A Review on SECS/GEM: A Machine-to-Machine (M2M) Communication Protocol for Industry 4.0

Shams A. Laghari, Selvakumar Manickam, and Shankar Karuppayah
National Advanced IPV6 Centre Universiti Sains Malaysia (USM), Pulau Pinang, Malaysia
Email: shamsularfeen@nav6.usm.my; {selva; kshankar}@usm.my

***Abstract*—Industry 4.0 has imminently emerged as the fourth industrial revolution. It garners emphasis primarily on the interaction between factory equipment and machines involved in entire value-chain activities for boosting efficiency and production with limited human intervention. The M2M communication protocols have gained significant prominence amidst sage minds in recent years, particularly in the manufacturing industries. Several M2M communication protocols have been developed for the industry, such as SECS/GEM, OPC UA, DDS, and MQTT. Among these protocols, SECS/GEM is a semiconductor's equipment interface protocol for equipment-to-host data communications. It is not a modern, but decades-long protocol, and has been again brought to the fore by Industry 4.0. Thereupon, it was imperative to review amenability of SECS/GEM protocol in the context of adaptation of features depicted by industry 4.0, and limelight plethora of shortcomings and limitations. In this paper, comparisons of prominent features and limitations of the aforementioned M2M communication protocols in general and review of the SECS/GEM protocol in particular, have been made. Findings include deficiencies in security, point to point communication, discovery mechanism, fixed message format, interoperability, extendibility, and manual integration.**

***Index Terms*—Industry 4.0, SECS/GEM, M2M, Machine-to-Machine, Semiconductor**

I. Introduction

The advent of mechanical equipment powered by thermal and kinetic energy has revolutionized industrial processes at the culmination of the 18th century. This was vocally termed as the first industrial revolution. This echoed for mass production, which was achieved with the emergence of electrical technology during the mid of 19th century. Thus, recognized as the second industrial revolution. The industries took flight with the invention of Programmable Logical Controller (PLC) in late 1960, which revolutionized industrial automation and termed as third industrial revolution [1], [2]. Technological advances have progressed so exponentially over the last few decades that we have entered the fourth industrial revolution dubbed Industry 4.0 [3]. As the turning point of this technological revolution, cyber-physical structures have been praised so that the physical world can be completely incorporated into the virtual world. additive manufacturing, Internet of Things (IoT), machine learning, big-data analytics, 5G networks, cloud computing, autonomous robots, and cybersecurity are the main developments of this modern technological age, which bring revolutionary changes in the economy, industry, society, and individuals [4]. Industry 4.0 is moving towards modernizing the production process and increasing industrial productivity with its emphasis on advanced robotics and automation, new forms of machine-to-machine interaction, real-time data collection, machine learning, and enhanced connectivity [5]. The growing population has growing demands for custom products in the shortest possible time at the cost of large-scale production.

Automation in semiconductor manufacturing is playing a vital role in day-to-day operations. The noteworthy focus is on boosting efficiency and productivity in the value-chain process by substituting human operators with automated machines in myriad situations, such as tasks that are recursive in nature, complex, prone to errors, dangerous and hazardous, and the like. The fab operations are categorized into three types, i.e., *manual*, *semi-automated*, and *fully-automated*, based on the attention required by the operator. The manual fab operations require an operator to run equipment, which makes it very astringent to find equipment operating without computer assistance in the modern semiconductor industries. The semi-automated fab operations have achieved cognizance in the modern factory installations; wherein, processing tools and equipment are assisted, monitored, and controlled by computers for major activities, while operators carry out tasks including material loading/unloading and other manual processes. The fully automated fab operations have eased processes in the modern factory systems, i.e., 12-in 300mm, by the installation of systems that carry out automated processing without the involvement of the human operator.

M2M communication refers to the interaction as well as data exchange between two or more interconnected machines without human intervention [6]. This encompasses almost everything, including smartphones, laptops, tablets, factory equipment, robots, automatic sensors, and enables these devices to react and adjust their internal processes based on external feedback [7]. Based on effective M2M communications and interactions, machines and factory equipment may

Manuscript received May 9, 2020; revised August 4, 2020; accepted August 15, 2020.
Corresponding author: Selvakumar Manickam (email: selva@ usm.my).

doi: 10.18178/ijeetc.10.2.105-114

provide information on patterns in usage (or misuse) and signal events to act immediately. Machines can be interconnected to produce operational performance statistics, predictive diagnostic data, inactivity analysis, and a host of related monitoring and control information. Thus, active decisions with simple, cost-saving advantages can be taken quickly. M2M can transform the conventional linear supply-chain into a feedback loop that continuously flows through dynamic business alliances that produce, distribute, and service the objects. Human intermediaries will in many cases, be entirely eliminated from the equation. With specified and managed parameters, equipment assets can make key decisions themselves, thus providing the end-user with maximum cost-effectiveness. Therefore, based on the relevant scientific works, the most popular M2M and industrial communication protocols including SECS/GEM are discussed in this paper.

The paper is further organized in sections. Section 2 introduces briefly discusses the most common Industry 4.0 protocols. Section 3 discusses SECS/GEM communication protocol and its associated key components; wherein, four primary SEMI standards, i.e., E4, E5, E30, and E37, are discussed in greater detail. Section 4 discusses the publish/subscribe model, message broker, and data packaging and encoding concepts. Section 5 highlights the issues and weaknesses found in current SECS/GEM implementations. Section 5 is the concluding paragraph.

II. Messaging and M2M Communication Standards in Industry 4.0

A. OPC UA (Open Platform Communications) Unified Architecture

OPC UA is a machine-to-machine, service-oriented open-source communication protocol that is defined, developed, and maintained by Open Foundation under IEC 62541 specifications [8], [9]. OPC UA is predominantly designed for and heavily used in industrial automation. OPC UA communicates and interacts with industrial machines, equipment, and systems for data collection, monitoring, and control. Its main goals are to provide a platform-independent communication protocol using an information model to describe the data exchanged between communicating entities in the industrial network. OPC UA primarily obeys client/server architecture; however, lately, the support for publish/subscribe has been extended in the protocol specifications. Unlike many other publish/subscribe protocols[10], OPC UA provides no support for the Quality of Services (QoS) in newly published protocol specifications. OPC UA uses TCP/UDP port 4840 for its binary protocol by default. To realize security, OPC UA provides authentication, authorization, as well as integrity and confidentiality protection.

