TU Delft

Delft University of Technology

PRIDE

A Privacy-Preserving Decentralised Key Management System

Kester, David; Li, Tianyu; Erkin, Zekeriya

DOI
10.1109/WIFS55849.2022.9975379

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 2022 IEEE International Workshop on Information Forensics and Security (WIFS)

Citation (APA)
Kester, D., Li, T., & Erkin, Z. (2022). PRIDE: A Privacy-Preserving Decentralised Key Management System. In *Proceedings of the 2022 IEEE International Workshop on Information Forensics and Security (WIFS)* (pp. 1-6). IEEE. https://doi.org/10.1109/WIFS55849.2022.9975379

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

PRIDE: A Privacy-Preserving Decentralised Key Management System

David Kester
Cyber Security Group
Delft University of Technology
Delft, The Netherlands
d.m.kester@student.tudelft.nl

Tianyu Li
Cyber Security Group
Delft University of Technology
Delft, The Netherlands
tianyu.li@tudelft.nl

Zekeriya Erkin
Cyber Security Group
Delft University of Technology
Delft, The Netherlands
z.erkin@tudelft.nl

2022 IEEE International Workshop on Information Forensics and Security (WIFS) | 979-8-3503-0967-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/WIFS55849.2022.9975379

***Abstract*—There is an increase in interest and necessity for an interoperable and efficient railway network across Europe, creating a key distribution problem between train and trackside entities' key management centres (KMC). Train and trackside entities establish a secure session using symmetric keys (KMAC) loaded beforehand by their respective KMC using procedures that are not scalable and prone to operational mistakes. A single system would simplify the KMAC distribution between KMCs; nevertheless, it is difficult to place the responsibility for such a system for the whole European area within one central organization. A single system could also expose relationships between KMCs, revealing information, such as plans to use an alternative route or serve a new region, jeopardizing competitive advantage. This paper proposes a scalable and decentralised key management system that allows KMC to share cryptographic keys using transactions while keeping relationships anonymous. Using non-interactive proofs of knowledge and assigning each entity a private and public key, private key owners can issue valid transactions while all system actors can validate them. Our performance analysis shows that the proposed system is scalable when a proof of concept is implemented with settings close to the expected railway landscape in 2030.**

***Index Terms*—blockchain, key management, privacy-preserving, proofs of knowledge, ertms**

I. Introduction

Railway signalling systems are designed to control railway traffic alongside preventing collisions and unsafe situations. The European signalling and speed control system, ERTMS, replaces different national legacy systems to increase the interoperability and capacity of the European rail network. Trains and trackside equipment are assets maintained by different Key Management Centers (KMC) across Europe and it is estimated that there are 100 KMCs operational in Europe. Moreover, it is expected that by 2030, 38 500 trains will use the European signalling system [1]. ERTMS is based on a wireless link between train and trackside entities known as Radio Block Centre (RBC) in charge of a geographic area or a specific rail line. An RBC provides speed boundaries and track information, and the train sends periodic status updates, such as position. Before a train and an RBC can establish a session, mutual identification and authentication takes place using a preloaded triple-DES key known as KMAC. This key needs to be shared beforehand between the respective KMCs and is unique per train and RBC combination.

KMAC distribution between KMCs is done using an *offline* method, where the key is exchanged using storage media such as compact discs and flash drives [2]. The procedure is known to be not scalable, prone to operational mistakes and the actual implementation differs per KMC as the scheme either lacks definition on key distribution or is simply not suitable for use in certain cases [3]. An *online* method, specified in 2015 to overcome the shortcomings of the offline method, prescribes TLS-PKI for communication between KMC [4] but is not used in practice. In personal interviews, KMC administrators explained that KMAC distribution is a costly and long process, consisting of thousands of euros per KMAC and taking days or even weeks, resulting in that KMACs are generated without an expiration date and are hardly refreshed once installed in trains and RBCs.

