Secured SECS/GEM: A Security Mechanism for M2M Communication in Industry 4.0 Ecosystem

Ashish Jaisan, Selvakumar Manickam*, Shams A. Laghari, Shafiq Ul Rehman, Shankar Karuppayah
National Advanced IPv6 Centre (NAv6)
Universiti Sains Malaysia (USM)
Pulau Pinang, Malaysia

***Abstract*—The manufacturing industry has been revolutionized by Industry 4.0, vastly improving the manufacturing process, increasing production quality and capacity. Machine-to-Machine (M2M) communication protocols were developed to strengthen and bind this ecosystem by allowing machines to communicate with each other. The SECS/GEM protocol is at the heart of the manufacturing industry, thriving as a communication protocol and control system for years. It is a manufacturing equipment protocol used for equipment-host data communications. However, it is not without drawbacks, despite being a widely adopted communication protocol used by leading industries. SECS/GEM does not offer any type of security features as it was designed to work in a closed network. Such shortcomings in the protocol will allow attackers to steal secrets such as manufacturing processes by looking at recipes, perform reconnaissance prior to sabotage attempts, and can have severe implications on the entire industry. This paper proposes a mechanism to secure SECS/GEM data messages with AES-GCM encryption and evaluate the performance with the standard SECS/GEM protocol. The results from our evaluations showed that the proposed mechanism achieves data confidentiality and authenticity with a negligible overhead of 0.8 milliseconds and 0.37 milliseconds when sending and receiving a message, respectively, compared to the standard protocol.**

Keywords—SECS/GEM; HSMS; cybersecurity; industry-4.0; machine-to-machine communication; AES-GCM

I. Introduction

Industry 4.0 is bringing forth significant changes to the manufacturing industry. Industry 4.0 aims to take the manufacturing industry to the next level of technological advancement for an interconnected manufacturing ecosystem where machines communicate through the network to exchange messages, instructions, and data. With sophisticated machinery and automation, more integrated machine-to-machine communication, real-time monitoring and data collection, machine learning, and enhanced inter-connectivity, Industry 4.0 is changing the existing manufacturing process for the better and improving overall production [1]. As machines are interconnected, they generate activity analysis, predictive diagnostic data, performance statistics, and other monitoring and control information. Thus, real-time decisions can be made quickly with advantages such as time and cost-saving. In many circumstances, human interaction will be removed from the factory environment. With predefined and maintained settings and parameters, the factory equipment can make crucial decisions by itself, ensuring maximum cost-effectiveness for the industry.

Industry 4.0 deals with large volumes of data. Therefore, data security is a major concern when trying to achieve the true potential of Industry 4.0. It is essential to implement end-to-end encryption to fix vulnerabilities against various attacks [2]. With Industry 4.0's increased data density and the convergence of information and operational technologies, new issues emerge, particularly in the field of cybersecurity [3]. Cybers-attacks are the most critical problem that all countries are concerned about. It is a method of safeguarding digitally stored corporate data and valuable information about a system or subject from misuse, unauthorized access, and theft. Cyberattacks have become more common as network connections have grown, owing to a growing tendency to exploit data for various reasons, including financial gain and strategic reasons [4]. It is especially true in the case of cyberattacks against the manufacturing industry.

Although the manufacturing industry has been gradually updating and improving its IT security over the years, it can be seen in the Verizon Data Breach Investigation Report 2019, detailing 352 cyberattack incidents, out of which 87 were against the manufacturing industry. Recent attacks and security breaches against the manufacturing industry are alarming, making it a highly targeted and vulnerable entity for attackers [5]. A survey by the Engineering Employers' Federation (EEF) shows that 60% of manufacturers were victims of cyberattacks at some point in time, and one-third of the affected manufacturers have suffered financial losses and market loss. A 2021 study by Cybersecurity Ventures predicts that corporations worldwide will suffer losses up to $10.5 trillion yearly by 2025 resulting from cyber-attacks, estimated in 2015 to be $3 trillion [6]. The cyberattack on Taiwan Semiconductor Manufacturing Company (TSMC) was in Taiwan's history, the worst data security infringement to befall them. It completely exposed data security vulnerabilities at TSMC's production foundries. These cyber-attack incidents are happening as the manufacturing industry embraces the shift to Industry 4.0, with more and more machines becoming connected for communications and automation [7].

The SECS/GEM protocol is at the heart of the semiconductor industry in companies such as Intel, Samsung, TSMC, IBM, Qualcomm, and many more [8]. It has been profoundly used as a Machine-to-Machine (M2M) communication protocol and control system for decades. The SECS/GEM protocol is a specially designed semiconductor

*Corresponding Author

manufacturing equipment protocol used for equipment-host data communications.

A study by A. Laghari et al. [8] reveals that although SECS/GEM has been widely adopted and is critical to the semiconductor manufacturing industry, it is not without drawbacks. It does not offer any type of security features. It uses binary encoded messages to communicate between machines, making it open for anyone on the network to read the data; thus, data confidentiality is lost. Attackers on the network can modify data inside the messages as there are no authenticity checks. SECS/GEM protocol standards were designed in an era where machines were not required to be connected to the network [9]. The machines were initially expected to be working behind an air-gapped network and therefore did not require security features like network and data security. Thus, the focus was only needed on physical security. With Industry 4.0, the machines are required to be connected to the network for accessing data, analysis, and much more. These requirements open up air-gapped networks, and hence, the attack surface is enlarged in the process. Thus, cybersecurity in the manufacturing ecosystem is of the most importance. Data confidentiality, authenticity, and availability in machinery connected to the industrial network are all considered in the context of cybersecurity [10]. SECS/GEM is susceptible to these issues since it does not provide any kind of security features.

As SECS/GEM is a widespread M2M communication protocol used all around the world, we cannot simply introduce a new protocol. This paper proposes a security mechanism for the SECS/GEM protocol to attain data confidentiality and authenticity in SECS/GEM communications. We propose to encrypt the data payload of SECS/GEM messages to protect the data from attackers. In addition, we propose to use a hash-based tag to verify message data authenticity to ensure data has not been modified or corrupted by attackers.

To our knowledge, no prior research has been done on the security aspects of the SECS/GEM protocol. Hence, this is a novel field of study and has no known related works for comparison.

The rest of the paper is structured as follows. In Section II, the SECS/GEM protocol standards are described briefly. Section III discusses the security issues found in SECS/GEM protocol standards. In Section IV, we present the proposed mechanism in detail. Section V presents the implementation and testbed details. In Section VI, we present our evaluation and results for the proposed mechanism. Section VII concludes this work and discusses future work.

