Blockchain-enabled balise data security for train control system

Ganesan Muniandi

Digital & Integrated Systems Technology Centre, ALSTOM Transport India Limited, Bengaluru, India

Correspondence
Ganesan Muniandi, 10th Floor, Block B, ITC GREEN CENTRE, Doddabanaswadi Main Road, Jeevanhalli, Bengaluru 560005, India.
Email: m.ganesan@alstomgroup.com

Abstract
The train control system is the railway system’s brain, ensuring the train protection and the railway network’s operating performance. This control system relies on the balise data received via near-field wireless communication to localize the train and update real-time operating conditions and constraints. The attackers can capture these data through their balise antenna and corrupt them through replay or false data injection attacks. This corrupted data can compromise train safety and jeopardize the entire train fleet operations. To overcome these attacks and to ensure the trustworthiness of balise and data integrity, this study proposes the permissioned blockchain-enabled two novel architectures for data transactions. Besides, each balise is equipped with a novel built-in blockchain cryptography algorithm to create its secret and public keys to provide the key integrity without needing third-party Certification Authority of conventional Public-Key Infrastructure based security systems. Each balise can create the unique signature using a ‘nonce’ signal sent by the train control system in the static architecture or using a self-generated ‘k-once’ secret key per train control system in dynamic architecture. Finally, the case studies are carried out to confirm the security sufficiency of the proposed architectures.

1 | INTRODUCTION

1.1 | Background

The railway is considered one of the most important modes of environment-friendly and safer mobility for long-distance interstate and intercity travels. The railway operators worldwide are deploying European Railway Traffic Management System (ERTMS) Level 2 or Level 1 [9] to ensure interoperability, safety, and reliability in train operations. The European Train Control System (ETCS) [10] is the brain and the control, command, and signalling component of the ERTMS. In ETCS Level 1, trackside to onboard ETCS communication is established only through the track-mounted Euro-balises or Euro loops. The balises are used just for the train’s precise localization on the railway track in ETCS Level 2. In this case, the trackside to ETCS communication is established by the Radio Block centre through Global System for Mobile Communications—Railway (GSM-R). The main difference between ETCS Level 3 [11] and Level 2 is that there is no need for lineside signals or other trackside train detection systems than Euro-balises.

In any of the ETCS, ranging from Level 1 to Level 3, Euro-balise [1, 22] is the primary component in which the train control system requires the train’s immediate localization. Euro-balise is the passive or active wireless near field communication device that can send its data to ETCS whenever the train passes over it, as shown in Figure 1. The passive balises do not need any external power from the trackside [22]. Balise Transmission Module (BTM) with a balise antenna powers them radiofrequency energy broadcast to power them as shown in Figure 1. Euro-balise data [12] is composed of one or several telegrams. A Euro-balise telegram contains one header and an identified and coherent set of Packets. The interlocking subsystem is the safety-critical subsystem responsible for preventing trains’ conflict movements and sets the railway route for the specific train. In this case, LEU powers the active balise. Irrespective of active or passive balises, the BTM of ETCS extracts the data by processing signals received by the onboard balise antenna.

The balise to ETCS transmission [16] can consist of critical data related to signalling data, control, geographical location, power collection, train target running, the route, permanent speed restrictions, temporary speed restrictions etc. The fixed obstructions such as buffer stops, movement authority, gradients, support to cable and radio in-fill functionality, linking data, other data, etc., are also included. Similarly, the ETCS to balise (Down-link) [16] may consist of the train’s makeup, changes to train status that might affect the maximum safe speed, track...
to train adhesion, the status of the traction control, other data etc. In summary, Euro balise is one of the subsystems intended for use in all of the levels of applications defined within the ERTMS/ETCS called Level 1, Level 2, Level 3, respectively [13, 14, 16].

To ensure the interoperable deployment of balises with any onboard equipment, ETCS defined the detailed specifications in the form of SUBSET-036 [16] on balise telegram. As the transmitted data from the balises can also be the safety-relevant transmission, the balise data should be secured [17]. However, the balise requirements [16] do not specify any explicit security mechanism in the communication link [5]. Due to this shortcoming, any compromised balise data can affect the entire train safety and the train fleet's operational efficiency. The background about the possible security threats and how the proposed novel balise architectures overcome these threats are detailed in Section 1.2.

1.2 Motivation and key contributions

ETCS SUBSET-041 defines the accuracy requirement of onboard distance measurement as “for every measured distance’s, the accuracy shall be better or equal to $\pm(5+5\%\ 's')$ meters” [15, 25]. If the train stops at a different location due to its location uncertainty induced by tampered balise data, it can lead to unnecessary travel time delays [2]. To do this, as shown in Figure 2, the attackers can capture the balise data through their antenna. Then, the attackers can replay or modify this data. Later, they can program this corrupted data into their attacker balise with an attacker balise programming tool. These attacker balises transmit the replayed or modified data to ETCS when placed within $\pm(5+5\%\ 's')$ distance from the actual balise as shown in Figure 2, and ETCS will not be able to identify the valid balise. This kind of malicious tampering of the balise transmission in the form of a telegram replaying or replay attack [2, 3] or other forms of attacks such as physical replacement [17] of balises or the tampering of balise data [4, 6, 7, 18], i.e. false data injection attack is possible. The vital trackside infrastructure data, such as permanent or temporary speed restriction of the track section or the gradient data, can be corrupted by the attacker [2]. Such data corruptions can lead to train derailment or collision [5]. They can even produce discomfort to the passengers (for example, injury due to falling from the seat or standing place) [8].