B. DDS (Data Distribution Service)

DDS is a data-centric, publish-subscribe, machine-to-machine communication protocol for real-time systems [11]. Unlike OPC UA and MQTT (Message Queuing Telemetry Transport), DDS is a distributed, decentralized, and peer-to-peer protocol; hence, it does not require a centralized broker to publish-subscribe its messages [11]. Instead, Publishers and Subscribers can directly communicate and thereby allow asynchronous data exchange amongst participating nodes. The centralized broker has the advantage of having a network-wide view; however, often centralized brokers become the bottleneck and single point of failure [12]. DDS is more resilient and reliable in terms of system availability as devices participate in a distributed fashion, and there are many brokers, hence, not a single point of failure in this case. In DDS publishers and subscribers are separated from each other so that a publisher is allowed to publish data even if there is no interested subscriber. Publishers are not required to identify their data sinks beforehand; thus, consumers may subscribe anonymously to the intended data [13].

DDS is a peer-to-peer protocol, so it requires to find out the presence of data producers and consumers. This is achieved with a dynamic discovery process built-in discovery protocol. Unlike many other protocols where brokers are required to be configured for notifications and message forwarding, in DDS, it is not required to configure anything, publishers and subscribers will automatically be discovered and data will flow in real-time. Hence, discovery service makes DDS highly scalable because there is no single point of failure and devices may join and leave the system at any time.

The key functional components in DDS architecture are Domain, Domain Participant, Publisher, Subscriber, Topic, Data Reader and a data writer. A Domain represents an address space where topic and type definitions are defined. Domains are autonomous and each domain is assigned a unique domain id; therefore, two applications can only communicate with each other if they are registered within the same domain [14]. Domain participants are an entry point where topics, data readers, data writers, and other objects are created and destroyed in the global data space (GDS). GDS is fundamental abstraction in DDS; wherein, data is gathered, shared, analyzed, and acted upon [15].

DDS can run on both TCP and UDP, so security mechanisms can be selected accordingly [16]. When DDS run over TCP, TLS (Transport Layer Security) becomes an obvious choice, whereas, DTLS (Datagram Transport Layer Security) is more appropriate for UDP as a transport protocol. Both TLS [17] and DTLS [18] put heavy computational overhead on resource-constrained devices which makes them the expensive choice for IoT systems.

C. MQTT (Message Queuing Telemetry Transport)

MQTT is an open-source, lightweight, publish/ subscribe [19], M2M communication protocol. MQTT performs well in networks where connectivity conditions are not ideal such high latency and low bandwidth. It is specifically designed for resource-constrained devices having limited computational power, memory, storage, and battery backup. MQTT protocol provides reliable

communication as it runs over the TCP transport protocol, with a small code footprint, it becomes the most suitable protocol for M2M communications and IoT-based networks [9]. An MQTT system is consists of two communicating parties based on the publish-subscribe messaging principles, client and server (server is also known as a broker in MQTT paradigm). A client can be a source to generate data/traffic and publish it, or it can be a subscriber to receive messages of interest from a broker, or it may be a publisher as well as a subscriber at the same time. When a client takes the role of publisher, it is required to know the broker it is going to connect, whereas, it must know the topic it needs to subscribe when it takes the role of a subscriber. Multiple clients are allowed to be subscribers of the same topic, and they receive updates whenever a new message is published. The broker plays a vital role and takes responsibility to register bind clients with topics, filters arrived messages, and delivers them to the subscribed clients [13]. MQTT defines three QoS levels and services offered at each level are given as under:

- QoS 0 requires no confirmation or acknowledgment from the receiver and delivers messages on a best-effort basis. This QoS level may be a choice in cases where some sensors gather telemetry information over a longer time duration and where the sensors' values do not change significantly.
- QoS 1 guarantees message delivery and sends acknowledgments when messages are received or if the publisher does not receive an acknowledgment within a predefined time duration, it will retransmit (publish) the message again.
- QoS 2 guarantees exactly-once and offers duplication-free message delivery.

MQTT runs over TCP as a transport protocol, which makes it inappropriate for constrained devices [9]. Furthermore, MQTT does not provide encryption and transfers data as plaintext, which is clearly an issue from the security standpoint.

D. SECS/GEM

Semiconductor Equipment and Material International (SEMI) is an association enjoying membership of more than two thousand companies around the world. It deals in materials, services, and equipment required by the manufacturing industries. It unleashes various standards, including E4, E5, E30, and E37, which arrange communication between host and the factory equipment. The SEMI communication standards, as described in Table I, are collectively known as SECS/GEM (Semiconductor Equipment Communication Standard / Generic Model for Communications and Control of Manufacturing Equipment). The SECS/GEM is an industry protocol and is in profound use for decades in almost all manufacturing industries [20]. The SECS/GEM of SEMI's is the jugular vein in the semiconductor industry, such as Intel, Samsung, TSMC, IBM, Qualcomm, Broadcom, UMC, SK Hynix, Micron, TXN, Toshiba, NXP, and so on, proving as a communication protocol and control system since years [21].

TABLE I: SEMI'S SECS/GEM COMMUNICATION STANDARDS

Year	SEMI Standard	Description
1978	**E4** SECS-I	**SEMI Equipment Communications Standard-I**: It is a communication protocol that helps establish communication between various equipments and a host on RS-232 cable. This works at physical layer.
1982	**E5** SECS-II	**SEMI Equipment Communications Standard-II**: It helps exchanging information between equipments and host as a series of streams and function messages on a defined format.
1992	**E30** GEM	**Generic Equipment Model**: It helps defining usage of SECS-II messages and monitor behaviour of the equipment while exchanging messages with the host.
1994	**E37.1** HSMS-SS	**High-Speed SECS Message Service – Single Session**: It defines a communication protocol managing point-to-point communication between a equipment and a host on TCP / IP.
	E37.2 HSMS-GS	**High-Speed SECS Message Service – Global Session**: It is akin to E37.1 with the addition of capacity to handle multiple sessions by retaining state of equipment.

The SECS/GEM communication protocol is feature rich and offers two type of capability sets 1) The fundamental requirements and 2) the additional capabilities. All SECS/GEM compliant equipment must support at least fundamental requirements. The fundamental requirements include State models, Equipment processing states, host-initiated, Event notification, Error messages, Online identification, documentation, and Control.

