An Efficient Key Management and Multi-Layered Security Framework for SCADA Systems

Darshana Upadhyay, Marzia Zaman, Rohit Joshi, and Srinivas Sampalli, *Member, IEEE*

***Abstract*—Supervisory Control and Data Acquisition (SCADA) networks play a vital role in industrial control systems. Industrial organizations perform operations remotely through SCADA systems to accelerate their processes. However, this enhancement in network capabilities comes at the cost of exposing the systems to cyber-attacks. Consequently, effective solutions are required to secure industrial infrastructure as cyber-attacks on SCADA systems can have severe financial and/or safety implications. Moreover, SCADA field devices are equipped with microcontrollers for processing information and have limited computational power and resources. This makes the deployment of sophisticated security features challenging. As a result, effective lightweight cryptography solutions are needed to strengthen the security of industrial plants against cyber threats. In this paper, we have proposed a multi-layered framework by combining both symmetric and asymmetric key cryptographic techniques to ensure high availability, integrity, confidentiality, authentication and scalability. Further, an efficient session key management mechanism is proposed by merging random number generation with a hashed message authentication code. Moreover, for each session, we have introduced three symmetric key cryptography techniques based on the concept of Vernam cipher and a pre-shared session key, namely, random prime number generator, prime counter, and hash chaining. The proposed scheme satisfies the SCADA requirements of real-time request response mechanism by supporting broadcast, multicast, and point to point communication.**

***Index Terms*—SCADA Systems, random number generator, symmetric key cryptography, public key algorithm, cyber security, network attacks, key management.**

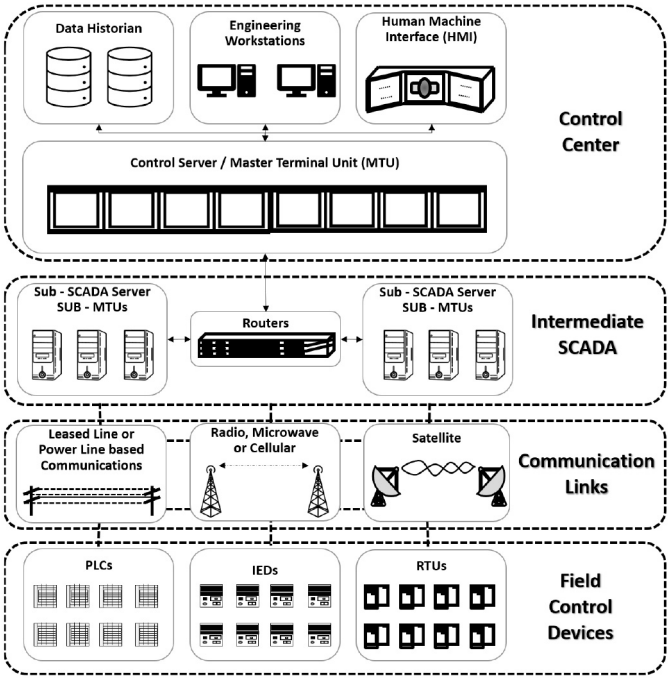

Fig. 1. Block diagram of a SCADA system, Legend: MTU: Master Terminal Unit, PLCs: Programmable Logic Controllers, RTUs: Remote Terminal Units, IEDs: Intelligent Electronic Devices.

I. Introduction

THERE has been a surge in the deployment of Supervisory Control and Data Acquisition (SCADA) systems to control and monitor industrial infrastructure over the Internet [1]. Organizations such as oil and natural gas, power stations, water & sewage systems, chemical plants, manufacturing units, railway, and other transportation use SCADA systems to monitor and control their infrastructure such as oil pipelines, solar panels, water pipelines, boilers, railway tracks, and plant floor components across open access networks [2], [3].

A SCADA system typically includes a control server (also known as Master Terminal Unit (MTU)), SUB-MTUs, communication links (e.g., satellite, radio or microwave links,cellular network, switched or lease lines and power-lines), and geographically dispersed field control devices, namely, Programmable Logic Controllers (PLCs), Remote Terminal Units (RTUs), and Intelligent Electronic Devices (IEDs) [2], [4]. The block diagram of a typical SCADA system is depicted in Figure 1.

For continuous monitoring and control of plant floor devices, sensors and actuators are used to measure different attributes of machinery and transmit that information to field devices [5]. Further, the field control devices, namely, PLCs, RTUs, and IEDs supply digital status information to the MTU (typically placed at the remote location) to determine the acceptable ranges according to parameters set in the server. This information will then be transmitted back to the field control device(s) where actions may be taken to optimize the

Manuscript received October 29, 2020; revised March 9, 2021 and June 22, 2021; accepted August 4, 2021. Date of publication August 17, 2021; date of current version March 11, 2022. The authors gratefully acknowledge the support in part by the Natural Sciences and Engineering Research Council (NSERC), Canada, through a Collaborative Research Grant. The associate editor coordinating the review of this article and approving it for publication was J. Zhang. *(Corresponding author: Srinivas Sampalli.)*

Darshana Upadhyay and Srinivas Sampalli are with the Faculty of Computer Science, Dalhousie University, Halifax, NS B3H 1W5, Canada (e-mail: srini@cs.dal.ca).

Marzia Zaman and Rohit Joshi are with the Research and Development Department, Cistel Technology Inc., Ottawa, ON K2E 7V7, Canada.

Digital Object Identifier 10.1109/TNSM.2021.3104531

performance of the system. Moreover, the status information is stored in a database and is displayed on a Human Machine Interface (HMI) at the control center, where operators can interact with the plant floor machinery for centralized monitoring and system control [6]. Large SCADA networks such as those on a power plant require hundreds of field devices and dedicated subsystems to reduce the load on the centralized server [2].

SCADA communication messages have sensitive information as they are used to monitor and control the plant floor devices. For example, in water and sewage systems, the communication messages are used to raise and lower water tank levels or open and close the safety valves. Since these control devices are operated and monitored remotely, they can make them high-value targets for attackers to launch various cyber-attacks that can compromise the control systems, communication, and emergency services. Consequently, one of the critical aspects of the SCADA systems is secure transmission of messages so that they cannot be tampered during the communication. Moreover, the SCADA devices must be authenticated and maintain confidentiality of the information during the transmission so that no interceptor can misuse the system.

In the last few years, many key management techniques have been published to secure SCADA communication, namely, SCADA key establishment (SKE), SCADA Key Management Architecture (SKMA), Advanced SCADA Key Management Architecture (ASKMA), Hybrid Key Management Architecture (HKMA) and Advanced Hybrid SCADA Key Management Architecture (AHSKMA), Limited Self-Healing key distribution (LiSH) [7], [8], [9], [10], [11], [12]. These techniques fall under two main categories, namely, centralized key management and decentralized key management schemes. Moreover, each of these categories uses three approaches to generate and extract the session key, namely, symmetric, asymmetric, and hybrid. The drawback of the centralized scheme is that if the key distribution center (KDC) is down, the communication is cut off, which is not acceptable in SCADA systems. In a decentralized approach, the keys are created using keying material and may only affect the single communication link in case of a breakdown.

The symmetric key based approach is efficient in terms of message integrity and high availability, but does not provide authentication and confidentiality. On the other end, asymmetric key provides message integrity, authentication, and privacy, but may compromise availability. Hence, hybrid techniques are more suitable for SCADA systems. A few key management techniques have been proposed using hybrid methods. For example, Rezai *et al.* [10] propose an advanced Hybrid key management architecture (HSKMA), which improves the key management architecture proposed by Choi *et al.* [11]. However, it uses a centralized KDC to distribute the keys. Moreover, the communication between the MTU and the sub-MTU is established using Elliptic-Curve Cryptography (ECC) based asymmetric key cryptography while the sub-MTU and the RTU communicate using Rivest–Shamir–Adleman (RSA) asymmetric key cryptography. The same approach has been used to enhance the scheme proposed by Rezai *et al.* [13] using a decentralized system in [9]. In this scheme, the master keys are refreshed using ECC and symmetric cryptography is used for encryption, decryption, and session key updates. However, this scheme does not validate the message integrity and authentication. Moreover, none of the previous methods has practical implementation proof that it provides immunity against quantum attacks [14]. Furthermore, it has been known that RSA does not guarantee perfect forward secrecy [11]. In summary, none of the techniques covers all the security aspects.

The forgoing discussion brings in the need for an effective cryptography solution that will prevent these systems from potential breaches. The objective of this paper is to propose a robust & low-cost security framework for automated industries to mitigate various security flaws and cyber-attacks. The proposed work aims to offer a multi-layered security framework for industrial infrastructures by combining both symmetric and asymmetric key cryptography techniques. This novel approach follows a layered architecture, where the MTU and sub-MTU can communicate using a hybrid technique for an entire session while the sub-MTU and RTU can communicate using symmetric key cryptography once the session key is securely exchanged. Also, we have proposed a novel approach to generate symmetric keys using vernam cipher rather than using existing methods such as 3DES, AES, etc. Furthermore, the proposed scheme satisfies SCADA requirements of real-time request-response mechanism by supporting broadcast, multicast, and point-to-point communication.

A. *Contributions of the Paper*

1) We propose a secure session-key agreement scheme according to SCADA protocol standards to ensure the security amongst MTU, sub-MTUs and RTUs. For that, a true random number generator based on current date and time (CDT) and a fraction of the square root of a prime number (FSRP) are used to generate the session key. Moreover, these elements are shared by XORing them to enhance the privacy of the shared secrets. Furthermore, the dynamic HMAC is derived using the value of FSRP. Moreover, using these same elements, the HMAC is derived to validate the integrity of the message. This reusability of the elements increases the computational speed of session key, symmetric key and HMAC derivation. The randomness of key and HMAC offers immunity against various attacks such as correlation attacks, length extension attacks, etc.
2) We propose a novel approach to generate symmetric keys in the Vernam cipher by combining prime counter and hash chaining techniques. The mathematical property of the fraction square root of prime number (FSRP) is used, which returns a non-terminating, non-repeating irrational number. In a recent publication by Manjunatha *et al.* [15], the authors propose Vulgar fractions to generate a complex key with secured seed exchange for the Vernam cipher. This fraction is generated by dividing a small number by a large

prime number, resulting in a fraction number [15]. For example, frac(1/7) = 0.1428571428571 generates long strings with a repetitive sequence of digits. However, our proposed approach advances that method by generating completely random and non-repeating decimal numbers using the concept of FSRP. For example frac(sqrt (7)) = 0.645751311064590590501615753639 3 returns long strings without repetitive sequence of digits.

3) We propose a multi-layered framework by integrating the concept of symmetric and asymmetric key cryptography that ensures various security mechanisms, namely, authentication, confidentiality, message integrity, availability, and scalability for SCADA systems. The proposed method for symmetric key cryptography is based on the Vernam cipher, which provides protection against all the cryptographic attacks while the NTRU based post-quantum public-key algorithm resists quantum and data harvest attacks.
4) We identify an efficient cipher suite by comparing and analyzing various private and public key algorithms for the proposed framework by considering multiple factors, namely, prevention mechanism against classical and quantum attacks, key storage cost, the randomness of key and computational speed. The proposed cipher suite overcomes the weaknesses of the cipher suite offered by the American Gas Association (AGA) security standards [14], [16].

B. Outline of the Paper

The rest of this paper is organized as follows. Section II describes related research in the areas of key management and encryption schemes foe SCADA systems. Section III, presents the reasoning of choice of the algorithms. The proposed multi-layered framework for secure SCADA communication is introduced in Section IV, which covers secure key and information exchange. Section V presents the complete experimental setup which includes algorithm selection for cipher suites, computational speed of proposed framework, randomness evaluation of symmetric key, and calculation of the cost of the keys. Section VI presents the comparative studies with the state-of-the-art techniques in terms of security analysis, storage cost, and execution speed. Section VII concludes the paper.

II. Background and Related Work

A. Literature Survey

SCADA networks are typically configured using proprietary protocols such as Modbus, IEC 61850, IEC 60870, DNP3, and Profinet, which do not support secure data communication. Moreover, the remote procedure call (RPC) follows open link communication and one of the real-time examples of the consequent vulnerability was the Blaster worm [2]. Furthermore, many network sniffing tools are freely available to view and gather the network traffic [17]. Therefore, secure data transmission is one of the important requirements for SCADA systems. Key management and encryption play a vital role in securing SCADA communication. Typically, in a SCADA communication, the MTU sends control signals to the RTUs to control the plant floor devices, which require three types of communication, namely, broadcast, multicast, and point to point. However, controller RTUs may need to operate other field RTUs. In case of an emergency shutdown, to acquire the clock information or synchronization, MTUs broadcast the signal to all the control devices such as RTUs, IEDs, and PLCs. To operate a specific substation device, the MTU requires multicast communication, whereas monitoring and controlling the plant for machinery typically requires point-to-point communication. Therefore, while designing a secure framework for SCADA networks, it is crucial to cover all three types of communication.

During the last two decades, many key management schemes have been proposed, which typically fall into two categories, namely, centralized key distribution such as [4], [7], [18], [19], and decentralized key distribution scheme such as [9], [20], [21], [22]. In the centralized scheme, the Key Distribution Center (KDC) plays a vital role in generating and distributing the secret keys to establish secure communication between the communication parties. In contrast, the decentralized scheme requires pre-shared keying material that is used to create the session key. Once the session key is derived using keying essence, further communication takes place using that key. Furthermore, some key management schemes use the public key-based technique to establish secure transmission. Although this method is time-consuming and power-consuming, various research studies suggest that ECC is a suitable public-key cryptosystem [4], [9], [11].

