A Novel Secure Root Key Updating Scheme for LoRaWANs Based on CTR_AES DRBG 128

NUR HAYATI, (Member, IEEE), KALAMULLAH RAMLI, (Member, IEEE), SUSILA WINDARTA, (Member, IEEE), AND MUHAMMAD SURYANEGARA, (Senior Member, IEEE)
Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia, Depok, Jawa Barat 16424, Indonesia
Corresponding author: Kalamullah Ramli (kalamullah.ramli@ui.ac.id)

ABSTRACT A long-range wide area network (LoRaWAN) has a weakness in terms of key management: its root key is static, meaning that it never changes. Since all cryptographic keys are derived from the root key, such a weakness endangers LoRaWAN security. This paper proposes a novel secure root key updating scheme for LoRaWAN that involves periodically changing the root key value based on the CTR_AES DRBG 128 algorithm. The scheme consists of two sequential phases: the initialization process that occurs at the end device and the root key update process that occurs at the join server. To validate the proposed scheme, we conduct randomness and communication protocol tests. The results indicate that the proposed scheme has a high degree of randomness, passes all 15 statistical tests in the NIST suite, and has secure communication protocols. The analyses verify that the new scheme has a mechanism to resist replay attacks and protects data integrity. The main advantage of the scheme is that it has a perfect forward secrecy feature that enhances the root key updating scheme with a lightweight computational load for the end device; additionally, root key updating can be performed automatically from a remote distance within the LoRaWAN coverage network. The proposed scheme also supports simultaneous updates for implementation in a large area with many devices. Finally, the overall results demonstrate how our proposed scheme may compensate for the weakness of LoRaWANs in key management and improve their security performance.

INDEX TERMS LoRaWAN security, cryptographic key, root key update, CTR_AES DRBG, key management.

I. INTRODUCTION
Long-range wide-area networks (LoRaWANs) are Internet of Things (IoT) technologies that have become the low-power wide-area (LPWA) standard through Recommendation ITU-T Y.4480 [1]. In the last few years, many academic researchers have studied the potential applications of LoRaWANs with respect to various fields, such as healthcare [2]–[4] or smart cities [5]–[7]. LoRaWANs are also widely applied in the industrial IoT (IIoT) [8]–[10], where there are more than 2.7 million LoRa-based gateways used worldwide, more than 163 public and private network operators in various countries, and a total of 225 million units of implemented LoRa-based end devices [11]. However, LoRaWANs have a weakness in terms of their security key management [12]–[15]. This weakness has become a technical challenge since data security and privacy are essential aspects of IoT implementation [16]–[20].

The weakness of LoRaWAN key management is associated with the life cycle of the root key. The root key is the primary cryptographic key used to calculate integrity parameters and derive all other cryptographic keys [21]. In LoRaWANs, the root key is static [22]. The value remains the same [23] throughout the device’s lifespan, implying that its crypto period exceeds the recommended value [24]–[26]. Even the LoRaWAN specification document from version 1.0 to 1.1 does not provide a scheme to update the root key [27]–[30]. Consequently, one of the adverse effects of such an un-updated root key is the potential for key compromises. The LoRaWAN root key is stored in the end device,
located remotely [31], so that the attacker has plenty of time to learn the existing key.

Based on the concept described above, a new scheme for periodically updating the root key is essential for improving LoRaWAN security [32], [33]. Some studies in [24], [31], [34] have proposed schemes to update the static root key. However, they come with some limitations, such as providing neither a standard validation for the randomness of the newly generated root key nor a formal security analysis validation for the communication protocol.

In this paper, we propose a novel secure root key updating scheme for LoRaWANs by utilizing the CTR_AES DRBG 128 bits algorithm. Our proposed scheme works in two sequential phases: the initialization process that occurs at the end device and the root key update process that occurs in the join server. To validate the proposed scheme, we conduct 2 (two) security tests, including a randomness test based on the 15 parameters in the NIST statistical test suite and communication protocol tests by modeling the scheme in secure network testing using Scyther validation tools. Furthermore, we analyze the proposed scheme in terms of aspects of replay attack resistance, integrity protection, perfect forward secrecy, lightweight key updating, dynamic-remote-and-simultaneous root key updates, and implementation compatibility.

The main contribution of our work is a novel secure root key updating scheme for LoRaWANs that improves the security of LoRaWAN root key management. We demonstrate that our proposal is utterly secure by utilizing a formal analysis approach. In addition, the generated root-key satisfies the NIST randomness test. The proposed scheme’s performance evaluation shows better results than the existing schemes in terms of lightweight protocol steps from the end device point of view.

The remainder of the paper is organized into sections addressing the theories of LoRaWAN cryptographic keys and related works (Section II), our proposed novel secure root key updating scheme for LoRaWANs (Section III), the validation test, the results and discussion (Section IV), and the conclusions and future work (Section V).

II. THEORY AND RELATED WORK
A. LoRaWAN CRYPTOGRAPHIC KEY
LoRaWAN infrastructure has security features to protect communication and data privacy [35]. These security features involve AES encryption algorithms with two types of symmetric keys: root keys and session keys [36]–[38]. A root key is a pre-shared key distributed during commissioning [39]. Moreover, a session key is a derivative key of a root key. A LoRaWAN session key is used to secure communications and payload transmission [36].

In LoRaWAN standardization version 1.0, the root key consists of the network key (NwkKey) only, and in LoRaWAN version 1.1, the root key consists of the NwkKey and the application key (AppKey). The LoRaWAN root key has some essential roles other than being used as an end-to-end encryption key between the pair of end devices and the join server. The root key acts as a key in message integrity code (MIC) calculation of the LoRaWAN. The LoRaWAN data transmission and authentication processes through over-the-air activation (OTAa) mode are protected by MIC [38]. Illustrations of the LoRaWAN infrastructure and its cryptographic keys are displayed in Fig. 1.

Another essential role of the root key is that it is utilized as a security parameter to derive other LoRaWAN cryptographic keys. JSEncKey and JSIntKey are both keys derived from the root key. JSEncKey is the encryption key of the Join-accept message in response to a Rejoin-request. Additionally, JSIntKey is the key used to calculate the MIC value of Join-accept and Rejoin-request type 1 messages. The subsequent cryptographic key derived from the root key is the session key. The session key is generated through the key derivation function (KDF) process using the AES_ECB algorithm. In LoRaWAN version 1.1, the session key resulting from the AppKey derivation is the application session key (AppSKey). Within the NwkKey derivation, there are various types of network session keys [27], [36]. These session keys are then directly used to protect data transmission from the end device to the network server and application server, and vice versa.

The static LoRaWAN root key makes the same key that is used to derive all session keys and other cryptographic keys [21]. Then, if the root key is compromised, it can compromise all derived cryptographic keys and other security parameters. When this accident occurs, it endangers the LoRaWAN infrastructure. Hence, a scheme to update the root key is necessary to keep LoRaWANs secure.