For the effective and efficient prevention of replay attacks, the balise data containing the nonce-based random signals were introduced [2, 4]. The authors [5] introduced the one-time
usable nonce code transmitted by the Train Control System (TCS) to the balise, and balise generates the variable data to improve security. The message authentication code with a timestamp and uniform crypto-key authentication process can also mitigate balise data attacks [5, 18]. The resilient control algorithm for train stop control applications was proposed in [6, 7, 8]. A TCS can share the security key with route LEUs and synchronize with all the LEUs before the train trip starts [5]. In such a case, both balise and TCS can use symmetric security keys [5, 19, 18]. The major drawback of all these solutions is that the integrity of symmetric security keys can be easily compromised as any entity can remember them.

The various cryptographically enabled balises were well studied in the literature [2, 4–8, 18, 19]. Overall, these balise keys are one-time programmable using a dedicated balise data programming tool. The major drawback of these one-time programmable balises is that the key is locked, preventing further changes. Besides, these balises are programmed with the externally created keys that can compromise the key, i.e. they do not have the built-in key generation module. Due to this probably compromised key, it is possible to corrupt the balise data or duplicate the balise, later leading to potential security attacks.

Compared to literature references, the motivational questions of this work are,

a. Is it possible to maintain the “secrecy of the balise keys,” i.e. key integrity?
b. Is it possible to generate a “unique signature” for each TCS, i.e. data integrity?
c. Is it possible to use a “unique security private key” for each TCS, i.e. identity/trustworthiness?

The Public Key Infrastructure (PKI) with security private/public key pairs [34] can be the answer to these questions (b) and (c). In ERTMS [35], Key Management Centre (KMC) performs key management, and PKI supports data signature creation. The KMC can also play a Certification Authority (CA) role in PKI [36]. It can generate valid certificates that will guarantee the ERTMS entities’ identity, for example, balise. The major drawback of the inclusion of balise PKI is it can introduce the additional complexity of key management in KMC by considering the tens of thousands of balises deployed across the country compared to few hundred trains. As there is another entity (KMC/PKI) than TCS and balise, the possible compromise of secret keys cannot be ruled out for question (a). Hence, the existing KMC and PKI approach will not solve the problem of balise data security completely. One way to overcome CA’s dependency and the question of (a) is to have self-generated keys within balise itself in addition.

All the concerns and questions linked with symmetric and asymmetric keys, PKI/CAs, i.e. (a), (b), and (c), can be addressed in an integrated way through the use of blockchain technology [20, 21]. Though the evolution of blockchain [20, 21], for similar applications [28, 29, 30, 31], especially in railway applications [26, 27] is well studied in the literature, still the adaptation into balise data security is the open problem. By considering the possible improvements in the balise data security and the concerns in KPI/CAs, this paper proposes a novel balise blockchain that incorporates the self-generation of balise keys and blockchain-based transactions between balise and TCS. Each balise is equipped with a built-in blockchain cryptography algorithm in this study. The significant advantage of this balise blockchain is the key integrity; the proposed solution does not need an external KMC like CAs to issue certificates [36]. Besides, the secret keys are not distributed. These equivalent certificates are self-generated by the balise themselves, and trust about the balise data has arrived from a network of validating train control systems in this study.

Depending on the access right limits, the blockchain can be broadly categorized as permissionless or public, permissioned or private, and hybrid types [32]. In the case of permissionless blockchain, any participant can read, write, and participate in transactions and the consensus without the need for special authentication or authorization. Examples of permissionless blockchains are Bitcoin and Ethereum [32]. The major drawback is that it is more prone to attacks such that the anonymous participants can have several identities to influence the consensus. In the case of permissioned blockchains, the authenticity is mandatory for any participant to gain access and to use the system resources; such examples include Multichain, Hyperledger Fabric, Parity, BigChainDB, InterPlanetary, Corda, and Quorum [32]. Permissioned blockchains have a dedicated hardware module to manage the identity, privacy, and confidentiality within the system [32]. The hybrid blockchain is a combination of the public and private blockchain. In the context of balise and TCS, the permissioned blockchain is appropriate [34] due to (a) the participation of the balise in the data transaction with TCS is based on invite-only, i.e. the registered balises only can participate in the proposed method. Thus, each balise will be identified using the unique pair of private and public keys [34] in the permissioned blockchain. (b) As permissioned blockchains need their nodes to validate transactions, the proposed work involves the TCS to participate in the validation of balise data.