Sandia Labs proposed a SCADA key establishment (SKE) method for managing cryptographic keys in the network [7]. This scheme is proposed for point-to-point communication amongst MTU, sub-MTU, and RTU and uses the symmetric key technique to establish secure communications between sub-MTUs and RTUs, while sub-MTUs and MTUs communicate using public key cryptography. For the symmetric key, the session key is generated using three types of keys, namely, long term key (LTK), general seed key (GSK), and general key (GK) [7]. KDC assigns public and private key pair to each sub-MTU and MTU. However, this method does not support broadcast, multicast, and RTU to RTU communication. Moreover, it increases the overall key storage overhead and complexity as the long-term keys are managed manually. In [19], the authors propose a SCADA Key Management Architecture (SKMA) for secure session key management, which enhances the capability of SKE. While the SKE uses both a public key algorithm and a symmetric key algorithm, the SKMA uses only symmetric encryption algorithm. SKMA generates a session key using a pseudorandom function, keyed by the node-node key, and a timestamp that is based on the duration of the session. SKMA uses key establishment protocol based on ISO 11770-2 mechanism [8]. However, the scheme does not provide secure message broadcasting but supports RTU-RTU communication. Moreover, it does not provide any confidentiality and integrity.

Advanced SCADA Key Management Architecture (ASKMA) supports both message broadcasting and secure communications. Furthermore, evenly spreading the total amount of computation across the high power nodes (MTU or

SUB-MTU) significantly avoids the performance bottleneck and keeps minimal burden on the low power nodes (RTU). It uses the LKH (Logical Key Hierarchy protocol) to construct a logical tree of symmetric keys. Each member knows all the symmetric keys from its leaf to the root, and if any new node joins the group, LKH updates the entire set of symmetric keys from its leaf to the root. Although the overall performance of ASKMA has many advantages, it can be less efficient during the multicast communication process. To solve this issue, ASKMA+ was proposed [7]. ASKMA+ divides the key structure into two classes, by applying the IoLus framework to construct each class as a logical key hierarchy (LKH) structure. Through this key structure, the authors proposed a more efficient key-management scheme supporting efficient multicast communication by considering the number of keys stored in a remote terminal unit (RTU). However, ASKMA+ does not address the availability issue in SCADA.

To satisfy the availability requirement, Hybrid Key Management Architecture (HKMA) and Advanced Hybrid Scada Key Management Architecture (AHSKMA) were proposed [10], but there is a chance that field devices will stop working during the replacement of field control devices. To solve this issue, Choi *et al.* propose a hybrid key management scheme [11]. A centralized key distribution (CKD) protocol is applied between the sub-MTU and MTU, and LKH protocol is applied between sub-MTU and RTU. However, if the centralized key distribution server breaks down, the entire approach fails to execute the protocol. Rezai *et al.* [9] also use a hybrid key management method using ECC. Jiang *et al.* [12] propose Limited Self-Healing key distribution (LiSH), which offers revocation capabilities along with collusion-resistance for group communication in SCADA systems. The LiSH+ is used to address the dynamic revocation mechanism, which enhances the base method of LiSH. Kang *et al.* [21] propose a scheme for radial SCADA systems based on a pre-shared session key that relies on symmetric key cryptography. This solution enhances the performance of the radial SCADA system by using the master key concept.

AGA-12, Part 2, provides security features offering a new security protocol standard [23]. It uses cipher suites to secure communication amongst SCADA field devices, which covers authentication, confidentiality, and integrity. However, it fails to provide faster execution. Furthermore, it does not offer prevention against quantum and Denial of Service (DoS) attacks. In addition, AGA-12 uses the RSA algorithm for encryption, which was recently cracked and also does not provide key management [14]. The other security standards, such as IEC 62210, IEC 62351, fail to offer security against man-in-the-middle (MiM) attacks and also lack strong key management. A novel key distribution method was proposed for smart grids in [24] which uses identity-based cryptography. This method adopts a hybrid approach to counteract man-in-the-middle and replay attacks. However, this method does not cover the authentication of the SCADA components. The authors in [25] introduce the authentication and authorization roles for SCADA devices using attribute-based access control. The hybrid Diffie-Key exchange, along with the authentication scheme, was proposed in [26]. This scheme uses RSA and AES for session key generation and encryption. However, it does not provide high availability.

B. Research Gaps

Originally, the objective of SCADA systems was to focus on accurate and efficient process execution at the plant floor rather than aiming to secure communication. While accessing the plant machinery remotely through SCADA systems accelerates the industrial processes, it compromises the security by exposing the systems to the outside world [24]. Consequently, unauthorized parties such as hackers, intelligent foreign agents, and corporate saboteurs, can exploit the weaknesses to compromise industrial systems. Typically, general safeguards include restricted perimeters, patch management, strong cryptography and most importantly, separation of the control network and corporate network through the defense-in-depth mechanism [1], [22]. However, these security guards are difficult to deploy owing to legacy-inherited security weaknesses, and that significantly increases the chances of possible exploitation during real-time communication [2], [27].

Moreover, SCADA field devices such as PLCs, RTUs, and IEDs have resource and computational power limitations that make the deployment of sophisticated security features challenging [9]. Furthermore, availability, integrity, and confidentiality are the three fundamental security requirements of SCADA communication [19]. To circumvent threats against these security requirements, a robust security framework for key management schemes and lightweight encryption techniques are needed [9], [20]. Although many key management and encryption techniques have been proposed, few methods exist for secure key exchange for point-to-point communication while some are specifically intended for broadcast and multicast communication. Furthermore, none of the schemes satisfy all the requirements of secure SCADA communication and real-time request-response mechanism. Some private key based methods offer integrity and availability, while some public key based methods provide authentication and confidentiality. Hence, neither private nor public key based approach alone is sufficient [4], [28]. The development of a secured SCADA framework with hybrid efficient key management scheme and lightweight cipher is the primary research gap that is addressed in this paper.

C. Proposed Solution

The proposed system aims to provide a multi-layered security framework for industrial infrastructures by combining both symmetric and asymmetric key cryptography techniques. This novel approach covers major security aspects of the systems, namely availability, integrity, confidentiality, authentication and scalability. For that, an efficient session key management mechanism has been proposed besides lightweight ciphers by merging the concept of random number generator and Hashed Message Authentication Code (HMAC). Moreover, for each session, three symmetric key cryptography techniques are introduced, namely, random prime number generator,

prime counter, and hash chaining based on the concept of Vernam cipher and pre-shared session key. Furthermore, the proposed scheme satisfies SCADA requirements such as real-time request response mechanism by supporting broadcast, multicast, and point to point communication.

III. Reasoning of Choice of Algorithms

Many SCADA-based industrial systems, such as water & sewage control, energy and power plants, and gas pipelines, rely on real-time communication with limited computational resources. We have used the Vernam cipher for symmetric key cryptography because it is proven to offer an absolute secure solution theoretically, is easy to implement, and accelerates encryption & decryption by using low power and memory [29]. Therefore, it is an appropriate solution for embedded system devices. Moreover, the modulo-2 operator (XOR) used in the Vernam cipher provides faster execution and the flexibility in design of the onboard hardware [15]. By employing these features of the Vernam cipher, we can protect the data with low computational power and memory utilization.

The Vernam cipher provides complete secrecy as the key is unique and completely random for each message. An amount of time that is necessary to break any cipher and tamper with the data is based on the size and nature of the symmetric key. However, in the Vernam cipher, as the keys are random and unique for every message, an eavesdropper will be unable to guess the key even with unlimited computing power. Even asymmetric ciphers such as RSA can be broken with unlimited time and processing power [14]. Furthermore, the frequency analysis of the Vernam cipher is evenly distributed, and hence cryptanalysis will not produce any meaningful information.

The focus of the proposed framework is to provide high security along with high availability since SCADA communication depends on real-time request-response mechanisms. We can replace digital signature and asymmetric key cryptography by applying HMAC in symmetric key cryptography. This approach provides message authentication and integrity without compromising the execution speed during the communication between MTU and RTU.

Typically, HMAC depends on a shared secret key, which is exchanged using a trusted channel (in our case, we have used NTRU-based asymmetric key cryptography) between the sender and receiver to agree on the same key before starting the information exchange. The same secret key is combined with the MAC to generate HMAC at both the communication devices. However, the cryptographic strength of the HMAC depends on the size of the secret key, since brute force attacks are the most common attacks against HMAC.

In a typical key distribution scenario, the secret key is distributed over the trusted channel. Instead, in our proposed approach, we exchange the parameters of FSRP & CDT such as the index of FSRP and keysalt which are used to generate the secret key. Moreover, these parameters are reusable, and are not only used to generate the session key but also are applied to produce the key for HMAC. Furthermore, the key used in HMAC depends on the value of FSRP, which is generated by a random prime generator or a prime counter to produce a new key for each message. This makes brute force attacks computationally infeasible as the secret key used in HMAC is dynamically generated.

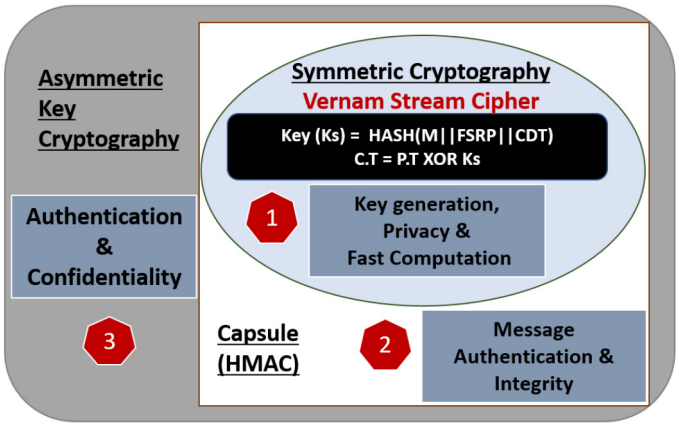

Fig. 2. Multi-layered framework for secure SCADA communication.

IV. Multi-Layered Framework for Secure SCADA Communication

This section presents the proposed multi-layered framework for secure SCADA communication. The framework uses three levels for robustness, namely, symmetric key cryptography, cryptographically secure HMAC function, and a public key algorithm. The security features of each phase are illustrated in Figure 2.

In our framework, a unique session key is generated for each connection between SCADA communication devices. The elements of this session key are securely shared using asymmetric key cryptography. This is called the key agreement stage. Furthermore, during this phase, the sender's authentication and recipient confidentiality are also validated using the private-public key pair. Moreover, HMAC is used for message authentication and integrity. Once both the communication parties agree on the reliable key exchange, further communications take place using symmetric key cryptography. The encryption of the original message is hashed, and subsequently, the symmetric keys are generated to encrypt the message using the lightweight Vernam cipher. After that, the cipher text and hash digest of this encrypted message are sent together over the communication channel. At the other end, the receiver device validates the message integrity using HMAC and then the cipher text is decrypted to receive sender's original message.

Since ICSs control field-site components at the plant floor, the activities related to controlling and monitoring of the elements should be done securely and efficiently [30]. For that, we have introduced two modules, namely, secure key exchange and secure information exchange. Moreover, secure information exchange consists of four methods, namely, Multi-Layered (ML) architecture, Random Prime Generator (RPG), Prime Counter (PC), and Hash Chaining (HC). While ML and HC offer very high security in SCADA networks, PC and HC

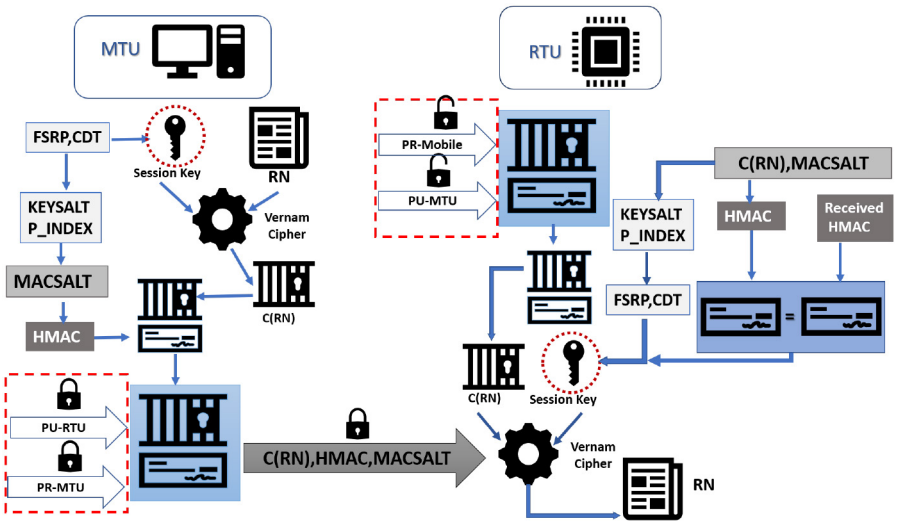

Fig. 3. Secure Key exchange mechanism for SCADA systems.

are proposed for time-critical applications. The RPG offers medium level security and availability.

A. Secure Key Exchange

The key agreement refers to three stages, namely, key generation at the sender side, key distribution over the communication channel, and key extraction at the receiver side.