B. RELATED WORKS
Researchers have conducted several studies to improve the security of LoRaWAN key management. One study [40] proposed a comprehensive LoRaWAN key management scheme, from key generation to key destruction. In a centralized key management system, where the join server acts as a centralized key management server (CKMS), the LoRaWAN key updates scheme can be performed based on if it is event-driven or time-driven [40]. Meanwhile, some studies on LoRaWAN key management focus on session key updating schemes, as conducted in research [15], [37], [41], while others focus on root key updating schemes.
Another study [24] proposed an 'assisted mode' key management scheme to improve LoRaWAN security. The assisted mode resolves the issue by freeing end devices from lifelong key management and delegating the task to the master device. The master device can be a laptop, where the master device and end device must be nearby during the key updating process through the 'charge security battery' procedure. The proposed scheme is proximity based and used for wearable end devices based on LoRaWANs. Thus, it is only suitable for limited environments or specific use cases, such as for healthcare. The proposed scheme’s shortcoming is that it is less effective for large-scale implementation at remote distances where the end devices simultaneously require a key updating scheme.

Subsequent research [31] proposed a lightweight root key updating scheme using the two-step KDF based on the rabbit stream cipher algorithm. The two-step KDF involves two phases: an extractor and an expander. It should be noted that the two-step KDF is carried out at the end device and the join server individually. The proposed solution has been claimed to provide a highly random newly generated key, but the research did not provide a sufficient randomness test. Moreover, to the best of our knowledge, the proposed KDF is not suitable for LoRaWAN end devices since an extra computational cost is accrued. The consequence of changing the root key value is changing the session key. Therefore, the end device must run two types of KDF to update the root key and the session key. Furthermore, the additional load of the end device emerges from the communication step. The end device should run four steps to update its root key by the two-step KDF scheme, including the 'Root Key Updating Request with the Context' step, the 'Confirmation' step, the 'New Key Ready' step, and the 'New Key Ready Confirmation' step.

In addition, the proposed root key updating scheme applied in [34] utilized an elliptic curve cryptography (ECC) algorithm. Each end device and server used a hierarchical deterministic (HD) wallet, where the key pair in the HD wallet was generated based on a key agreement with an ECC algorithm. The drawback of this scheme was that it had an extra computational load at the end device since the root key update was performed on the end device. Additionally, the proposed scheme consisted of several phases wherein each phase, starting from registration to agreement, had two to three communication steps.

These previous studies have resolved the static root key problem, but there are still some shortcomings. Therefore, the LoRaWAN root key updating scheme should resolve this fundamental problem using several optimization features. This feature includes a validation standard for the randomness of the root key since the LoRaWAN root key plays an essential role in deriving the other cryptographic key. The next feature is a security validated communication protocol to exchange messages between the involved entities during the root key update execution. Lightweight computation features for the end device should also be considered to ensure that the device has a longer lifetime. In addition, the root key updating scheme should support remote and simultaneous updates because many LoRaWAN end devices are generally implemented at a far distance, with vast LoRaWAN coverage areas.

This research proposes a novel secure root key update of LoRaWAN infrastructure based on the CTR_AES DRBG 128 bits algorithm. The proposed scheme has secure communication protocols and can produce new root keys with a high degree of random bit sequences. Additionally, the proposed scheme is also resistant to replay attack, supports perfect forward secrecy of the new root keys, has a lightweight computation load for the end device, and supports a dynamic-remote-and-simultaneous root key updates scheme.

### III. PROPOSED SECURE ROOT KEY UPDATING SCHEME FOR LoRaWANs

The proposed root key updating scheme is executed in the LoRaWAN join server, assuming that the join server is a trusted entity and located in a secure environment. A secure environment implies that the LoRaWAN join server has had a mitigation mechanism to prevent various attacks, including denial of service (DoS) or distributed denial of service (DDoS) attacks. Since we assume that the join server already has a mitigation mechanism as a mandatory prerequisite to avoid the possibility of an attack, time-driven DoS or DDoS is not a problem. Relevant works supporting our statement are found in [42]–[44]. Previous research [42] discussed the mitigation mechanism using a semisupervised machine learning algorithm and [43] applied an entropy-based solution based on a software-defined networking (SDN) architecture. Researchers [44] studied the approach by utilizing a Markov model.

From the end device perspective, the proposed scheme requires only a small additional computation process. At the same time, the join server can process many key update requests and provide a high degree of random root keys. Therefore, the proposed scheme can restrict attackers from obtaining information through a compromised LoRaWAN key.

#### A. CTR_AES DRBG 128

The proposed scheme utilized CTR_AES DRBG 128 bits as a root key updating algorithm. Such an algorithm works on the basis of the deterministic random bit generator (DRBG) system. The DRBG system utilized for the scheme operates based on a block cipher algorithm with a counter mode, CTR_DRBG [45]. The DRBG is used to generate a sequence of random bits with a high degree of randomness. Additionally, the sequence is difficult to predict, making it suitable for a cryptographic key [45]. To produce such an output, the CTR_DRBG system requires a cryptographic module that must comply with the FIPS-140 standard [46].

In our scheme, the block cipher-based AES algorithm with a counter mode, AES_CTR, is selected as the encryption algorithm in DRBG block encryption. The performance of the AES_CTR algorithm was tested in [47] to ensure
that it is suitable as the internal state algorithm in DRBG block encryption. However, some researchers found security vulnerability in the CTR_DRBG, as discussed in [48]–[50]. A practical attack occurred in CTR_AES DRBG 128, and the attacker could obtain the input of the internal state that was being used. However, we can address this vulnerability by applying a frequently reseeded mechanism to the internal state of the DRBG block encryption, as Cohney et al. [50] suggested.

**B. PROPOSED ROOT KEY UPDATING SCHEME**

The proposed root key updating scheme involves two LoRaWAN infrastructure entities, namely, the end device and the join server. The join server acts as an online key generation center [51]. This study determined that the root key updating scheme applies to LoRaWANs in OTAA join procedure mode. As illustrated in Fig. 1, in LoRaWAN field implementation, data transmitted from the end device passes through the gateway and is then forwarded to the network server and the join server. However, in the proposed design, we state that only two entities are involved because only the end device and the join server are directly involved in processing the root key updating scheme.

Our proposed scheme employs the time-driven approach based on a periodic update [40]. In this approach, the key updating scheme is carried out periodically at a specific time, the scheduled timestamp. The root key update period can be performed every n days, for example, once a week [36], [37]. The period selection approach should consider the computational load on the end device, where appropriate root key updates are less frequent [31], [40]. Additionally, the cryptoperiod of the key must be maintained. Then, since the proposed root key updating scheme is a time-driven approach, time synchronization between the end device and the join server should be performed during commissioning. To provide further explanation, we summarize these technical terms and their definitions in Table 1.

The proposed root key updating scheme for LoRaWANs consists of two phases. **Phase-1** is an initialization process that occurs in the ED and **phase-2** is the root key update process that occurs in the JS. Fig. 2 displays the highlighted design of the proposed root key updating scheme.

In the proposed scheme, we apply four communication protocols used to exchange messages between ED and JS during the root key updating process. ED sends two protocols, and JS sends two protocols. The protocols sent by ED are **New_Join-request** and **New_Rejoin-request**. **New_Join-request** is a modified protocol of the Join-request message of the LoRaWAN used by ED for authentication when joining the network. **New_Rejoin-request** is a modified protocol of the Rejoin-request type 1 message of the LoRaWAN. However, in this scheme, both the **New_Join-request** and **New_Rejoin-request** protocols are used to request a new root key update based on a scheduled time. Furthermore, two protocols are sent by JS to respond to the request for a new root key update: **New_Join-accept** and **New_Rejoin-accept**.