II. SECS / GEM Protocol Standards

SEMI (formerly Semiconductor Equipment and Materials International) has released five major protocols over the years. With the first release in the year 1978 and the latest revision being released in 2020. Though the SECS/GEM communication protocols were published two decades ago, they are regularly maintained and published. This section is an overview of the major SECS/GEM protocol releases. Table I gives a brief description of SECS/GEM standards.

TABLE I. SECS / GEM Protocol Standards

Year	SEMI Standard	Description
1978	**E4** SECS-I	**SEMI Equipment Communications Standard-I** protocol allows various equipment and a host to communicate over an RS-232 connection.
1982	**E5** SECS-II	**SEMI Equipment Communications Standard-II** facilitates data exchange between equipment and host as a specific stream and function message in a predefined format.
1992	**E30** GEM	**Generic Equipment Model** aids in specifying usage of any particular SECS-II message as well as the monitoring of equipment behavior when communicating with the host
1994	**E37.1** HSMS-SS	**High-Speed SECS Message Service – Single Session** is a TCP/IP-based communication protocol that manages a single machine-to-machine communication link between equipment and a host.
1994	**E37.2** HSMS-GS	**High-Speed SECS Message Service – Global Session** is an extension to E37.1 with handling multiple sessions and maintaining the state of the equipment as an additional feature.

A. *SEMI Equipment Communications Standard-I*

The SEMI Equipment Communications Standard-I (SECS-I), also referred to as SEMI E4 standard, is the oldest SECS/GEM standard. The exchange of communication messages between manufacturing equipment and a host computer is described in this standard. The equipment and host are not required to be familiar with one another to exchange messages [11] [12]. The SECS-I standard uses the RS-232-c standard for communication. Over RS-232, the SECS-I has a sluggish data transfer rate and does not offer support local area networks based on TCP/IP. The messages and data are exchanged asynchronously. The connection is bidirectional but limited to work in half-duplex mode. The rate of communication is generally between 9,600 baud and 19,200 baud. The protocol uses 256-byte blocks for multiblock data transfers. However, longer distances are not suitable for RS-232 transmission, and it has a low noise immunity. SECS-I is only used in old legacy production machinery and is not used in any newer machinery.

B. *SEMI Equipment Communications Standard-II*

The SEMI Equipment Communications Standard-II (SECS-II), also referred to as SEMI E5 standard, is a communication protocol that defines a generic messaging layer to send or receive any given data structure supported by the standard. Additionally, it specifies a collection of standard messages, each with its purpose, structure, and identity. It decodes the message type, message structure, data types, and message contents sent between the manufacturing equipment and the host. The message types are specified for various categories that cover a wide range of functions, generic as well as for specific purposes. The messages are divided into streams based on the particular category the message falls into (e.g., equipment status is dealt by Stream-1, whereas recipe management specifications are handled by Stream-7, etc.), with functions being individual messages within each stream [13].

The streams and functions are represented by numbers of size 1 byte. Since only one byte is used, the numbers start from 0 and can go up to 255. The combination of stream and function numbers can be represented as SnFm, wherein n represents stream number and m represents function number designated for data exchange. The request messages are represented by odd-numbered function codes, whereas response messages are represented with even-numbered function numbers. For example, a request message of Stream 1 and Function 13 (S1F13) is an "Establish Communication Request" message for a host/equipment. Upon receiving an S1F13 message, the equipment/host would send a stream-1 and function-14 (S1F14) message as a reply. A request message and its corresponding response message are called a transaction (i.e., S1F13/S1F14). A unique ID is assigned to each transaction. The sender specifies the SystemBytes, a field in the message header of size 4 bytes. The SystemBytes is used to link a request message with the respective response message.

The SECS-II standard provides data types for encoding data in a compact, bandwidth-efficient format. Integers, both unsigned and signed, can be stored in 1-byte, 2-byte, 4-byte, and 8-byte sized fields. Floating-point values can be stored in fields of size 4 and 8 bytes. On/off, values are represented using the 01-byte Boolean data type. Strings are described using the ASCII data type, while file data such as images and statistical plots are stored using the binary datatype. The List data item type can contain nested lists as well as a sequence of other primitive data items. The total number of items in a list is obtained from the length bits of the List data item. The maximum size for a data element within a SECS-II message is 16,777,215 bytes (approximately 16.5MB) long, according to the E5 standard. A message could contain only one single data element (for example, binary data or encoded text), or a large, sophisticated data structure (for example, lists stored within another list), or even no data at all.

C. Generic Equipment Model

The Generic Equipment Model (GEM), also referred to as SEMI E30 standard, defines a set of minimum requirements for describing factory equipment using a generic model, as well as optional features, use cases, and scenarios. A subset of SECS-II messages is used in the GEM model [14]. The GEM interface includes basic requirements as well as additional equipment capabilities. The GEM standard defines the generic model for equipment so that whatever the scale or sophistication of the production equipment, a generic interface (GEM) can be implemented for it. Some basic equipment, for example, does not require recipe management because it does not have any recipes for processing. For complex equipment, having many recipes to pick from, the requirement is that it must push/pull recipes to and from the host machine. GEM is also scalable in terms of data size. Simple devices with limited capabilities, for example, may publish a dozen different collection events. On the other hand, complex factory equipment may generate large amounts of events and data and publish many collection events in a short period. Yet, both can use the same GEM interface.

D. High-Speed SECS Message Service

High-Speed SECS Message Service (HSMS), also referred to as SEMI E37 standard, is a SEMI standard that defines the transport protocol for SECS/GEM message communications [15] [16]. HSMS is based on TCP/IP. It is, in fact, a derivation of TCP/IP with minor modifications and employs nearly the same techniques for creating connections as specified in RFC 793 [17]. One such change is that RFC 793 specifies to allow the communicating parties to connect to each other simultaneously. The HSMS protocol, on the other hand, restricts the connection-establishment procedure and defines two separate modes to establish connections, the passive and active modes. Devices running active mode can only initiate a request to establish a connection. The devices in passive mode can only accept connection establishment requests from other devices in active mode. HSMS carries SECS-II messages in binary encoding format to monitor status, control processes, report on events, and perform numerous other machinery operations after a communication link between equipment and host has been established. Between the communicating entities, the established connection is maintained for as long as required. Messages are exchanged between equipment and host until either device disconnects for some reason, such as hardware/software upgrades, machine additions or removals, or maintenance. The messages are sent as a data stream with a fixed header structure. The header fields are described in Table II. The first 4 bytes determine the encoded SECS-II message's total length, including the size of the header (10 bytes). The smallest HSMS message is 10 bytes (i.e., just the header size), while the largest conceivable size of the message is 4 GB. The structure of a HSMS message is depicted in Fig. 1.