1) Key Generation: During the key generation phase, a sender (MTU or RTU) uses three main elements, namely, a Random Number (*RN*), Current Date & Time (*CDT*), and Fraction of Square Root of Prime number (*FSRP*). Here, *CDT* and *FSRP* are used as secret elements to generate the session key. The choice of these two key elements is based on the property of generating true random numbers. *CDT* generates a random number every microsecond and to make it more random, we choose *FSRP*, which returns a non-terminating, non-repeating decimal number [31]. The session key (S_K) is derived by applying a hash function on both these elements by combining them, as in eq. (1).

$$S_K = \text{HASH}(CDT||FSRP) \tag{1}$$

These session key elements are securely distributed using *MACSALT*. The index of *FSRP* is combined with *KEYSALT* to generate *MACSALT*, where *KEYSALT* is derived by *XORing CDT* and *FSRP*. The formulas are given in eq. (2) & (3).

$$KEYSALT = CDT \oplus FSRP \tag{2}$$

$$MACSALT = KEYSALT||PRIME_{index} \tag{3}$$

Once S_K and *MACSALT* are generated, *RN* is encrypted using S_K which generates cipher of random number *C*(*RN*), as in eq. (4).

$$C(RN) = RN \oplus S_K \tag{4}$$

In this process, the algorithm produces a hash not only from the encrypted *RN* but also from the *CDT* & *FSRP* key elements. This derivation follows the procedure of *HMAC*, as given in the eq. (5) and is used to check message integrity.

$$HMAC_{sender} = \text{HASH}(C(RN), CDT||FSRP). \tag{5}$$

2) Key Distribution: The bundle of the *C*(*RN*), *HMAC* of *C*(*RN*), and *MACSALT* is securely sent over the communication channel using the private key of sender's (K_{spri}) and public key of receiver (K_{rpub}) that validate the sender's authentication and receiver's confidentiality as in eq. (6).

$$K_{rpub}\big(K_{spri}(C(RN), HMAC_{sender}, MACSALT)\big). \tag{6}$$

3) Key Extraction: At the receiver side, the private key of receiver and public key of sender is applied to validate authentication and confidentiality as in eq. (7).

$$Krpr(Kspu(C(RN), HMAC_{sender}, MACSALT))) \tag{7}$$

The elements of *MACSALT* are used to generate *FSRP* and *CDT*. $PRIME_{index}$ is used to extract the value of *FSRP* and *CDT* is obtained by XORing *FSRP* and *KEYSALT* as shown below in eq. (8)-(10).

$$MACSALT = KEYSALT||PRIME_{index} \tag{8}$$

$$FSRP = FRAC(SQRT(PRIME_{index})) \tag{9}$$

$$KEYSALT = CDT \oplus FSRP \tag{10}$$

Finally, the session key is derived by applying hash on *CDT* and *FSRP* as in eq. (11).

$$CDT = FSRP \oplus KEYSALT \tag{11}$$

$$HMAC_{receiver} = HASH(C(RN), CDT||FSRP) \tag{12}$$

HMAC is computed at the receiver using *C*(*RN*), *CDT* & *FSRP*, as in eq. (12) to compare with $HMAC_{sender}$ to check data integrity. The *HMAC* of the sender and receiver are checked, if both are equal it moves to the next step, else the message is discarded. The session key S_K is then validated using *CDT* and *FSRP* as in eq. (13). The session key is XORed with *C*(*RN*) to get the RN as shown in eq. (14).

$$S_K = \text{HASH}(CDT||FSRP) \tag{13}$$

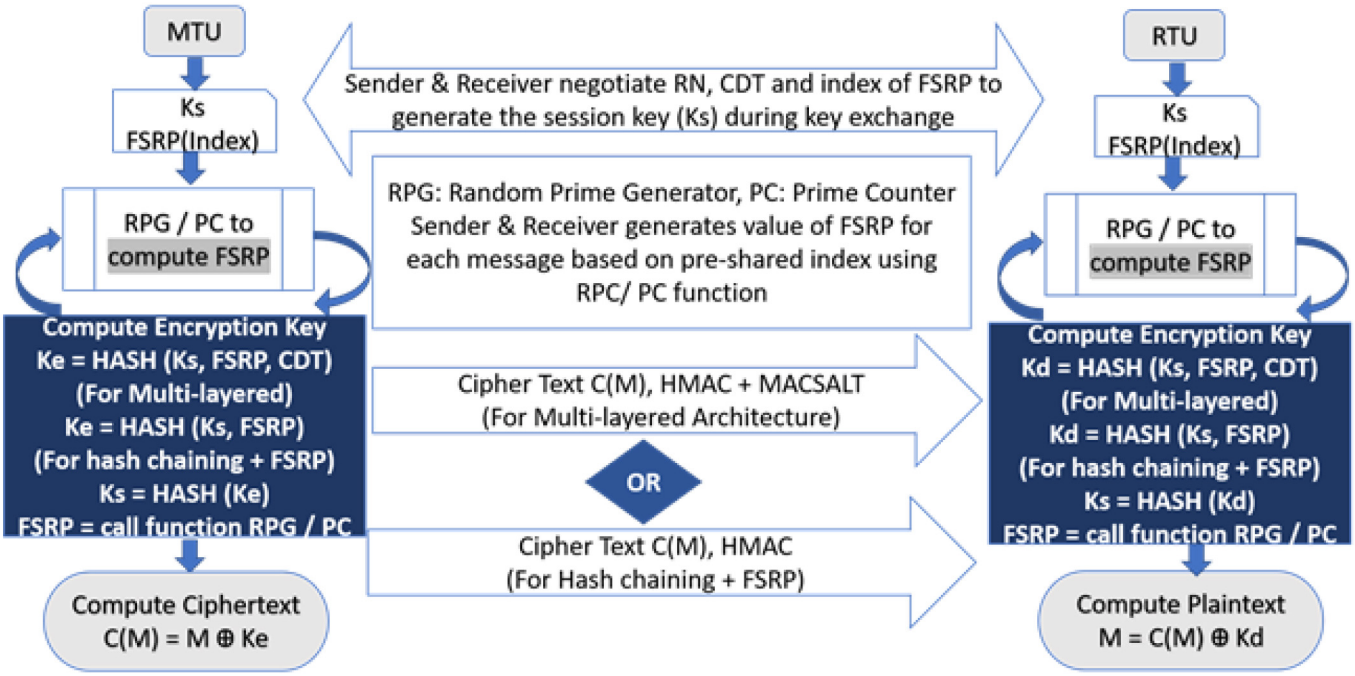

Fig. 4. Process diagram of encryption and decryption of data in secure SCADA communication.

$$RN = C(RN) \oplus S_K \tag{14}$$

The receiver will send an acknowledgement to the sender by encrypting $RN + 1$ using the same session key to validate secure key exchange. Figure 3 illustrates the secure key exchange mechanism between SCADA devices, namely, MTU and RTU.

The proposed scheme uses RN, CDT, and FSRP to generate the session key (Ks) for both the communication devices, namely, MTU and RTU. However, during the key exchange, these elements are not transferred openly, rather RN is encrypted by the key generated using the combination of CDT & FSRP. Moreover, the modulo-2 operator (XOR) is applied on CDT and FSRP to generate the keysalt which will be shared over the communication channel along with an index of FSRP and the cipher text of RN. The index of FSRP is considered as the root of trust for the entire scheme. Furthermore, the Vernam cipher is used for symmetric key cryptography, which requires a fresh key for each message during the encryption and decryption process. This symmetric key is generated using the session key (Ks) and key parameters, namely, CDT and FSRP, depending on the proposed approaches. The FSRP can be generated using a random prime generator (Method 2) or a prime counter (Method 3). Furthermore, hash chaining (Method 4) can be combined with any of these approaches to generate a new fresh symmetric key for the Vernam cipher.

B. Secure Information Exchange

In SCADA systems, the field site components are controlled and monitored using short messages communicated between RTU and MTU. Based on the reading obtained from the field control devices, namely, RTU, PLC, and IED, the SCADA master (MTU) makes a proper decision and sends an appropriate signal to the field components to operate plant machinery. Generally, the control messages are short in length (typically 256 bits), which control the sensors and actuators of plant machinery. For example, in water management systems, the signals used during communication include OPEN/CLOSE the valve, SWITCH_ON/SWITCH_OFF the devices, RAISE/LOW the water level tank, etc. [22]. Such systems operate using short messages. Hence the average length of the control message consists of 24 to 32 characters (192 to 256 bits) for one frame.

The Vernam cipher requires the same length for key and message. Moreover, each communication message requires a distinct key for encryption and decryption. To generate such a unique key every time, we have proposed two main approaches, namely, multi-layered architecture, and hash chaining with FSRP. Moreover, both these approaches are further divided in the multiple methods to generate a unique value of FSRP, namely, random prime generator (RPG), and prime counter (PC). Figure 4 illustrates the symmetric key generation process used to encrypt and decrypt the message at both the communication endpoints. Both the sender and receiver negotiate RN (random number), CDT (current date and time), and the index number of FSRP (which acts as a seed for random prime generator/ prime counter) to generate session key (Ks). Using RPG/PC, both the sender and receiver generate a distinct FSRP for each message. Moreover, Blake2s (cryptographically secure hash function [32]) is applied on the session key and FSRP to generate the encryption key (Ke). Similarly, the receiver produces the decryption key (Kd) using the pre-shared Ks and the value of FSRP. Note that, the value of the symmetric key not only depends on the previous key but also on the value of FSRP (which is generated using RPG/PC). In the case of our multi-layered architecture, instead of two parameters, both, MTU and RTU use three parameters, namely, CDT,

Algorithm 1: Multi-Layered (Hybrid)

Input: M = Input Message
begin
  Sender:
  while *(Session!=END)* **do**
    (1) Generate *CDT*
    (2) Generate *FSRP*
    (3) $K_e \leftarrow$ HASH(K_s, *CDT*, *FSRP*)
    (4) $C(M) \leftarrow M \oplus K_e$, $K_s \leftarrow K_e$
    (5) $HMAC_S \leftarrow$ HASH($C(M)$, *CDT* || *FSRP*)
    (6) *KEYSALT* $\leftarrow$ *FSRP* $\oplus$ *CDT*
    (7) *MACSALT* $\leftarrow$ *KEYSALT* || $Index$
    (8) $Bundle \leftarrow K_{rpub}(K_{spri}(C(M), HMAC_S,$ *MACSALT*))
  end
  Receiver:
  while *(Session != END)* **do**
    (1) $Bundle \leftarrow K_{rpri}(K_{spub}(C(M), HMAC_S,$ *MACSALT*))
    (2) $FSRP = Frac(Sqrt(PRIME(Index)))$
    (3) $CDT \leftarrow KEYSALT \oplus FSRP$
    (4) $HMAC_R \leftarrow$ HASH($C(M)$, *CDT* || *FSRP*)
    if *(($HMAC_S$, $HMAC_R$) == TRUE)* **then**
      (5) $K_d \leftarrow$ HASH(K_s,*CDT*,*FSRP*)
      (6) $M \leftarrow C(M) \oplus K_d$, $K_s \leftarrow K_d$
    else
      (7) Discard *M*
    end
  end
end

Algorithm 2: RPG & Prime Counter

Input: M = Input Message, K_s = Session Key
  (a) $FSRP = frac(Sqrt(RPG(Seed)))$ OR (b) $FSRP = frac(Sqrt(PC(Index)))$
begin
  Sender:
  while *(Session!=END)* **do**
    (1) Generate *CDT*
    (2) $K_e \leftarrow$ HASH(Ks, *CDT*, *FSRP*)
    (3) $C(M) \leftarrow M \oplus K_e$, $K_s \leftarrow K_e$
    (4) $HMAC_S \leftarrow$ HASH($C(M)$, *CDT* || *FSRP*)
    (5) *MACSALT* $\leftarrow$ *FSRP* $\oplus$ *CDT*
    (6) $Index \leftarrow Index + 1$
    (7) Transmit $C(M)$, *HMAC*,*MACSALT*
  end
  Receiver:
  while *(Session != END)* **do**
    (1) *CDT* $\leftarrow$ *MACSALT* $\oplus$ *FSRP*
    (2) $HMAC_R \leftarrow$ HASH($C(M)$, *CDT* || *FSRP*)
    if *(($HMAC_S$, $HMAC_R$) == TRUE)* **then**
      (3) $Kd \leftarrow$ HASH(Ks, *CDT*, *FSRP*)
      (4) $K_s \leftarrow K_d$, $Index \leftarrow Index + 1$
      (5) $M \leftarrow C(M) \oplus K_d$
    else
      (6) Discard M
    end
  end
end

FSRP, and Ks to generate the symmetric key. These parameters are exchanged securely using MACSalt and NTRUEncrypt public-key cryptography.

For our evaluation, we assume that the length of the key is 256 bits as Blake2s depends on a 32 byte word size. In the case of 256 bits $<$ input string $<$ 512 bits, we can replace Blake2s with Blake2b to generate the symmetric key, which consists of a 64 byte word size.

The following section describes four methods to implement secure SCADA framework for information exchange.

1) Hybrid Multi-Layered Architecture: We can use the same nomenclature of session key agreement for further secure communication in which after successful distribution of the session key the message is communicated between two parties using both symmetric and asymmetric key cryptography. The data encryption and decryption are obtained using Vernam cipher. The key generator of the Vernam cipher follows the same procedure of session key derivation to generate the symmetric key at the sender and receiver sides. The symmetric key, HMAC and MACSALT are derived using FSRP and CDT. Further encrypted message C(M), HMAC and MACSALT are shared securely using asymmetric key cryptography. Here, the complexity of the method is obtained by N * (Asymmetric Key + Symmetric Key) during each session which provides high security with moderate availability. N is the number of messages exchange during the session. The steps of this approach are shown in Algorithm 1.