Similarly, the built-in balise blockchain cryptography algorithm also must be compliant with the chosen permissioned blockchain. This study adopts one such cryptographic algorithm, i.e. Elliptic Curve Digital Signature Algorithm (ECDSA) [23], which is common in permissioned blockchains such as R3 Corda [39] and Multichain [37]. In Multichain [37], ECDSA ensures that only rightful owners can spend funds. It uses cryptographic elliptic curves (EC) to generate private/public key pairs [38]. The self-generated secure private key [24] is kept within balise and not disclosed to others. It is used only to sign the balise data digitally. A public key is an address of the balise blockchain [24] known to all those who want to access the balise data, i.e. TCS. Hence, the external system receives only the public key. Instead of sending the balise data alone, the proposed method also sends the blockchain-enabled balise data unique digital signature to TCS compared to the traditional way.

Now, the problem of frequency of blockchain cryptographic key generation arises, i.e. one-time or for every transaction. This question leads to two novel solution variants such as static and dynamic balise blockchain architectures. As the name suggests, in static architecture, the private security key is generated once
and permanently programmed as a ‘static private security key.’ Still, the unique signature is created based on a ‘nonce’ signal sent by the train control system. TCS generates a ‘nonce signal’ for each balise. The significant drawbacks of static architecture are: (a) each TCS needs to have ‘nonce’ algorithm (b) as ‘nonce’ signal is tele-powered, it needs additional hardware at TCS and processing power at balise (c) TCS has to remember transmitted ‘nonce’ signal every time for the appropriate extraction of received balise data.

The objective of dynamic balise blockchain is to overcome the disadvantages linked with static architecture. The dynamic balise blockchain is based on the balise self-generated one-time usable ‘unique key,’ i.e. ‘k-once’ (‘key once’) for each TCS to produce the unique balise digital signature every time. As there is no ‘nonce’ signal, the complexity at TCS is reduced relatively compared to static architecture. However, additional interfaces are required for the real-time transmission of keys to TCS. Overall, each balise can create a unique signature either using a ‘nonce’ signal sent by the train control system or using a self-generated ‘k-once’ secret key per train control system in the case of static and dynamic architectures respectively. One of the static and dynamic architectures can be chosen for the specific balise deployment by studying the pros and cons.

In summary, the study’s significant, novel contributions for balise data security in the context of the train control system are:

1. Novel built-in blockchain key generation module inside a balise using a permissionless blockchain cryptographic algorithm, i.e. ECDSA, to maintain the ‘secrecy of the keys.’ The proposed solution ensures the integrity of the ‘private security key’ as these keys are not distributed to TCS when compared to conventional ERTMS. Also, each balise does not rely on third-party CAs, for example, KMC in ERTMS.
2. Novel static balise blockchain architecture generates the “unique signature” for each TCS. The combination of self-generated ‘one-time’ balise secret private key and ‘nonce’ per transaction based unique balise signature ensures the balise data integrity and trustworthiness by avoiding replay and false data injection attacks.
3. Novel dynamic balise blockchain architecture which generates the “unique signature” for each TCS. The self-generated balise ‘secret private key’ per TCS per data transaction ensures the balise data integrity and trustworthiness by avoiding replay and false data injection attacks. As there is no ‘nonce’ signal, the software and hardware complexities at TCS and balise processing are reduced relatively.

1.3 Paper organization

The rest of the paper sections’ organization is as follows: Section 2 introduces the static balise blockchain architecture. The dynamic balise blockchain architecture is detailed in Section 3. Sections 4 presents the case studies of the proposed balise blockchain architectures. Furthermore, the MATLAB simulation results are provided. The security analysis for replay, false data injection, and fake balise attacks are detailed in Section 5. Finally, Section 6 concludes the summary of this study. Throughout this paper, the terms such as ETCS and TCS are used invariably to cover both specific European TCS and generic TCS.

2 | ARCHITECTURE OF STATIC BALISE BLOCKCHAIN

2.1 Description

As the name suggests, in static balise blockchain, the Security private Key (SK or $k_p$) is fixed and stored permanently internal to balise to maintain its secrecy. As the $k_p$ is only known to balise, the $k_p$ is much secured compared to conventional methods. Since $k_p$ is static or one-time generated, the public key (PK or $k_s$) of the balise and its programmed data is also constant. With the static stored data and keys, the digital signature of data will also become static. With the static signature, the security attacks are still possible in the form of replay attacks or fake balises. However, the secured keys rule out the false data injection attack. The dynamic signature using well known ‘nonce’ signal from the TCS [2, 4] over the static balise blockchain mitigates replay attacks or fake balises. In summary, static balise blockchain uses the fixed $k_s$, $k_p$ per balise, and for the signature, it adds the dynamic ‘nonce’ from the TCS to improve the overall data security.

As per the proposed architecture, each balise consists of an in-house key generation module that generates a pair ($k_p$, $k_s$). Balise signs the transmitted data to TCS using $k_p$. TCS uses the $k_p$ to verify the integrity of transmitted balise data. For this, TCS needs to know about the $k_p$ of each upcoming balise along its route. The balise sends the $k_p$ to Balise Data Programming Tool (BDPT) in the key distribution cycle, as shown in Figure 3. The BDPT acts as the $k_p$ distributor between balise and TCS through the traffic management system (TMS). The distribution of $k_p$ is managed either through the USB or other equivalent interfaces, as shown in Figure 3. The BDPT programs each balise once with its respective data.