TABLE II. HSMS HEADER FIELDS

Header Field	***Size***	***Description***
Session ID	2 bytes	It is used to associate reference between control messages and subsequent data messages
Stream	1 byte	Represents the Stream number of the message
Function	1 byte	Represents the Function number of the message
PType	1 byte	It is an enumerated type to define encoding used. HSMS defines PType with value zero to mean SECS-II message encoding. Non-zero PType values are reserved for subsidiary standards' future use.
SType	1 byte	It is an enumerated type to identify if the message is a control message (non-zero) or a data message (zero)
System Bytes	4 bytes	It is used to associate primary messages with the respective secondary message (reply)

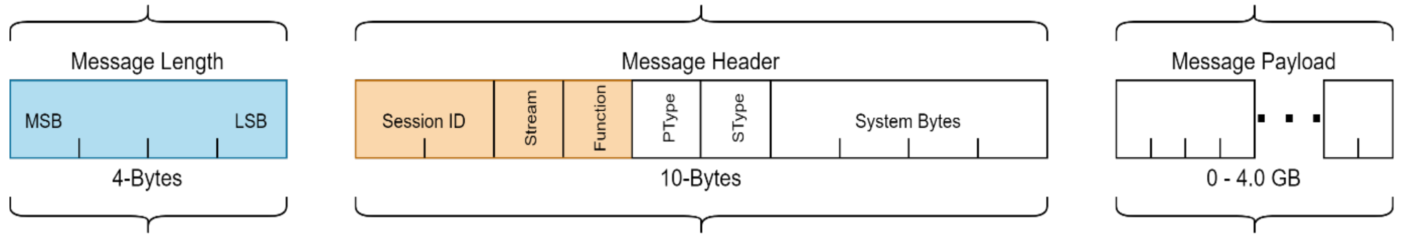

Fig. 1. Standard HSMS Message Structure.

The SECS/GEM interface allows factory hosts to monitor equipment actions and provides total equipment control. Everything happening on the machinery can be monitored, and enhanced logic can be put on the equipment to make better decisions. Various applications can be implemented using SECS/GEM to monitor and analyze statistical data, troubleshoot, predict possible maintenance requirements, control processes for feedback/feedforward, check usage, track materials, validate recipes, etc. These systems also eliminate the requirement for an operator-to-equipment interaction, resulting in fewer operators needed in the production environment. Factories can reduce material scrap and waste by using effective recipe management. For example, storing golden recipes in a centralized location via the SECS/GEM interface makes sure that the right recipes and materials are used.

III. Security Issues

The SECS/GEM protocols in its original standard do not specify any encryption for its message data and all messages between equipment and host are unencrypted binary encoded data. This shortcoming introduces opportunities for attackers to exploit and disrupt the health of the production environment. Attackers can launch attacks, disrupt communications, steal intellectual property belonging to the company, and more.

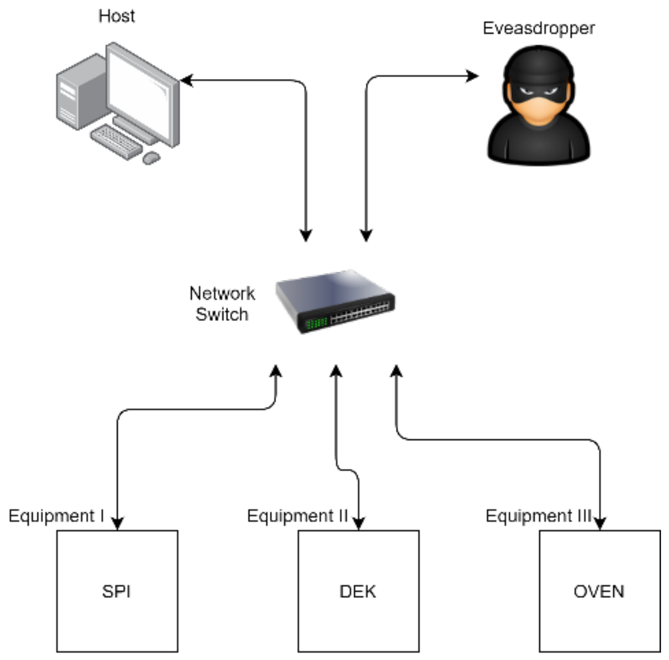

Fig. 2. Attacker on the network Eavesdropping Communications.

In this paper, we focus on data confidentiality and authenticity issues in SECS/GEM protocols. Due to data being transferred in binary encoded format, attackers can eavesdrop on equipment-host communication and lead to loss of data confidentiality. Fig. 2 shows how an attacker may position themselves on the network to eavesdrop on communication messages passively. Attackers can learn machine parameters and settings, product design information from communication messages. Intellectual property such as product designs, parameters, and settings for the manufacturing process can be stolen by attackers for monetary gain. Such an attack can be a life-or-death situation for companies as the industry is always competitive, and loss of IP can cost a company their leadership in the industry.

A study by A. Corallo et al. [10] shows that if a product's design information is no longer confidential, it could negatively affect the company's competitive advantage. The loss of unique knowledge about the items and their manufacturing methods may work in competitors' favor. Data confidentiality of machine settings and parameters or machinery status, if lost, could lead to a decline of the company's reputation. This information provides insight into the production ecosystem's health. Therefore, if sensitive information such as machine malfunctions are revealed, the company's reliability would be questioned and could lead to investors leaving and losing customers. The loss of confidentiality of product properties, like quality indicators, would negatively affect the company's leadership and change in favor of its competitors. In fact, in the event of a product fault, competitors may exploit the situation by using ad hoc styled marketing strategies to win a larger market share.

Attackers can also launch tailored attacks such as Man-In-The-Middle (MITM) attacks to disrupt production and cause financial losses as part of sabotage operations. Such attacks can range from disrupting communications, injecting false data, causing machine failures, etc. It is easy for attackers to launch such attacks as current SECS/GEM protocol implementations do not provide data confidentiality or perform authenticity checks on the data.

To further discuss the seriousness of this issue, consider the scenario where an attacker launches a passive eavesdropping attack on the network and listens to communications. Over time the attacker can gather enough data about the machine settings and parameters to launch MITM attacks. For example, attackers may learn parameters that can make machines run differently, cause malfunctions or create defective products that fail quality checks. Such knowledge will let attackers launch attacks that will look normal to the system and intentional disruptions of the manufacturing systems. Attacks of this kind can cause failure rates to be high and make it look like the machines malfunction or fail even when they are not. This type of attack is proven to be possible.