The following methods describe the approach of symmetric key cryptography instead of using a combination of public-private key pairs. After secure session key and prime seed distribution, further encryption process can be carried out using one of the three symmetric key based proposed methods as listed below.

2) Random Prime Number Generator: In this method, the seed of the prime index value is used to determine FSRP using next random prime number. Also, CDT and hash of the input message h(M) are determined to generate symmetric key, HMAC and MACSALT. This information is sent to the recipient over the communication channel. Using MACSALT and random number prime generator, the receiver can generate the symmetric key to decrypt the data using the Vernam cipher. Here the complexity of algorithm is measured by Asymmetric key + N * Symmetric key for every session where asymmetric and symmetric key are used during session key distribution while the symmetric key is used during secure communication. However, this approach is comparatively less secure as the adversary could intercept the MACSALT to derive the keys such as FSRP and CDT. Algorithm 2 summarizes the above process.

3) Prime Counter: In this method, instead of random prime generator, we have used prime counter which significantly increases the execution speed. The rest of the steps are same

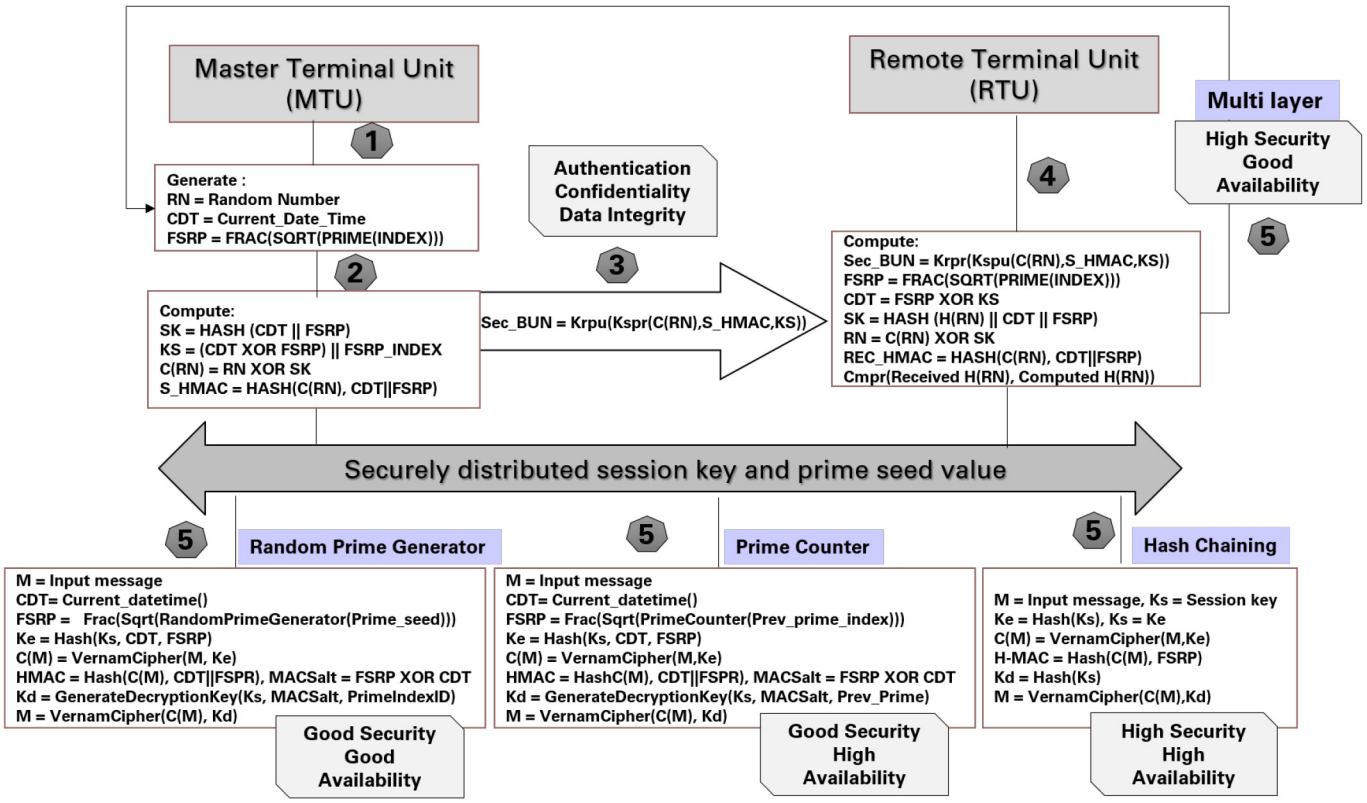

Fig. 5. Complete process diagram of secure communication between MTU and RTU.

Algorithm 3: HASH Chaining

Input: M = Input Message, K_s = Session Key
$FSRP = Frac(Sqrt(PC(Index)))$

begin
  Sender:
  while *(Session!=END)* **do**
    (1) $K_e \leftarrow$ HASH(K_s,*FSRP*)
    (2) $C(M) \leftarrow M \oplus K_e$, $K_s \leftarrow K_e$
    (3) $HMAC_S \leftarrow$ HASH($C(M)$,*FSRP*)
    (4) $Index \leftarrow Index + 1$
    (5) Transmit $C(M)$, *HMAC*
  end
  Receiver:
  while *(Session != END)* **do**
    (1) $HMAC_R \leftarrow$ HASH($C(M)$, *FSRP*)
    (2) $Index \leftarrow Index + 1$
    if *(($HMAC_S$, $HMAC_R$) == TRUE)* **then**
      (3) $K_d \leftarrow$ HASH(K_s, *FSRP*), $K_s \leftarrow K_d$
      (4) $M \leftarrow C(M) \oplus K_d$
    else
      (5) Discard *M*
    end
  end
end

and hence we have highlighted the difference in red font in Algorithm 2. The previous prime number is used to determine next FSRP. Similarly CDT and hash of the input message h(M) are used to determine symmetric key, HMAC and MACSALT. This information is sent to the recipient. Using MACSALT and prime counter, the receiver can generate symmetric key to decrypt the data using Vernam cipher. In this approach the adversary could also intercept the MACSALT to derive the essence of the keys such as FSRP and CDT. The complexity of algorithm is measured by Asymmetric key + N * Symmetric key for every session. Consequently, the model provides good security with high availability.

4) Hash Chaining: This proposed method is one of the robust solutions for SCADA systems which covers all the security mechanisms. This approach not only provides high security but also offers high availability. In this, the pre-shared session key is used as input of the hash function to generate the next symmetric key. Moreover, the previous FSRP is used to generate HMAC which can be derived independently at both the ends and is used to check the integrity of the message. The generated symmetric key is then used to encrypt and decrypt the message using the Vernam cipher, as mentioned in the Algorithm 3. The complexity of this method is based on the Asymmetric + N * Symmetric key cryptography.

The complete process diagram of the proposed framework of secure SCADA systems is shown in Figure 5.

V. Experiments

A. Algorithm Selection of Cipher Suite for Proposed Framework

The choice of the algorithms to design the security framework generally depends on the nature of the application.

The communication of SCADA systems relies on a real-time request-response mechanism. Moreover, SCADA field devices are equipped with micro controllers for processing information and have limited computational power and resources. Consequently, identifying the most appropriate algorithms for the proposed scheme is one of our implementation's crucial steps. The identified algorithms for our cipher suite should provide faster execution speed and be suitable for deploying in an embedded system environment. The comparative analysis of various algorithms was carried out using wolfSSL and libntru 0.5 cryptosystems on Linux subsystem of Windows 10 with Intel Core i5-8300H 2.30GHz processor and 8 GB RAM. The wolfSSL is a lightweight and portable embedded SSL library that is specially meant for IoT, embedded, and RTOS environments [33]. The libntru 0.5 is an open source library that supports the implementation of the public-key encryption scheme NTRUEncrypt in C language by following the IEEE P1363.1 standard [34]. Moreover, the proposed symmetric schemes are implemented on an integrated development environment for Python called IDLE on Windows 10 operating system.

1) HASH Functions: In this framework, the hash function plays a vital role as it acts as a message authentication code and is used to generate a symmetric key. To identify the cryptographically secure and computationally efficient function, we have compared various hash functions. Based on the comparative analysis of computational speed presented in Table I, Blake seems to be most prominent. There are three flavors of Blake's hash function, namely, Blake, Blake2, and Blake3. Furthermore, Blake2 is subcategorized in two types, namely, Blake2s and Blake2b. Blake2b is designed for 64 bits of word length while Blake2s and Blake3 are designed for 32 bits. Both the categories of Blake2 are cryptographically secure hash functions and used to target various applications such as cloud storage intrusion detection, version control systems, and Internet of Things. Moreover, it is computationally efficient like MD5, and provides security similar to SHA-3 [35]. We can also take advantage of Blake2 in multicore architectures for parallel processing. Furthermore, Blake2 uses 32% less RAM than Blake and has proven efficient MAC function [36]. These features make Blake2 a suitable candidate for SCADA systems. For the framework implementation, we have used Blake2s as one of our proposed cipher suite elements. A new version of Blake, namely Blake3, has been released recently [37]. Blake3 is comparatively faster than Blake2s as it uses seven rounds, whereas Blake2s uses ten rounds to compute the hash function [38]. One scope for future work for our research would be to implement our framework using Blake3.

2) Symmetric Key Cryptography: Advanced Encryption Standard (AES) is the well-known symmetric key cryptography used to design secure systems. AGA has used AES as a symmetric key component in its standard protocol suite [16]. Nowadays, AES modes are preferred to secure the systems owing to better security and faster execution speed. 3DES is also used in traditional cryptosystems. In Table II, we have compared the computational speed of various modes of AES and DES with the proposed hash-based Vernam Cipher. The computational speed of Vernam Cipher is calculated by adding the execution speed of Blake2S hash with the speed of Exclusive-OR operation. The comparative analysis shows that the hash-based symmetric key technique used in Vernam Cipher is faster than other algorithms.

TABLE I
COMPARATIVE ANALYSIS OF VARIOUS HASH FUNCTIONS

Algorithm	Ex.Speed (MB/sec)
MD5	340.729
RIPEMD	129.74
SHA	31.571
AES-256-CMAC	110.504
SHA2-256	152.863
SHA3-256	106.781
Blake2b	**172.2**
Blake2s	**169.78**

TABLE II
COMPARATIVE ANALYSIS OF COMPUTATIONAL SPEED OF VARIOUS SYMMETRIC KEY ALGORITHMS

Algorithm	Ex.Speed (MB/sec)
AES-256-CBC-enc, AES-256-CBC-dec	94.565 , 88.169
AES-256-GCM-enc, AES-256-GCM-dec	25.596, 24.318
AES-256-ECB-enc, AES-256-ECB-dec	55.69, 63.067
AES-256-CFB	86.329
AES-256-OFB	71.146
AES-256-CTR	64.576
3DES	14.542
Vernam Cipher with Blake2s	**157.45**

3) Asymmetric Key Cryptography: Asymmetric key cryptography not only offers the confidentiality but also ensures integrity, authentication, and non-repudiation during communication. Some public key algorithms such as Diffie-Hellman key exchange provide key distributions and secrecy, whereas some provide encryption and digital signature such as RSA, ECC, and NTRU [39]. We have compared various well-established public key algorithms, namely, RSA, DH, ECC, and NTRU by considering the key size and total operations performed per second. According to the output results presented in Table III, NTRU outperforms the other methods. NTRU public-key cryptography is also known as NTRUEncrypt. This is constructed using a lattice-based technique by applying the concept of the shortest vector problem. It depends on the factoring of certain polynomials in a polynomial ring into a quotient of two minimal coefficients. Both encryption and decryption follow simple polynomial multiplication, which makes NTRU faster than other asymmetric key cryptosystems [40]. Moreover, the points mentioned below represent the capabilities of the NTRU based public key cryptography. Therefore, we have chosen the NTRU public key algorithm for our proposed cipher suite.

- NTRU is the highest performing public key cryptograpic system for embedded devices [41].
- NTRU decryption is more than 92 times faster than RSA decryption at an equivalent security level [42].
- NTRU is nearly 60% faster than RSA at encryption and TLS with a 370 times improvement in key generation time [42].

TABLE III
COMPARATIVE ANALYSIS OF THE COMPUTATIONAL SPEED OF VARIOUS PUBLIC KEY CRYPTOGRAPHY

Algorithm	Key Size	Mode	Ops/sec
RSA	1024	Key Generation	12
RSA	2048	Key Generation	3
DH	2048	Key Generation	913
DH	2048	Key Agreement	500
ECC	256	Key Generation	523
ECDHE	256	Key Agreement	500
NTRU	**1026**	**Key Generation**	**2153**
NTRU	**1499**	**Key Generation**	**698**
NTRU	**1615**	**Key Generation**	**801**
NTRU	**2066**	**Key Generation**	**345**

- NTRU encryption and decryption are faster than the best-performing ECC algorithms at equivalent security levels [42].
- NTRU is only around 20 times slower than a recent AES implementation [42].
- Both RSA and ECC are vulnerable to quantum computing attacks where NTRU offers resistance to that [43].
- NTRU accomplishes TLS authentication and key negotiation by combining classic cryptography which offers quantum-safe cryptography [43].
- Parallel implementation of NTRU is possible on top of the existing crypto infrastructure [41].