TMS gets the $k_p$ of all the balises under its control, i.e. for the list of station/inter-station areas. TMS maintains the database of $k_p$. Based on the train operator’s request, TMS sends the list of $k_p$ of balises along the specific train route to TCS. TCS and TMS can use long-range communication to cover the hundreds/thousands of kilometres of the distance. After the reception of $k_p$, TCS stores them into the onboard ‘Balise Public key database.’

Thanks to static balise blockchain $k_p$ and $k_s$, the false data injection is directly ruled out as $k_p$ is kept as a secret within balise itself, as one needs to know $k_p$ to update the balise data. To overcome the replay attack and fake balises, as shown in Figure 3, the balise receives a ‘nonce’ modulated tele-powering signal from the TCS through the air gap [2, 4, 5]. It sends back the static data and dynamic signature through the near-field wireless communication link to the TCS. If the data is compromised, it will be rejected by TCS by verifying the dynamic signature.
Figure 3 shows all the steps involved in the static balise blockchain in circles. Similarly, Figure 4 shows the steps in a sequence diagram.

2.2 | Balise blockchain key distribution—offline

The balise static key management steps are performed offline as below:

Step-1: The built-in balise key generation module generates the static $k_s$ with 32 bytes or 256 bits using any “random number generator.” Balise stores encrypted $k_s$ in its internal storage. Simultaneously, the PK, $k_p$, with the size of 64 bytes or 512 bits is generated from $k_s$ using the Elliptic Curve Digital Signature Algorithm (ECDSA) by the same key generation module of balise. As similar to permissioned blockchain, for example, Multichain [38], ECDSA is currently used in this study. The expression for $k_p$ is as,

$$k_p = \text{ECDSA}(k_s)$$

(1)

The significant advantage of using the pair of $(k_s, k_p)$ is that the reverse generation of $k_s$ from $k_p$ is not possible, i.e.

$$k_s \neq \text{ECDSA}(k_p)$$

(2)

These two keys are mathematically related to each other so that the data that is digitally signed using $k_s$ can be verified using the corresponding $k_p$.

Step-2: Each balise sends its $k_p$ to Balise Data Programming Tool (BDPT). BDPT stores the $k_p$ into a “Balise public key database” (unique balise id, $k_p$) pair. At the same time, the respective balise data is sent from BDPT to balise for the programming. Later, BDPT interfaces with each balise to collect the static $k_p$ while programming the static data.

Step-3: The entire database of the ‘Balise Public Key Database’ or the $k_p$ of specific balises can be shared with TMS using USB drives.

Step-4: Based on the request from the train operator or the TCS, TMS sends the list of pairs of (balise ID, PK) through long-range communication links for a specific route of the train journey.

2.3 | Balise data security—real-time

Step-5: During the real-time operation, the TCS generates a “nonce” field (32-bit, i.e. 4-byte) using its “random number generation” function, with the anticipation of each balise. This nonce field is valid for a single transaction between TCS and balise. The nonce can be sent to balise in the form of the modulated tele-powering signal, for example.
Step-6: On the reception of the ‘nonce’ signal, the balise extracts the ‘nonce’ data. A typical balise dynamic data $D_d$ is generated by balise for balise stored data $D_s$ as,

$$D_d = (D_s + \text{nonce})$$  \hspace{1cm} (3)

The secured $HASH(D_d)$ is created by balise by calculating the double hash value, for example, SHA-256 (Secure Hash Algorithm-256) of $D_d$. SHA-256 generates an almost-unique 256-bit (32-byte) signature for the security of the data. The secured hash is expressed as,

$$\text{balise } HASH (D_d) = \text{SHA}_256 (\text{SHA}_256 (D_d))$$  \hspace{1cm} (4)

Then, the hash of the data is signed using $k_s$ and is called Dynamic or Digital Signature.

$$\text{balise } SIGN (D_d) = \text{ECDSA}_\text{Sign}(\text{balise } HASH (D_d), k_s)$$  \hspace{1cm} (5)

This $\text{balise } SIGN (D_d)$ is sent along with a $D_s$ by each balise to TCS through the uplink transmission shown in Figure 1. Even though the same $D_s$ will be sent; however, the $\text{balise } SIGN (D_d)$ will be different each time due to the “nonce” code. Even the attacker can capture this data and signature, as shown in Figure 2, for a possible replay attack; thanks to “nonce,” the TCS signature verification will fail since the ‘nonce’ sent by this train is different from the one in the signature. The confidential $k_s$ of balise rules out any modification of the balise signature, preventing the ‘false data injection attack.’

Step-7: The TCS verifies the “received balise data and its dynamic signature.” At first, it recreates the hash of the final dynamic data as,

$$TCS \quad HASH (D_d) = \text{SHA}_256 (\text{SHA}_256 (\text{received } D_s + \text{nonce}))$$  \hspace{1cm} (6)

Later, it decrypts the signature using $k_p$ of the balise,

$$TCS \quad \text{DEC}_\text{HASH} (D_d) = \text{Decryption} (\text{balise } SIGN (D_d), k_p)$$  \hspace{1cm} (7)

If the data’s calculated hash $TCS \quad HASH (D_d)$ and the decrypted hash of data $TCS \quad \text{DEC}_\text{HASH} (D_d)$ are the same, the data is verified as OK by the TCS. Else, the received balise data is marked as invalid in security attacks and reported back to the TMS for recording purposes.