A single and scalable key management system would make key distribution between KMCs easier and international operations more efficient. Nevertheless, a study from stakeholders brought to light that a centralised approach is difficult to implement because it is difficult to place the responsibility of a PKI for the entire European railway area within a single organisation [3]. Furthermore, roughly half of all KMCs have partially or not implemented cybersecurity measures such as risk assessments, audits or legacy systems upgrades [5], therefore it is assumed that KMCs do not trust each other in terms of security. KMC administrators pointed out that (freight) railway operators try to load as many KMACs as possible in their trains to service a larger number of clients and locations. Being able to look up which RBC and train pairs share a KMAC could reveal commercial strategies, such as intentions for a new route. As a result, a key management system should take privacy into account. Although there exists previous work regarding ERTMS key management [6], [7], none of them addresses privacy and the work is either build around a centralised approach or focused on reducing management overhead exclusively inside a KMC domain.

A. Our contribution

In this work, we present, to the best of our knowledge, first privacy-preserving decentralised key management system tailored for railway KMCs. The system is built around a permissioned blockchain and allows KMCs to exchange

Authorized licensed use limited to: TU Delft Library. Downloaded on January 02,2023 at 09:37:19 UTC from IEEE Xplore. Restrictions apply.

symmetric keys for entity pairs in a confidential, integer and authentic way using transactions. After registration, a KMC can exchange KMACs with any other registered KMC for every possible train and RBC combination using $n+1$ private keys, where n is the number of assets under its domain.

Privacy is preserved by assigning each possible train and RBC pair an identifier *tag* which does not reveal the entities involved and can only be computed by both asset owners or managers. Anonymising users in a system opens the door to misbehaving activities such as denial of services attacks. To mitigate this risk, we enforce the use of authenticated but anonymous transactions. The tag is included in every transaction, and each KMC can verify it is from the set of possible pairs using non-interactive proofs of knowledge. The construction allows actors in a permissioned blockchain to verify the authenticity of each transaction without deducing the involved KMCs, trains, or RBCs involved. Because existing procedures require logging activities concerning key generation, distribution, and revocation, our proposal ensures with blockchain's append-only nature that all transaction data is appropriately recorded even though KMCs do not trust each other.

The rest of the paper is structured as follows. Section II summarises relevant concepts used in this work and Section III reviews related work. Section IV presents PRIDE, our proposed decentralised key management system, analysed afterwards in Section V. The performance is evaluated by implementing a proof-of-concept, the results are presented in Section VI. The paper ends with concluding remarks.

II. Preliminaries

A. ERTMS Key Distribution overview

The generalised key distribution procedure between two KMCs is summarised in Figure 1, where a Railway Operator KMC requests a KMAC for a train under its domain to an Infrastructure Manager in charge of a specific RBC. Upon acceptance, the Infrastructure Manager KMC generates a KMAC for the train-RBC pair and shares it with the Railway Operator KMC. Both KMC will load the KMAC into their respective assets, followed by an operation test, where the train establishes a session with the trackside entity using the corresponding KMAC for the first time [2].

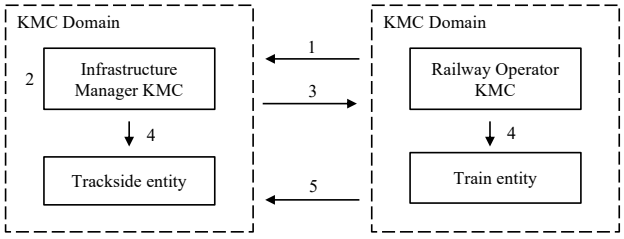

Fig. 1. Key distribution procedure between KMCs: 1) Railway Operator requests a KMAC for a train and an RBC combination 2) Upon acceptance, the Infrastructure Manager generates a KMAC 3) The KMAC is shared with the Railway Operator KMC 4) Both KMC load the KMAC into their assets 5) the train is able to establish a session with the RBC.

B. Blockchain

A blockchain is an append-only tamper-proof ledger that facilitated the development of decentralised applications, such as cryptocurrencies [8]. In a permissioned blockchain, users need to be explicitly admitted to the system. Such a registration process blocks Sybil attacks and allows the implementation of finite consensus protocols such as practical Byzantine Fault Tolerance (pBFT).

C. Elliptic Curve Cryptography

This work is based on Elliptic Curves (EC) where solving the Discrete Logarithm (DL) is thought to be intractable. A related problem is the Decision Diffie–Hellman problem.

Definition 1 (Discrete Logarithm Problem (DLP)): Given a generator G and and point P, it is computationally infeasible to find an $x \in \mathbb{Z}_q$, such that $P = xG$.