B. Computational Speed of Proposed Framework

This section represents the calculation of the overall computational speed of the proposed framework. We have considered the execution time of the major four elements, namely, session key, symmetric key, HMAC, and asymmetric key. First, we have calculated the time to generate and extract the session key. After that, we have computed the execution time of symmetric and asymmetric key generation, distribution, encryption, and decryption. Finally, we have calculated the overall time by combining it with execution time to generate and extract the HMAC.

1) Execution Time of Session Key Generation and Extraction: We have generated the session key, KEYSALT, and MACSALT using two random parameters CDT and FSRP, along with Blake2s HASH function. These parameters are securely exchanged between two communication SCADA devices and extracted back at the receiver side. The average execution time and total execution time to generate and extract these elements are shown in Figure 6 and Figure 7. We observed that it takes approximately 0.15 milliseconds average execution time to create and extract a 256-bit session key.

2) Execution Time of Symmetric Key Cryptography: This section presents the execution time of three symmetric key cryptography methods, namely, Random Prime Generator (RPG), Prime Counter (PC) and Hash Chaining (HC). To calculate the execution time of each method we have considered the overall time of each module to generate and extract the symmetric key along with encryption and decryption time taken by the Vernam stream cipher. In Table IV, we present the time of three proposed symmetric key cryptography methods

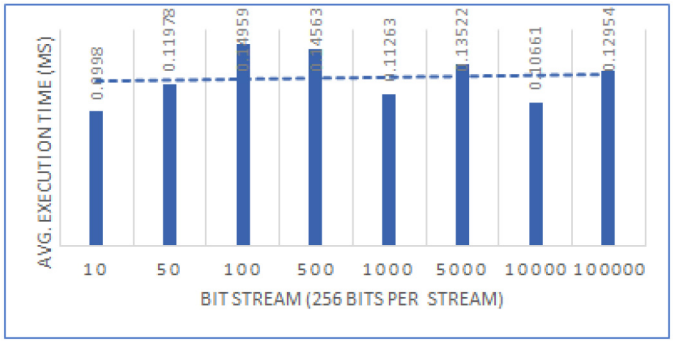

Fig. 6. Average Execution Time for Session Key Generation and extraction.

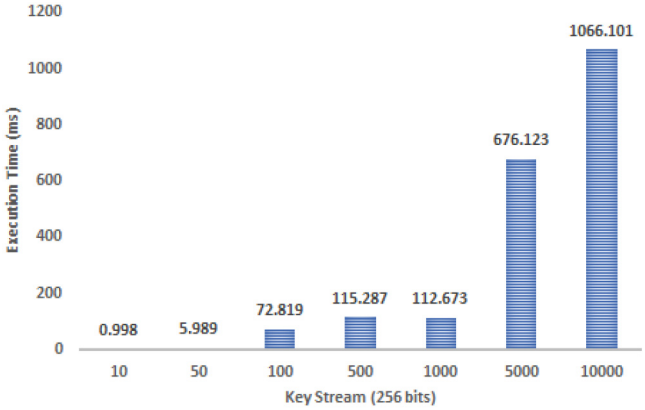

Fig. 7. Overall Time (Session Key).

TABLE IV
EXECUTION TIME OF PROPOSED SYMMETRIC KEY METHODS

Input stream (256 bits)	Random Prime (second)	Prime Counter (second)	Hash Chaining (second)
1	0.0050	0.000022	0.0000009
10	0.040	0.0004	0.0001
50	0.280	0.0027	0.0003
100	0.411	0.0040	0.0012
500	2.325	0.254	0.0062
1000	6.566	0.439	0.0129
5000	32.929	2.499	0.0685
10000	50.883	4.392	0.1325

(in seconds) for various sizes of input streams. Based on the results, hash chaining seems to be the most efficient in terms of computational speed amongst the three proposed methods.

3) Execution Time of NTRU Based Public Key Cryptography: We have compared NTRU based implementations based on security levels, namely, moderate, standard, high, and highest security. Each security level is defined considering the size of cipher text, a public key, and private key. In most applications, the standard security level is used to avoid lattice-based, brute force, and man-in-the-middle attacks. The observation is carried out using total execution time by considering key generation, encryption, and decryption as shown in Table V. Moreover, we have computed the average execution time of public-key pair generation, which is around 1.51 ms with an encryption time of 0.073 ms and a decryption time of 0.106 ms.

TABLE V
EXECUTION TIME OF NTRU BASED PUBLIC KEY CRYPTOGRAPHY

NTRU (INPUT=256 bits)	Key_Gen (ms)	Enc (ms)	Dec (ms)	Total (ms)
Moderate Security (CT=1022, Kpub=1026,Kpr=111)	0.468	0.03	0.047	0.545
Standard Security (CT=1495,Kpub=1499,Kpr=1339)	1.432	0.073	0.096	1.601
High Security (CT=1611,Kpub=1615,Kpr=301)	1.248	0.066	0.138	1.452
Highest Security (CT=2062,Kpub=2066,Kpr=227)	2.893	0.123	0.145	3.161
Average Execution Time (ms)	1.510	0.073	0.106	1.689

TABLE VI
CONSIDERABLE PARAMETERS OF DIFFERENT CRYPTOGRAPHIC COMPONENTS

Notation	Description	Cost in ms
Tskg	Time for a session key generation	0.06485
Tpc	Time to generate Prime number Counter	0.00997
Trpg	Time to generate random prime generator	0.5063
Tudt	Time to generate universal date and time	0.00010
Tse	Time for a symmetric encryption	0.000199
Tsd	Time for a symmetric decryption	0.000996
Thash	Time to generate HMAC	0.000001
Tex	Time for a session Key extraction	0.0992
Takg	Time for a asymmetric key generation	1.51025
Tae	Time for a asymmetric encryption	0.073
Tad	Time for a asymmetric decryption	0.1065

TABLE VII
TOTAL EXECUTION TIME CALCULATION

Method	Delay
Multi-layered Approach	N * (Tskg + Tex + Takg + Tae + Tad + Tsym + Tse + Tsd + Thash)
Symmetric Key Approach	Tskg + Tex + Takg + Tae + Tad + N * (Tsym + Tse + Tsd + Thash)
Random Prime Generator	Tsym = Trpg + Tudt
Prime Counter	Tsym = Tpc + Tudt
Hash Chaining	Tsym = Thash

4) Total Execution Time: In order to achieve consistent results, we have measured the execution time of each cryptographic components. The execution time of these elements is listed in Table VI.

Moreover, Table VII presents the mathematical equations that calculate the total execution time of all the four methods, namely, ML, RPG, PC, and HC. In hybrid approach, both symmetric and asymmetric algorithms are used to secure the information. In contrast, in the other three approaches, once the session key has been shared between two communication devices, only the symmetric key algorithm is used for performance improvement. Furthermore, the execution time of these three symmetric key algorithms is varied due to how they generate the keys to secure the information.

Table VIII represents the total execution time of all the four proposed methods by considering the major four parameters, namely, key generation, key extraction, encryption and decryption. According to the results, the execution time of HC is lower than the other three methods and has proven most efficient amongst all. Moreover, PC and ML approaches are more prominent than RPG. Comparatively, RPG takes more time because of its intricate design to generate a random prime number based on a seed value.

TABLE VIII
TOTAL EXECUTION TIME IN SECONDS

BIT STREAM (256 bits)	Hybrid	RPG	PC	HC
1	0.0026	0.0052	0.0019	0.0016
10	0.0072	0.0437	0.0056	0.0026
50	0.0373	0.2806	0.0285	0.0046
100	0.0592	0.4111	0.0415	0.0136
500	0.3453	2.3248	0.2557	0.0635
1000 (32KB)	0.6203	6.5663	0.4410	0.1309
5000 (160KB)	3.3980	32.9295	2.5007	0.6867
10000 (320KB)	6.1889	50.8831	4.3941	1.3271
50000 (16MB)	30.2294	257.9262	21.2546	2.2456
100000 (32MB)	61.1824	560.3114	43.2326	10.4327

C. Calculation of Key Storage Cost

Storage cost is another important parameter to evaluate the performance of SCADA networks. Field control devices such as RTUs, PLCs, and IEDs are typically located at the plant floor and remote from the MTU. Hence they require to update the session keys periodically. On the other hand, if field control devices have many static keys, and if any of them is compromised, it can expose the entire network communication. Consequently, the session key update process is a very crucial step. Since the key generation, distribution, and extraction are periodic and costly operations, the SCADA network should have fewer stored keys on each field control device. For this reason, we have identified the storage cost of our proposed key management scheme. Table IX summarizes the storage cost by considering the three types of communication, namely, point-to-point, broadcast, and multicast amongst MTU, Sub-MTU, and RTU. The total cost of keys is calculated at each SCADA location, where m denotes the number of sub-MTU's keys, and r represents the maximum number of RTU's keys.

D. Randomness Evaluation

Many cryptography applications may need to meet more robust random number generator requirements when the randomness of the keys is one of the most critical factors for that system. We have used the Vernam stream cipher for our proposed framework, which requires a distinct and random key to secure the information. In particular, the key generator's output must be unpredictable. Hence, we have evaluated the proposed symmetric key generator using the National Institute of Standards and Technology (NIST) statistical toolkit. We have configured this tool in the Linux subsystem of the Windows 10 operating system. This toolkit offers a total of sixteen different statistical tests to determine whether a generator is suitable for a particular cryptosystem. Each test evaluates the randomness based on specific criteria by considering the number of 1's and 0's in the binary stream and accordingly produces the P-value. If the test has P-value

TABLE IX
KEY MANAGEMENT : STORAGE COST OF KEYS

Types	MTU	SUB-MTU	RTU
Point to Point Communication			
MTU to SUB-MTU SUB-MTU to RTU RTU to SUB-MTU	Private Key of MTU= 1, (D) Public Key = m, (E) m = Number of Sub-MTUs	Private Key of Sub-MTU = 1, (D) Public Key = r, (E) r = Number of RTUs	Private Key of RTU= 1, (D) Public Key = 1, (E) Public key of Sub-MTU
RTU to RTU	NA	NA	Private Key of Controller RTU = 1, (D) Public Key = r, (E) r = Number of Slave-RTUs
Broadcast Communication			
MTU to RTU	Private Key of MTU= 1, (E)	Private Key of Sub-MTU = 1, (E) Public Key of MTU = 1, (D)	Public Key of sub-MTU = 1, (D)
RTU to RTU	NA	NA	Private Key of Controller RTU = 1, (E) Public Key of Controller RTU = 1, (D)
Multicast Communication			
MTU to RTU	Private Key of MTU= 1, (E)	Private Key of Sub-MTU = 1, (E) Public Key of MTU = 1, (D)	Public Key of sub-MTU = 1, (D)
RTU to RTU	NA	NA	Private Key of Controller RTU = 1, (E) Public Key of Controller RTU = 1, (D)
Total Keys	1(PK) + m(PUB-SUBMTU) = **m + 1**	1(PK)+r(PUB-RTUs) + 1(PUB-MTU) = **r + 2**	Controller RTU: **m + 2** 1 (PK) + 1 (PUB-SUBMTU) + m (PUB-RTU) Slave RTUs: **2** 1(PK) + 1(PUB-CRTU)

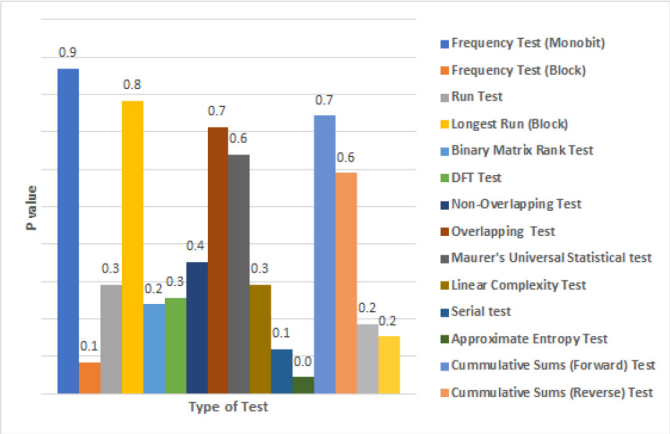

Fig. 8. Randomness assessment of symmetric key.

≥ 0.001, that means an input binary sequence would be random with a 99.9% confidence. Figure 8 presents all the 16 tests and corresponding P-values for the proposed symmetric key generator for Vernam cipher. Our proposed key generator passes all the statistical tests and proven to be random.

VI. PERFORMANCE ANALYSIS

A. Formal Analysis of Protocol

Researchers currently use two main approaches to verify security protocols, namely, provable security and the formal method approaches [44], [45], [46]. Provable security defines a rigorous framework to describe and prove cryptographic properties from a mathematical point of view. However, the formal method approach proposes a model to describe and analyze cryptographic protocols by abstracting basic properties. Dalal *et al.* [47] discusses various tools such as Avispa, ProVerif, and Scyther that are useful for the formal verification of the cryptographic protocols. Scyther outperforms the state-of-the-art Avispa tools. Although Scyther uses no abstraction techniques, it still offers a performance level similar to the abstraction-based ProVerif tool [47]. In Scyther, small (e.g., Needham-Schroeder, Yahalom, Otway-Rees) to medium-sized (e.g., TLS, Kerberos) protocols are usually verified in less than a second. Moreover, Scyther is currently the fastest protocol verification tool that does not use approximation methods [48].

