Message Size (bytes)* | *Key size (bytes)* | *Total* | *Control Overhead (%)* |
|---|---|---|---|---|
| **Control msg header-only** | 10 | 512 | 522 | 98.084% |
| **Data msg (1KB)** | 1024 | 512 | 1536 | 33.333% |
| **Data msg (1MB)** | 1048576 | 512 | 1049088 | 0.049% |
| **Data msg (10MB)** | 10485760 | 512 | 10486272 | 0.005% |

**TABLE 7.** Experimental results - attacks prevention success rate.

| Attack Type | Message Type | Experiment Count [N] | Attack Success [S] | APSR |
|---|---|---|---|---|
| DoS | Control | 20 | 0 | 1 |
| | TCP [RST] | 20 | 0 | 1 |
| Replay | Control | 20 | 0 | 1 |
| | Data | 20 | 0 | 1 |
| FDIA | Control | 20 | 0 | 1 |
| | Data | 20 | 0 | 1 |

### *D. PROPOSED MECHANISM: DEFENSE AGAINST CYBER-ATTACKS*

The experiments conducted in this study aimed to measure the ability of the proposed mechanism to prevent potential cyber-attacks carried on SECS/GEM communications. The Scapy-tool [42] was used to launch cyberattacks against SECS/GEM communications in these experiments. The attacks were counted as a success if the attacker was able to inject a ***separate.req*** or malicious data in the message into the stream of already ongoing communication between the host and the equipment. Attack will be considered failed if the receiver entity detects the injected message and it does not comply with the request and retains the connection. For the higher confidence intervals, the attacks were repeated 20 times to ensure the ability to prevent DoS attacks, replay attacks, and FDIA attacks, and the attack prevention success rate (APSR) was then measured for both standard SECS/GEM and the proposed mechanism. The APSR was calculated by using Equation 2 [1]:

$$APSR = 1 - s/n \tag{2}$$

where $s$ is the number of successful attack attempts, and $n$ is the total number of attempts, which in this case is 20. TABLE 7 shows that the proposed mechanism successfully


**FIGURE 15.** Proposed mechanism: Detection and prevention of malicious packets.

detected and prevented cyber-attacks carried out on SECS/GEM communications. Thus, the experiments prove that the proposed mechanism is a suitable candidate mechanism to be deployed in existing industrial installations in order to avoid cyber-attacks. FIGURE 15 depicts the proposed mechanism in action on factory equipment in a testbed environment, wherein the equipment was attacked with a DoS attack; however, the equipment successfully detected and dropped the packets due to the packet failing the verification process and thus being considered an attack on the system.

## VII. CONCLUSION AND FUTURE WORK

Industry 4.0 is already a reality in the contemporary world, and industrial sectors are transforming at a breakneck pace. It is a basic need of industry 4.0 that various functional units within the industry be interconnected with one another in order to get real-time insights into the different production processes for efficient decision making and higher productivity. In today's increasingly interconnected digital world, communication protocols without security features will eventually be deprecated and removed unless they are patched up with sophisticated safeguards against cyber threats.

Since its inception, SECS/GEM has been regarded as the backbone of the semiconductor industry, and it has been in widespread usage ever since. Its goal is to provide machine-to-machine communication and real-time insights into industrial equipment to facilitate efficient decision-making and increased efficiency in the factory. The standard SECS/GEM protocol transmits messages in plaintext without authentication or encryption, making it vulnerable to numerous cyber-attacks. It is critical to highlight that the machines have a very long lifespan, and once deployed, and they stay functioning for at least a couple of decades. This indicates that the presently deployed machines and SECS/GEM compliant machines purchased in the future would lack security features, which may result in a catastrophe if these security concerns are not addressed and fixed in a timely manner. Therefore, it is essential that in order to exploit the benefits of Industry 4.0 fully, the appropriate precautions be taken to secure communications and propose a defensive system that

may be used to protect SECS/GEM communications from well-anticipated cyber-attacks.

Based on the experiments conducted in this study, it is revealed that the SECS/GEM processes are subject to a variety of attacks, including DoS attacks, replay attacks, and FDIA attacks. In order to cater to the situation, we have proposed a security mechanism that authenticates SECS/GEM entities, ascertains message integrity, and prevents cyber-attacks. In conjunction with the proposed mechanism, the results indicated that even though the proposed mechanism successfully prevented cyber-attacks, it requires two different steps to hash and then encrypt the generated hash using public-key cryptography. Further research is needed to provide comparable security without relying on public-key-based cryptographic approaches.