Definition 2 (Decision Diffie–Hellman problem (DDH)): Given a generator G, xG and yG, for $x, y \in \mathbb{Z}_q$, it is not possible to distinguish xyG from a random element in $\mathbb{Z}_q$.

A Digital Signature Scheme provides authenticity, integrity and non-repudiation of a message. The Elliptic Curve Digital Signature Algorithm (ECDSA) consists of a private and public key pair (x, Y), such that $Y = xG$ and $x \in_R \mathbb{Z}_q$.

D. Stealth Addresses

Stealth addresses are used to anonymise both transaction issuer and reciever [9]. Given a generator G and public keys A and B, an issuer computes $R = rG$ and $P = \mathcal{H}(rA)G + B$ where $r \in_R \mathbb{Z}_q$. The tuple (R, P) is embedded in the transaction and the receiver, who knows the private view key a and private spend key b such that $A = aG$ and $B = bG$, checks for every incomming transaction if $P = \mathcal{H}(aR)G + B$ holds. If true, the transaction is destinated to him/her and only he/she can compute the corresponding one-time private key $x = \mathcal{H}(rA) + b$.

E. Sigma protocols

Zero-Knowledge Proofs (ZKP) are used by a prover to convince a verifier that he/she knows information satisfying a given relationship without revealing anything else. Sigma protocols are used to create ZKPs which are made non-interactive using the Fiat-Shamir Transformation and a cryptographic secure hash function $\mathcal{H}(\cdot)$. Given a group of prime order q and generators G and H, Protocol 1 presents a Non-Interactive ZKP (NIZKP) to prove knowledge of $w \in \mathbb{Z}_q$ satisfying $Y = wG$ and $Z = wH$. The result is used to prove the equality of discrete logarithms and we say (G, Y, H, Z) is a Diffie-Hellman (DH) tuple if $H = xG$ for $x \in \mathbb{Z}_q$. Cramer et al. [7] presented a technique to combine two or more sigma protocols to create disjoint statements, allowing a prover to convince a verifier that he/she knows information satisfying t out of n relationships without revealing which ones. The result is a witness indistinguishable sigma protocol.

Authorized licensed use limited to: TU Delft Library. Downloaded on January 02,2023 at 09:37:19 UTC from IEEE Xplore. Restrictions apply.

Prover (w, G, H)		Verifier (G, H)
$v \in_R \mathbb{Z}_q$		
$R = vG$		$R' = rG + cY$
$S = vH$	$\xrightarrow{(Y,Z,c,r)}$	$S' = rG + cH$
$c = \mathcal{H}(R \parallel S)$		$c \stackrel{?}{=} \mathcal{H}(R' \parallel S')$
$r = v - cw \mod q$		

Protocol 1. A non-interactive[1] proof of knowledge to prove equality of a discrete logarithm.

III. Related work

Thomas et al. [6] addressed ERTMS key management by presenting an alternative key hierarchy where the advantage is mainly noticeable by Infrastructure Managers. Franeková et al. [11] addressed ERTMS key management using a centralised approach. These works do not consider privacy and are built arround a centralised approach. Therefore, decentralised key management systems proposed for applications that resemble the ERTMS structure, such as Internet-of-Things and vehicle-to-vehicle communication systems were studied. Even though these works resemble the ERTMS structure, they do not explicitly consider hiding relationships between system actors [12], [13]. Platforms that facilitate the creation of industry-grade decentralised applications, like Hyperledger Fabric and Corda, were also investigated. Fabric offers *private channels* and *private data collections* for sharing private information between organizations but at the price of administrative overhead and redundancy reduction. Corda does not offer standalone privacy features and a choice needs to be made between privacy and security [14]. Hyperledger Indy offers a decentralised key management system based on self-sovereign identities (SSI), allowing users to manage *how* they authenticate themselves [15]. These platforms do not meet our requirements *out-of-the-box*, for that reason we look at building a blockchain from scratch and investigating anonymisation techniques based on group signatures. A linkable spontaneously anonymous group (LSAG) signature scheme achieves anonymity, linkability, and spontaneity [16]. Spontaneity refers to the absence of a group secret, group manager or group secret sharing setup and the linkability property exposes two or more signatures made using the same private key.