**FIGURE 2.** Highlighted design of the proposed root key updating scheme of LoRaWAN.

**Fig. 3** depicts the details of the proposed root key updating scheme involving a pair ED and JS, with the following explanations.

1) **PHASE-1: INITIALIZATION PROCESS**

The initialization process of the proposed root key updating scheme consists of the following four steps:

1) In the first step, **ED** retrieves the scheduled timestamp, \( T_s \). The root key updating schedule is predetermined during commissioning and is synchronized with that of JS, and JS has a database to keep each device’s root key updating schedule under its care.

2) In the second step, **ED** retrieves its counter value and stores it as **Count**. If the **Count** = 0, then **ED** denoted the value as **DevNonce**. This value indicates that **ED** newly joins the LoRaWAN network via the OTAA mode. Then, if the **Count** = 0 ≤ 216 − 1, it is denoted as **RJcount**. The counter’s upper limit is adjusted based on the counter value of the LoRaWAN’s message field, as defined by the specification. Next, if the counter value reaches the maximum limit, then the **ED** needs to reset the counter 36.

3) The third step: **ED** calculates **MIC** to secure messages sent to the **JS**. The algorithm used for **MIC** calculation is the same as the standard, **AES-CMAC 128 bits**. However, there is an additional timestamp value on the calculated component. **MIC** \( E_j \) is the **MIC** calculated by each **ED**. The codes \( j \) and \( r \) are added to distinguish the **MIC** type of the **New_Join** and **New_Rejoin** messages, respectively. When **Count** = 0, the **DevNonce** message field is used, and the calculation of **MIC** \( E_j \) is as follows:

\[
\begin{align*}
\text{cmac}_j &= \text{aes128_cmac}(\text{NwkKey}, \text{MHDR}_{ED} | \text{JoinEUI} | \text{DevEUI} | \text{DevNonce} | T_s) \\
\text{MIC}_{E_j} &= \text{cmac}_j[0..3]
\end{align*}
\]
TABLE 1. Technical term and definition (abbreviation) used in this study.

| Term                        | Definition                                                                 |
|-----------------------------|-----------------------------------------------------------------------------|
| $ED$                        | End Device                                                                  |
| $JS$                        | Join Server                                                                 |
| $Ts$                        | End Device’s Timestamp                                                      |
| $Ts'$                       | Join Server’s Timestamp                                                     |
| $\Delta T$s                 | Deviation time between $JS$ and $ED$, $\Delta T$s = $Ts'$ − $Ts$           |
| $\text{NwkKey}_{\text{old}}$| NwkKey obtained from the current value before the new root key is applied |
| $\text{JSEncKey}_{\text{old}}$| JSEncKey obtained from the current value before the new root key is applied |
| $\text{Count}$             | The counter value of each end device, it can be DevNonce or RJCount1        |
| $\text{New}_\text{Join-request}$ | A new format of the Join-request message sent from $ED$ to $JS$ as an initial request in processing the root key update procedure |
| $\text{New}_\text{Rejoin-request}$ | A new format of the Rejoin-request type 1 message sent from $ED$ to request root key update’s procedure |
| $\text{MIC}_{E_{\text{EJ}}}^J$ | Message Integrity Code of the $\text{New}_\text{Join-request}$             |
| $\text{MIC}_{E_{\text{EJ}}}^{R}$ | Message Integrity Code of the $\text{New}_\text{Rejoin-request}$          |
| $\text{MHDR}_{ED}$          | End Device’s Message Header                                                 |
| $\text{MIDR}_{JS}$          | Join Server’s Message Header                                                |
| $\text{Key}$               | 128 bits Pseudorandom Bit Sequence generated by Random Bit Generator (RBG) |
| $\text{Nonce}_{\text{Count}}$ | 112 bits Pseudorandom Bit Sequence as Nonce of the counter, it generated by RBG |
| $\text{New}_\text{Root}_{\text{Key}}$ | LoRaWAN new root key, LoRaWAN dynamic root key which is periodically updated |
| $\text{JContext}$          | Context data related to Join-accept                                          |
| $\text{RContext}$          | Context data related to Rejoin-accept                                        |
| $\text{MIC}_{E_{\text{R}}}^J$ | Message Integrity Code of the $\text{New}_\text{Rejoin-request}$          |
| $\text{Message}$            | Default Join-accept data, containing join-accept related information        |
| $\text{New}_\text{Join-accept}$ | Message sent from $JS$ to $ED$ in response to the $\text{New}_\text{Join-request}$ |
| $\text{New}_\text{Rejoin-accept}$ | Message sent from $JS$ to $ED$ in response to the $\text{New}_\text{Rejoin-request}$ |

Moreover, when $\text{Count} = 0 \leq 2^{16} − 1$, the $ED$ must ensure that $\text{JSIntKey}$ has been derived from the existing root key before the MIC is calculated. The default algorithm is used to process the derivation and utilize $\text{JSIntKey}$, but with the updated root key value. Thus, the MIC calculation is as follows:

- $\text{JSIntKey} = \text{aes128_encrypt(NwkKey, } 0 \times 06|\text{DevEUI}|\text{pad16})$
- $\text{CMAC}_{r} = \text{aes128_cmac(}\text{JSIntKey, MHDR}_{ED}|\text{ReJoinType1}|\text{JoinEUI}|\text{DevEUI}|\text{RJCount1}|\text{Ts})$

4) After completing the $\text{MIC}_{E_{\text{EJ}}}^J$ or $\text{MIC}_{E_{\text{EJ}}}^{R}$ calculation, the fourth step is to concatenate the $ED$ header, the format join request, the rejoin request, and the $ED$ trailer to form a PHY payload. The payload then sent to $JS$ has the following format:

- $\text{New}_\text{Join} − \text{request} = \text{MHDR}_{ED}, (\text{JoinEUI}, \text{DevEUI}, \text{DevNonce}, \text{Ts}), \text{MIC}_{E_{\text{EJ}}}^J$
- $\text{New}_\text{Rejoin} − \text{request} = \text{MHDR}_{ED}, (\text{ReJoinType1}, \text{JoinEUI}, \text{DevEUI}, \text{RJCount1}, \text{Ts}), \text{MIC}_{E_{\text{EJ}}}^R$

The details of the PHY payload format of the message field sent during the initialization process of the root key updating scheme are illustrated in Fig. 4 and Fig. 5. The figures show the different formats of the message fields.

There is a different format of the message field between the original version of the Join-request and the Rejoin-request specified by the LoRaWAN [36] and the proposed ones. The difference is that there is an additional 4-byte timestamp at the end of the default message. Four bytes is the standard size of the LoRaWAN timestamp [36]. In addition, the message field format can be addressed as a differentiator for whether $ED$ supports the root key update feature or not.
2) PHASE-2: ROOT KEY UPDATING PROCESS

Upon receiving the message from ED, JS processes the root key update using the CTR_AES DRBG 128 algorithm with the following steps:

1) The JS calculates the $MIC_{EJ}$ value. Depending on the received message type, the MIC can be either $MIC_{EJj}$ or $MIC_{EJr}$. JS continues to check whether the value is correct, goes to the next step or is incorrect, and then...
sends a notification to the associated ED to repeat the process in predetermined time intervals.