Therefore, we have used the Scyther tool to formally verify our security protocol, which performs the evaluation under the cryptographic assumption. We define all the cryptographic functions completely. Moreover, the entire assessment is carried out by considering the presence of an adversary. This tool uses an unbounded model checking approach that demonstrates the soundness of a protocol for all the possible behaviors in the presence of an adversary [49]. The language used in Scyther is called Security Protocol Description Language (SPDL). It is also known as role-based language that describes the entire protocol using roles and sending/receiving events.

SPDL provides expressions for encryption and hashing. Furthermore, we can verify authentication, confidentiality and message integrity using claims in the Scyther. We have mainly focused on three types of goals, namely, non-injective synchronization, non-injective agreement, and secrecy for our proposed approach. We have generated a trace pattern route that represents the packet forwarding from RTU to MTU, as illustrated in Figures 9 and 10. Figure 11 illustrates the protocol design code for Scyther to analyze the attacks by considering all the participants, namely, MTU, RTU, and

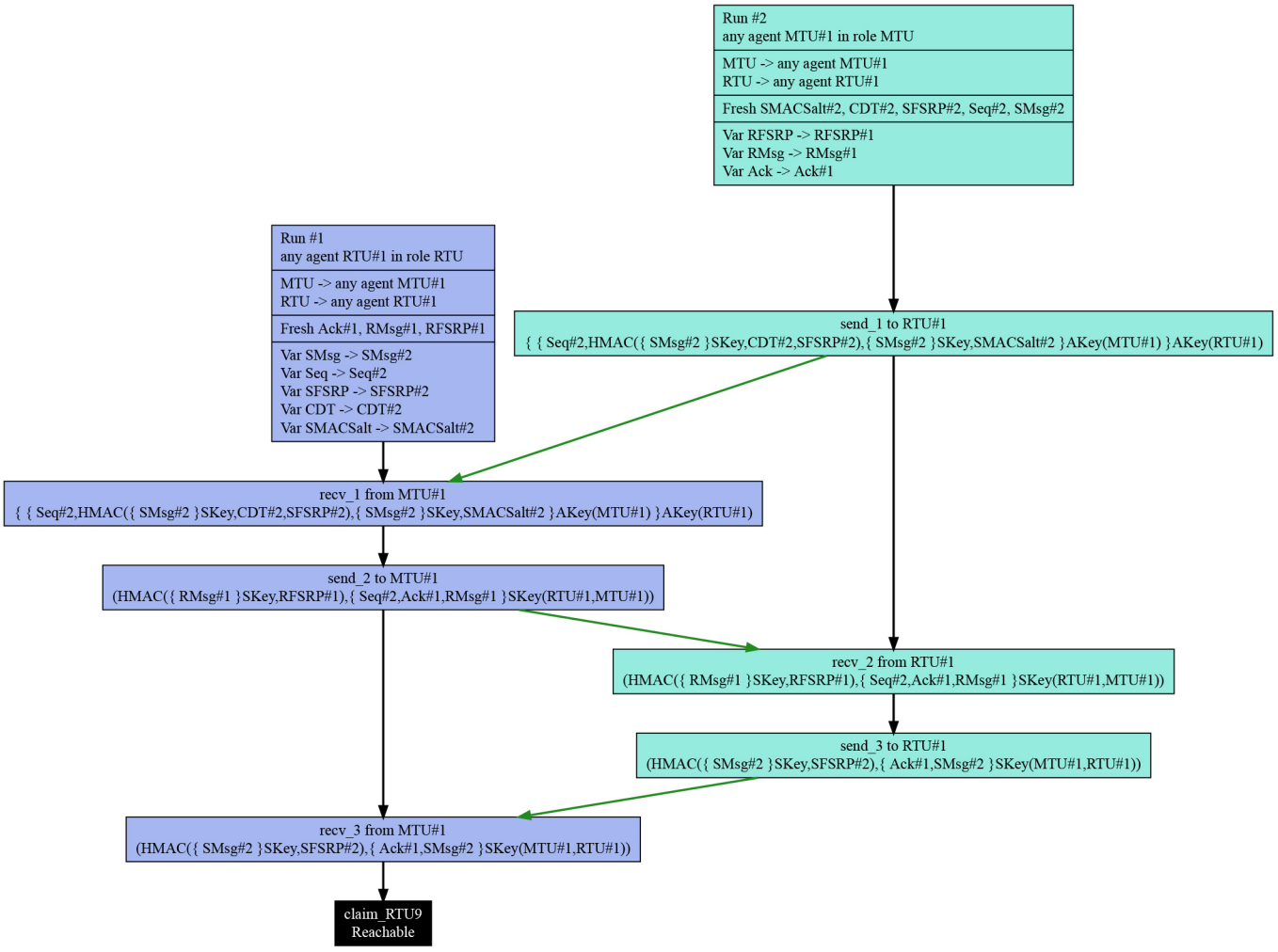

Fig. 9. Formal Analysis of proposed protocol for secure communication between MTU and RTU using Scyther Tool.

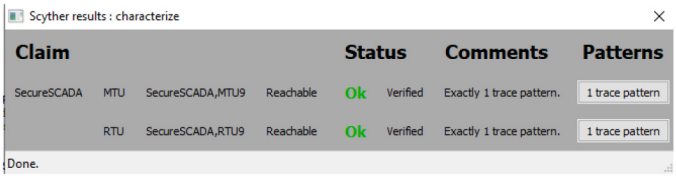

Fig. 10. Claims & trace pattern validation of proposed protocol for secure communication between MTU and RTU using Scyther Tool.

the attacker. We have verified the protocol using "automatic claim" and "verification claim" procedures. As illustrated in Figure 12, our proposed framework is resistant to all the attacks over the communication channel.

B. Attack Analysis on Hash Function

Generally, a hash function can be broken by three types of attacks, namely, collision attack, preimage resistance attack, and length extension attack [35], [50], [51], [52], [53]. A brief description of each attack is described below.

1) *Collision Attack:* This attack aims to identify two different inputs that will generate the same hash value to create a collision with transmitted data over the communication channel. For example, the attacker will try to find messages m1 & m2, leading to the same hash function, i.e., Hash(m1) = Hash(m2). In general, for two different precedes, p1 & p2, the intruder chooses two appendages m1 & m2 such that Hash(p1||m1) = Hash(p2||m2) which leads to the chosen-prefix collision attack.
2) *Preimage Resistance Attack:* This attack is intended to find out the message for the particular hash value. That means, given a hash value h, the attacker will find a message m such that Hash(m) = h.
3) *Length Extension Attack:* In this attack, an attacker can use Hash(m1) and the length of m1 to calculate Hash(m1||m2), where an attacker will control m2 without knowing the content of m1.

Three types of approaches are used to check the strength of the hash function to test if the given hash function can be broken practically, theoretically or partially as listed below [54].

1) *Practically Broken:* The attack has been demonstrated in practice and able to break the entire hash function.

Protocol description | Settings

```
secret AKey: Function;
secret  SKey: Function;
hashfunction HMAC;
protocol SecureSCADA(MTU,RTU)
{
    role MTU
    {
        fresh SMACSalt: Nonce; fresh SMsg: Nonce;
        fresh CDT: Nonce;fresh SFSRP: Nonce; fresh Seq: Nonce;
        var Ack: Nonce;var  RMsg: Nonce;var RFSRP: Nonce;

        send_1(MTU,RTU,
        {{Seq,HMAC({SMsg}SKey,CDT,SFSRP),{SMsg}SKey,SMACSalt}AKey(MTU)}AKey(RTU));
        read_2(RTU,MTU,
        HMAC({RMsg}SKey,RFSRP),{Seq,Ack,RMsg}SKey(RTU,MTU) );
        send_3(MTU,RTU,
        HMAC({SMsg}SKey,SFSRP),{Ack,SMsg}SKey(MTU,RTU) )

        claim_i1 (MTU,Secret,CDT);
        claim_i2 (MTU,Secret,SFSRP);
        claim_i3 (MTU,Secret,SMsg);
        claim_i4 (MTU,Secret,SMACSalt);
        claim_i5 (MTU,Alive);
        claim_i6 (MTU,Weakagree);
        claim_i7 (MTU,Niagree);
        claim_i8 (MTU,Nisynch);
    }
    role RTU
    {
        var SMACSalt: Nonce;var CDT: Nonce;var SFSRP: Nonce;var Seq: Nonce;var SMsg: Nonce;
        fresh Ack: Nonce;fresh RMsg: Nonce;fresh RFSRP: Nonce;

        read_1(MTU,RTU,
        {{Seq,HMAC({SMsg}SKey,CDT,SFSRP),{SMsg}SKey,SMACSalt}AKey(MTU)}AKey(RTU));
        send_2(RTU,MTU,
        HMAC({RMsg}SKey,RFSRP),{Seq,Ack,RMsg}SKey(RTU,MTU) );
        read_3(MTU,RTU,
        HMAC({SMsg}SKey,SFSRP),{Ack,SMsg}SKey(MTU,RTU) );

        claim_i1 (RTU,Secret,RMsg);
        claim_i2 (RTU,Secret,RFSRP);
        claim_i3 (RTU,Secret,Ack);
        claim_i4 (RTU,Secret,Seq);
        claim_i5 (RTU,Alive);
        claim_i6 (RTU,Weakagree);
        claim_i7 (RTU,Niagree);
        claim_i8 (RTU,Nisynch);
    }
}
```

Fig. 11. Claims & Protocol Design Code for Scyther for Analysis of attacks.

Scyther results : verify

Claim				Status		Comments
SecureSCADA	MTU	SecureSCADA,MTU1	Secret CDT	Ok	Verified	No attacks.
		SecureSCADA,MTU2	Secret SFSRP	Ok	Verified	No attacks.
		SecureSCADA,MTU3	Secret SMsg	Ok	Verified	No attacks.
		SecureSCADA,MTU4	Secret SMACSalt	Ok	Verified	No attacks.
		SecureSCADA,MTU5	Alive	Ok	Verified	No attacks.
		SecureSCADA,MTU6	Weakagree	Ok	Verified	No attacks.
		SecureSCADA,MTU7	Niagree	Ok	Verified	No attacks.
		SecureSCADA,MTU8	Nisynch	Ok	Verified	No attacks.
	RTU	SecureSCADA,RTU1	Secret RMsg	Ok	Verified	No attacks.
		SecureSCADA,RTU2	Secret RFSRP	Ok	Verified	No attacks.
		SecureSCADA,RTU3	Secret Ack	Ok	Verified	No attacks.
		SecureSCADA,RTU4	Secret Seq	Ok	Verified	No attacks.
		SecureSCADA,RTU5	Alive	Ok	Verified	No attacks.
		SecureSCADA,RTU6	Weakagree	Ok	Verified	No attacks.
		SecureSCADA,RTU7	Niagree	Ok	Verified	No attacks.
		SecureSCADA,RTU8	Nisynch	Ok	Verified	No attacks.

Done.

Fig. 12. Claims & Attack Analysis of proposed protocol for secure communication between MTU and RTU using Scyther Tool.

TABLE X
COMPARATIVE ANALYSIS OF THE VARIOUS HASH FUNCTIONS (COL: COLLISION ATTACK, PR: PREIMAGE RESISTANCE ATTACK, LE: LENGTH EXTENSION ATTACK)

Hash Function	COL	PR	LE	Partial	Practical	Theory
MD5 [51]	YES	YES	NO	NO	YES	YES
PANAMA [56]	YES	NO	NO	NO	YES	NO
RIPEMD [51]	YES	YES	NO	YES	YES	NO
SHA-1 [52]	YES	YES	NO	YES	YES	YES
SHA256 [53]	YES	YES	NO	YES	NO	NO
BLAKE2s [54]	YES	YES	NO	YES	NO	NO
SHA3 [54]	YES	YES	NO	YES	NO	NO
BLAKE2b [36]	YES	YES	NO	YES	NO	NO

2) *Theoretically Broken:* Attack demonstrates in theory by proof of concept which is able to break all the rounds of the hash function.
3) *Partially Broken:* No attack has demonstrated to break the entire function successfully. However, only a reduced version of the hash is broken and requires more work than the claimed security level.

Table X compares the types of attacks and the breaking mechanisms of various popular hash functions. As illustrated in Table X, Blake2 is comparatively better than other functions. Blake2 can be partially broken and fragile due to collision and preimage resistance attacks. To overcome this issue, we have incorporated two approaches, namely, PNG (Prime number generator) and HMAC. To prevent the system from collision attacks, we have introduced the parameters FSRP and CDT, which generate a unique key at each iteration. In this case, even if the attacker identifies a similar input which generates the same hash function as the transmitted data, it will not help in successfully launching a correlation attack. In our proposed solution, we use HMAC, which not only relies on the hash of the message but also uses CDT & FSRP. Hence, during the validation process, the authentication and message integrity are identified at the receiver end and can prevent the system from correlation and preimage resistance attacks. The following discussion gives the security proof of our proposed approach against correlation and preimage resistance attack.