3 | ARCHITECTURE OF DYNAMIC BALISE BLOCKCHAIN

3.1 | Description

Like static blockchain architecture, in the dynamic balise blockchain architecture also, the $k_s/k_p$ generation module is kept inside the balise. The significant differences in key distribution are (a) balise transmits the $k_p$ to Lineside Electronic Unit (LEU) as shown in Figures 5 and 6. (b) The received $k_p$ is forwarded to the Interlocking (IXL) subsystem and is stored
FIGURE 5 Components involved in dynamic balise blockchain

into a balise public key database at IXL. (c) IXL manages the distribution of keys to the traffic management system via wireless communication or other equivalent interfaces. Thanks to LEU, it is possible to program the balise with static or dynamic data in real-time. Unlike the static balise blockchain architecture, this dynamic blockchain doesn’t need a ‘nonce’ modulated tele-powering signal from the TCS; instead, it uses a one-time usable ‘k-once’ $k_p$ for a specific train or TCS. TMS gets the ‘k-once’ $k_p$ from IXL for all the balises associated with the train route by providing a station/inter-station IDs list.

The key distribution from TMS to train is the same as static balise blockchain architecture: (a) TMS maintains the database
of $k_p$. (b) Based on the request from the train operator, TMS sends the list of public keys of the list of balises along with the train route. (c) TMS and train can use long-range communication such as GSM-R, 4G, 5G to send the $k_p$ of all the balises along the train route. (d) After the reception of $k_p$, ETCS stores the decrypted $k_p$ into the onboard ‘Balise Public key database.’

The following databases or other equivalent are used as shown in Figures 5 and 6.

- “Balise database” at IXL: stores the list of (unique balise ids).
- “Balise Public Key database” at IXL/TMS: stores the list of pair of (unique balise ids, $k_p$) for all the routes
- “Balise Public Key database (Route)” at TCS: stores the list of pairs of (unique balise ids, $k_p$) along the specific train route.

3.2 | Balise blockchain key distribution—real-time

The steps involved in the balise real-time key management are shown in Figure 6 in circles. Similarly, Figure 7 shows the steps in the sequence diagram.

Step-1: The train operator or the TCS can request the TMS to send the information related to balises along a specific train route. Then, TMS connects with the list of IXLs along the route. The long-range communication link is used for this exchange.

Step-2: Each IXL forwards the request to the respective one or multiple LEUs attached to it. LEU places the “Request new key for a specific TCS” to balise.

Step-3: Balise Blockchain uses the same key generation module as static blockchain architecture to generate the pair of $(k_p, k_s)$ using the ECDSA algorithm. The encrypted $k_s$ is stored at balise in its internal storage along with the train number.

Step-4: Each balise sends its $k_p$ to LEU. LEU forwards it to IXL, which will store the keys from all the associated LEUs into the “Balise public key database” as (unique balise id, $k_p$) pair.

Step-5: The entire database of the ‘Balise Public Key Database’ or the $k_p$ of specific balises can be shared with TMS from IXL through wireless communication.

Step-6: Based on the request from the train operator or the TCS, TMS sends the list of encrypted pairs of (balise ID, $k_p$) through long-range communication links for a specific route of the train journey.

3.3 | Balise data security—real-time

Step-7: TMS informs IXL about the upcoming train with ‘train Id.’ IXL forwards the same to respective LEUs. LEUs deliver the same to balise.

Step-8: As there is no “nonce” data required in this step, the balise directly generates the specific TCS data in advance. Balise has the already stored lookup table for (train Id, $k_s$). Based on the actual ‘train Id’ from LEU, the balise prepares the secured data in advance before the train arrival. The secured signature is created by balise by calculating the SHA-256 (Secure Hash Algorithm-256) of “balise stored data” similar to static balise blockchain. Balise sends this “signature” along with a “balise stored data” to TCS through the air gap. Even though the same “balise stored data” may be sent every time (unless there is a specific change in data from LEU to balise) still, the “Secured Balise Signature” will be different due to the unique “k-once $k_p$” for a specific TCS.

Step-9: The TCS verifies the “balise stored data $D_s$” in a two-step process as similar to static balise blockchain expect that ‘k-once’ $k_p$ is unique for each balise data:
the first step is to regenerate the hash of the received balise data $D_s$ using

$$TCS\ HASH\ (D_s) = SHA\_256(\ SHA\_256\ (received\ D_s))$$ \ (8)

the second step is to create the decrypted hash by decrypting the signature using $k_p$ of the balise (for signature with $k_s$).

$$TCS\ DEC\_HASH\ (D_s) = ECDSA\_Decrypt(\ balise\ SIGN\ (D_s), k_{one}, k_p)$$ \ (9)

If the data’s calculated hash $TCS\ HASH\ (D_s)$ and the decrypted hash of data $TCS\ DEC\_HASH\ (D_s)$ are the same, the data is verified as OK by the TCS.