In addition to the fundamental GEM requirements, the SECS/GEM specifies a myriad list of optional capabilities that can be implemented and supported by a GEM interface based on the complexity and requirements. The additional capabilities include Establishment of Communication, Event Notification, Dynamic Event report configuration, variable data collection, trace data collection, status data collection, alarm management, remote control, equipment constants, process program management, material movement, equipment terminal services, clock, limit monitoring, spooling, and host initiated control [22].

III. SEMI'S SECS/GEM STANDARDS

A. SECS-I

The standard E4 alias SECS-I describes the exchange of messages between semiconductor equipment and host computer without requiring equipment and host to have acquaintance with each other [23], [24]. The SECS-I standard defines point-to-point communication by utilizing the RS-232-c standard. SECS-I utter a slow transmission data rate over RS-232; whereas, it is deficient in supporting TCP/IP based local area networks. The communication is bidirectional, asynchronous, and half-duplex. The pace of communication usually ranges between 9,600 and 19,200 baud-rate. SECS-I protocol establishes multiblock transfers based on blocks of 256 bytes. The communication over RS-232 is inappropriate for longer distances and project scarce noise immunity.

The SECS-I protocol is currently only available in legacy factory equipment and is not supplied with new machines.

B. SECS-II

The SEMI E5 standard, commonly termed as SECS-II, is a message content protocol that specifies a generic message layer to transmit any data structure supported by the specifications. It also describes a set of standard messages with each specific message identity, purpose, and format. SECS-II reveals an interpretation of message types, data types, message structure, and message contents exchanged between intelligent factory equipment and host. The message types are defined for various categories, which cover a wide array of functions - specific or general. The SECS-II [25] messages are classified into various categories referred to as streams (i.e., stream-1 deals with equipment status, stream-7 covers specifications related to recipe management), and so on, whereas, the functions are specific messages within a particular stream category. Both streams and functions are defined by a combination of a stream number and a function number, which are single-byte values ranging among 0 to 255. This combination is symbolized as ***Msg=SnFm***, where *n* and *m* represent a specific stream/function number designated for data exchange. The odd-numbered function codes represent primary messages (request message), whereas even-numbered function codes signify secondary message (reply message). For example, a primary message S1F13 refers to Stream 1 and Function 13, which enables an entity to send an Establish Communication Request message to the intended entity/equipment; and on receiving such a message, the equipment/host will reply with S1F14. The pair of each primary message and secondary message (i.e., S1F13/S1F14) is called a transaction. Each transaction is identified with a unique transaction Id. The sender sets a special 4-byte integer header field called "system bytes," which is used to pair the primary message with its secondary message. Fig. 1 demonstrates two different scenarios where the exchange of pair of messages (S1F13/S1F14) is exchanged between equipment/host.

SECS-II provides primitive data types to encode messages in a highly compact bandwidth-efficient format. The signed and unsigned integers are stored in 1, 2, 4, and 8 byte-sized variables. The floating-point numbers are stored in 4 and 8 byte-sized variables. The 01-byte Boolean data type is used to represent on/off values. ASCII data type is used to describe strings, and Binary data type is used to store binary data such as images and graphs. The List data item type is supported to contain a sequence of other primitive data items as well as nested lists. The length bits of List data item determine the number of total data items in the list. The E5 standard limits a single data element within a SECS-II message to be of a maximum of 16,777,215 bytes (about 16.5MB) long. A message could be a simple data element (e.g. a binary response, an ASCII string), a complex list structure (e.g. multiple levels of lists in the hierarchy), or without data.

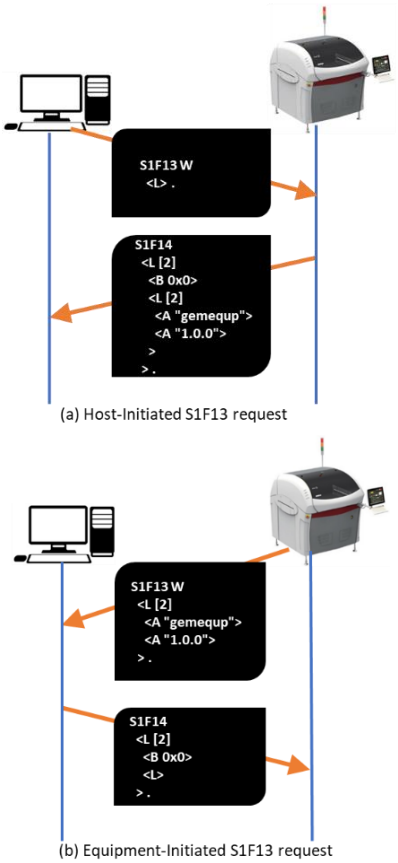

Fig. 1. Scenario to illustrate the exchange of (S1F13/S1F14) message pair.

C. HSMS

HSMS is a transport protocol for SECS/GEM communications [26], [27]. The HSMS protocol is derived from TCP/IP protocol and uses almost identical procedures for connection establishment as defined in RFC 793 with slight differences [28]. The RFC 793 specifications permit intended communicating parties to establish a connection with each other simultaneously. However, the HSMS protocol restricts connection establishment procedure and defines two different modes for connection establishment, namely ***Active Mode*** and ***Passive Mode***. The active mode devices can only initiate the connection establishing requests. Once the communication link is established between the host and equipment, HSMS transports binary encoded SECS-II messages to control recipe change, monitor, report, and various other functions of the semiconductor equipment. The connection remains established and maintained for longer durations between communicating entities. The data flows back and forth until one or both entities are intentionally taken offline for some specific reasons such as software upgrades, add/remove machines, and maintenance. The HSMS message format is depicted in Fig. 2. The HSMS message is transported as a stream of bytes and the first 4-bytes determine the total length of the encoded SECS-II message along with the header. The minimum HSMS message size is 10 byte (i.e., header-only), and the maximum theoretical message size is 4.3 gigabytes.

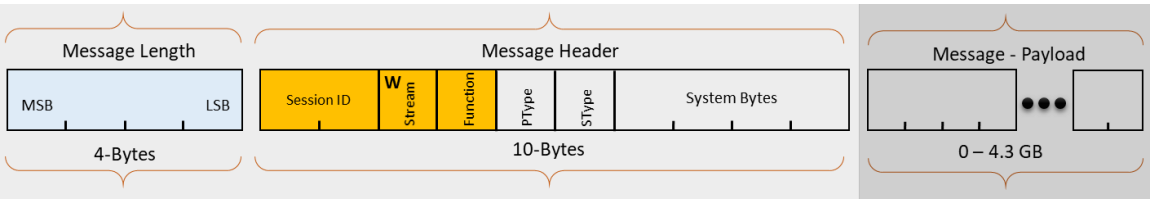

Fig. 2. HSMS message format.