2) The JS retrieves its server timestamp, $T_s'$, and calculates the deviation between the received ED timestamp and the JS timestamp, $T_s' - T_s$ $\leq \Delta T_s$; the $\Delta T_s$ value should be in a range with the scheduled root key update of the associated ED. If the $\Delta T_s$ value is a mismatch, it sends a notification to the associated ED. The root key update is processed when the MIC and $\Delta T_s$ calculation results are correct. When the $\Delta T_s$ and $M_{IC_{JE}}$ calculations are matched, JS updates its database by labeling the current NwkKey as NwkKeyold. Similarly, when the $\Delta T_s$ and $M_{IC_{JE}}$ calculations are matched, JS changes the label of JSIntKey to JSIntKeyold. To summarize, when both the $\Delta T_s$ and $M_{IC_{JE}}$ calculations are correct, the JS labels all current associated ED keys as old keys, including labeling JSEncKey as JSEncKeyold. The last task of this step is that the JS retrieves the counter value from DevNonce or RJCount1 message fields.

3) JS instructs the random bit generator (RBG) to provide the algorithm's input parameters. The RBG generates the main input parameters of the root key updating algorithm. The JS then asks the RBG to generate two pseudorandom bit sequence values of 128 and 112 bits.

4) JS arranges the CTR_AES DRBG 128 bits algorithm's input parameters since the algorithm requires two different inputs: a key and a counter. The key value is a 128-bit pseudorandom bit sequence, while the counter is a concatenation of a 112-bit pseudorandom bit sequence stored as Nonce_Count and 16-bit DevNonce or RJCount1 as follows:

- $Key = 128$ bitpseudorandombitsequence
- $Counter = 112$ bitNonce_Count
  16bitDevNonce or RJCount1

5) The JS processes the root key update utilizing the CTR_AES DRBG 128 algorithm. Fig. 8 visualizes the root key updating algorithm, and the formula is expressed as follows:

- $New\_Root\_Key = CTR\_AES\_DRBG\_128(Key, Nonce\_Count|DevNonce)$ or
- $New\_Root\_Key = CTR\_AES\_DRBG\_128(Key, Nonce\_Count|RJCount1)$

6) After the key update process is complete, JS prepares a context to calculate $M_{IC_{JE}}$ in this sixth step. The context value is adjusted to the counter value as follows:

- $JContext = JoinEUI|DevNonce|MHRDJ_S\_128\_{JoinNonce}$
  $NetID|DevAddr|DLSettings|RxDelay|CFList$
- $RContext = JoinEUI|RJCount1|MHRDJ_S\_128\_{JoinNonce}$
  $NetID|DevAddr|DLSettings|RxDelay|CFList$

The algorithm used for the $M_{IC_{JE}}$ calculation is aes128cmac; then, the value of $M_{IC_{JE}}$ has the following format:

- $cmac_j = aes128cmac($JSIntKey_old, 0xFF$|JContext|New\_Root\_Key)$
- $M_{IC_{JE}} = cmac_j\{0..3\}$

$M_{IC_{JE}}$ is the MIC used to secure the $New\_Join-accept$ message, and $M_{IC_{JE}}$ is used to secure the $New\_Rejoin-accept$ message. The calculation of $M_{IC_{JE}}$ is as follows:

- $cmac_r = aes128cmac($JSIntKey_old, 0 × 01$|RContext|New\_Root\_Key)$
- $M_{IC_{JE}} = cmac_r\{0..3\}$

7) The JS prepares a $Jmessage$, the original version of the $Join-accept$ message, then encrypts it along with the $New\_Root\_Key$ and the $M_{IC_{JE}}$. The encryption process is defined as follows:

- $JMessage = JoinNonce|NetID|DevAddr|DLSettings|RxDelay|CFList$
- $Enc\_Join – accept = aes12decrypt(NwkKey\_old, JMessage|New\_Root\_Key|M_{IC_{JE}})$
- $Enc\_Rejoin – accept = aes12decrypt(JSEncKey\_old,\newline JMessage|New\_Root\_Key|M_{IC_{JE}})$

AES – ECB 128 bits decryption is the algorithm used to encrypt the message. It is the default algorithm for encrypting the $Join-accept$ message [24], but the key and message components of the algorithm have been modified to support the proposed scheme.

8) In the last step, the JS concatenates $MHDRJS$ to $Enc\_Join-accept$ or $Enc\_Rejoin-accept$ and then sends the message to the associated ED. The format of the $New\_Join-accept$ or $New\_Rejoin-accept$ is as follows:

- $New\_Join-accept = \{MHDRJS, aes12decrypt(NwkKey\_old, JMessage|New\_Root\_Key|M_{IC_{JE}})\}$
- $New\_Rejoin-accept = \{MHDRJS, aes12decrypt(JSEncKey\_old, JMessage|New\_Root\_Key|M_{IC_{JE}})\}$

The $New\_Join-accept$ is a message sent by JS to reply to the $New\_Join-request$, while the $New\_Rejoin-accept$ is a reply message to the $New\_Rejoin-request$. The size and PHY payload format of the $New\_Join-accept$ and $New\_Rejoin-accept$ message fields are depicted in Fig. 6 and Fig. 7, respectively.

The $CTR\_AES\_DRBG$ 128 algorithm, which is used to update the LoRaWAN root key and is described in the fifth phase of step 2 has a design structure displayed in Fig. 8.
In the algorithm design, we assume that the module used as the RBG system has complied with the FIPS-140 standard as a key generation module [46]. Module compliance is needed for the RBG to ensure that it can produce unpredictable bit sequences used as the key material. The key material is a critical part of the input parameter for the security process in the DRBG internal state.

Every time the root key update is processed, it requires two input parameters: key and counter. The key parameter is obtained from the direct output of the RBG system, while the counter should be calculated first. In this scheme, the counter-parameter is obtained from a combined value between the RBG output and each ED’s counter. Then, both input parameters eventually acquire the same size, 128 bits. The 128 bit input parameter size is selected considering the cryptographic key security strength [52] and the size of the LoRaWAN root key on LoRaWAN, which has a length of 128 bits [36]. Thus, each root key update process takes only one iteration.

The reseed counter process is a function to update the key and counter values. In the proposed algorithm design, the reseed counter is executed every $2^{16} - 1$. When the LoRaWAN counter reaches the maximum value, both the ED and JS pairs reset the counter value, and the RBG regenerates Key and Nonce_Count. Thus, the block encrypted as the internal state has new unpredictable inputs.

The internal state executes the AES-128 bits algorithm with the relevant input parameters, and there are ten rounds with operations: Addroundkey, Subbytes, ShiftColumn, and MixColumn [53]. The result of the CTR_AES DRBG 128 algorithm is random bit sequences of the LoRaWAN New_Root_Key.

Furthermore, at the end of the proposed root key updating scheme, there is a LoRaWAN rekeying process conducted by ED and JS separately. The rekeying process is a follow-up of the updated root key value. New_Root_Key replaces the old root key; then, rekeying is carried out. The rekeying process includes the session keys KDF, cryptographic key derivation processes such as JSIntKey and JSEncKey, and other MIC calculations. All the cryptographic key involving the New_Root_Key should be updated, and the previous root key and other cryptographic keys should be stored as the old value.

IV. VALIDATION TEST: RESULT AND DISCUSSION

This section presents the validation of the proposed scheme by conducting the randomness test and communication protocol test. We also analyze the proposed scheme based on the aspects of replay attack resistance, integrity protection, perfect forward secrecy, lightweight key updating scheme, dynamic-remote-and-simultaneous root key updates, and implementation compatibility.