4.1 Static balise blockchain

This case study’s objective is to demonstrate numerically the steps involved in static blockchain described in Figures 8 and 10 in MATLAB. This study makes the following assumptions:

a. Two trains, i.e. train-1 and train-2 with TCS-1 and TCS-2 respectively, run along the railway line.

b. A single balise, balise-1 is considered.

c. Balise-1 uses the same $k_{p,1}$ for both TCSs.

d. Each TCS generates the unique ‘nonce’ for each balise.

While producing a digital signature, balise first receives the ‘nonce’ from the TCS and extracts the ‘stored data.’ Then, it applies a double SHA-256 hashing algorithm to get a unique string of numbers. These strings of numbers are then digitally signed using the ECDSA algorithm using balise’s $k_{s,1}$. The TCS receives the digital signature and the balise data using the balise antenna. It verifies the signature using the corresponding balise’s $k_{p,1}$ and checks the received data retains its integrity.

The calculations involved in TCS-1 with balise-1 is elaborated in Figures 8a and Figure 10a. In Figure 8b, the commonalities between TCS-1 and TCS-2 with balise-1 are provided in ‘blue’ boxes, and differences are highlighted in ‘green’ boxes.

In the end, the ECDSA verification is performed by TCS-1 using $k_{p,1}$ as,

$$TCS\_1\ VERIFY\ (D_s) = ECDSA\_Verify(\ SHA\_256(\ SHA\_256((D_s + TCS\_1\ Nonce), balise\ SIGN\ (D_s), k_{p,1})) = true$$ \ (10)

to confirm the blockchain transaction as the valid one. Once the signature verification is successful, the ‘TCS-1’ confirms the balise identification. Then, TCS extracts the received balise data zone id = ‘zone_01’. It further processes the data for the track id and balise location for the ‘up’ direction.

Similarly, the calculations involved in TCS-2 with balise-1 is elaborated in Figures 8b and 10b.
Then, the ECDSA verification is performed by TCS-2 using \( k_{A1} \) as,

\[
\text{TC\_2 \text{ \text{\textsc{verif}}} (D_2)} = \text{ECDSA\_\text{\textsc{verif}}} (\text{SHA\textunderscore 256} (\text{SHA\textunderscore 256} (D_2 + \text{TC\_2 \text{ \textsc{ nonce}}}), \text{balise SIGN} (D_2), k_{A1}) = \text{true} \quad (11)
\]

to confirm the blockchain transaction as the valid one.

### 4.2 Dynamic balise blockchain

This case study’s objective is to numerically demonstrate the steps involved in dynamic blockchain as detailed in Figures 9 and 11 in MATLAB. This study makes the following
assumptions:

a. Two trains, i.e. train-1 and train-2 with TCS-1 and TCS-2 respectively, travel along the railway line.
b. Trains come across a balise, balise-1.
c. Balise-1 generates and uses the unique ‘k-once’ private key for each TCS.
d. Each TCS does not send the ‘nonce.’

Like static architecture, balise first extracts the ‘balise stored data’ and applies a double SHA-256 hashing algorithm to get a unique string of numbers in this architecture. The significant difference is that these strings of numbers are then digitally signed using a unique ‘k-once’ for each TCS. When the TCS receives the digital signature using a balise antenna, it verifies the signature using ‘k-once’ that corresponds to the specific TCS. The calculations involved in TCS-1 with balise-1 are elaborated in Figures 9a and 11a. In Figure 9b, the commonalities between TCS-1 and TCS-2 with Balise-1 are provided in ‘blue’ boxes, and differences are highlighted in ‘green’ boxes. The grey box is not applicable in dynamic architecture compared to static architecture.

At the TCS-1, the proper reception of balise-1 data is verified. The double SHA-256 of the received balise-1 data is recreated, and the signature verification is performed as:

\[
\text{TCS}_1 \text{ VERIFY} \ (D_i) \\
= ECDXS_\text{A Verify}(SHA_{256}(SHA_{256}(D_i)), \\
\text{balise SIGN} (D_i), \ 'k\_once' \ k_{P1}) = true \tag{12}
\]

Once the signature verification is successful, the received balise data is split using the ‘|’ separator, and the zone id is extracted as ‘zone_01’. Then, the track id and balise location data are pulled out for the ‘up’ direction.

The calculations involved in TCS-2 with balise-1 are elaborated in Figures 9b and 11b. The TCS-2 verifies the proper balise-1 data reception as,

\[
\text{TCS}_2 \text{ VERIFY} \ (D_i) \\
= ECDXS_\text{A Verify}(SHA_{256}(SHA_{256}(D_i)), \\
\text{balise SIGN} (D_i), \ 'k\_once' \ k_{P2}) = true \tag{13}
\]

5 | SECURITY ANALYSIS

5.1 | Static balise blockchain

5.1.1 | Replay attack

If the data is invalid or can’t be verified with any \(k_p\), then the received data is rejected. Further, the analysis is done to check whether any malicious data attack was attempted. Though two trains receive the same balise data, due to different “nonce,” the transmitted signature will be unique, as shown in Figure 10. Thanks to this unique balise signature, every time due to “nonce,” the replay attack is overcome by the static balise blockchain inherently.