D. GEM (E30): Generic Equipment Model

The SEMI E30 standard, also referred to as GEM, specifies a minimum set of specifications to describe a generic factory equipment model along with additional optional capabilities, use cases, and user scenarios. The GEM model is built upon a subset of SECS-II messages [29]. The GEM interface encompasses fundamental requirements and additional equipment capabilities [30]. The GEM interface can be implemented on any factory equipment irrespective of its size and complexity. Some simple equipment, for example, has no recipes for processing; hence, no support for recipe management is needed for such equipment. On the other hand, complex equipment has several recipes to choose from; thus, such equipment needs to upload/download recipes from the factory host. SECS/GEM also scales well with data size. For example, simple devices with basic functionality may publish a dozen separate collection events. In contrast, complex factory equipment may generate a massive bulk of data and publish tens of thousands of collection events in a brief time period, but both can still use the same SECS/GEM interface.

TABLE II: COMPARISON OF THE PROTOCOLS AND THEIR MAIN FEATURES

Feature	SECS/GEM	MQTT	OPC UA	DDS
Infrastructure	Ethernet RS-232	Ethernet	Ethernet	Ethernet
Network layer	IPv4 or IPv6	IPv4 or IPv6	IPv4 or IPv6	IPv4 or IPv6
Transport layer	TCP	TCP	TCP, UDP	TCP/UDP
Transport port	5000	1883, 8883	4840	7400, 7401
Message Pattern	pub/sub	pub/sub	Req/Res, pub/sub	Req/Res, pub/sub
Mechanism	One-to-one	One-to-many	One-to-many	Many-to-many
Methodology	Message-Oriented	Message-oriented	Service-Oriented	Message-oriented
Paradigm	Event-based	Event-based	Polling based	Event Based
Reliability Mechanism	One Level	3 QoS Levels	One Level	20+ QoS Levels
Standard	SEMI.org	ISO/IEC, OASIS	OPC Foundation	OMG
Encoding	Binary	UTF-8 (Binary)	Binary, XML	Binary
security	No security	SSL, TLS	User, PKI	PKI TLS/DTLS
Developed By	HP	IBM	OPC Foundation	RTI
Header Size	10 bytes	2 bytes	8 bytes	56 [31]
Fault Tolerance	SPoF*	SPoF*	SPoF*	Decentralized
Discovery	No	No	No	YES

The SECS/GEM interface offers complete equipment control and allows factory hosts to monitor equipment activities. The SECS/GEM interface offers complete equipment control and allows factory hosts to monitor equipment activities. Anything that happens on the equipment can be monitored, and advanced control policies on the equipment can be implemented for effective decisions. A SECS/GEM framework allows applications for statistical process monitoring, trouble-shooting, predictive maintenance, process controls for feedback/feedforward, equipment utilization, material tracking, recipe validation, etc. to be implemented. These systems also reduce the need for an interface between the operator and the equipment, resulting in the reduced number of operators required in the factory. Effective recipe management schemes enable factories to reduce material scrap and waste. For example, using the SECS/GEM interface to store golden recipes in the centralized location ensures that the correct material recipe is used. Table II summarizes the salient properties and features of all protocols discussed in this paper.

IV. SECS/GEM FEATURES

A. SECS/GEM Connection Establishment Procedure

The SECS/GEM standard limits equipment to be connected with only one host device at a time. Current protocol specifications and implementation permit only one active point-to-point communication link with factory equipment and a host. It is imminent for a successful connection establishment to configure both communicating entities in different modes, i.e., passive and active mode. The Passive entity opens a TCP port and listens for the incoming connection requests, whereas the Active entity initiates the connection request. Under some particular circumstances and special requirements, it is hard to choose a specific mode for the end device. SECS/GEM provides Alternating mode to deal with such situations. An entity configured with Alternating mode permits equipment to act as an Active or Passive entity as per need. The entities can be configured in different modes, as mentioned in Table III.

TABLE III: EQUIPMENT/HOST CONFIGURATION MODES

	Active	**Passive**	**Alternating**
Active	Not allowed	Active Initiates	Active Initiates
Passive	Active Initiates	Not Allowed	Alternating Initiates
Alternating	Active Initiates	Alternating Initiates	Either can Initiate

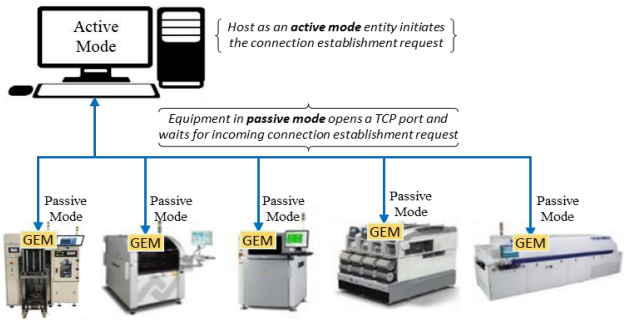

Fig. 3. Equipment/Host configuration scenario.

Fig. 3 depicts a scenario where equipment is configured passive mode and they open and listen for incoming connections on a TCP port (usually port 5000). The factory host generally operates in active mode and initiates a connection with the equipment. Upon a successful exchange of S1F13/S1F14 messages, the connection is successfully established between the host and equipment, and the state-machine of equipment change from CONNECTING to CONNECTED. Afterward, the equipment will reject all subsequent incoming connection requests.

B. Message Encoding and Packaging

SECS-II uses 1-byte as a format code, and data items are represented with different codes. For example, six bits are used to represent a data type and often identified with octal values such as 00, 10, 20, and 31 represent List, Binary, ASCII and 1-byte signed integer, respectively. The length of the data item is coded in 2-bits, whereas values 1, 2, and 3 represent a number of length bytes used to carry data values. A zero-length in the format byte is illegal and produces an error. Fig. 4 underneath represents the Format-byte structure. A complete list of supported data types and associated codes can be found in the E5 Standard documentation.

The encoding and packaging a single item (as an array) with multiple values is extremely compact and requires only 1 Format code to encode an entire array of any length. The total storage required to encode different SECS-II data types can be calculated as under:

$$\text{Storage} = T_{\text{size}} \times n_{\text{Items}} + F_c + \text{LB}_{1\text{-}3} \qquad (1)$$

where T_{size} represents the size of a single data item (1, 2, 4, 8 bytes-long int, etc.), n_{Items} is the total number of values encoded within a single data item (as an array), F_c is 1-byte format code, and it indicates the data type; and LB is 1-3 length bytes to package the data item.