IV. PRIDE

This section introduces PRIDE, our privacy-preserving key management system for the European signalling and speed control system based on a permissioned blockchain. The system addresses the distribution of KMAC between KMCs (step 3 in Figure 1). We assume that Registration Centers (RC) are established beforehand with the sole role of granting KMC access to the system. Registered KMC create a peer-to-peer network and register a set of assets. An asset, either a train or an RBC, can only belong to the domain of one KMC, its home KMC and KMCs use transactions to distribute KMACs for a particular train and RBC combination.

We present our design in six steps: 1) Initialization. 2) Data Encryption using AES block cypher with Galois/Counter Mode (AES-GCM). 3) Train and RBC pair identifier. 4) Traceable relationships. 5) Transaction creation and verification. 6) Block and blockchain creation and verification.

Step 2 allows KMACs to be transferred in a confidential, authentic and confidential way, as required by the ERTMS specifications [2]. Each train and RBC pair is assigned a *tag* in step 3, which can only be computed by the owners of the entities. The tag does not disclose the entities it identifies, creating an anonymous identifier which is included in each transaction. Step 4 explains how the tag links all transactions for a given train and RBC pair together. Finally, step 5 and 6 describes how all elements fit together to create the blockchain. The relevant notation is summarized in Table I.

TABLE I
Notation

Symbol	Description
G	Elliptic curve base point
$\mathcal{Y}$	Set of assets (train and/or RBC) public keys
$\mathcal{U}$	Set of train public keys, $\mathcal{U} \subset \mathcal{Y}$
$\mathcal{V}$	Set of RBC public keys, $\mathcal{V} \subset \mathcal{Y}$
x_a	Private key for asset a, $x_a \in \mathbb{Z}_q$
Y_a	Public key for asset a, $Y_a = x_a G$, $Y_a \in \mathcal{Y}$
K_{ab}	Tag for asset combination a and b
π	Index for asset combination a and b
k	Mixin count
u	Transaction one-time private key
$\lfloor \cdot \rfloor$, $\lceil \cdot \rceil$	Floor and ceil functions
$\lvert \cdot \rvert$	Set cardinality
$A \times B$	Cross product between set A and B

A. Initialization

RCs are established to grand KMC access to the system. More than one RC can be appointed to avoid centralization; for example, each Member State could designate a national RC. Each KMC has a unique identifier and generates a KMC private and public key pair. The RC signs the KMC public key creating a certificate as proof of admittance, and once registered, a KMC creates a domain by registering assets, i.e., trains or RBCs, into the system. For each asset in its domain, the KMC generates a private and public key. Similar to the previous process, the public key and asset identifier are signed by the KMC using the KMC private key, creating a certificate. Registered KMC establishes a peer-to-peer network of nodes where each node has a list of RC, registered KMC, trains and RBCs with their corresponding identifier, public key, and certificate. Each KMC has a $n + 1$ private keys: one KMC private key and a set of n private keys, one for each asset in its domain, where n is the number of assets in the domain. $\mathcal{Y}$ is the set of public keys of all registered trains and RBCs.

B. Hiding Confidential Information

KMACs are encrypted using AES-GCM consisting of four inputs: the plaintext message, a secret key, an Initialization

[1]The strong Fiat-Shamir Transformation requires $c = \mathcal{H}(R \parallel S \parallel Y \parallel K)$ [10] but was omitted in this case due to space constraints

Authorized licensed use limited to: TU Delft Library. Downloaded on January 02,2023 at 09:37:19 UTC from IEEE Xplore. Restrictions apply.

Vector (IV) and optional additional authentication data. The output is a ciphertext that ensures confidentiality, and a Message Authentication Code (MAC) providing integrity. The secret key used for encryption and decryption is obtained from an authenticated Diffie-Hellman key exchange using the KMC public and private keys. Symmetric encryption allows both KMCs to retrieve the KMAC from the same transaction.