A. RANDOMNESS TEST

Randomness is an essential property in a cryptographic key update since it affects LoRaWAN security [54]. To verify the randomness, we simulate the proposed scheme by running the CTR_AES DRBG 128 algorithm using OpenSSL version 1.1.1. The two pseudorandom bit sequences that are used can also be obtained from the online RBG module [55], which has passed the NIST randomness test and passed the industrial standard test series [56], [57].

The simulation is executed on a computer server with the Ubuntu 18.04.5 LTS operating system, Processor Intel(R) Core(TM) i7-4790 CPU @1.80 GHz, 992 Mhz, and 16.00 GB RAM. The input values are stored in the local server database and recalled every time the CTR_AES DRBG 128 algorithm is executed. The reseed function to replace entire input parameters is executed when the counter value has reached 65,535.

The CTR_AES DRBG 128 algorithm is executed with $500 \times 65,536$ iterations. Each iteration produces 128 bits of ciphertext to ensure that the overall iteration is equivalent to 4.194 Gbits of data representing the New_Root_Key value. Many iterations are executed because we need to meet the criteria of the number of bits to measure the bit sequence randomness. NIST specifies that the minimum number of bits
tested is 1000 sequences, where each sequence consists of 1 million bits [58].

After completing the CTR_AES DRBG 128 algorithm execution, we convert the New_Root_Key value into a binary file. Then, the binary file is validated based on 15 parameters in the NIST statistical test suite [58] using Fast_NIST_STS_v6.0.1 tools [59], [60]. The tool is an optimized version of the NIST statistical test suite [61], [62].

The results with $\alpha = 0.01$ are presented in Table 2. Table 2 shows that the New_Root_Key value generated by the CTR_AES DRBG 128 algorithm passed the randomness test.

| No. Parameter | Parameter of the NIST Statistical Test Suite | The Proportion of Successful Sequences $\geq 0.01$ | Result |
|---------------|---------------------------------------------|-----------------------------------------------|--------|
| 1             | Frequency                                   | 0.9928                                        | Success |
| 2             | BlockFrequency                              | 0.9902                                        | Success |
| 3             | CumulativeSums (1)                          | 0.9924                                        | Success |
| 4             | CumulativeSums (2)                          | 0.9924                                        | Success |
| 5             | Runs                                        | 0.9926                                        | Success |
| 6             | LongestRun                                  | 0.9924                                        | Success |
| 7             | Rank                                        | 0.9881                                        | Success |
| 8             | NonOverlapping-Template (Min)               | 0.9855                                        | Success |
| 9             | NonOverlapping-Template (Max)               | 0.9931                                        | Success |
| 10            | OverlappingTemplate                         | 0.9876                                        | Success |
| 11            | Universal                                   | 0.9876                                        | Success |
| 12            | ApproximateEntropy                          | 0.9890                                        | Success |
| 13            | RandomExcursions (Min)                      | 0.9861                                        | Success |
| 14            | RandomExcursions (Max)                      | 0.9928                                        | Success |
| 15            | RandomExcursionsVariant (Min)               | 0.9864                                        | Success |
|               | RandomExcursionsVariant (Max)               | 0.9936                                        | Success |
| 16            | Serial (1)                                  | 0.9924                                        | Success |
| 17            | Serial (2)                                  | 0.9914                                        | Success |
| 18            | LinealComplexity                            | 0.9862                                        | Success |

**TABLE 2.** The 15 parameters of the NIST statistical test suite based on the obtained New_Root_Key bit sequence.

**B. COMMUNICATION PROTOCOL TEST**

A formal security analysis can be used to confirm that the proposed communication protocols support the root key updating scheme. The communication protocol pairs, New_Join-request and New_Join-accept and New_Rejoin-request and New_Rejoin-accept, are validated using Scyther tools [63]–[65]. Scyther is a validation tool that can be used to test the security of protocols and communication networks under enemy control [66], [67]. Modeling with the Scyther tool allows us to prove that the attacker is unable to interfere with or learn any secrets of the protocols [68].

The source code for each pair New_Join-request and New_Rejoin-request is executed with the Scyther tool over a computer server with an Ubuntu 18.04.5 LTS operating system and an Intel(R) Core(TM) i7-4790 CPU @ 1.80 GHz and 1992 MHz with 16.00 GB RAM. Fig. 9 shows the validation result for each pair of the New_Join-request and the New_Rejoin-request protocols.

Scyther proved that the communication protocols exchanged between ED and JS are secure and pass other property tests. In our scheme, “Alive” represents recent aliveness properties because we use time-based parameters. The noninjective agreement (Niagree) parameter shows that both ED and JS are convinced that they agree to exchange their information. Then, the noninjective synchronization (Nisynch) parameter indicates that the message received by the pairing entity is exactly the same as the transmitted message, both sequence and content. The validation result also ensures secrecy during the communication process with no attacker being present. The new root key transfer also proves to be secure, and the entities can maintain confidentiality during the key update process.

Furthermore, the Scyther validation results suggest that the characteristics of our protocols have exactly one pattern during the exchange communication from ED to JS, or vice versa. The only one-race pattern occurs when ED sends a New_Rejoin-request message to JS, which means that JS directly receives a message without it being intercepted by the attacker. At the same time, when JS sends a reply message, ED also directly receives the message without an interception that may occur during transmission.

In the case that a forgery attack occurs, the Scyther validation tool proves that the proposed scheme can resist the attack. Scyther models the forgery attack resistance by assuming that the threat model applied is the Dolev-Yao intruder. The threat model assumes that the attacker has full control against the
communication network except accessing the entities’ secret keys. Thus, since ED and legitimate JS share the same secret key, the forgery attack can be resisted. Then, if a fake JS sends a forgery New Join-accept or New Rejoin-accept message with a fake root key, ED cannot decrypt the received message. ED cannot calculate the MIC since the MIC is encrypted within the message. Therefore, in case a forgery attack may occur, the fake JS must obtain the secret key with efforts equal to brute force attack in $2^{128}$ possible keys.

However, ED implements a verification procedure to determine the validity of a received message. Once ED receives the New Join-accept or New Rejoin-accept message, ED decrypts and verifies the message by calculating MIC’. Then, ED compares MIC’ with MIC. If MIC’ = MIC, then ED executes the LoRaWAN rekeying process. Otherwise, ED decides the message is invalid.

C. REPLAY ATTACK RESISTANCE

The replay attack that may occur during transmission of New Join-request and New Rejoin-request is resisted based on the timestamp $Ts$, which is attached as a message component. Any ED that supports the proposed scheme will retrieve $Ts$ based on the scheduled key update period and embed the $Ts$ value to the message sent to the JS. Furthermore, $Ts$ is part of the input parameter for the MIC$_{EJr}$ and MIC$_{EJJ}$ calculations, which is the MIC and a part of the message sent by ED.

Upon receiving the message from ED, JS retrieves its timestamp $Ts^\prime$ and calculates the difference from the received $Ts$, where $\Delta Ts = T^s - Ts$. When $\Delta Ts$ does not match the updated schedule in the JS database, the root key update is not processed. In a situation where a replay attack occurs, an attacker will duplicate the New Join-request or New Rejoin-request message. However, when the message is received, the JS determines the value of the $Ts^\prime - Ts > Ts$, so the root key update is not processed. At the same time, when an attacker intends to change the timestamp value so that $Ts^\prime - Ts \leq \Delta Ts$, then the MIC$_{EJr}$ or MIC$_{EJJ}$ value of the received message will be invalid because the MIC calculation requires a key known only by the pair of associated ED and its JS. Thus, the proposed scheme can be declared resistant to replay attacks.