The infamous STUXNET virus is a real-life example of the previously described scenario which targeted the Iranian nuclear program and caused equipment to malfunction [18], [19]. The STUXNET virus had the pattern described in the example scenario. It recorded data from the Supervisory control and data acquisition (SCADA) systems controlling the equipment during its incubation period. It then starts actively attacking the facility by sending malicious parameters to make the equipment fail. It replayed the previously recorded data during the attacks to trick the operators from knowing the actual status of the equipment [20]. It appeared to the operators as just equipment malfunction. In this case, a slight change in the rotation speed of certain parts of the equipment caused them to malfunction and explode. STUXNET is considered to be an attack on a nation. If a nation is at risk from such attacks, it only makes it more apparent that a manufacturing company is even more susceptible to such attacks.

An attack such as STUXNET on the manufacturing industry may be a targeted attack against a company's sustainability. An attacker will launch attacks to disrupt operations until the company is forced to stop operations due to substantial financial losses. If the attacker cannot see the data being transmitted through the communication messages, it can help protect against the attacker’s reconnaissance attempts and stall following attacks. Unsolicited messages from attackers with possibly malicious instructions can be blocked if the data is checked for authenticity.

Hence, data confidentiality and authenticity have a significant impact on the industry's ecosystem. The SECS/GEM protocol is at the heart of the semiconductor industry, and therefore these issues are of serious nature. With the leap of the manufacturing industry into Industry 4.0, machines will need to communicate with other machines through the production network. With the ongoing Corona Virus Disease 2019 (COVID-19) global pandemic, during the time of this research, the need for remote access and communication with production machines has become more necessary due to work-from-home scenarios [21]. Managers and operators overlooking factory equipment require remote access to check equipment status all the time. However, allowing machines to connect to the network and operations personnel further increases the attack surface for cybercriminals to gain access to the production environment. Therefore SECS/GEM protocol's method of communication with binary encoded data becomes a major security issue and must be addressed to thwart attack attempts from cybercriminals.

All major industries are attempting to bring their factories up to the Industry 4.0 standards to reap the benefits. Machine-to-Machine communication is essential to automate the processes in every industry. For example, machines can communicate with other machines when they need more components, a change of recipe, or when an error occurs, the previous machine on the production line needs to stop sending more batches to process. Such coordination between factory equipment can help a lot with automation and with the overall efficiency of the manufacturing process as Industry 4.0 compatible factories would need lesser human interaction.

IV. Proposed Mechanism

This section describes, in detail, the proposed mechanism for preserving data confidentiality and authenticity in SECS/GEM during transmission in production networks. Data confidentiality is a critical part of the production network since data may go through several hops. This can be ensured using a secure encryption mechanism. This is necessary due to the wide range of devices, services, and networks that communicate/operate with a lot of data and thus present sufficient opportunity for data confidentiality violations as well as modifications due to the ease with which data may be accessed in SECS/GEM communication [22]. A data transfer mechanism for secure and efficient SECS/GEM communication is proposed in response to this requirement. The proposed mechanism is designed for the HSMS protocol in the SECS/GEM protocol stack. HSMS was chosen as it is the latest SECS/GEM protocol and is supplied with the latest machines.

We propose to use the Advanced Encryption Standard Galois/Counter Mode (AES-GCM) encryption scheme to achieve data confidentiality in SECS/GEM communication. Galois/Counter Mode (GCM) is one of several modes available for symmetric-key cryptographic block ciphers. It is adopted widely for its performance and throughput rates. With inexpensive hardware resources, throughput rates as high as 10 Gbps can be achieved [23]. It is an authenticated encryption algorithm that provides both data confidentiality and authenticity. GCM is defined for block ciphers that operate on a block size of 128 bits, and hence AES-GCM is used.

A. Proposed Mechanism Design

The proposed mechanism is designed to encrypt the data payload of the HSMS packet and verify its authenticity at the receiver end. The proposed mechanism's packet structure is depicted in Fig. 3. The message has 4 bytes of message length denoting the size of the HSMS message, including the header and payload length. The header of the message consists of 10 bytes. The header fields are described in Table II.

The header and the length bytes follow the same structure as in the standard HSMS message. The message payload, however, has a different structure from a standard HSMS message data payload. The data message payload of the proposed mechanism has a structure, as depicted in Fig. 3. It has fixed sizes for certain data at the beginning and end of the payload structure. The first 16 bytes of the message payload is the nonce. Next comes the ciphertext data message of variable length up to maximum payload size in bytes minus sum of nonce size and tag size. The last 16 bytes of the message payload is the message tag.

The nonce is a pseudorandom value of length 16 bytes. It is generated by the encryption mechanism as an input for internal use. The nonce is similar to an initialization vector (IV) used in various encryption schemes. The same nonce is required to decipher the ciphertext back into plaintext.

The tag is a hash of length 16 bytes generated by the encryption mechanism. It is used to verify the message's authenticity. The tag is computed during the deciphering process and checked with the sender's tag to verify message authenticity.

B. *Proposed Mechanism Flow*

Fig. 4 illustrates the flow of the proposed mechanism. Three inputs are required for the encryption mechanism to work, the pre-shared key, a nonce, and the plaintext data. The pre-shared key is a 256-bit symmetric encryption key (32 bytes). The nonce is a pseudorandom value of size 128 bits (16 bytes). It is used as an IV for the encryption scheme and the hashing function used to generate the message verification tag. The plaintext data is the HSMS message's original payload. The encryption scheme is AES-GCM 256, as the key is 256 bits in length, and a longer key implies increased security against exhaustive brute force attacks [24].

The algorithm used to encrypt the payload and generate the tag is shown in Fig. 5. The plaintext data is passed into the encryption mechanism along with the pre-shared key. A pseudorandom nonce is generated on the fly and is used as an initialization vector for the encryption mechanism's internal counter. The same nonce is required at the receiver end to decipher the ciphertext and is written to the data payload as it is safe to share nonce along with the message. The nonce is written to the first 16 bytes of the message payload. The encryption scheme then encrypts the data as 128-bit blocks using the provided key and part of the nonce as an IV for its internal counter. The ciphertext data is then appended to the message payload after the nonce. Upon completing the encryption process, a tag is generated by the encryption mechanism and written to the last 16 bytes of the message payload. This tag is essentially a hash generated by the encryption mechanism. The tag is used to verify the message authenticity on the receiver's end.