C. Anonymous Relationships

The KMAC generated and encrypted in step 2 corresponds to a specific train and RBC combination, identified by a tag. Let the set of registered trains public keys be $\mathcal{U} \subset \mathcal{Y}$ and the set of registered RBCs public keys be $\mathcal{V} \subset \mathcal{Y}$, such that $\mathcal{U} \cap \mathcal{V} = \emptyset$ and $\mathcal{U} \cup \mathcal{V} = \mathcal{Y}$. The number of possible combinations $\{(Y_i, Y_j) \mid Y_i \in \mathcal{U} \wedge Y_j \in \mathcal{V}\}$ is $|\mathcal{V} \times \mathcal{U}|$. Each train and RBC combination is uniquely mapped to a tag K using a Diffie-Hellman key exchange (Equation 1). The tag resulting from this one-way function is used as an identifier rather than a secret and is included in each transaction having two important properties: 1) the tag can only be computed by the private key owners, and 2) the tag does not reveal the involved public and private keys.

$$K_{ij} = x_i Y_j = x_j Y_i = K_{ji} \tag{1}$$

Apart from being an identifier, a valid tag authenticates a transaction. A transaction is considered authentic if it can be proven to be issued by a registered KMC and for a combination of a registered train and RBC. The tag forms a DH tuple (G, Y_i, Y_j, K_{ij}) for which a NIZKP, as presented in Protocol 1, can be created. The result would convince all system actors that the tag is computed by a registered KMC using Equation 1 but comes at the cost of revealing the public keys involved, instantly revealing the relationship between the two KMC. To prove authenticity while remaining anonymous, Protocol 1 is extended to prove a tag is related to *one* out of all $|\mathcal{V} \times \mathcal{U}|$ possible tags, in other words, one of the DH tuples from the set $\{(G, Y_i, Y_j, K_{ij}) \mid Y_i \in \mathcal{U} \wedge Y_j \in \mathcal{V}\}$ satisfy Equation 1. The tag and proof generation for RBC a and train b is created using Algorithm 1 and RBC private key x_a. Any system actor can verify the authenticity of the tag by running Algorithm 2. The size of the proof is proportional to $|\mathcal{U} \times \mathcal{V}|$, which could become a problem for a large set of assets. A smaller result is obtained by introducing a mixin count $k < |\mathcal{U} \times \mathcal{V}|$. Let $\mathcal{S}$ be a set of k elements from a uniform random sample without replacement from $\{1, ..., |\mathcal{U} \times \mathcal{V}|\} \setminus \{\pi\}$, where π is the index of a particular RBC and train combination (Y_a, Y_b). The ordered multisets fed into Algorithm 1 and 2 are then

$$\mathcal{U}' = \{U_{\lfloor i \div |\mathcal{V}| \rfloor} \mid U_{\lfloor i \div |\mathcal{V}| \rfloor} \in \mathcal{U} \wedge i \in \mathcal{S}\} \cup \{Y_b\}$$

and

$$\mathcal{V}' = \{V_{i \mod |\mathcal{V}|} \mid V_{i \mod |\mathcal{V}|} \in \mathcal{V} \wedge i \in \mathcal{S}\} \cup \{Y_a\}.$$

The value of k acts as a measure of ambiguity: a small value results in a smaller proof, but increases the chance of guessing the underlying relationship from a single transaction. Setting $k = 0$ results in Protocol 1, instantly exposing the relationships. Algorithm 1 is only executed once when presenting a tag for the first time in a transaction. Subsequential transactions with the same tag can be traced back to the first transaction, as will become clear in the next section.

Algorithm 1 Generation

G: Elliptic curve base point
function GENERATE($a, b, \mathcal{U}, \mathcal{V}$)
  $K = x_a Y_b$
  $v \in_R \mathbb{Z}_q$
  $R, S, c, r = \{\}, \quad n = 1, \quad \pi = 0$
  for each $i : Y_i \in \mathcal{U}$ **do**
    for each $j : Y_j \in \mathcal{V}$ **do**
      if $i = a \wedge j = b$ **then**
        $R_n = vG, \quad S_n = vY_b$
        $\pi = n$
      else
        $c_n, r_n \in_R \mathbb{Z}_q$
        $R_n = r_n G + c_n Y_i, \quad S_n = r_n Y_j + c_n K$
      $n = n + 1$
  $s = \mathcal{H}(G \parallel R_1 \parallel S_1 \parallel ... \parallel R_n \parallel S_n \parallel K)$
  $c_\pi = s - \sum c \mod q$
  $r_\pi = v - x_a c_\pi$
  return $c, r, \mathcal{U}, \mathcal{V}, K$