Equation (1) cannot be applied on list data type as the list data type serves the purpose of grouping related data items under one hood. Unlike other data items, the list does not have length-byte to indicate its size in bytes; instead, the list treats the length-byte as number items in the list. Each list member can be of any data type, including lists, which allows lists to be a data member within a list (i.e., nested list). The list structure allows grouping items of related information, which may have different formats into a useful structure. For example, a list with 500 data items would have the format code and length bytes, as shown in Fig. 5:

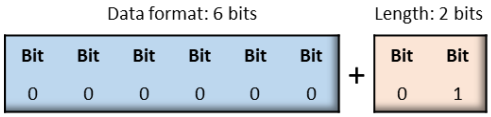

Fig. 4. SECS-II format code-byte.

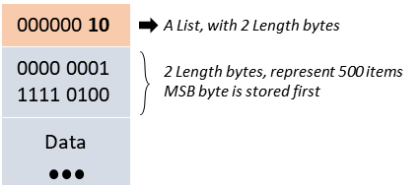

Fig. 5. A list with 500 data items.

TABLE IV: THE SECS/GEM'S DATA PACKAGING DENSITY

	format	Length	single item	1000 items	100000 items
Binary	1-byte	1-3 bytes	1x1+1 = 2	1x1000+1+2 = 1003	99.70%
ASCII: 1	1-byte	1-3 bytes	1x1+1 = 2	1x1000+1+2 = 1003	99.70%
±int: 2	1-byte	1-3 bytes	1x2+1+ 1 = 4	2x1000+1+2 = 2003	99.85%
int: 4	1-byte	1-3 bytes	1x4+1+ 1 = 6	4x1000+1+2 = 4003	99.93%
float: 8	1-byte	1-3 bytes	1x8+1+ 1 = 10	8x1000+1+2 = 8003	99.96%

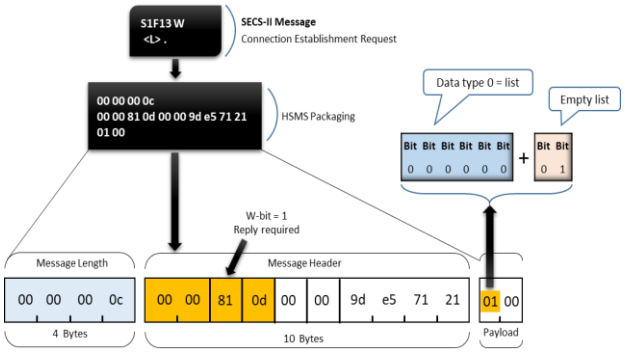

Fig. 6. S1F13 message structure and packaging.

The encoding of related data items in lists provide ease of use and increase the readability and productivity. However, lists with single-valued items consume more bandwidth as compared to an array of single-valued items. Consider a list with 500 1-byte unsigned integer items, where each item has a format code, a length byte and value. Hence, each list item would take 3 bytes and the list would take 3 bytes (as explained in the aforementioned example) which would result in 3×500+ 3 = 1503 bytes; whereas only 503 bytes would have been taken, if a single item would have been encoded with 500 different values. In other words, a list with 500 single-byte items incurs 66.73% more control bytes as compared to a single-item containing 500 1-byte values. The SECS/GEM's data packaging density in Table IV shows SECS/GEM data types and their associated data density.

The simple S1F13 (Connection Establishment Request) message from the host with an empty list is depicted in Fig. 6. The W-bit in S1F13 message is present, which signifies that the host computer is expecting a reply message from equipment. The message size is then computed based on that data passed from the SECS-II layer, and in this particular example, the data portion is consisting of an empty list; therefore, that payload size is just 2-bytes. The total message size is 16 bytes (i.e., 4 length bytes, 10 header bytes, 2 data bytes). It can be observed that each data item (including lists) has a Format Code and length bytes required to encode data.

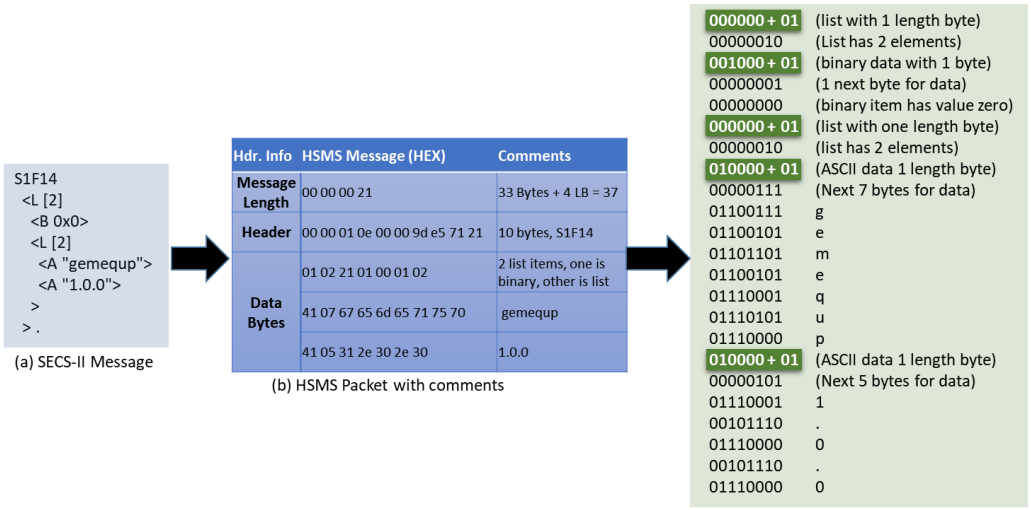

Fig. 7. S1F14 message structure and packaging.