This paper focused on attacks against SECS/GEM communications through the HSMS protocol. The DoS attack was conducted against HSMS using the separate.req command that caused the receiver to terminate the connection immediately without waiting for an acknowledgment. It should be noted that transmitting any TCP segment with the RST flag set, valid IP addresses, and port numbers, and a sequence number within the TCP connection's window may lead to a receiving node that implements TCP correctly in accordance with RFC 793 to terminate the TCP connection. Therefore, attacks using the TCP RST flag remain effective and will continue to be successful against all TCP connections if not mitigated.

Considering a couple of decade-long usable lifespans and the high cost of industrial equipment, it is anticipated that SECS/GEM will continue to operate for a substantial amount of time. Thus, it is critical to provide a comprehensive framework that incorporates security elements such as authentication, confidentiality, and integrity for secure M2M communication in manufacturing under the Industry 4.0 landscape. The authentication mechanism is needed because it will guarantee that only authorized SECS/GEM-enabled factory equipment are permitted to participate in communications. The integrity mechanism will ensure that the contents of messages are not changed during transit, and the confidentiality mechanism will prevent messages from being read during transit. Our future work is focused on addressing all of these concerns and providing a comprehensive security framework that will safeguard SECS/GEM communications from the cyberattacks described in this article.

## REFERENCES


[1] S. A. Laghari, S. Manickam, S. Karuppayah, A. Al-Ani, and S. U. Rehman, "Cyberattacks and vociferous implications on SECS/GEM communications in industry 4.0 ecosystem," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 7, pp. 329–336, 2021.

[2] E. Oztemel and S. Gursev, "Literature review of industry 4.0 and related technologies," *J. Intell. Manuf.*, vol. 31, no. 1, pp. 127–182, Jan. 2020.

[3] S. A. Laghari, S. Manickam, and S. Karuppayah, "A review on SECS/GEM: A machine-to-machine (M2M) communication protocol for industry 4.0," *Int. J. Electr. Electron. Eng. Telecommun.*, vol. 10, no. 2, pp. 105–114, 2021.

[4] J. Johnson. (2021). *Cyber Espionage: Most-Targeted Industries 2020*. Statista. Accessed: Nov. 8, 2021. [Online]. Available: https://www.statista.com/statistics/221293/cyber-crime-target-industries/

[5] N. Tuptuk and S. Hailes, "Security of smart manufacturing systems," *J. Manuf. Syst.*, vol. 47, pp. 93–106, Apr. 2018.

[6] R. Gupta, R. K. Phanden, S. Sharma, P. Srivastava, and P. Chaturvedi, "Security in manufacturing systems in the age of industry 4.0: Pitfalls and possibilities," in *Advances in Industrial and Production Engineering* (Lecture Notes in Mechanical Engineering). Singapore: Springer, 2021, pp. 105–113.

[7] S. Morgan, "Cyberwarfare in the C-suite cybercrime facts and statistics," Cybersecur. Ventures, Tech. Rep., Jan. 2021.

[8] *The Real Reason Behind the TSMC Cyber Attack| Industry| 2018-11-21| Web Only*. Accessed: Dec. 18, 2020. [Online]. Available: https://english.cw.com.tw/article/article.action?id=2194

[9] F. Chen, Y. Huo, J. Zhu, and D. Fan, "A review on the study on MQTT security challenge," in *Proc. IEEE Int. Conf. Smart Cloud*, Nov. 2020, pp. 128–133.

[10] S. Hernández Ramos, M. T. Villalba, and R. Lacuesta, "MQTT security: A novel fuzzing approach," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–11, Feb. 2018.

[11] C. Patel and N. Doshi, "'A novel MQTT security framework in generic IoT model,'" *Proc. Comput. Sci.*, vol. 171, pp. 1399–1408, Jan. 2020.

[12] A. Rahman, S. Roy, M. S. Kaiser, and M. S. Islam, "A lightweight multi-tier S-MQTT framework to secure communication between low-end IoT nodes," in *Proc. 5th Int. Conf. Netw., Syst. Secur. (NSysS)*, Dec. 2018, pp. 1–6.

[13] M. H. Schwarz and J. Borcsok, "A survey on OPC and OPC-UA: About the standard, developments and investigations," in *Proc. 24th Int. Conf. Inf., Commun. Autom. Technol. (ICAT)*, Oct. 2013, pp. 1–6.

[14] S. Profanter, A. Tekat, K. Dorofeev, M. Rickert, and A. Knoll, "OPC UA versus ROS, DDS, and MQTT: Performance evaluation of industry 4.0 protocols," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Feb. 2019, pp. 955–962.