Algorithm 2 Verification

G: Elliptic curve base point
function VERIFY($c, r, \mathcal{U}, \mathcal{V}, K$)
  $R, S = \{\}, \quad n = 1$
  for each $i : Y_i \in \mathcal{U}$ **do**
    for each $j : Y_j \in \mathcal{V}$ **do**
      $R_n = r_n G + c_n Y_i, \quad S_n = r_n Y_j + c_n K$
      $n = n + 1$
  $s = \mathcal{H}(G \parallel R_1 \parallel S_1 \parallel ... \parallel R_n \parallel S_n \parallel K)$
  return $s \stackrel{?}{=} \sum c \mod q$

D. Traceable Relationships

The tag introduced in the previous step allows all system actors to link transactions for a given combination but without deducing the train or RBC that the tag identifies. As the system is used and KMACs need to be updated, a tag will be associated with more than one KMAC transaction. We consider only the most recent KMAC transaction for a given tag to have a valid KMAC, all other KMACs are considered to be revoked. To facilitate this procedure, while keeping the issuer and receiver KMC anonymous, we use a stealth address. Before creating a transaction, the issuer looks up the KMC public key B of the receiver, generates $r, s \in_R \mathbb{Z}_q$ and computes $u = \mathcal{H}(rB) + s$. The resulting stealth address is the tuple $(R, P = uG, S = sG)$ which is included in the transaction as the destination. Every KMC checks for every incoming transaction if $P = \mathcal{H}(bR)G + S$, where b is its public KMC private key. If it holds, the KMC knows that it has a new

Authorized licensed use limited to: TU Delft Library. Downloaded on January 02,2023 at 09:37:19 UTC from IEEE Xplore. Restrictions apply.

KMAC for an asset in its domain. The transaction private key u is only known to the issuer KMC, which is used to update the KMAC from the corresponding transaction by disclosing u in a new KMAC transaction. KMACs in transactions with disclosed private keys are considered to be revoked, ensuring that every asset combination has only one valid KMAC at any time. Transactions with undisclosed transaction private keys are known as unspent transactions.

E. Transaction Creation & Validation

A transaction contains the elements of Table II and groups together the results from step 2, 3 and 4. The result from Algorithm 1 is computed and attached only once in a transaction presenting a new tag: the original transaction for an asset combination. Such a situation accounts, for example, for a new asset in operation, reassignment, or relocation. Even though two KMCs can compute a tag for the same combination and create a valid proof, we assume that only the Infrastructure Manager KMC will issue transactions (see Figure 1). KMAC transactions corresponding to known tags in the system are referred to as update transactions and are considered authentic only if it discloses the private key of the current unspent transaction for the corresponding tag. All update transactions can be traced back to an original transaction. After a transaction is created, it is broadcasted to the peer-to-peer network and every node verifies its correctness and validity of it before adding it to its pool of unconfirmed transactions. Only registered KMCs can publish valid transactions whose authenticity can be verified without deducing the involved KMCs, trains, or RBCs involved.

TABLE II
TRANSACTION STRUCTURE

Item	Description
timestamp	Date and time of transaction creation
destination	Stealth address (see Part IV-D)
tag	Train and RBC combination identifier (see Part IV-C)
payload	Encrypted KMAC, MAC and IV (see Part IV-B)
proof	Result from Algorithm 1 or transaction private key
hash	The digest of the transaction from a hash function

F. Block Creation & Validation

The last step is to create the blockchain by grouping transactions created in step 5 into blocks. A new block is proposed by a *leader* node to the network for acceptance. A block contains a list of transactions, a block number, the hash of the previous accepted block, issuer identifier and signature over the block's digest. Each node will verify the correctness and validity of the included transactions as explained in step 5. The block issuer's signature is verified using the issuer's public key and certificate. If a majority of nodes accept the block, it will be added to the blockchain and the process can start again. Once the block is added to the blockchain, the transactions are confirmed and cannot be removed. The blockchain therefore ensures transaction integrity in a network of nodes that do not fully trust each other. The first block is known as the *genesis* block, agreed upon during the initialization phase.