5.1.2 | False data injection attack

For the false data injection attack study, let the original balise data for temporary speed restriction is 25 kmph, in addition to its location data as,
Balise-1 data (without False injection) = ‘zone_01 | track_01 | 501.88 | up | 50’

Let the attacker injects the false balise data for a temporary speed restriction of 50 kmph,
Balise-1 data (with False injection) = ‘zone_01 | track_01 | 501.88 | up | 25’

This false data of temporary speed restriction is 50 kmph can lead to train derailment as only the maximum allowed speed is around 25 kmph. This attack not only affects the balise data security, but it can also cause a potential safety issue due to derailment or significant discomfort to sitting or standing passengers. With the conventional balise data without any cryptographic signature, the attacker can capture this data as shown in Figure 2, and the false data injection is possible.

However, in the case of the proposed architecture, even if the attacker captures two instances of the static balise blockchain data by sending their ‘nonce’ as, like Figure 8, it is still impossible to extract the "k-once" keys from the signature. Even if the attacker knows "k-once" yet the verification of balise data only can be performed. In case the balise data is captured and modified to inject false data, i.e. speed restriction of 50 kmph, its signature still will be invalid due to “nonce.” This way, the security attacks are overcome with the help of static balise blockchain architecture.

5.2 Dynamic balise blockchain

Once the signature verification is successful, TCS-2 confirms the balise identification. Though two TCSs receive the same balise data, the transmitted signature will be unique due to “unique keys,” i.e. ‘k-once’ keys, as shown in Figure 11. Thanks to the dynamic balise signature due to “unique k-once keys,” the attacker balise cannot execute replay and false data injection attacks as actual balises use the pair of “k_{pub}/k_{priv}” only once. Even if the attacker places the corrupted balises at other locations, TCS ignores them automatically as it will not have k_{priv} of these balises. This way, the security issues due to fake attacker balises are also eliminated.

6 CONCLUSION

This case study presents two architectures to enable the balise data security in the train control system application. A novel combination of built-in blockchain key generation module inside the balise and blockchain cryptography enabled balise architectures is studied to solve security attacks and fake balises. Instead of relying on the conventional, externally generated cryptographic keys, each balise generates the pair of security private and public keys within itself and transmits only the public keys externally on a request basis. The train sends the ‘nonce’ to balise in static architecture. The dynamic architecture introduces ‘k-once’ for each transaction between the balise and the train. The results of case studies suggest that it is possible to overcome security attacks and issues of fake balises. The simulation results confirm the proposed architectures’ effectiveness, ensuring the balise data security in the train control system. The future direction can explore further studies towards the possible blockchain extension with other key generation schemes.