[15] R. Matischek and B. Bara, "Application study of hardware-based security for future industrial IoT," in *Proc. 22nd Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2019, pp. 246–252.

[16] M. Iglesias-Urkia, A. Orive, and A. Urbieta, "Analysis of CoAP implementations for industrial Internet of Things: A survey," *Proc. Comput. Sci.*, vol. 109, pp. 188–195, Jan. 2017.

[17] Z. Shelby, K. Hartke, and C. Bormann. (2014). *The Constrained Application Protocol (RFC-7252)*. Accessed: Jun. 2014. [Online]. Available: http://www.rfc-editor.org/info/rfc7252

[18] M. Friesen, G. Karthikeyan, S. Heiss, L. Wisniewski, and H. Trsek, "A comparative evaluation of security mechanisms in DDS, TLS and DTLS," in *Kommunikation und Bildverarbeitung in der Automation*. Berlin, Germany: Springer-Verlag, 2020, pp. 201–216.

[19] X. Yu and H. Guo, "A survey on IIoT security," in *Proc. IEEE VTS Asia Pacific Wireless Commun. Symp. (APWCS)*, Aug. 2019, pp. 1–5.

[20] A. Esfahani, G. Mantas, R. Matischek, F. B. Saghezchi, J. Rodriguez, A. Bicaku, S. Maksuti, M. G. Tauber, C. Schmittner, and J. Bastos, "A lightweight authentication mechanism for M2M communications in industrial IoT environment," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 288–296, Feb. 2019.

[21] A. Karati, S. K. H. Islam, and M. Karuppiah, "Provably secure and lightweight certificateless signature scheme for IIoT environments," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3701–3711, Aug. 2018.

[22] Y. Zhang, R. Deng, D. Zheng, J. Li, P. Wu, and J. Cao, "Efficient and robust certificateless signature for data crowdsensing in cloud-assisted industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 15, no. 9, pp. 5099–5108, Sep. 2019.

[23] W. Yang, S. Wang, X. Huang, and Y. Mu, "On the security of an efficient and robust certificateless signature scheme for IIoT environments," *IEEE Access*, vol. 7, pp. 91074–91079, 2019.

[24] K. Mahmood, S. A. Chaudhry, H. Naqvi, T. Shon, and H. F. Ahmad, "A lightweight message authentication scheme for smart grid communications in power sector," *Comput. Elect. Eng.*, vol. 52, pp. 114–124, May 2016.

[25] M. Mumtaz, J. Akram, and L. Ping, "An RSA based authentication system for smart IoT environment," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun., IEEE 17th Int. Conf. Smart City; IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 758–765.

[26] T. Shah and S. Venkatesan, "Authentication of IoT device and IoT server using secure vaults," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./ 12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2018, pp. 819–824.

[27] S. F. Aghili and H. Mala, "Breaking a lightweight M2M authentication protocol for communications in IIoT environment," *Cryptol. ePrint Arch.*, vol. 1, no. 1, p. 9, 2019.
[28] E. Lara, L. Aguilar, M. A. Sanchez, and J. A. García, "Lightweight authentication protocol for M2M communications of resource-constrained devices in industrial Internet of Things," *Sensors*, vol. 20, no. 2, p. 501, Jan. 2020.
[29] K. K. Kolluru, C. Paniagua, J. van Deventer, J. Eliasson, J. Delsing, and R. J. DeLong, "An AAA solution for securing industrial IoT devices using next generation access control," in *Proc. IEEE Ind. Cyber-Phys. Syst. (ICPS)*, May 2018, pp. 737–742.
[30] *Introduction to SECS/GEM*. Accessed: Feb. 6, 2020. [Online]. Available: http://www.hume.com/secsintro.htm
[31] O. A. Amodu and M. Othman, "Machine-to-machine communication: An overview of opportunities," *Comput. Netw.*, vol. 145, pp. 255–276, Nov. 2018.
[32] L. Ma, N. Zhang, and Z. Zhang, "Tool Efficiency Analysis model research in SEMI industry," in *Proc. E3S Web Conf.*, vol. 38, 2018, pp. 1–5.
[33] E. F. Terng, S. C. Yeoh, K. C. Tong, and K. S. Yeo, "Data analysis on SMT reflow oven with SECS/GEM communication protocol," in *Proc. IEEE 10th Symp. Comput. Appl. Ind. Electron. (ISCAIE)*, Apr. 2020, pp. 118–124.
[34] S. Azaiez, F. Tanguy, and M. Engel, "Towards building OPC-UA companions for semi-conductor domain," in *Proc. 24th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2019, pp. 142–149.
[35] A. B. Nelson, P. M. K. Chow, V. A. Snowball, S. Chung, and A. Majumdar, "High-speed SECS message services (HSMS) pass-through including bypass," U.S. Patent 8 102 844, Jan. 24, 2012.
[36] J. Postel, *RFC 793: Transmission Control Protocol*. Fremont, CA, USA: Internet Engineering Task Force, Sep. 1981, pp. 1–85.
[37] S. Pfrang and D. Meier, "On the detection of replay attacks in industrial automation networks operated with profinet IO," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy*, Feb. 2017, pp. 683–693.
[38] D. Diaz-Sanchez, A. Marin-Lopez, F. Almenarez, P. Arias, and R. S. Sherratt, "TLS/PKI challenges and certificate pinning techniques for IoT and M2M secure communications," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3502–3531, 4th Quart., 2019.
[39] P. W. D. Charles, *Simple Python SECS/GEM Implementation*. San Francisco, CA, USA: GitHub Repository, GitHub, 2021.
[40] A. Al-Ani, M. Anbar, A. K. Al-Ani, and I. H. Hasbullah, "DHCPv6Auth: A mechanism to improve DHCPv6 authentication and privacy," *Sādhanā*, vol. 45, no. 1, pp. 1–11, Dec. 2020.
[41] K. A. Abuhasel and M. A. Khan, "A secure industrial Internet of Things (IIoT) framework for resource management in smart manufacturing," *IEEE Access*, vol. 8, pp. 117354–117364, 2020.
[42] S. Bansal and N. Bansal, "Scapy a Python tool for security testing," *J. Comput. Sci. Syst. Biol.*, vol. 8, no. 3, pp. 140–159, 2015.