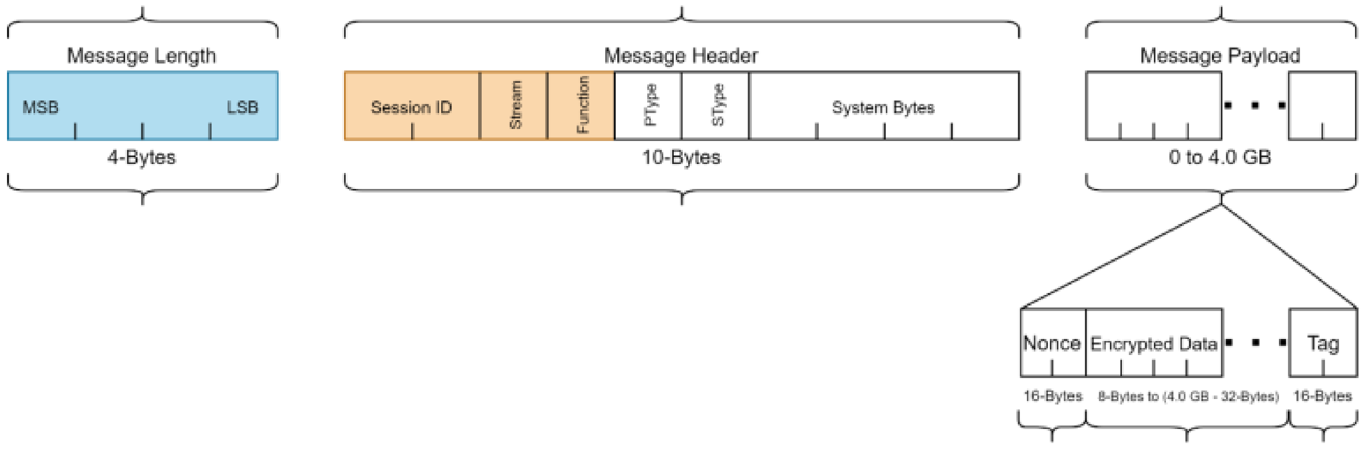

Fig. 3. Proposed HSMS Message Structure with Encrypted Data.

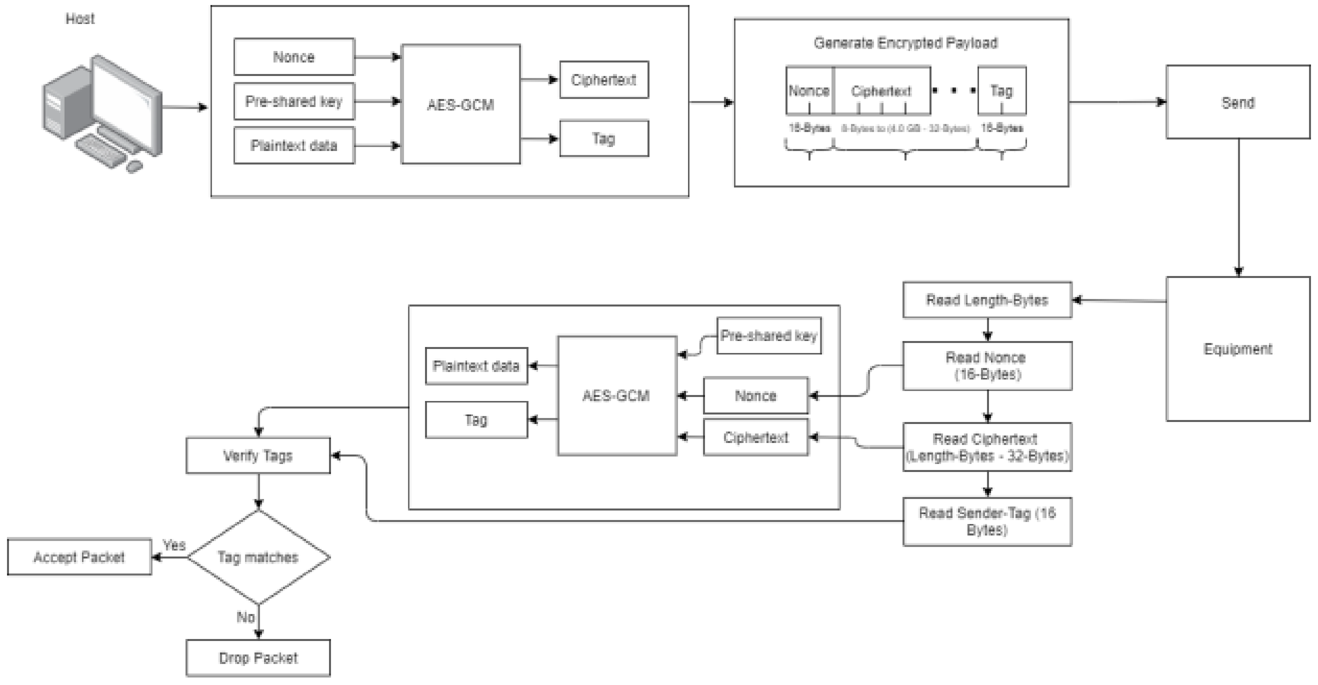

Fig. 4. Proposed Mechanism.

The algorithm used to decrypt the payload and verify its authenticity is shown in Fig. 6. On the receiver end, the data payload of the HSMS message is read as in the packet structure for the proposed mechanism. The first 16 bytes of the message payload are read as the nonce. Since the last 16 bytes of the encrypted payload is the tag generated by AES GCM, 16 bytes are subtracted from the length of the remaining payload, and the data is read for that length. Equation (1) can be used to calculate the size of the ciphertext data within the payload:

$$C_{len} = P_{len} – N_{len} + T_{len} \quad (1)$$

C_{len} is the ciphertext length computed by calculating the difference between P_{len}, the payload length, and the sum of N_{len} and T_{len}, where N_{len} is the size of nonce and T_{len} is the size of the tag. The nonce, ciphertext, and the pre-shared key are passed in as inputs to the decryption mechanism. The decryption mechanism takes 128-bit blocks of cyphertext and decrypts them. After decryption, a tag is generated by the decryption mechanism. This tag would be the same as the tag obtained from the encryption mechanism. If the tags match, the message is accepted; otherwise, the payload's authenticity fails, and the message is rejected.

```
Algorithm: Send HSMS message with encrypted data

1   Start
2   If the message length is 10, then
            Go to step 7
3   nonce = generate random value.
4   ciphertext = encrypt the payload with preshared-key, nonce and
    get ciphertext output
5   tag = Get message authentication tag from AES-GCM output
6   Replace message payload with nonce + ciphertext + tag
7   Send message
9   End
```

Fig. 5. Algorithm to Encrypt Payload and Generate Tag.

```
Algorithm: Receive HSMS message with encrypted data

1   Start
2   If the message length is 10, then
            Go to step 9
3   nonce = read first 16 bytes of payload
4   ciphertext = read payload size – 32 bytes of data
5   plaintext = decrypt ciphertext with preshared-key, nonce and get
    plaintext output
6   tag = Get message authentication tag from AES-GCM output
7   sender-tag = read last 16 bytes of payload
8   If sender-tag is the same as tag, then
            Replace message payload with plaintext
    Else
            Drop the message and go to step 10
9   Accept and process the message
10  End
```

Fig. 6. Algorithm to Decrypt the Payload and Verify the Authenticity.