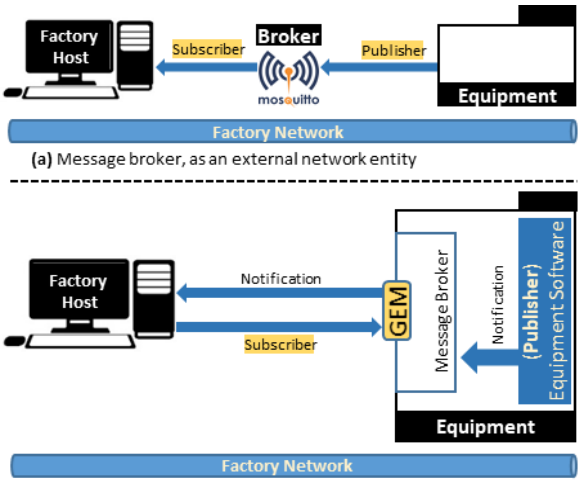

Fig. 8. Internal (embedded) and external message broker architectures.

The simple S1F13 (connection establishment request) message from the host with an empty list is depicted in Fig. 6. The W-bit in S1F13 message is present, which signifies that the host computer is expecting a reply message from equipment. The message size is then computed based on that data passed from the SECS-II layer, and in this particular example, the data portion is consisting of an empty list; therefore, the payload size is just 2-bytes. The total message size is 16 bytes (i.e., 4-length bytes, 10-header bytes, 2-data bytes). It can be observed that list data item has a Format Code and length bytes required to encode data.

The equipment will reply to the S1F13 message with the S1F14 message and fill the data portion with the necessary information. The reply message shown in Fig. 7 contains two lists, one binary data item and two ASCII data items. The memory packaging of the data portion is shown in Fig. 7 (c). It must be observed that the presence of a list data item increases the payload size and adversely impacts on the data density.

Fig. 9 shows SECS-II reply message S1F14 captured and visualized in Wireshark. It can be seen in the figure that the message contains a list of two items. Surprisingly, the HSMS protocol only shows 33 bytes as packet length; whereas, the TCP shows 37-bytes payload. The 4-bytes are prefixed with HSMS packet, and these 4-bytes determine packet length. These 4-length bytes are prefixed in all HSMS packets and are stripped-off while packet processing at HSMS level.

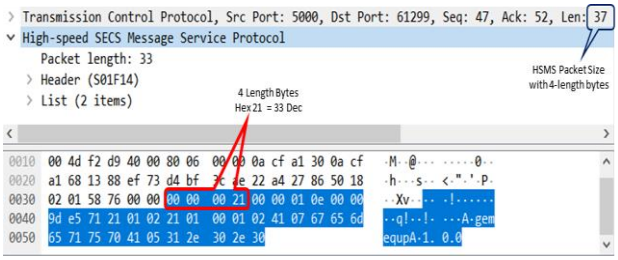

Fig 9. Wireshark capture of S1F14 reply message

C. SECS/GEM Message Broker

SECS/GEM is a publish-subscribe messaging protocol. One concern with a design model for publisher/subscriber in an industrial application is, irrespective of subscriptions, a publisher usually needs to send everything to a message broker. Unlike other messaging protocols, such as DDS and MQTT [10], where entities (i.e., host, equipment) communicate with the external broker over the network all the time. The SECS/GEM broker is typically built and integrated within the equipment's GEM interface software and runs on one of the equipment's computers. In this manner, the communication between broker and equipment software is confined to an internal network that is usually isolated from the industry's communication network. The built-in and integrated broker allows a SECS/GEM interface to send only subscribed messages to the host over the network. The embedded broker within the GEM equipment becomes bandwidth-efficient and requires much less bandwidth in comparison with protocols having message brokers running as a separate entity.

The host and equipment communicate with each other through GEM interface, which acts as a broker. The Publish/Subscribe model is utilized with dynamic subscriptions. Only the host is destined to receive message notifications [32]. The same goes with GEM

data collection features, such as alarms, trace data collection, and event reporting. The design is so flexible that engineers could design a unified GEM interface to circulate messages according to requirements. Without upsetting the existing host system, the equipment manufacturer could also upgrade a GEM interface with novel attributes. The MQTT broker architecture as an external entity is depicted in Fig. 8 (a). Contrarily, the SECS/GEM broker architecture embedded inside equipment is illustrated in Fig. 8 (b).

D. SECS/GEM Alarms and Data Collections

Real-time data collection is a principal requirement, which is achieved through a GEM interface with the help of status variables, data variables and equipment constants. A brief treatise on each variable types is given below:

- *Status Variables* contain equipment details that shall be accurate and valid at all times. A host can collect values of status variables using messages such as S1F3, S6F19, and S2F23.
- *Data Variables* contain useful information when an event occurs. For instance, when a recipe change is triggered, a PPChanged event will occur, which will eventually update the value of the ChangedRecipe data variable. It is important to note that the data variables are relevant only when an event takes place and may otherwise yield invalid results.
- *Equipment Constants* are global variables used to dictate the behavior of equipment. These variables include device configurations and settings. The host may set or modify these system variables, which can alter the equipment's behavior.

In addition to the variables mentioned above, a GEM interface provides other means of data retrieval, such as Collection Events and Alarms. With collection events and alarms, the host can monitor and track equipment operations in detail. A GEM alarm follows the publish/subscribe model and alerts the host, if any untoward situation jeopardize the equipment.

V. Issues, Limitations and Challenges

The manifold issues, limitations, and challenges of SECS/GEM are highlighted below:

Closed Firewall: Even though SECS/GEM is a publish/subscribe protocol and equipment publish data at a very fine grain level, the broker is implemented within the equipment itself, which requires subscribers to establish a connection with the equipment to subscribe for a specific topic. Implementing a message broker embedded within the equipment itself has an advantage that it requires minimal network bandwidth consumption as only data for the subscribed messages actually leaves the equipment, and everything else continues to communicate inside the equipment itself. However, SECS/GEM requires equipment/host to open and listen to incoming requests on a port (usually port 5000) for message subscriptions. Keeping inbound ports open on equipment poses a potential threat and makes the system vulnerable. Keeping all inbound firewall ports closed at the plant resolves many security issues for industry; however, such a solution may not work in current protocol specifications.

Single-Session Architecture: The implementation of SECS limits equipment to be connected with only one host device at a time. Contemporary protocol specifications and implementation permit only an active point-to-point communication link with factory hosts and restrain equipment from accepting connections from other entities. The imposed restriction makes it impossible to connect equipment with preceding or succeeding machines on the shop floor for an effective machine to machine communication to facilitate a decision-support process. For example, it is not possible for equipment configured in passive mode to instruct its preceding equipment directly to adjust recipe parameters because all equipment in the assembly line are usually configured in passive mode. Under such barricades and limitations, the newly developed factory applications, such as advanced process control (APC), factory automation, and advanced planning and scheduling (APS) systems require a steady stream of accurate real-time data, even in the form of peer-to-peer communications to facilitate a decision-support process [13].