**SHAMS UL ARFEEN LAGHARI** received the B.Sc. (Hons.) and M.Sc. degrees in computer science from the University of Sindh, Jamshoro, Pakistan, and the M.S. degree in computer science from PAF-KIET, Karachi, Pakistan. He is currently pursuing the Ph.D. degree in network security with the National Advanced IPv6 Centre, Universiti Sains Malaysia. His research interests include cybersecurity, industry 4.0, distributed systems, cloud computing, and mobile cloud computing.

**SELVAKUMAR MANICKAM** is currently an Associate Professor working in cybersecurity, the Internet of Things, industry 4.0, and machine learning. He had ten years of industrial experience before joining academia. He also has experience building IoT, embedded, server, mobile, and web-based applications. He is a member of technical forums at national and international levels. He has graduated 13 Ph.D. students. He has authored or coauthored almost 200 articles in journals, conference proceedings, and book reviews.

**AYMAN KHALLEL AL-ANI** received the Ph.D. degree in advanced computer networks from Universiti Sains Malaysia (USM). He is currently working as a Postdoctoral Research Fellow (PDRF) at the University of Malaya (UM). His current research interests include malware detection, web security, intrusion detection systems (IDS), intrusion prevention systems (IPS), network monitoring, the Internet of Things (IoT), IPv6 security, artificial intelligence, machine learning, data mining, and optimization algorithms.

**SHAFIQ UL REHMAN** received the Ph.D. degree from the University of Science Malaysia (USM), in 2017. He is currently an Assistant Professor at Amity University Rajasthan (AUR), Jaipur, India. He has previously worked as a Postdoctoral Research Fellow at the ST Engineering Electronics-SUTD Cyber Security Laboratory, Singapore University of Technology and Design (SUTD), from 2017 to 2020. He is also involved with various research projects related to the fields of networking and communication, cybersecurity, internet, and other emerging technologies. His current research interests include cybersecurity, machine learning, the Internet of Things, cloud computing, and industry 4.0.

**SHANKAR KARUPPAYAH** received the B.Sc. degree (Hons.) in computer science from Universiti Sains Malaysia, in 2009, the M.Sc. degree in software systems engineering from the King Mongkut's University of Technology North Bangkok (KMUTNB), in 2011, and the Ph.D. degree from TU Darmstadt, in 2016. His Ph.D. dissertation titled "Advanced Monitoring in P2P Botnets." He has been a Senior Lecturer at the National Advanced IPv6 Centre (NAv6), Universiti Sains Malaysia, since 2016. He has also been a Senior Researcher/Postdoctoral Researcher at the Telecooperation Group, TU Darmstadt, since July 2019. He is currently working on several cyber-security projects and groups, such as the National Research Center for Applied Cybersecurity (ATHENE), formerly known as the Center for Research in Security and Privacy (CRISP).