D. INTEGRITY PROTECTION

The integrity protection of the proposed scheme can be accomplished by the MIC protection to every message exchanged between the associated ED and JS. Messages sent by ED to initiate root key update include New Join-request equipped with the MIC$_{EJr}$ and New Rejoin-request equipped with the MIC$_{EJJ}$. Meanwhile, the reply message is from the JS to the ED, New Join-accept is equipped with the MIC$_{EJr}$, and New Rejoin-accept is equipped with the MIC$_{EJJ}$. Every time ED calculates the MIC message to initialize the root key update process, ED uses a different root key and different message components, depending on the message type. The root key used by ED for MIC calculation is the newest key value, where only the pair of ED and its JS know the key.

E. PERFECT FORWARD SECRECY

The perfect forward secrecy of the proposed scheme is related to the root key value. All ED – JS related keys are updated by the FIPS standard RBG system within a specified period, which each ED – JS pair always has different key values. Every time a root key update is processed, ED receives the new update. The JS freshly generates the new root key. Thus, when the root key located at ED is compromised, it occurs only in one period of time. An attacker cannot retrieve the previous root key or predict the future root key. The attacker cannot retrieve the root key parameter since it is stored and processed in JS.

F. LIGHTWEIGHT KEY UPDATING SCHEME

In the IoT, devices that require light processing are EDs; thus, we analyze the number of processes as the computation load that occurs during root key updates from an ED’s point of view. This paper calculates the computation load by adding up the KDF processed in ED and the number of communication processes or protocol steps involving ED. The variables $M$ and $N$ represent both processes. The comparison of the computation load covered by ED during the root key update process is displayed in Table 3.

| Scheme   | Computation Process | ED’s Total Computation Load |
|----------|---------------------|----------------------------|
|          | KDF (M) Protocol Step (N) |                                |
| [31]     | 1M 4N               | 1M + 4N                     |
| [34]     | 1M 3N               | 1M + 3N                     |
| Proposed | 0M 2N               | 2N                           |

The proposed scheme provides a lightweight process compared to other schemes [31], [34]. The proposed scheme only needs a pair of protocol steps to update the root key, such as when ED sends New Rejoin-request to JS, the reply is sent by New Rejoin-accept. We ignore the initialization and post-update process in calculating the computational load since every scheme has a similar condition. In our proposed scheme, the initialization process is that ED retrieves its device scheduled update timestamp $Ts$, counter $DevNonce$ or $RJCount1$, and calculates MIC$_{EJr}$ or MIC$_{EJJ}$. The proposed scheme provides the process efficiency, yet also produces the same outcome; it is lightweight and secure, and the new root key has random values.

G. DYNAMIC, REMOTE, AND SIMULTANEOUS ROOT KEY UPDATES

Dynamic root key updates can be acquired because the proposed scheme applies a time-driven approach, namely, a timestamp. The LoRaWAN root key is updated automatically by its infrastructure based on a predetermined schedule. During commissioning, each device that supports the root key update should determine its update.
schedule. The ED automatically sends a New_Join-request or New_Rejoin-request message as an initial root key update at a scheduled time. The proposed scheme does not need any additional devices, such as the assisted mode [24], so that the infrastructure, along with the proposed scheme can be dynamically scalable.

The proposed scheme supports the remote update feature because the device placed at a remote distance can perform the auto-update after commissioning is done. Unlike another proposed root key update in [24], which requires user intervention during the update procedure, our proposed scheme can perform the remotely automatic update based on the scheduled time. Therefore, the proposed update is suitable for a vast area where ED is located far from JS, but still within the LoRaWAN network range.

A simultaneously updated feature is needed when undertaking a large-scale implementation that involves multiple end devices. When the update process is set simultaneously, the key updating scheme can still be carried out, considering that JS is a server that, by nature, can execute many requests at once. In addition, setting the process simultaneously can also be realized because no user intervention and additional devices are needed to run the proposed scheme, in contrast to [24].

H. IMPLEMENTATION COMPATIBILITY

Regarding the field implementation of the proposed scheme, engineers should consider the spreading factor (SF) of the LoRaWAN infrastructure. The SF should be noticed because it is closely related to the size of the data transmitted by the LoRaWAN network. In the LoRaWAN network, the data size is adjusted to the SF value. For example, in Indonesia, LoRaWANs work at a frequency of 923–925 MHz following the AS923 standard. It has been determined that the maximum downlink MAC payload size for SF2 is 19 bytes. Thus, the proposed root key updating scheme can be applied if the LoRaWAN data rate is set to SF3 or the downlink MAC payload size = 61 bytes. The maximum payload capacity is different for each region and has been specified in standardization [69].

V. CONCLUSION AND FUTURE WORK

A. CONCLUSION

In this research, a novel secure root key updating scheme for LoRaWANs is proposed to change the root key from static to dynamic by updating it periodically. We utilize CTR_AES DRBG 128 as an algorithm to update the root key value. Validation results show that the new root key passed tests based on all 15 considered parameters, which means it has a high degree of randomness. The proposed scheme consisted of the initialization process executed at the ED and the key update process executed at the JS. Pairs of communication protocols are exchanged between the ED and JS to support the execution process, namely, New_Join-request and New_Rejoin-request paired with New_Join-accept and New_Rejoin-accept. Formal analysis tools validate both pair protocols with the following result. The communication process can guarantee data confidentiality during transmission, and the attacker cannot control the protocols. The communication protocol also passes other formal analysis parameter tests, including the noninjective agreement and the noninjective synchronization tests.

The proposed scheme provides a mechanism to resist replay attacks and protect data integrity. The scheme can also accommodate perfect forward secrecy for dynamic root keys. Moreover, root key generation is a low-computational-burden process for the ED. The ED only needs to retrieve the relevant timestamp and counter value and then calculate the MIC message to run the protocols and receive the newly updated root key.

With the proposed scheme, updating the root key remotely is also possible, where the ED is placed in a distant area within LoRaWAN’s coverage area. Next, since the root key updating process is executed in the JS, a simultaneous update for many devices is achievable. Finally, by considering the analysis mentioned above, it can be concluded that the proposed scheme can improve the security of LoRaWAN key management, especially for root keys.

B. FUTURE WORK

Future work on the proposed scheme should address the issue regarding backward compatibility. Compatibility implementation with previous LoRaWAN versions, such as version 1.0, is essential because the earlier versions have differences in the backend infrastructure, root key type, and message format [27]. The LoRaWAN version 1.0 backend infrastructure does not have a JS, where the JS is an entity used to execute the key updating algorithm. Therefore, instead of adding JS, the execution of the root key updating task is transferred to the network server, as in the LoRaWAN version 1.0 infrastructure. Furthermore, the key format and the message format used in New_Join-request are adjusted to the format of the LoRaWAN version by adding a timestamp value to the trailer.