V. Security Analysis

A. Anonymous relationships

An honest prover using Algorithm 1 will always succeed in constructing a valid proof satisfying Algorithm 2, since correctness is trivial for R_i and S_i for $i \neq \pi$, and

$$\begin{aligned} R_\pi &= r_\pi G + c_\pi Y_a & S_\pi &= r_\pi Y_b + c_\pi K \\ &= (v - x_a c_\pi)G + c_\pi Y_a & &= (v - x_a c_\pi)Y_b + c_\pi K \\ &= vG - c_\pi Y_a + c_\pi Y_a & &= vY_b - c_\pi K + c_\pi K \\ &= vG, & &= vY_b. \end{aligned}$$

Assume that a cheating prover who does not know one of the private keys x was able to compute a valid proof. Because of the second pre-image resistance property of secure hash functions, we can assume that R_π and S_π were fixed before s was computed. Then, for $c_\pi = s - \sum_{i \neq \pi} c_i$, r_π is chosen such that $r_\pi G = R_\pi - c_\pi Y_a$, solving the DL problem, which is infeasible. Therefore a cheating prover, without knowing the private key, will fail to convince the verifier. A verifier cannot infer which train and RBC pair corresponds to π, therefore the best it can do is guess the correct assets involved with a probability of $1/(k+1)$.

B. Stealth address

A stealth address creates anonymous and unlinkable transactions. We refer the reader to [9] for the security proofs. A receiver of a stealth address can give *view* control to another party without giving *spend* control. Our modified stealth address is no different than the original proposal in the sense that an issuer sends a transaction to itself and gives the actual receiver only *view* control.

VI. Performance and evaluation

The complexity analysis is based on 256-bit EC cryptography. We have selected EC secp256k1, hash function sha256 and AES-256 using a 96-bit IV.

A. Time Complexity

Transaction generation and verification times are bounded linearly by the mixin count k. Each value of k introduces 4 EC point multiplications and 2 EC point additions. Algorithm 1 and 2 were implemented in the Rust programming language and results for the generation algorithm with k ranging from 3 to 300, averaged over 100 iterations and obtained using an Apple Mac mini (M1, 2020), are plotted in Figure 2. The verification protocol has two additional EC point multiplications and two additional EC point additions, regardless of k.

B. Communication Complexity

A proof-of-concept using pBFT was implemented in Node.js simulating 100 fully connected nodes spread geographically over 3 Dutch cities (Delft, Rotterdam and Gouda). The nodes communicate using WebSockets over TLS and throughput is analysed by selecting a node at random to propose a new block (assuming that each node has the same set

Authorized licensed use limited to: TU Delft Library. Downloaded on January 02,2023 at 09:37:19 UTC from IEEE Xplore. Restrictions apply.

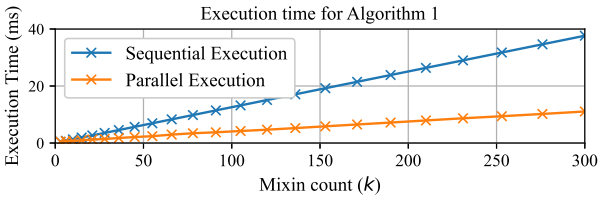

Fig. 2. Sequential and parallel execution times for Algorithm 1. The execution time increases linearly as a function of the mixin count, as expected.

of unconfirmed transactions). For each node, time is measured from the moment it receives the block proposal request until it agrees to add the block to the blockchain. After 10 blocks, on average, all nodes reached block consensus within 4,2 seconds.

C. Space Complexity

The size of original transactions is determined by the result of Algorithm 1, which increases linearly as a function of the mixin count k, and update transactions have a constant size. The blockchain is made out of blocks, containing transactions and, in the case of pBFT, also depend on the number of nodes N in the network. Each confirmed block has $\lceil (2N-1)/2 \rceil$ signatures. Figure 3 depicts the blockchain size maintained by 100 nodes after 7.7 million original transactions for different numbers of transactions per block and transaction size. A larger block size equals a smaller blockchain but comes at the price of a longer average confirmation time.

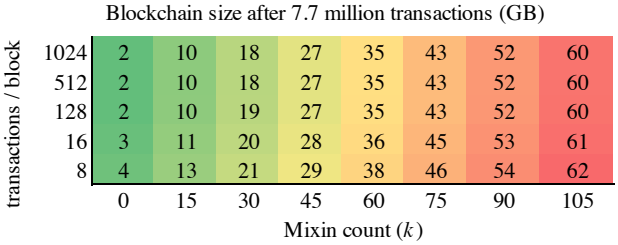

Fig. 3. Blockchain size after 7.7 million original transactions as a function of transactions per block and mixin count in a network mantained by 100 nodes.