Security Proof: With reference to the proposed framework, let us denote the original Message as C(M1) and the key parameters used to generate HMAC as CDT & FSRP.

$$HMAC_{Sender} = Hash((C(M1), CDT||FSRP)) \quad (15)$$

Let us assume, over the communication channel, the attacker identifies another message C(M2) and replaces C(M1) with C(M2) where, Hash(C(M1)) = Hash(C(M2)). The receiver computes HMAC based on received message C(M2) as follows.

$$HMAC_{Receiver} = Hash((C(M2), CDT||FSRP)) \quad (16)$$

$$HMAC_{Sender} \neq HMAC_{Receiver} \quad (17)$$

The difference in signature of the HMAC identifies if the integrity is compromised and in such a case M1 is discarded.

The above proof illustrates that the proposed security framework prevents the collision attack. Similarly, even though the intruder can identify message C(M1) which generates Hash(C(M1)), the message integrity or authentication cannot be broken owing to the key parameters CDT and FSRP. Thus, the pre-image attack is prevented.

C. Analysis of Avalanche Effect for Hash Function

Confusion and diffusion techniques have traditionally been used to evaluate the security of cryptographic primitives [56]. In the context of the hash function, confusion is defined using the relation between the secret key and a hash value for a given input message. Confusion is obtained naturally due to the inherited property of chaos [57]. Diffusion, also known as the avalanche effect, is a desirable property for cryptographically secure hash functions [57]. This is one of the factors to check the randomization capability of the given function. The ideal hash function should exhibit the evidence of the avalanche effect up to the significant level which supports the randomization and make difficult to predict by cryptanalysis [58]. Generally, the butterfly effect and large data blocks are used to generate the avalanche effect [59], in which a small change to an input value will make a significant change in the output hash value. Moreover, there is no correlation between current and previous hash outputs. In our proposed approach, we have used the Blake hash function, which demonstrates a higher-order avalanche effect in that there is a probability of 50% of data alteration in the hash output if a single bit is modified in the input [60]. The example in [32] demonstrates the avalanche effect of Blake and is proven to generate random hash output that doesn't rely on the previous hash value.

D. Randomness Analysis of Keys

Session Key Generation (Parameters): A session key is derived and communicated to both parties during initial authentication. This key is derived using three parameters, namely, random number (RNi), where i = 1,2,3, ... n, index of the function of the fraction of square root of a prime number (FSRP(index)), where index = 1,2,3, ... n, and CDT = current date and time in a microsecond. These parameters are generated at each session and exchanged securely using NTRUEncrypt (public-key cryptography). We have analyzed the following test cases concerning the values of these three parameters.

Case 1: MTU/RTU generates unique values for RN, Index of FSRP, and CDT at every session:

$SessionKey_1 : Hash(RNi, FSRP(index), CDT)$ returns unique value

$SessionKey_2 : Hash(RNi, FSRP(index - k), CDT)$ returns unique value, where k is any random number

Case 2: MTU/RTU generates the same value of RN & seed of FSRP for two or more consecutive sessions, however, CDT is always unique:

$SessionKey_1 : Hash(RNi, FSRP(index), CDT)$ returns unique value as CDT is always distinct

$SessionKey_2 : Hash(RNi, FSRP(index - k), CDT)$ returns unique value as CDT is always distinct, where k = 0.

Here we have used the Blake2 hash function which is proven to be a cryptographically secure function [32] and hence in both the above cases, our proposed approach always generates unique and random session keys.

Symmetric key Generation: The symmetric key is derived using two parameters, namely, session key (Ks) and fraction square root of a prime number. As mentioned earlier, the session key is derived using three randomly generated parameters and distributed over the secure communication channel using public-key cryptography. Moreover, the value of FSRP is generated randomly using a random prime number generator or prime counter. In this case, the seed of the prime number is distributed to both the communication ends, namely, control center, and field site components during session key exchange. These parameters are further computed by combining the concept of hash chaining and FSRP. This is how the proposed approach generates unique and random parameters for the symmetric key used in the Vernam cipher for every message.

Parameters: Here we have used two parameters to derive a symmetric key for the Vernam cipher, namely, FSRP(index), where index = 1,2,3, ..., n (FSRP is generated using a random prime generator or prime counter, the index value is distributed during session key exchange), and session key SKi = Hash(RNi, FSRP(index), CDT), where i, index = 1,2,3, ... n.

Case 1: MTU/RTU generates distinct values of Ks and FSRP for every message:

SKi = Hash(SKi-1, FSRP(index)) returns unique value

SKi + 1 : Hash(SKi, FSRP(index-n)) returns a unique value, where n is any random number, and SKi is updated with the previous session key.

Case 2: MTU/RTU generates the same value of FSRP for two or more consecutive messages, however, Ks is always unique:

SK_i = Hash(SKi, FSRP(index)) returns a unique value as SK is always unique for all messages

$SK_i + 1$ = Hash(SKi-1, FSRP(index-n)) returns unique value as SK is always unique, where n = 0.

In both the above cases, the key is unpredictable and random as the index value of FSRP is only known to MTU and RTU. Moreover, the value of the symmetric key is different even though the value of the FSRP is the same for two consecutive messages as the current key depends on two parameters, namely, SK and FSRP, and is generated using a cryptographically secure hash function.

E. Security Analysis

In this section, the proposed framework is analyzed by considering various security mechanisms, namely, authentication, confidentiality, integrity, availability, and scalability. Moreover, the evaluation is extended by targeting various attacks and corresponding prevention mechanisms.

1) Message Integrity
   - Multi-layered hybrid architecture using symmetric and asymmetric key cryptography offers integrity.
   - Vernam stream cipher provides resistance to cryptography attacks [39].

TABLE XI
COMPARATIVE ANALYSIS OF STORAGE COST OF KEYS (M = NUMBER OF SUB-MTU'S KEYS, R = NUMBER OF RTU'S KEYS)

Key Management Schemes	MTU	Sub-MTU	RTU
SKE [4]	m(1+r)	1+r	1
SKMA [18]	m(1+r)	1+r	1
ASKMA [19]	2m+mr	r+logm	2+logr
ASKMA+ [7]	m	1+r+logm	1+logr
Symmetric [21]	r+1	-	2
Symmetric [21]	r+1	-	2
Hybrid [11]	m+2	2r+1	1+logr
CKMI [22]	2+r+m	2+r	1
Proposed Algo	**m+1**	**r+2**	**2**

TABLE XII
COMPARATIVE ANALYSIS OF VARIOUS CIPHER SUITES

Cipher Suite	Avg_Time (ms)
AGA_ECDHE, RSA, AES-128, GCM, SHA256	4.14
AGA_ECDHE, ECDSA, AES-128, GCM, SHA256	3.94
RSA, AES-128, CBC, SHA1	3.81
RSA, AES-128, CBC, SHA256	3.83
RSA, AES-256, CBC, SHA1	3.82
RSA, AES-256, CBC, SHA256	3.85
Multi-layered (NTRU, Vernam Cipher, Blake2s)	**2.61**
Random Prime Generator (NTRU, Vernam Cipher, Blake2s)	**5.25**
Prime Counter (NTRU, Vernam Cipher, Blake2s)	**1.91**
Hash Chaining (NTRU, Vernam Cipher, Blake2s)	**1.68**

- Randomness of Key offers immunity to collision and preimage resistance attacks [61].
- Dynamic Salt offers resistance to rainbow table attack and dictionary attack [61].
- NTRU based public key cryptography offers resistance to quantum attacks, brute force, and meet-in-the-middle attacks. It also prevents the system against data harvest attacks [62].
- HMAC provides immunity against length extension attacks [63].

2) Authentication, Confidentiality
   - Public key of sender and private key of receiver of NTRU based public key cryptography provides sender's authentication and recipient's confidentiality.
   - HMAC offers message integrity and authentication.
3) High Availability—Faster execution
   - Once the session key distribution is established using hybrid method, further communication will take place using symmetric key cryptography that increases the computation speed.
   - Symmetric key generation using hash chaining and prime counter offers high execution speed.
   - Use of Vernam stream cipher uses modulo operation for encryption and decryption which requires only 4 cycles in hardware implementation [64].
   - NTRU is one of the fastest public key cryptographic systems compared to well-known methods such as RSA and ECC [41].
   - HMAC is derived using the same components used to generate the key. This reusability of elements reduces the computational time.
4) Scalability
   - Same symmetric key cryptography (Vernam cipher) is used for both encryption and decryption.
   - Authentication and confidentiality are established using public-private key pairs amongst communication parties.

F. Storage Cost

The periodic session key agreement is a crucial step in SCADA communication that offers key refreshment. However, field control devices have limited power and memory requirements. Hence, an effective key agreement scheme with fewer stored keys can significantly improve the efficiency of SCADA networks. Many key management and agreement schemes have been proposed to address the problem of key storage costs. We have compared the key storage cost of our proposed scheme with various published techniques, as presented in Table XI.

G. Execution Speed

Table XII depicts the comparative analysis of the proposed scheme with various state-of-the-art techniques by implementing various cipher suites using the wolfSSL library. AGA has proposed two cipher suites for secure SCADA communication including the bundle of ECDHE, AES, RSA, and SHA256 and ECDHE, AES, ECC and SHA256 for authentication, confidentiality, message integrity and digital signature [16].The cipher suite RSA, AES, CBC and SHA is used in TLS communication, whereas we have used the NTRU, Vernam Cipher and Blake2s for our proposed framework. The average execution time of our proposed cipher suite is comparatively better than other protocol standards.

VII. CONCLUSION

The protection of critical industrial infrastructure against cyber-attacks is crucial for ensuring public safety, security, and reliability. SCADA system are used to control and monitor such industrial control systems. A robust solution to strengthen the security of these systems against cyber-attacks is a crucial requirement in the design of SCADA systems. Through this work, we aim to cover the protection of the industrial control system landscape by offering a low cost and robust framework for SCADA networks, which protects them against various cyber-attacks. In this paper, we have proposed a session key agreement in addition to lightweight multi-layered encryption techniques. The framework combines both symmetric and asymmetric cryptography to achieve high computational speed by covering all the security mechanisms. This security model is proposed to enhance the security of various industrial sectors such as water and sewage plants, power stations, chemical plants, oil industries, product manufacturing units, and transportation systems. The successful deployment of this model will allow operators and technicians to monitor and control the plant devices remotely as it will protect the entire system from potential breaches.

REFERENCES

[1] D. Upadhyay, S. Sampalli, and B. Plourde, "Vulnerabilities' assessment and mitigation strategies for the small linux server, Onion Omega2," *Electronics*, vol. 9, no. 6, p. 967, 2020.

[2] D. Upadhyay and S. Sampalli, "SCADA (supervisory control and data acquisition) systems: Vulnerability assessment and security recommendations," *Comput. Security*, vol. 89, Feb. 2020, Art. no. 101666.

[3] Y. Cherdantseva *et al.*, "A review of cyber security risk assessment methods for SCADA systems," *Comput. Security*, vol. 56, pp. 1–27, Feb. 2016.

[4] A. Rezai, P. Keshavarzi, and Z. Moravej, "Key management issue in SCADA networks: A review," *Int. J. Eng. Sci. Technol.*, vol. 20, no. 1, pp. 354–363, 2017.

[5] F. M. Salem, E. Ibrahim, and O. Elghandour, "A lightweight authenticated key establishment scheme for secure smart grid communications," *Int. J. Safety Security Eng.*, vol. 10, no. 4, pp. 549–558, 2020.

[6] D. Upadhyay, J. Manero, M. Zaman, and S. Sampalli, "Gradient boosting feature selection with machine learning classifiers for intrusion detection on power grids," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 1104–1116, Mar. 2021, doi: 10.1109/TNSM.2020.3032618.

[7] D. Choi, S. Lee, D. Won, and S. Kim, "Efficient secure group communications for SCADA," *IEEE Trans. Power Del.*, vol. 25, no. 2, pp. 714–722, Apr. 2010.

[8] T. C. Pramod and N. R. Sunitha, "Polynomial based scheme for secure SCADA operations," *Procedia Technol.*, vol. 21, pp. 474–481, Nov. 2015.

[9] A. Rezai, P. Keshavarzi, and Z. Moravej, "Secure SCADA communication by using a modified key management scheme," *ISA Trans.*, vol. 52, no. 4, pp. 517–524, 2013.

[10] A. Rezai, P. Keshavarzi, and Z. Moravej, "Advance hybrid key management architecture for SCADA network security," *Security Commun. Netw.*, vol. 9, no. 17, pp. 4358–4368, 2016.

[11] D. Choi, H. Jeong, D. Won, and S. Kim, "Hybrid key management architecture for robust SCADA systems," *J. Inf. Sci. Eng.*, vol. 29, no. 2, pp. 281–298, 2013.

[12] R. Jiang, R. Lu, C. Lai, J. Luo, and X. Shen, "Robust group key management with revocation and collusion resistance for SCADA in smart grid," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2013, pp. 802–807.

[13] A. Rezai, P. Keshavarzi, and Z. Moravej, "A new key management scheme for SCADA networks," in *Proc. 2nd Int. Symp. Comput. Sci. Eng.*, 2011, pp. 373–378.

[14] S. Ghosh and S. Sampalli, "A survey of security in SCADA networks: Current issues and future challenges," *IEEE Access*, vol. 7, pp. 135812–135831, 2019.

[15] V. Manjunatha, A. Rao, and A. Khan, "Complex key generation with secured seed exchange for vernam cipher in security applications," *Mater. Today Proc.*, vol. 35, no. 3, pp. 497–500, 2021.

[16] R. Chandia, J. Gonzalez, T. Kilpatrick, M. Papa, and S. Shenoi, "Security strategies for SCADA networks," in *Proc. Int. Conf. Crit. Infrastruct. Protect.*, 2007, pp. 117–131.