Manual Integration: The factory host system cannot possibly scan the SECS/GEM interface for its full capacities because different vendors may have configured the same message pairs for different purposes as best suited to their applications. Consequently, the manual integration of the SECS/GEM model specifications cannot possibly be avoided. Thus, it requires significant manual customization efforts for application to function as desired.

Security: Previously, the focus of cybersecurity in the manufacturing and automation industry was primarily aiming to secure organizational perimeters, i.e., preventing unauthorized access to the industrial network. However, the importance of cybersecurity has recently been realized, and numerous studies have been conducted to propose security models suitable for industry 4.0 [33] and the Industrial Internet of Things (IIoT). For the well-functioning of an industrial network, a trust relationship with all participating machines is essential for communication devices, because a single affected machine/device may become malicious and trigger disasters, including property theft and loss of lives. Therefore, it is of utmost importance that equipment/machines communicating in the IIoT environment must establish a trust relationship and communicate with authorized devices only.

Unfortunately the SECS/GEM protocol is pregnable in its security mechanism and offers no security features to establish a network connection with other entities in the network, which provides opportunities to adversaries to launch a DoS attack and sabotage reputation and productivity. Furthermore, the host computer and factory equipment exchange binary-encoded SECS/GEM messages without any integrity check. This provides opportunities to adversaries to launch cyber-attacks such

as DoS attack, false data injection attack, time delay attack, data tempering attack, replay attack, and spoofing attack by manipulating message contents with fraudulent information to disrupt the standard processing and functioning of factory equipment. The potential damage caused by cyber-attacks is significant and devastating in terms of business continuance, theft of confidential data, and reputational damage.

Discoverability: The SECS/GEM interfaces are not discoverable, i.e., the factory automation system cannot query the SECS/GEM interface to determine its capabilities. Regardless of the complexity of the factory automation program, the SECS/GEM model needs an enormous manual integration effort for factory equipment to integrate with the existing machinery.

Interoperability: The main focus of the SECS-II standard is on defining messages and their contents, and it does not specify how messages are put together to accomplish a specific task. The equipment suppliers are required to decide which messages to combine to automate the intended tasks. Despite producing devices with similar functionalities and features, the GEM interface of each vendor is somewhat different from one another. Different vendors may choose different message combinations to automate a task, which makes it difficult to create a translator program to communicate with ERP and other external systems in the industrial network.

Extendibility: The SECS/GEM interface offers a vast pool of user-defined streams and functions to expand factory equipment's operational capabilities. Streams are numbered from 1-127 and each stream has functions from 1-255. It is of paramount importance to note that the standard reserves streams 1-63 and functions 1-63 in each such stream for standard specific messages and operations. However, functions from 64-255 in streams 1-63 and functions from 1-255 in streams 64-127 are available for customized user-specific messages. The primary objective of implementing the SECS/GEM specification was the creation of a generic equipment model with characteristics and functions to communicate and exchange information with the equipment manufactured by different suppliers. However, problems arise when different vendors use the same message-pair for different equipment operations, which results in incompatibility among inter-vendor equipment.

VI. Conclusion and Future Outlook

The manufacturing industries vouched to remain productive and competitive rather than impoverished by recognizing benefits of Industry 4.0, which includes enhanced productivity and performance, increased versatility and agility, higher profitability, and better customer service. Industry 4.0 technologies offer discreet knowledge of the manufacturing process, supply chains, distribution chains, and key performance indicators about business. The plethora of the advantages offered by Industry 4.0 would still be a burning topic for healthy debate in the context of exciting innovations portrayed by Smart Factory.

Critical analysis of essential M2M communication protocols, i.e., SECS/GEM, DDS, OPC UA, and MQTT, available to the manufacturing industry, was carried out in this treatise. The majority of these protocols offered connectivity and performance but were deficient in protection, interoperability, discoverability, extensibility, and so on. In this research, the outstanding qualities and eccentricities of the SECS / GEM protocol were made conspicuous. The SECS/GEM was observed to be insecure, bare, and vulnerable to Replay, Impersonation, and Denial of Service attacks. This could unleash potential damage in terms of business continuity, theft of confidential data, and reputational loss.

Notwithstanding, the door to research might be open in the security arena, where anything like authentication and encryption features could be implemented in the SECS/GEM to make it impregnable to attacks. Augmented further, the SECS/GEM was a point-to-point communication protocol offering Active/Passive modes for connection establishment. The equipment configured in a passive mode could only exchange information with the host rather than communicate with other equipment on the production line. This suppresses timely decision making, which results in reduced production and wastage of material. Hereafter, prospects might be there to amend design for many-to-many communication so that timely decisions and better productivity could be obtained.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Shams A. Laghari conducted the research, performed testing, and wrote the paper; Selvakumar Manickam supervised the research and co-wrote the paper. Shankar Karuppayah analyzed the data and reviewed the paper. All authors have approved the final version.

References

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

[2] O. Taburiaux. Simulating industrial control systems using mininet dissertation presented by Bertrand MASSET. 2018. [Online]. Available: http://hdl.handle.net/2078.1/thesis:14706.

[3] Y. Aheleroff, S. Xu, X. Lu, Y. Aristizabal, M. P. Velásquez, J. Joa, and B. Valencia, "IoT-enabled smart appliances under industry 4.0: A case study," *Advanced Engineering Informatics*, vol. 43, #101043, Jan. 2020.

[4] S. Galli, "Towards a new technological paradigm based on Industry 4.0: Opportunities and challenges for innovation policies," Doctoral dissertation, University of Trento, 2017.

[5] S. Azaiez, F. Tanguy, and M. Engel, "Towards building OPC-UA companions for semi-conductor domain," *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, 2019, pp. 142–149.

[6] O. A. Amodu and M. Othman, "Machine-to-machine communication: An overview of opportunities," *Comput. Networks*, vol. 145, pp. 255–276, Nov. 2018.

[7] H. Yang, S. Kumara, S. T. S. Bukkapatnam, and F. Tsung, "The internet of things for smart manufacturing: A review," *IISE Trans.*, vol. 51, no. 11, pp. 1190–1216, 2019.

[8] R. Matischek and B. Bara, "Application study of hardware-Based

security for future industrial IoT," in *Proc. of Euromicro Conf.*, 2019, pp. 246–252.