The proposed secure version of the HSMS protocol runs on a different port from the standard HSMS protocol. For example, if the standard version runs on port 5000, the proposed version can run on port 5001. The secure version is thus distinguishable from standard communication protocol. A different port is required because the standard protocol would not be expecting an encrypted payload and may run into errors when trying to parse the payload. The proposed mechanism acts as an overlay protocol. It handles data confidentiality and authenticity on both ends and then forwards the message to the next layer, where the message is processed.

V. Implementation and Testbed Setup

A. Implementation

We used secsgem from [25], a python implementation of SECS/GEM protocols, as the base for our implementation. The implementation is free and available online on GitHub. For implementing AES-GCM encryption over secsgem, we used the Python Pycryptodome library from [26]. Pycryptodome is a library of implementation for cryptographic algorithms.

B. Experimental Testbed Setup

Our testbed consists of two machines running SECS/GEM simulator with Machine-I acting as the host and Machine-II as the equipment. Both machines have the configuration as stated in Table III.

TABLE III. Experimental Testbed Machine Configuration

	Specification
Processor	Intel Core i3-9100F @ 4.2Ghz
Memory (RAM)	2 GB
Operating System	Ubuntu 18.04 LTS
Network	100Mbps ethernet

Machine-I (host) was set up to be the active device initiating connections to machine-II. Machine-II (equipment) was set up to be a passive device listening for connections from Machine-I.

VI. Performance Evaluation and Results

For the evaluation of the proposed mechanism, the host machine was configured to connect to the equipment machine and send over 1000 SECS/GEM messages at regular intervals. The SECS/GEM implementation was configured to compute the time taken for processing while sending and receiving messages and store it in a log file. The experiments were conducted for both the standard HSMS protocol and the proposed mechanism. The processing times were then obtained from the log files for each experiment. The processing time obtained from the log files was labeled as described in Table IV.

TABLE IV. Processing Time: Label Description

Label	***Description***
Host-S	the processing time taken by the host to process the initial message to be sent (to equipment).
Equip-R	the processing time taken by the equipment to process the message received (from the host).
Equip-S	the processing time taken for the equipment to process the reply to be sent (to the host).
Host-R	the processing time taken by the host to process the reply received (from equipment).

C. *Processing Time*

The evaluation of the performance of the standard HSMS protocol and the proposed mechanism experiments and obtained the following results.

The standard HSMS protocol experiment's results are plotted in Fig. 7. It can observed that the host takes the longest time to send a message (Host-S), followed by the reply being sent from the Equipment (Equip-S). This variation is due to the differences in the size of the data payload. Host-S is the initial message, and Equip-S is the reply, essentially two different messages. The time taken for the host to process the response from the equipment (Host-R) takes the least amount of time, whereas processing time for the equipment to receive data (Equip-R) is slightly higher. This shows that the pattern in Host-S and Equip-S is the same in Equip-R and Host-R due to varying payload sizes of the initial message and reply message.

The processing times taken for the proposed mechanism are plotted in Fig. 8. The graph shows that the processing time for the host to send data (Host-S) was the longest. Following Host-S, the second-longest was the time processing time taken for the equipment to send a reply (Equip-S). The time taken for the equipment to process the message from the host (Equip-R) and the time taken for the host to process the reply from the equipment (Host-R) were similar. However, the processing time Equip-S was below the processing time Host-R for the most part. The pattern in the standard HSMS experiments is also seen in this experiment, meaning very well that the different sizes in the initial messages and the replies influence the processing time.

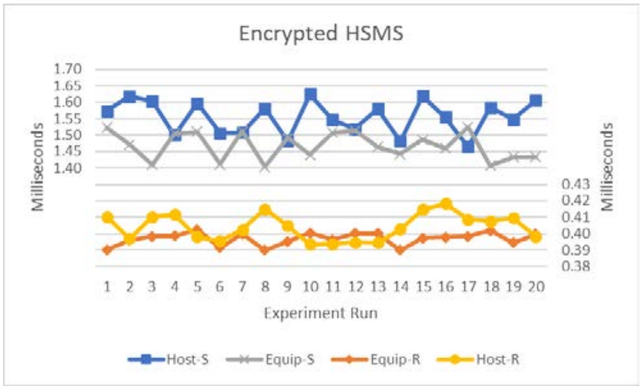

Fig. 7. HSMS with Data Encryption.

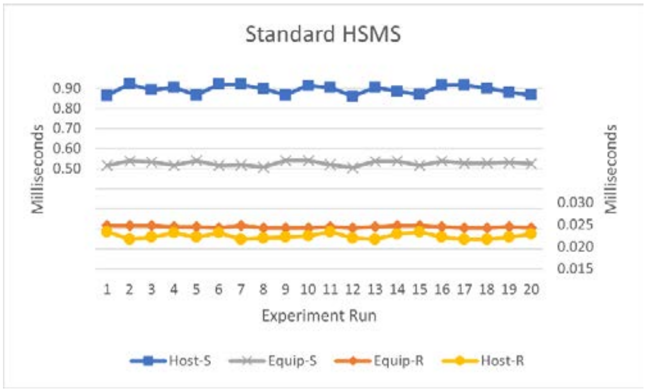

Fig. 8. Standard HSMS Experiment.

We computed the minimum, maximum, mean, and standard deviation in processing time for both the standard protocol and the proposed mechanism from the data we obtained in our experiments. Table V shows the mentioned metrics for the processing times of the standard protocol. It is seen that the mean processing time for Host-S and Equip-S is between half millisecond and one millisecond and Equip-R, and Host-R is below 1 microsecond.

TABLE V. PROCESSING TIME (MILLISECONDS): STANDARD HSMS

	Host-S	*Equip-R*	*Equip-S*	*Host-R*
Min	0.8607	0.0243	0.5054	0.0217
Max	0.9217	0.0248	0.5400	0.0234
Mean	0.8941	0.0245	0.5260	0.0224
SD	0.0218	0.0002	0.0112	0.0006

Table VI shows the metrics calculated for the processing time taken by the proposed mechanism. The results show that the mean processing time for Host-S and Equip-S is between 1.6 milliseconds and 1.4 milliseconds, and approximately 0.4 milliseconds for Equip-R and Host-R.