VII. Discussion and Conclusion

We have presented PRIDE, a key management system aimed at railway KMCs across Europe. PRIDE is designed to be maintained in a decentralised way, where each KMC contributes to the decentralisation of the network, as stakeholders point out that it is not realistic to have a central body in charge of key management [3]. A blockchain is selected to store the transactions as all transaction data is required to be recorded in an integer way where cybersecurity measures differ per KMC. Relationships between KMC are hidden by assigning each train and RBC combination an identifier meaningful only to their respective KMC, but verifiable by all KMC using a NIZKP. The construction allows weighing anonymity against space and time complexity using a mixin count. Results from our proof-of-concept showed that KMACs can be distributed between KMCs within seconds, a process that currently takes days. After a KMAC transaction is confirmed, KMCs can load the KMAC into their assets as usual (see Figure 1). With a mixin count of 105, 128 transactions per block and 7.7 original transactions (the expected number of train and RBC pairs by 2030), the blockchain requires around 60 GB of storage. Forward secrecy, block size, and trade-off between anonymity and complexity are the subject of future work. Nevertheless, the complexity analysis and implementation results show that PRIDE is a feasible key management system that fits the European Union's railway vision for 2030.

References

[1] M. Ruete, "Work plan 2020 of the european coordinator for ertms," 2020. [Online]. Available: https://ec.europa.eu/transport/sites/default/files/work_plan_ertms_2020.pdf

[2] *Off-line Key Management FIS*, European Union Agency for Railways, 12 2015, set of specifications 3 (ETCS B3 R2 GSM-R B1).

[3] M. Rogier, "Visie op ertms key management van vervoerders en infrabeheerders," 2021.

[4] *On-line Key Management FFFIS*, European Union Agency for Railways, 12 2015, set of specifications 3 (ETCS B3 R2 GSM-R B1).

[5] R. N. Dimitra Liveri, Marianthi Theocharidou, "Railway cybersecurity: Security measures in the railway transport sector," 2020. [Online]. Available: https://www.enisa.europa.eu/publications/railway-cybersecurity/

[6] R. J. Thomas, M. Ordean, T. Chothia, and J. de Ruiter, "Traks: A universal key management scheme for ertms," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ser. ACSAC 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 327–338. [Online]. Available: https://doi.org/10.1145/3134600.3134631

[7] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Advances in Cryptology — CRYPTO '94*, Y. G. Desmedt, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 174–187.

[8] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[9] N. van Saberhagen, "Cryptonote v 2.0," 2013. [Online]. Available: https://bytecoin.org/old/whitepaper.pdf

[10] D. Bernhard, O. Pereira, and B. Warinschi, "How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios," in *Advances in Cryptology – ASIACRYPT 2012*, X. Wang and K. Sako, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 626–643.

[11] M. Franeková, P. Lüley, K. Rástočný, and J. Ždánsky, "Proposal of on-line key management system solutions for railway applications based on asymmetric cryptography," in *Tools of Transport Telematics*, J. Mikulski, Ed. Cham: Springer International Publishing, 2015, pp. 188–197.

[12] S. Hameed, S. A. Shah, Q. S. Saeed, S. Siddiqui, I. Ali, A. Vedeshin, and D. Draheim, "A scalable key and trust management solution for iot sensors using sdn and blockchain technology," *IEEE Sensors Journal*, vol. 21, no. 6, pp. 8716–8733, 2021.

[13] S. Naoui, M. E. Elhdhili, and L. A. Saidane, "Security analysis of existing iot key management protocols," in *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, 2016, pp. 1–7.

[14] T. Koens, S. King, M. van den Bos, C. van Wijk, and A. Koren, "Solutions for the corda security and privacy trade-off: Having your cake and eating it." ING, 2019. [Online]. Available: https://mondovisione.com/_assets/files/Corda_DoSt_v1.6.pdf

[15] A. Preukschat, *Self-Sovereign Identity decentralized digital identity and verifiable credentials*. S.l: O'Reilly Media, 2021.

[16] J. K. Liu, V. K. Wei, and D. S. Wong, "Linkable spontaneous anonymous group signature for ad hoc groups," in *Information Security and Privacy*, H. Wang, J. Pieprzyk, and V. Varadharajan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 325–335.

Authorized licensed use limited to: TU Delft Library. Downloaded on January 02,2023 at 09:37:19 UTC from IEEE Xplore. Restrictions apply.