[9] 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. of IEEE Int. Conf. on Ind. Technol.*, 2019, pp. 955–962.

[10] D. Soni and A. Makwana, "A survey on mqtt: a protocol of internet of things (IoT)," presented at Int. Conf. on Telecommunication, Power Analysis and Computing Techniques, 2017.

[11] G. Pardo-castellote, B. Farabaugh, and R. Warren, "An introduction to DDS and data-centric communications," *Real-Time Innov.*, August 2005.

[12] Z. Y. Thean, V. Voon Yap, and P. C. Teh, "Container-based MQTT broker cluster for edge computing," in *Proc. of 4th Int. Conf. on Recent Adv. Innov. Eng. Thriving Technol.*, 2019, pp. 1–6.

[13] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–29, 2019.

[14] G. Baptista, F. Carvalho, S. Colcher, and M. Endler, "A middleware for data-centric and dynamic distributed complex event processing for iot real-time analytics in the cloud," presented in 34th Brazilian Symposium on Computer Networks and Distributed Systems, Salvador, Brazil, 2016.

[15] J. Yang, K. Sandström, T. Nolte, and M. Behnam, "Data distribution service for industrial automation," in *Proc. IEEE 17th Int. Conf. on Emerging Technologies & Factory Automation.*, 2012, pp. 1–8.

[16] M. Friesen, G. Karthikeyan, S. Heiss, L. Wisniewski, and H. Trsek, "A comparative evaluation of security mechanisms in DDS, TLS and DTLS," *Kommun. und Bild. der Autom.*, pp. 201–216, 2020.

[17] T. Dierks and E. Rescorla. (2008). RFC 5246-the transport layer security (TLS) protocol version 1.2. [Online]. Available: https://tools.ietf.org/html/rfc5246

[18] E. Rescorla and N. Modadugu. (2012). RFC 6347: Datagram transport layer security version. [Online]. Available: https://tools.ietf.org/html/rfc6347

[19] P. Nenninger, M. Gierl, and R. Kriesten, "A contribution to publish-subscribe based communication in industrial applications," in *Proc. IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conf.*, 2019, pp. 424–429.

[20] F. Stoop, G. Ely, R. Menna, G. Charache, T. Gittler, and K. Wegener, "Smart factory equipment integration through standardised OPC UA communication with companion specifications and equipment specific information models," *Int. J. Mechatronics Manuf. Syst.*, vol. 12, no. 3–4, pp. 344–364, 2019.

[21] S. Standard. Features & benefits of GEM: A guide to some of the main features and benefits of GEM – SEMI Standard E30. *cimetrix*. [Online]. Available: https://www.cimetrix.com/blog/features-and-benefits-of-the-secs-gem-communication-standards

[22] K. M. Goh, *et al.*, "Semiconductor equipment and materials international interface and communication standards: An overview with case studies," in *Industrial Communication Technology Handbook*, CRC Press, 2017, pp. 61–66.

[23] "SECS Messaging Primer," 2016.

[24] Introduction to SECS/GEM. [Online]. Available: http://www.hume.com/secsintro.htm

[25] K. Jung, J. S. Han, Y. M. Lim, and W. S. Kim, "XML format design for SECS-II message monitoring," in *Proc. Sixth Int. Conf. on Advanced Language Processing and Web Information Technology*, 2007, pp. 548–552.

[26] 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," Google Patents, 2012.

[27] L. Ma, N. Zhang, and Z. Zhang, "Tool efficiency analysis model research in SEMI industry," in *Proc. E3S Web Conf.*, 2018, pp. 1–5.

[28] J. Postel. (1981). RFC 793: Transmission control protocol. [Online]. Available: https://tools.ietf.org/html/rfc793

[29] T. Transfer. (1997). Generic equipment model (GEM) specification manual : The GEM specification as viewed from the host. [Online]. Available: https://www.academia.edu/27932413/Generic_Equipment_Model_GEM_Specification_Manual_The_GEM_Specification_as_Viewed_from_the_Host

[30] J. Wang and B. Wang, "Application of GEM in semiconductor equipment," *Mech. Electr. Eng. Mag.*, vol. 55, no. 7, pp. 34–36, 2008.

[31] B. Almadani, M. N. Bajwa, S. H. Yang, and A. W. A. Saif, "Performance evaluation of DDS-based middleware over wireless channel for reconfigurable manufacturing systems," *Int. Journal of Distributed Sensor Networks*, vol. 11, no. 7, 2015.

[32] B. Esmaeilian, S. Behdad, and B. Wang, "The evolution and future of manufacturing: A review," *Journal of Manufacturing Systems*, vol. 39, pp. 79–100, Apr-2016.

[33] S. Michel, L. Dalpra, T. Wagner, and P. Llerena, Status report Industry 4.0., Upper Rhine 4.0, pp. 1–65, 2020.

Shams A. Laghari received his B.Sc (Hons). and M.Sc. degrees in computer science from University of Sindh, Jamshoro, Pakistan. He received his M.S. degree in computer science from PAF-KIET Karachi, Pakistan. He is currently pursuing his Ph.D. degree in network security at National Advanced IPv6 Centre, Universiti Sains Malaysia. His research interests include Network Security, Industry 4.0, distributed systems, cloud computing, and mobile cloud computing.

Selvakumar Manickam is an associate professor working in cybersecurity, the Internet of Things, Industry 4.0, and Machine Learning. He has authored or co-authored more than 160 articles in journals, conference proceedings, and book reviews, and has supervised 13 Ph.D. students. He had 10 years of industrial experience prior to joining academia. He is a member of technical forums at national and international levels. He also has experience building IoT, embedded, server, mobile and web-based applications.

Shankar Karuppayah is a currently a Senior Researcher / Postdoc in the Telecooperation group, TU Darmstadt since July 2019. He is also a Senior Lecturer at the National Advanced IPv6 Centre (NAv6), Universiti Sains Malaysia since 2016. He obtained his B.Sc. (HONS) Computer Science from Universiti Sains Malaysia in 2009 and his M.Sc. Software Systems Engineering from King Mongkut's University of Technology North Bangkok (KMUTNB) in 2011. In 2016, Shankar Karuppayah obtained his Ph.D. degree from TU Darmstadt with his dissertation titled "Advanced Monitoring in P2P Botnets". He is currently working actively on several cybersecurity projects and working groups, e.g., the National Research Center for Applied Cybersecurity (ATHENE), formerly known as Center for Research in Security and Privacy (CRISP).