REFERENCES

[1] S. Fewkes. (Dec. 7, 2021). LoRaWAN® Formally Recognized as ITU International Standard for Low Power Wide Area Networking. Accessed: Dec. 9, 2021. Fremont, California. [Online]. Available: https://lora-alliance.org/lora-alliance-press-release/lorawan-formally-recognized-as-itu-international-standard-for-low-power-wide-area-networking/

[2] N. Hayati and M. Suryanegara, “The IoT LoRa system design for tracking and monitoring patient with mental disorder,” in Proc. IEEE Int. Conf. Commun., Netw. Satell. (Comnetsat), Semarang, Indonesia, Oct. 2017, pp. 135–139, doi: 10.1109/COMNETSAT.2017.8263587.

[3] A. T. Nugraha, N. Hayati, and M. Suryanegara, “The experimental trial of LoRa system for tracking and monitoring patient with mental disorder,” in Proc. Int. Conf. Signals Syst. (ICSigSys), Bali, Indonesia, May 2018, pp. 191–196, doi: 10.1109/ICSIYSYS.2018.8372663.

[4] A. T. Nugraha, R. Wibowo, M. Suryanegara, and N. Hayati, “An IoT-LoRa system for tracking a patient with a mental disorder: Correlation between battery capacity and speed of movement,” in Proc. 7th Int. Conf. Comput. Commun. Eng. (ICCCE), Kuala Lumpur, Malaysia, Sep. 2018, pp. 198–201, doi: 10.1109/ICCCE.2018.8593316.
Recommendation for Key Management Part 1: General

E. Barker, media@lora-alliance.org. (Feb. 2020). LoRaWAN Security Frequently Asked Questions. LoRa Alliance, Inc. Accessed: Apr. 25, 2021. [Online]. Available: https://lora-alliance.org/resource_hub/lorawan-specification-v1-0/

N. Sornin et al. (Feb. 2016). LoRaWAN Specification Version: V1.0.1. LoRa Alliance, Inc. Accessed: May 5, 2021. [Online]. Available: https://lora-alliance.org/resource_hub/lorawan-specification-v1-0-1/

D. Sornin, M. Luis, T. Kramp, and O. Hersent. (Jul. 2016). LoRaWAN Specification Version: V1.0.2. LoRa Alliance, Inc. Accessed: May 5, 2021. [Online]. Available: https://lora-alliance.org/resource_hub/lorawan-specification-v1-0-2/

N. Sornin et al. (Jul. 2018). LoRaWAN 1.0.3 Specification. LoRa Alliance, Inc. Accessed: May 5, 2021. [Online]. Available: https://lora-alliance.org/resource_hub/lorawan-specification-v1-0-3/

J. Han and J. Wang. “An enhanced key management scheme for LoRaWAN.” Cryptography, vol. 2, no. 4, p. 34, Nov. 2018. doi: 10.3390/cryptography2040003.

I. Butun, N. Pereira, and M. Giolund. “Security risk analysis of LoRaWAN and future directions,” Future Internet, vol. 11, no. 1, p. 3, Dec. 2018. doi: 10.3390/hi110403.

I. Butun, N. Pereira, and M. Giolund. “Analysis of LoRaWAN v1.1 security: Research paper,” in 4th ACM MobiHoc Workshop Experiences Design Implement. Smart Objects (SMARTOBJECTS), Los Angeles, CA, USA, 2018, pp. 1–6, doi:10.1145/3213299.3213304.

J. Xing, L. Hou, K. Zhang, and K. Zheng. “An improved secure key management scheme for LoRa system,” in Proc. IEEE 19th Int. Conf. Comm. Technol. (ICCT), Xi’an, China, Oct. 2019, pp. 296–301, doi:10.1109/ICCT46805.2019.8947215.

Francis Edward Heath. (2017). LPWA Technology Security Comparison. Accessed: Feb. 17, 2021. [Online]. Available: https://www.semantischolar.org/paper/LPWA-Technology-Security-Comparison-A-White-Paper-Heath/f5b1d1907ab3f64cee1500cd6f4ceef1500ebca3f9259c8906f

LoRa Alliance Technical Committee. (Oct. 11, 2017). LoRaWAN™ 1.1 Specification. Accessed: Sep. 25, 2019. LoRa Alliance. [Online]. Available: https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/

K.-L. Tsai, Y.-L. Huang, F.-Y. Liu, Y.-I. Yeh, and C.-H. Tsai. “AES-128 based secure low power communication for LoRaWAN IoT environments,” IEEE Access, vol. 6, pp. 45325–45334, 2018, doi:10.1109/ACCESS.2018.2852563.

Gemalto, Actility, and Semtech. LoRaWAN™ Security A White Paper. Accessed: Dec. 3, 2021. [Online]. Available: https://lora-alliance.org/sites/lora_security_whitepaper.pdf

The Things Industries. Activate Your Device. Accessed: Sep. 12, 2021. [Online]. Available: https://www.thethingsnetwork.org/docs/devices/uno/quick-start.html#your-device-expensive.

X. Chen, M. Lecuyer, and W. White. “A complete key management scheme for LoRaWAN V1.1.” Sensors, vol. 21, no. 9, p. 2962, Apr. 2021.

K.-L. Tsai, F.-Y. Liu, Y. Jou, S.-W. Chang, S.-J. Hu, and H. Park. “Low-power AES data encryption architecture for LoRaWAN,” IEEE Access, vol. 6, pp. 146348–146357, 2019, doi:10.1109/ACCESS.2019.2941972.

N. Ravi and S. M. Shalinie, “Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture.” IEEE Internet Things J., vol. 7, no. 4, pp. 3559–3570, Apr. 2020, doi:10.1109/JIOT.2020.2937316.

J. Galeano-Braiones, J. Carmona-Murillo, J. F. Valenzuela-Valdés, and F. Luna-Valero, “Detection and mitigation of DoS and DDoS attacks in IoT-based scalable SDN: An experimental approach,” Sensors, vol. 20, no. 3, p. 816, Feb. 2020, doi:10.3390/s20030816.

M. Kolinsky, V. Kharchenko, and P. Iryna, “IoT server availability considering DDoS-attacks: Analysis of prevention methods and Markov model,” in Proc. 10th Int. Conf. Dependable Syst., Services Technol. (DESSERT), Leeds, U.K., Jun. 2019, pp. 51–56, doi:10.1109/DESSERT.2019.8770012.

E. B. Barker and C. R. Kelsey, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, Standard NIST SP 800-90A, National Institute of Standards and Technology, Jun. 2015. doi:10.6028/NIST.SP.800-90A1.

Security Requirements for Cryptographic Modules, Standard NIST FIPS 140-3, National Institute of Standards and Technology, Gaithersburg, MD, Apr. 2019. doi:10.6028/NIST.FIPS.140-3.
[47] M. Aljohani, I. Ahmad, M. Bashter, and M. O. Alassafi, “Performance analysis of cryptographic pseudorandom number generators,” IEEE Access, vol. 7, pp. 39794–39805, 2019, doi: 10.1109/ACCESS.2019.2907079.

[48] V. T. Hoang and Y. Shen, “Security analysis of NIST CTR-DRBG,” in Advances in Cryptology-CRYPTO, vol. 12170, D. Micciancio and T. Ristenpart, Eds. Cham, Switzerland: Springer, 2020, pp. 218–247, doi: 10.1007/978-3-030-56784-2_8.

[49] J. Woodage and D. Shumow, “An analysis of NIST SP 800-90A,” in Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn., 2019, pp. 151–180.