ORCID
Ganesan Muniandi https://orcid.org/0000-0003-4681-0167

REFERENCES
1. Eurobalise. https://en.wikipedia.org/wiki/Eurobalise (2020) Accessed 29 Dec 2020
2. Hanshan J., et al.: Securing balise-based train control systems using cryptographic random fountains. In: 2017 IEEE Conference on Communications and Network Security (CNS). Las Vegas, NV, pp. 405–410 (2017)
3. Replay attack. https://en.wikipedia.org/wiki/Replay_attack (2020). Accessed 29 Dec 2020
4. Ronald L.: Security improvement on balise-based train control system. Int. J. Adv. Res. Sci. Eng. Technol. 6(6) 9829–9836 (2019)
5. Wu Y., et al.: Vulnerabilities, attacks, and countermeasures in balise-based train control systems. IEEE Trans. Intell. Transp. Syst. 18(4), 814–823 (2017)
6. Lim H.W., et al.: Data integrity threats and countermeasures in railway spot transmission systems. ACM Trans. Cyber-Phys. Syst. 4(1), 26 (2019).
7. Temple W.G., et al.: On train automatic stop control using balises: Attacks and a software-only countermeasure. In: 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC). Christchurch, pp. 274–283 (2017)
8. Wu Y., et al.: Position manipulation attacks to balise-based train automatic stop control. IEEE Trans. Veh. Technol. 67(6), 5287–5301 (2018)
9. European Rail Traffic Management System. https://en.wikipedia.org/wiki/European_Rail_Traffic_Management_System (2020). Accessed 29 Dec 2020
10. European Train Control System: https://en.wikipedia.org/wiki/European_Train_Control_System (2020). Accessed 29 Dec 2020
11. ERTMS - Levels and Modes. https://ec.europa.eu/transport/modes/rail/ertms/what-is-ertms/levels_and_modes_en (2020). Accessed 29 Dec 2020
12. Significant “terms and abbreviations” used in the ETCS (European Train Control System). https://www.etf-europe.org/wp-content/uploads/2019/03/ERTMS_Appendix-2_Glossary-EN.pdf (2020). Accessed 29 Dec 2020
13. Biagi M., et al.: Performability evaluation of the ERTMS/ETCS – Level 3. Transp. Res. Part C Emerging Technol. 82, 314–336 (2017)
14. ERTMS Level 3 Risks and Benefits to UK Railways. https://rri.co.uk/media/773946/cptr798___erms_level_3_risks_and_benefits_to_uk_railways.pdf (2020). Accessed 29 Dec 2020
15. Performance requirements for interoperability. https://www.cera.europa.eu/sites/default/files/filesystem/erms/erms/rtcs/tsi_annex_a___mandatory_specifications/set_of_specifications_3_ets_b3_r2_gsm-r_b1/index014___subset-041_v320.pdf (2020). Accessed 29 Dec 2020
16. FFIIS for Eurobalise. https://www.cera.europa.eu/sites/default/files/filesystem/erms/erms/rtcs/tsi_annex_a___mandatory_specifications/set_of_specifications_1_ets_b2_gsm-r_b1/index009___subset-036_v241.pdf (2020). Accessed 29 Dec 2020
17. Gabriel A., et al.: The determination of critical components of European Rail Traffic Management systems towards cyber-attacks. In: Boersma K., Tomaszewski B. (eds.) ISCRAM 2018 Conference Proceedings – 15th International Conference on Information Systems for Crisis Response and Management. pp. 291–303. Rochester Institute of Technology, Rochester, NY (2018)
18. Guo H., et al.: Protecting train balise telegram data integrity. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). Maui, HI, pp. 806–811 (2018)
19. Deichmann U., et al.: Checking the authenticity of a balise. https://patents.google.com/patent/DE102014212516A1/en (2020). Accessed 29 Dec 2020
20. Jesus E.F., et al.: A survey of how to use blockchain to secure Internet of Things and the stalker attack. Secur. Commun. Netw. 2018, 1 (2018). https://doi.org/10.1155/2018/9675050
21. Cheikhrouhou O., Koubâa A.: BlockLoc: Secure localization in the Internet of Things using blockchain. In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC). Tangier, Morocco, pp. 629–634 (2019). https://doi.org/10.1109/IWCMC.2019.8766440
22. Balise. https://en.wikipedia.org/wiki/Balise (2020). Accessed 29 Dec 2020
23. Elliptic-curve cryptography. https://en.wikipedia.org/wiki/Elliptic-curve_cryptography (2020). Accessed 29 Dec 2020
24. Blockchain. https://en.wikipedia.org/wiki/Blockchain (2020). Accessed 29 Dec 2020
25. Ganesan, M., Ezhilarasi, D.: Train distance and speed estimation using multi sensor data fusion. IET Radar Sonar Navig. 13(4), 664–671 (2019)
26. Ganesan, M.: Blockchain enabled virtual coupling of automatic train operation fitted mainline trains for railway traffic conflict control. IET Intell. Transp. Syst. 14(6), 611–619 (2020)
27. Blockchain: The Wheels Of Future Railway. http://www.businessworld.in/article/Blockchain-The-Wheels-of-Future-Railway/05-04-2020-168924/ (2020). Accessed 29 Dec 2020
28. Reyma, A., et al.: On blockchain and its integration with IoT: Challenges and opportunities. Future Gener. Comput. Syst. 88, 173–190 (2018)
29. Dai, H., Zheng, Z., Zhang, Y.: Blockchain for Internet of Things: A survey. IEEE Internet Things J. 6(5), 8076–8094 (2019)
30. Blockchain for the IoT: Opportunities and challenges. https://arxiv.org/abs/1805.02818 (2020). Accessed: 29 Dec 2020
31. Ali, M.S., et al: Applications of blockchains in the Internet of Things: A comprehensive survey. IEEE Commun. Surv. Tutorials 21(2), 1676–1717 (2019)
32. Ali Syed T., et al.: A comparative analysis of blockchain architecture and its applications: Problems and recommendations. IEEE Access 7, 176838–176869 (2019). https://doi.org/10.1109/ACCESS.2019.2957660
33. How Blockchain Addresses Public Key Infrastructure Shortcomings. https://www.infosecurity-magazine.com/opinions/blockchain-pki-shortcomings-1-1/#:~:text=Application%20design%20has%20changed%20dramatically%20since%20PKI%20emerged.&text=Since%20Blockchain%20runs%20on%20tens,read%20all%20of%20its%20contents (2020). Accessed 29 Dec 2020
34. Saxena S., et al.: Blockchain based transactive energy systems for voltage regulation in active distribution networks. IET Smart Grid 3, 646–656 (2019). https://doi.org/10.1049/iet-stg.2019.0286
35. The ERTMS/ETCS signalling system. http://www.railwaysignalling.eu/wp-content/uploads/2016/09/ERTMS_ETCS_signalling_system_revF.pdf (2020). Accessed 17 Sept 2020
36. Lopez L., Agudo M.: Cyber security study of the European Rail Traffic Management System. In: ERTMS Conference 2016A. Brussels (2016). 10.13140/RG.2.1.4832.8084
37. Multichain. https://www.multichain.com (2020). Accessed 17 Sept 2020
38. Address and key format. https://www.multichain.com/developers/address-key-format/ (2020). Accessed 17 Sept 2020
39. The Corda Platform: An Introduction. https://www.corda.net/content/corda-platform-whitepaper.pdf (2020). Accessed 17 Sept 2020

How to cite this article: Muniandi G. et al.: Blockchain-enabled balise data security for train control system. IET Blockchain. 2021;1–13. https://doi.org/10.1049/blc2.12003