[17] M. F. Moghadam, M. Nikooghadam, A. H. Mohajerzadeh, and B. Movali, "A lightweight key management protocol for secure communication in smart grids," *Electr. Power Syst. Res.*, vol. 178, Jan. 2020, Art. no. 106024.

[18] R. Dawson, C. Boyd, E. Dawson, and J. M. G. Nieto, "SKMA—A key management architecture for SCADA systems," in *Proc. 4th Aust. Symp. Grid Comput. e-Res. (AusGrid) 4th Aust. Inf. Security Workshop (Network Security) (AISW-NetSec)*, vol. 54, 2006, pp. 183–192.

[19] D. Choi, H. Kim, D. Won, and S. Kim, "Advanced key-management architecture for secure SCADA communications," *IEEE Trans. Power Del.*, vol. 24, no. 3, pp. 1154–1163, Jul. 2009.

[20] D. Wu and C. Zhou, "Fault-tolerant and scalable key management for smart grid," *IEEE Trans. Smart Grid*, vol. 2, no. 2, pp. 375–381, Jun. 2011.

[21] D. J. Kang, J. J. Lee, B. H. Kim, and D. Hur, "Proposal strategies of key management for data encryption in SCADA network of electric power systems," *Int. J. Electr. Power Energy Syst.*, vol. 33, no. 9, pp. 1521–1526, 2011.

[22] T. C. Pramod, G. S. Thejas, S. S. Iyengar, and N. Sunitha, "CKMI: Comprehensive key management infrastructure design for industrial automation and control systems," *Future Internet*, vol. 11, no. 6, p. 126, 2019.

[23] T. M. D. Hadley and K. A. Huston. *AGA-12, Part 2 Performance Test Results*. Accesed: Oct. 12, 2020. [Online]. Available: https://www.energy.gov/sites/prod/files/oeprod/DocumentsandMedia/9-AGA-12_Part_2_Performance.pdf

[24] D. Abbasinezhad-Mood, A. Ostad-Sharif, and M. Nikooghadam, "Novel anonymous key establishment protocol for isolated smart meters," *IEEE Trans. Ind. Electron.*, vol. 67, no. 4, pp. 2844–2851, Apr. 2020.

[25] N. Saxena, B. J. Choi, and R. Lu, "Authentication and authorization scheme for various user roles and devices in smart grid," *IEEE Trans. Inf. Forensics Security*, vol. 11, pp. 907–921, 2015.

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

[27] M. Keshk, E. Sitnikova, N. Moustafa, J. Hu, and I. Khalil, "An integrated framework for privacy-preserving based anomaly detection for cyber-physical systems," *IEEE Trans. Sustain. Comput.*, vol. 6, no. 1, pp. 66–79, Jan.–Mar. 2021.

[28] J. Qian, C. Hua, X. Guan, T. Xin, and L. Zhang, "A trusted-id referenced key scheme for securing SCADA communication in iron and steel plants," *IEEE Access*, vol. 7, pp. 46947–46958, 2019.

[29] D. G. Brosas, A. M. Sison, and R. P. Medina, "Modified OTP based Vernam Cipher algorithm using multilevel encryption method," in *Proc. IEEE Eurasia Conf. IOT Commun. Eng. (ECICE)*, 2019, pp. 201–204.

[30] R. C. B. Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," in *Proc. 7th Int. Symp. Resilient Control Syst. (ISRCS)*, Aug. 2014, pp. 1–8.

[31] R. Zazkis, "Representing numbers: Prime and irrational," *Int. J. Math. Educ. Sci. Technol.*, vol. 36, nos. 2–3, pp. 207–217, 2005.

[32] Wikipedia. *Blake (Hash Function)*. Accesed: May 12, 2021. [Online]. Available: https://en.wikipedia.org/wiki/BLAKE_(hash_function)

[33] WolfSSL. *Embedded TLS Library for Applications, Devices, IoT, and the Cloud*.Accessed: Aug. 12, 2020. [Online]. Available: https://www.wolfssl.com/download

[34] Libntru. *The NTRU Project*. Accessed: Aug. 12, 2020. [Online]. Available: https://tbuktu.github.io/ntru/

[35] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein, "BLAKE2: Simpler, smaller, fast as MD5," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Security*, 2013, pp. 119–135.

[36] J. O'Connor and J.-P. Aumasson. *BLAKE2: Simpler, Smaller, Fast as MD5*. Accesed: Feb. 15, 2021. [Online]. Available: https://www.blake2.net/blake2.pdf

[37] J. O'Connor, S. Neves, and Z. Winnerlein. *Blake3—One Function, Fast Everywhere*. Accesed: Feb. 12, 2021. [Online]. Available: https://github.com/BLAKE3-team/BLAKE3-specs/raw/master/blake3.pdf

[38] J. O'Connor, S. Neves, and Z. Winnerlein. *Blake3 is an Extremely Fast, Parallel Cryptographic Hash*. Accesed: Feb. 15, 2021. [Online]. Available: https:/www.infoq.com/news/2020/01/blake3-fast-crypto-hash/

[39] H. Delfs, H. Knebl, and H. Knebl, *Introduction to Cryptography*, vol. 2. New York, NY, USA: Springer, 2002.

[40] A. A. Kamal and A. M. Youssef, "An FPGA Implementation of the NTRUEncrypt cryptosystem," in *Proc. Int. Conf. Microelectron.*, 2009, pp. 209–212.

[41] J. Hermans, F. Vercauteren, and B. Preneel, "Speed records for NTRU," in *Proc. Cryptogr. Track RSA Conf.*, 2010, pp. 73–88.

[42] J. N. Gaithuru and M. Bakhtiari, "Insight into the operation of NTRU and a comparative study of NTRU, RSA and ECC public key cryptosystems," in *Proc. 8th. Malaysian Softw. Eng. Conf. (MySEC)*, 2014, pp. 273–278.

[43] D. Stehlé and R. Steinfeld, "Making NTRU as secure as worst-case problems over ideal lattices," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2011, pp. 27–47.

[44] C. Jacomme and S. Kremer, "An extensive formal analysis of multi-factor authentication protocols," *ACM Trans. Privacy Security*, vol. 24, no. 2, pp. 1–34, 2021.

[45] N. Mouha and A. Hailane, "The application of formal methods to real-world cryptographic algorithms, protocols, and systems," *Computer*, vol. 54, no. 1, pp. 29–38, Jan. 2021.

[46] S. Szymoniak, "Security protocols analysis including various time parameters," *Math. Biosci. Eng.*, vol. 18, no. 2, pp. 1136–1153, 2021.

[47] N. Dalal, J. Shah, K. Hisaria, and D. Jinwala, "A comparative analysis of tools for verification of security protocols," *Int. J. Commun. Netw. Syst. Sci.*, vol. 3, no. 10, p. 779, 2010.

[48] A. H. Shinde, A. Umbarkar, and N. Pillai, "Cryptographic protocols specification and verification tools—A survey," *ICTACT J. Commun. Technol.*, vol. 8, no. 2, pp. 1533–1539, 2017.

[49] C. J. Cremers, "The scyther tool: Verification, falsification, and analysis of security protocols," in *Proc. Int. Conf. Comput. Aided Verification*, 2008, pp. 414–418.
[50] X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD," IACR, Lyon, France, Rep. 2004/199, 2004.
[51] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The first collision for full SHA-1," in *Proc. Annu. Int. Cryptol. Conf.*, 2017, pp. 570–596.
[52] Y. Sasaki, L. Wang, and K. Aoki, "Preimage attacks on 41-step SHA-256 and 46-step SHA-512," IACR, Lyon, France, Rep. 2009/479, 2009.
[53] D. A. Osvik, "Fast embedded software hashing," IACR, Lyon, France, Rep. 2012/156, 2012.
[54] J. Vidali, P. Nose, and E. Pašalić, "Collisions for variants of the BLAKE hash function," *Inf. Process. Lett.*, vol. 110, nos. 14–15, pp. 585–590, 2010.
[55] J. Daemen and G. Van Assche, "Producing collisions for PANAMA, instantaneously," in *Proc. Int. Workshop Fast Softw. Encrypt.*, 2007, pp. 1–18.
[56] M. Coutinho, R. T. De Sousa, and F. Borges, "Continuous diffusion analysis," *IEEE Access*, vol. 8, pp. 123735–123745, 2020.
[57] N. Abdoun. (2019). *Design, Implementation and Analysis of Keyed Hash Functions Based on Chaotic Maps and Neural Networks*. [Online]. Available: https://hal.archives-ouvertes.fr/tel-02271074/document
[58] N. Abdoun, S. E. Assad, T. M. Hoang, O. Deforges, R. Assaf, and M. Khalil, "Designing two secure keyed hash functions based on sponge construction and the chaotic neural network," *Entropy*, vol. 22, no. 9, p. 1012, 2020. [Online]. Available: https://www.mdpi.com/1099-4300/22/9/1012
[59] S. Al-Kuwari, J. H. Davenport, and R. J. Bradford, "Cryptographic hash functions: Recent design trends and security notions," IACR, Lyon, France, Rep. 2011/565, 2011. [Online]. Available: https://eprint.iacr.org/2011/565
[60] H. Feistel, "Cryptography and computer privacy," *Sci. Amer.*, vol. 228, no. 5, pp. 15–23, 1973. [Online]. Available: http://www.jstor.org/stable/24923044
[61] M. Stevens, "Attacks on hash functions and applications," Ph. D. dissertation, Math. Inst., Fac. Sci., Leiden Univ., Leiden, The Netherlands, 2012.
[62] H. Wang, Z. Ma, and C. Ma, "An efficient quantum meet-in-the-middle attack against NTRU-2005," *Chin. Sci. Bull.*, vol. 58, nos. 28–29, pp. 3514–3518, 2013.
[63] S. N. Kumar, "Review on network security and cryptography," *Int. Trans. Electr. Comput. Eng. Syst.*, vol. 3, no. 1, pp. 1–11, 2015.
[64] S. Ghosh, M. LeMay, D. M. Durham, and M. R. Sastry, "Processor hardware and instructions for SHA3 cryptographic operations," U.S. Patent 16 709 837, Apr. 16 2020.

Darshana Upadhyay received the master's degree in computer science from Nirma University, Ahmedabad, India. She is currently pursuing the Ph.D. degree with the Faculty of Computer Science, Dalhousie University. She also served as a Lecturer with Nirma University, before moving to Canada to pursue her Ph.D. degree. She was awarded the Gold Medal for securing the first position during her graduate study. Her primary research includes algorithm conceptualization, hardware design in the field of embedded systems, vulnerability assessments, and intrusion detection techniques for IoT/SCADA based systems. She is the co-recipient of the Indo-Canadian Shastri Research Grant in the field of wireless security and intrusion detection systems. She has been invited to be one of the Women in International Security–Canada's 2020 Emerging Thought Leaders.

Marzia Zaman received the M.Sc. and Ph.D. degrees in electrical and computer engineering from the Memorial University of Newfoundland, Canada, in 1993 and 1996, respectively. She started her career with Nortel Networks, Ottawa, ON, Canada, in 1996, where she joined the Software Engineering Analysis Lab and later joined the Optera Packet Core Project as a Software Developer. She has many years of industry experience as a Researcher and a Software Designer with Accelight Networks, Excelocity, Sanstream Technology, and Cistel Technology. Since 2009, she has been working closely with the Centre for Energy and Power Electronics Research, Queen's University, Canada, and one of its industry collaborators, Cistel Technology, on multiple power engineering projects. Her research interests include renewable energy, wireless communication, IoT, cyber security, machine learning, and software engineering.

Rohit Joshi received the bachelor's degree in mechanical engineering from the Birla Institute of Technology, Mesra, India, and the master's degree in innovation and technology management from the University of New Brunswick Saint John, Canada. He has over 20 years of experience in the domains of information security, risk management, and networking across multiple geographies. He has worked with organizations, such as Cistel Technology, Inc., Mariner Partners, HCL Technologies, Ramco System, and Sify Technologies Limited handling a variety of roles and providing end-to-end, IT security management consulting and solutions to large clients across various industry verticals. At Sify Technologies Limited, he was associated with Safescrypt which was the first licensed certifying authority in India that was set up in association with Verisign. His research interest include wireless communication, IoT, and cyber security.

Srinivas Sampalli (Member, IEEE) received the Bachelor of Engineering degree from Bangalore University and the Ph.D. degree from the Indian Institute of Science, Bangalore, India. He is currently a Professor and a 3M National Teaching Fellow with the Faculty of Computer Science, Dalhousie University. He has led numerous industry-driven research projects on Internet of Things, wireless security, vulnerability analysis, intrusion detection and prevention, and applications of emerging wireless technologies in healthcare. He currently oversees and runs the Emerging Wireless Technologies (MYTech) Lab and has supervised over 150 graduate students in his career. His primary joy is in inspiring and motivating students with his enthusiastic teaching. He has received the Dalhousie Faculty of Science Teaching Excellence Award, the Dalhousie Alumni Association Teaching Award, the Association of Atlantic Universities' Distinguished Teacher Award, the Teaching Award Instituted in his name by the students within his Faculty, and the 3M National Teaching Fellowship, Canada's most prestigious teaching acknowledgement. Since September 2016, he holds the honorary position of the Vice President (Canada), of the International Federation of National Teaching Fellows, a consortium of national teaching award winners from around the world.