TABLE VI. PROCESSING TIME (MILLISECONDS): PROPOSED MECHANISM

	Host-S	*Equip-R*	*Equip-S*	*Host-R*
Min	1.464	0.390	1.404	0.394
Max	1.625	0.402	1.525	0.418
Mean	1.555	0.397	1.468	0.404
SD	0.051	0.004	0.043	0.008

Table VII shows the difference in processing times between the standard HSMS protocol and the proposed mechanism. The differences show that Host-S and Equip-S are between a half millisecond to one millisecond. Equip-R and Host-R are below a half millisecond. Analysis shows that encrypted SECS/GEM messages have a slight overhead. The time required for processing each message (sending and receiving) is increased by an average of 0.8 milliseconds due to encryption and decryption of data bytes in the HSMS message. The Encrypted HSMS message also has 32-Bytes of overhead. The nonce and tag are attached to the data bytes for the receiver to decrypt the ciphertext data. Thus, the maximum payload size is slightly reduced by 32 bytes as the proposed mechanism uses 32 bytes for the nonce and tag.

TABLE VII. PROCESSING TIME (MILLISECONDS): DIFFERENCE

	Host-S	*Equip-R*	*Equip-S*	*Host-R*
Min	0.603	0.365	0.899	0.372
Max	0.703	0.378	0.985	0.395
Mean	0.661	0.372	0.942	0.382

However, no encryption is performed for control messages such as "Link-Test Messages" as these messages do not contain any data bytes. The proposed mechanism checks for the length of data bytes and only performs encryption and decryption if the size of data bytes is greater than zero.

D. Control Overhead

Table VIII shows the control overhead for the proposed mechanism for messages with various payload sizes. For control messages without data, there is no added overhead. For a message of size 1 KB, we see an overhead of a 3% increase in payload size over the standard protocol. This is as a result of the nonce and tag being added to the payload. However, with bigger messages such as 1 MB and 10MB, we see the overhead is reduced drastically to a point where it is negligible as the size of the nonce and tag have fixed size for all messages.

TABLE VIII. CONTROL OVERHEAD

	Message Size (bytes)	*Control data (bytes)*	*Total*	*Control Overhead*
Control msg (header-only)	10	-	10	0.00%
Data msg (1KB)	1024	32	1056	3.03%
Data msg (1MB)	1048576	32	1048608	0.0031%
Data msg (10MB)	10485760	32	10485792	0.0003%

The processing time overhead observed is also negligible, considering that data confidentiality and authenticity are achieved in SECS/GEM communication. Furthermore, AES-GCM is a block cipher algorithm widely adopted for its performance. The experiments were conducted were on a general-purpose computer where the encryption was software-based. In a real industry scenario, this would be done on a dedicated yet inexpensive hardware-based encryption module, leading to even better performance of up to 10Gbps speeds of encryption.

E. Security Analysis of Brute-Force Attack

The proposed mechanism encrypts the plaintext data into ciphertext, making it meaningless to anyone monitoring the ciphertext data. Thus, passive attacks such as eavesdropping and reconnaissance are rendered useless as attackers will not be able to get the plaintext data. For an attacker to obtain plaintext data, the secret key is required for decryption. Without the key, the attackers can only try to make an exhaustive brute force attack to guess the key. The proposed mechanism uses a 256-bit pseudorandom key, and thus it would require the attacker to try at least half of the keys on average to find the correct one. Therefore, on average, the attacker will need to try 2^{255} different keys.

The latest processor with special instructions for AES operations uses about 0.16 cycles to process 1 byte of plaintext [27]. Table IX shows the time taken in years to crack the encryption with an exhaustive brute force attack. Equation (2) was used to compute the time required (in years) to break AES-GCM for various computers [28]. The results are shown in Table IX. T is the time complexity to break AES-GCM. $K_{possibilities}$ is the average number of keys the attacker has to try before finding the correct key. For the proposed mechanism, it is 2^{255} possibilities, as discussed previously. C_{sec} is the number of cycles or operations the CPU can perform in a second. C_{byte} is the number of cycles required to process one byte of plaintext, while B_{size} is the size of one block of plaintext in bytes. $B_{size,}$ in this case, is 128 bits (16 bytes) as AES-GCM operates on 128-bit blocks. Y_{sec} is the total number of seconds in a year (60 × 60 × 24 × 365.25 = 31,557,600 seconds).

$$\mathrm{T} = \frac{K_{possiblities}}{\left({C_{sec}}/{C_{byte} \times B_{size}}\right) \times Y_{sec}} \qquad (2)$$

TABLE IX. YEARS REQUIRED TO BREAK AES-GCM WITH 256-BIT KEY

Computer	*Speed*	*Time required in years*
Intel Core i7-10870H	280 Gflop/s	1.677362236307178100432348431475e+58
Fugaku (Japanese supercomputer)	442 Pflop/s	1.0625824121402938192784107710701e+52
All computers in the world	200 Gflop/s × 2 billion	2.2932686824512200591848513711572e+55

For our security analysis of the proposed mechanism, we calculated the time complexity of cracking AES-256-GCM on the latest Intel Core i7 processor and Fugaku, the world's most powerful supercomputer at the time of this research [29] and all the computers in the world combined. The total number of computers in the world is around 2 billion [30]. The results presented in Table IX show the number of years required to successfully brute force the key is in multiples of trillions of trillions of years. Thus, an attacker cannot decipher the ciphertext with the technology available as of now. It remains safe to assume that the proposed mechanism would not be broken anytime soon.

Using AES-GCM, the proposed mechanism attains data confidentiality. It can prevent passive attacks such as eavesdropping and reconnaissance by attackers. The data is encrypted, and thus, attackers are unable to read the data. As the data authenticity is checked, the proposed mechanism protects against MITM attacks where attackers try injecting false data or modify the data. An attacker cannot modify the data as it is encrypted. Even if the attacker has altered parts of the encrypted data in the message payload, the authenticity of the data will fail as every message has a tag to verify message authenticity. Thus, the message's authenticity is verified.

VII. CONCLUSION AND FUTURE WORK

In this paper, we proposed a mechanism for SECS/GEM's HSMS Protocol to attain data confidentiality and check data authenticity in its data communication messages by encrypting the data payload using the AES-GCM encryption scheme. We also evaluated the performance of the proposed mechanism with the standard protocol. The results indicate that AES-GCM encryption of HSMS data messages has a slight overhead of 0.8 milliseconds and 0.37 milliseconds when sending and receiving a message, respectively, compared to the insecure standard HSMS protocol. However, this overhead is negligible considering that encrypting HSMS data messages makes the protocol secure from eavesdropping attackers seeing the data transferred in the messages while also checking the authenticity of messages. Thus, the proposed mechanism achieves data authenticity and confidentiality. This will be a step further towards Industry 4.0 for the HSMS protocol-enabled machines.