[50] S. Cohen, A. Kwong, S. Paz, D. Genkin, N. Heninger, E. Romen, and Y. Yarom, “Pseudorandom block swans: Cache attacks on CTR, DRBG,” in Proc. IEEE Symp. Secur. Privacy (SP), San Francisco, CA, USA, May 2020, pp. 1241–1258, doi: 10.1109/SP40000.2020.00046.

[51] C. Boyd, A. Mathuria, and D. Stebila, Protocols for Authentication and Key Establishment. Berlin, Germany: Springer, 2020, doi: 10.1007/978-3-662-58146-9.

[52] E. Barker, Recommendation for Key Management Part 1: General, Standard NIST SP 800-57p1r4, National Institute of Standards and Technology, Jan. 2016, doi: 10.6028/NIST.SP.800-57p1r4.

[53] Advanced Encryption Standard (AES). Standard FIPS 197, 2001.

[54] C. Kenny, N. Hayati, R. Suryanegara, and Y. Suryanto, “Potential development of AES 128-bit key generation for LoRaWAN security,” in Proc. 2nd Int. Conf. Commun. Eng. Technol. (ICCET), Nagoya, Japan, Apr. 2019, pp. 57–61, doi: 10.1109/ICCET.2019.8726884.

[55] M. Eldefrawy, I. Butun, N. Pereira, and M. Gidlund, “Formal security analysis of LoRaWAN,” in IEEE Access, vol. 8, pp. 76173–76184, 2020, doi:10.1109/ACCESS.2020.3002576.

[56] M. Sýs, Z. Říha, V. Matyáš, K. Márton, and A. Suciu, “On the interpretation of results from the NIST statistical test suite,” Math. Softw., vol. 43, no. 3, pp. 1–11, Jan. 2017, doi: 10.1145/2988228.705415.

[57] M. Sýs and Z. Z. Říha, “Faster randomness testing with the NIST statistical test suite,” in Security, Privacy, and Applied Cryptography Engineering, vol. 8804, R. S. Chakraborty, V. Matyas, and P. Schaumont, Eds. Cham, Switzerland: Springer, 2014, pp. 272–284, doi:10.1007/978-3-319-12060-7_18.

[58] M. Sýs, Z. Říha, and V. Matyáš, “Algorithm 970: Optimizing the NIST statistical test suite and the Berlekamp–Massey algorithm,” ACM Trans. Math. Softw., vol. 43, no. 3, pp. 1–11, Jan. 2017, doi:10.1145/2988228.705415.

[59] M. Sýs, Z. Říha, V. Matyáš, K. Marton, and A. Suciú, “On the interpretation of results from the NIST statistical test suite,” ROMJIST J., vol. 18, no. 1, pp. 18–23, 2015.

[60] C. J. F. Cremers, “The Scyther tool: Verification, falsification, and analysis of security protocols,” in Computer Aided Verification, vol. 5123, A. Gupta and S. Malik, Eds. Berlin, Germany: Springer, 2008, pp. 414–418, doi: 10.1007/978-3-540-70545-1_38.

[61] M. Eldefrawy, I. Butun, N. Pereira, and M. Gidlund, “Formal security analysis of LoRaWAN,” Comput. Netw., vol. 148, pp. 328–339, Jan. 2019, doi: 10.1016/j.comnet.2018.11.017.

[62] K.-L. Tsai, F.-Y. Lee, L.-L. Hung, and C.-Y. Ko, “Secure session key generation method for LoRaWAN servers,” IEEE Access, vol. 8, pp. 54631–54640, 2020, doi: 10.1109/ACCESS.2020.2978100.

[63] C. Cremers and S. Mauw, Operational Semantics and Verification of Security Protocols. Berlin, Germany: Springer, 2012, doi: 10.1007/978-3-540-78636-8.

[64] F. A. Putra, K. Ramli, N. Hayati, and T. S. Gunawan, “PURA-SCIS protocol: A novel solution for cloud-based information sharing protection for sectoral organizations,” Symmetry, vol. 13, no. 12, p. 2347, Dec. 2021, doi:10.3390/sym13122347.

[65] C. J. F. Cremers, “Scyther: Semantics and verification of security protocols,” Technische Univ. Eindhoven, Eindhoven, The Netherlands, 2006. Accessed: Nov. 21, 2021, doi: 10.6018/R614943.

[66] LoRa Alliance Technical Committee Regional Parameters Workgroup. LoRa Alliance, Inc. (Jan. 2018). LoRaWAN™ 1.1 Regional Parameters. Revision: B, LoRa Alliance, Inc. Accessed: Aug. 21, 2021. [Online]. Available: https://iotas.ru/files/documents/LoraWAN%20Regional%20Parameters%20v1.1B.pdf

NUR HAYATI (Member, IEEE) received the bachelor’s degree in applied science from the Politeknik Elektronika Negeri Surabaya as a telecommunication engineering major, in 2010, and the master’s degree in computer engineering from the Universitas Indonesia, in 2015, where she is currently pursuing the Ph.D. degree. She is an Electrical Engineering Lecturer for undergraduate students with the Faculty of Engineering, Universitas Muhammadiyah Yogyakarta. Her research interests include embedded systems, computer networks, and the IoT security.

KALAMULLAH RAMLI (Member, IEEE) received the master’s degree in telecommunication engineering from the University of Wollongong, Wollongong, NSW, Australia, in 1997, the Ph.D. degree in computer networks from the Universiteit Duisburg–Essen (UDE), NRW, Germany, in 2000, and the Ph.D. degree, in 2003. He has been a Lecturer at the Universitas Indonesia (UI), since 1994, and a Professor of computer engineering, since 2009. The government of Indonesia has awarded him several competitive research grants and international collaborative grants with UDE and UI. He currently teaches advanced communication networks, embedded systems, object-oriented programming, and engineering and entrepreneurship. His research interests include embedded systems, information and data security, computers and communications, and biomedical engineering. He is a prolific author, having published over 123 journals/conference papers and having written four books.

SUSILA WINDARTA (Member, IEEE) received the degree in cryptography from the National Crypto Academy, Bogor, Indonesia, the bachelor’s degree in information systems from Gunadarma University, Indonesia, and the master’s degree in mathematics from the Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Indonesia, Depok, Indonesia. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia. Since 2013, he has been working as a Lecturer with the Department of Cyber-Security Engineering, National Cyber and Crypto Polytechnic, Indonesia. His research interests include cryptography and information security-related topics, especially cryptographic hash functions and security protocols.

MUHAMMAD SURYANEGERA (Senior Member, IEEE) received the bachelor’s degree in electrical engineering from the Universitas Indonesia, in 2003, the master’s degree from University College London, London, U.K., in 2004, and the Ph.D. degree from the Tokyo Institute of Technology, Japan, in 2011. He is currently an Associate Professor of telecommunications management with the Universitas Indonesia. He is also a Lecturer with the Graduate Program in Telecommunications Management, part of the Department of Electrical Engineering, Universitas Indonesia. He has published more than 70 academic articles as a main author or coauthor. He is the principal investigator in the research area of ICT policy and technology management, the IoT, 4G/5G, and wireless communication technology, concerning both technical research and regulatory aspects. He engaged in international regulatory activities, among others, as a Drafting Group Chairman of the Asia Pacific Telecommunity (APT) Preparatory Group for ITU’s World Radiocommunications Conference (WRC-15) and WRC-19.

* * *

VOLUME 10, 2022 18819