The proposed mechanism aimed to protect data confidentiality and check data authenticity. However, SECS/GEM protocol has other shortcomings that need to be addressed to secure it completely. The proposed mechanism only encrypts the data payload part of a message. The header is still visible to the network. Although it does not expose sensitive data such as parameters, settings, or confidential data, an entity on the network can still see the frequency of each type of message sent on the network. Furthermore, SECS/GEM is still vulnerable to attacks such as replay and Denial of Service (DoS) attacks. Future research to enhance SECS/GEM security may include investigations into the implications of these problems and potential remedies. Future studies may potentially look at problems such as authentication and privacy for SECS/GEM communications in Industry 4.0 ecosystem.

References

[1] S. Azaiez, F. Tanguy, and M. Engel, "Towards building OPC-UA companions for semi-conductor domain," IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, vol. 2019-Septe, pp. 142–149, 2019, doi: 10.1109/ETFA.2019.8869171.

[2] M. Gadre and A. Deoskar, "Industry 4 . 0 – Digital Transformation , Challenges and Benefits," International Journal of Future Generation Communication and Networking, vol. 13, no. 2, pp. 139–149, 2020.

[3] Frost & Sullivan, "Cyber Security in the Era of Industrial IoT," A Frost & Sullivan White paper, 2017.

[4] B. C. Ervural and B. Ervural, "Overview of Cyber Security in the Industry 4.0 Era," no. September 2018, pp. 267–284, 2018, doi: 10.1007/978-3-319-57870-5_16.

[5] N. Tuptuk and S. Hailes, "Security of smart manufacturing systems," Journal of Manufacturing Systems, vol. 47, pp. 93–106, Apr. 2018, doi: 10.1016/j.jmsy.2018.04.007.

[6] S. Morgan, "CYBERWARFARE IN THE C-SUITE CYBERCRIME FACTS AND STATISTICS," 2021.

[7] S. Peng, "The Real Reason Behind the TSMC Cyber Attack," CommonWealth Magazine, Nov. 2018.

[8] S. A. Laghari, S. Manickam, and S. Karuppayah, "A Review on SECS/GEM: A Machine-to-Machine (M2M) Communication Protocol for Industry 4.0," International Journal of Electrical and Electronic Engineering and Telecommunications, vol. 10, no. 2, pp. 105–114, 2021, doi: 10.18178/ijeetc.10.2.105-114.

[9] 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," International Journal of Advanced Computer Science and Applications, vol. 12, no. 7, p. 2021, Sep. 2021, doi: 10.14569/ijacsa.2021.0120737.

[10] A. Corallo, M. Lazoi, and M. Lezzi, "Cybersecurity in the context of industry 4.0: A structured classification of critical assets and business impacts," Computers in Industry, vol. 114, p. 103165, 2020, doi: 10.1016/j.compind.2019.103165.

[11] T. O. F. Contents, "SECS Messaging Primer," pp. 1–14, 2016.

[12] "Introduction to SECS/GEM." http://www.hume.com/secsintro.htm (accessed Jun. 29, 2021).

[13] K. Jung, J. S. Han, Y. M. Lim, and W. S. Kim, "XML format design for SECS-II message monitoring," in Proceedings - ALPIT 2007 6th International Conference on Advanced Language Processing and Web Information Technology, 2007, pp. 548–552, doi: 10.1109/ALPIT.2007.69.

[14] K. Uriga, "Generic Equipment Model (GEM) Specification Manual : The GEM Specification as Viewed from the Host. Technology Transfer 97093366A-XFR," 1997. Accessed: Jun. 29, 2021. [Online]. Available: https://www.academia.edu/27932413/Generic_Equipment_Model_GEM_Specification_Manual_The_GEM_Specification_as_Viewed_from_the_Host.

[15] "US8102844B1 - High-speed SECS message services (HSMS) pass-through including bypass - Google Patents." https://patents.google.com/patent/US8102844B1/en (accessed Jun. 29, 2021).

[16] L. Ma, N. Zhang, and Z. Zhang, "Tool Efficiency Analysis model research in SEMI industry," in E3S Web of Conferences, Jun. 2018, vol. 38, p. 02027, doi: 10.1051/e3sconf/20183802027.

[17] "RFC: 793 TRANSMISSION CONTROL PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION," 1981.

[18] I. Jamai, L. Ben Azzouz, and L. A. Saidane, "Security issues in Industry 4.0," 2020 International Wireless Communications and Mobile Computing, IWCMC 2020, vol. 0, pp. 481–488, 2020, doi: 10.1109/IWCMC48107.2020.9148447.

[19] N. Benias and A. P. Markopoulos, "A review on the readiness level and cyber-security challenges in Industry 4.0," South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference, SEEDA-CECNSM 2017, 2017, doi: 10.23919/SEEDA-CECNSM.2017.8088234.

[20] J. Prinsloo, S. Sinha, and B. von Solms, "A review of industry 4.0 manufacturing process security risks," Applied Sciences (Switzerland), vol. 9, no. 23, 2019, doi: 10.3390/app9235105.

[21] A. Georgiadou, S. Mouzakitis, and D. Askounis, "Working from home during COVID-19 crisis: a cyber security culture assessment survey," Security Journal, pp. 1–20, Feb. 2021, doi: 10.1057/s41284-021-00286-2.

[22] N. N. Hurrah, S. A. Parah, J. A. Sheikh, F. Al-Turjman, and K. Muhammad, "Secure data transmission framework for confidentiality in IoTs," Ad Hoc Networks, vol. 95, p. 101989, 2019, doi: 10.1016/j.adhoc.2019.101989.

[23] D. A. Mcgrew and J. Viega, "The Galois/Counter Mode of Operation (GCM)," 2005.

[24] D. Yehya and M. Joudi, "AES Encryption : Study & Evaluation," no. November, 2020.

[25] "GitHub - bparzella/secsgem: Simple Python SECS/GEM implementation." https://github.com/bparzella/secsgem (accessed Jul. 28, 2021).

[26] "GitHub - Legrandin/pycryptodome: A self-contained cryptographic library for Python." https://github.com/Legrandin/pycryptodome (accessed Jul. 28, 2021).

[27] N. Drucker, S. Gueron, and V. Krasnov, "Making AES great again: The forthcoming vectorized AES instruction," in Advances in Intelligent Systems and Computing, 2019, vol. 800 Part F, pp. 37–41, doi: 10.1007/978-3-030-14070-0_6.

[28] "How long would it take to brute force AES-256? | ScramBox," Scrambox, 2016. https://scrambox.com/article/brute-force-aes/ (accessed Aug. 20, 2021).

[29] "Fugaku Holds Top Spot, Exascale Remains Elusive | TOP500." https://www.top500.org/news/fugaku-holds-top-spot-exascale-remains-elusive/ (accessed Jul. 26, 2021).

[30] SCMO, "How many computers are there in the world? — SCMO," SCMO, 2019. https://www.scmo.net/faq/2019/8/9/how-many-compaters-is-there-in-the-world (accessed Jul. 27, 2021).