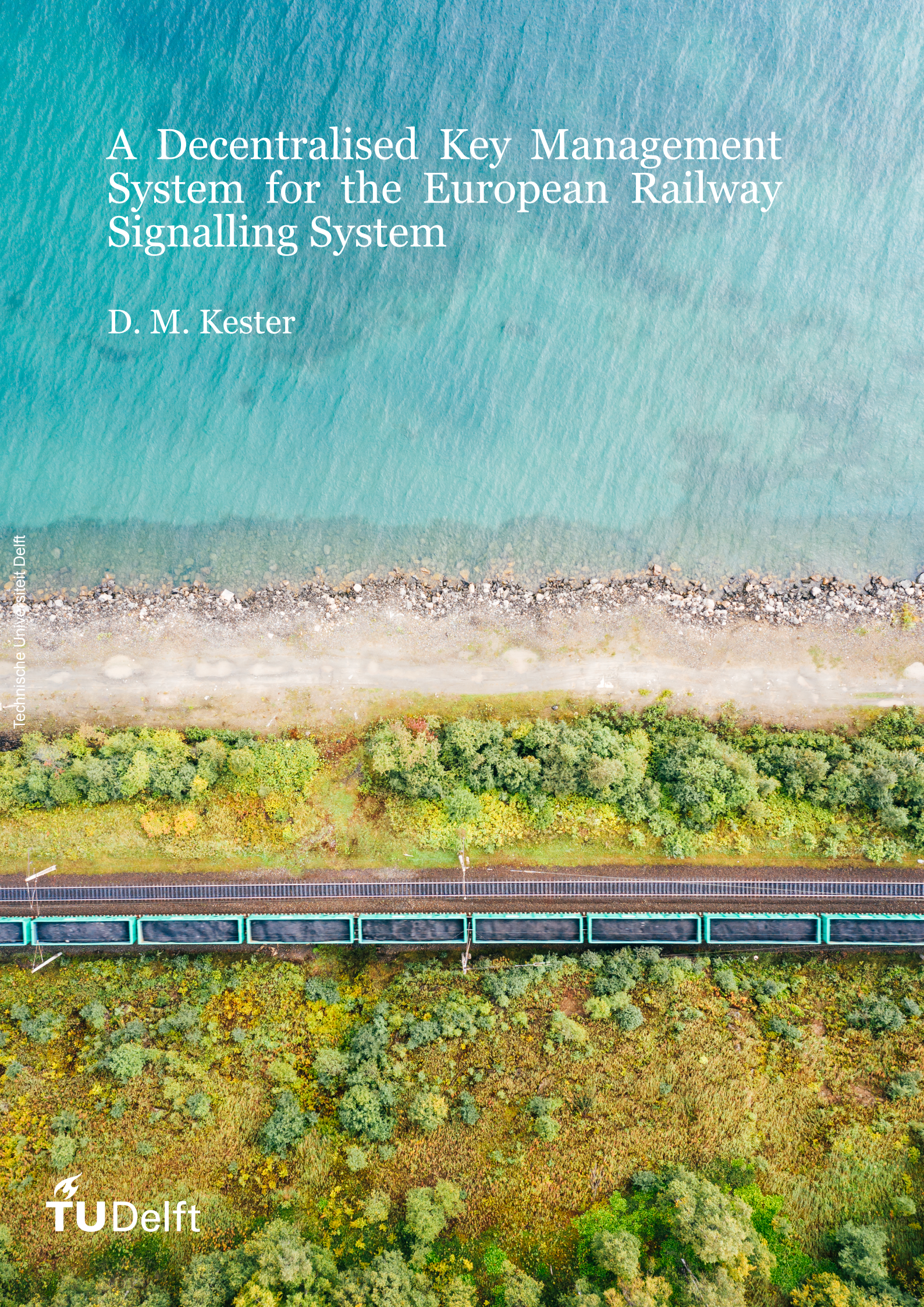
A Decentralised Key Management System for the European Railway Signalling System

D. M. Kester



Technische Universiteit Delft

TU Delft

A Decentralised Key Management System for the European Railway Signalling System

D. M. Kester

to obtain the degree of Master of Science
at the *Delft University of Technology*,
to be defended publicly on Friday December 17, 2021 at 9:20 AM.

Student number:	4221745	
Project duration:	February 1, 2021 – December 17, 2021	
Thesis committee:	Dr. Z. Erkin,	TU Delft, supervisor
	Prof. dr. ir. A.J. van der Veen,	TU Delft
	Dr. ir. E. Schrik CEng,	Mott MacDonald
	Ir. M. van Hesse CEng,	Mott MacDonald

TU Delft

Abstract

Modern railway signalling systems are based on a wireless communication link between train and trackside entities. Secure communication between these entities is established using cryptographic symmetric keys loaded beforehand. Train and trackside entities across Europe are maintained by different Key Management Centres (KMC), making the distribution of the symmetric keys challenging. The involved parties would benefit from using a single European key distribution system. Nevertheless, a recent study concluded that a centralised approach of such a single system is not feasible.

This work presents, to the best of our knowledge, the first decentralised key management system to be used by railway KMC across Europe. Existing procedures mandate that key distribution activities concerning key generation, distribution and deletion must be logged. To meet this requirement, the proposed decentralised system is based on a private and permissioned blockchain. The network is maintained by the KMC making use of the system and access to the system is granted by Registration Centres.

During the design of a single system to replace several one-to-one solutions between KMC, it came into light that train and/or trackside equipment owners might not accept revealing certain types of relationships, as these could, for example, reveal commercial strategies. To overcome this, the proposed decentralised system introduces privacy-preserving and verifiable combinations of train and trackside entities. The protocol is based around the decisional Diffie–Hellman assumption witness indistinguishable proofs.

The proposed design enables European railway KMC to use a single decentralised and scalable system to exchange cryptographic material in a secure and privacy-preserving way. Scalability is shown by building a proof-of-concept based a Byzantine Fault Tolerance consensus protocol. 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.

Preface

This master thesis was done in collaboration with Mott MacDonald, an engineering, management and development consultancy active in 140 countries, including The Netherlands. The Dutch branch is specialised in sectors ranging from mobility, infrastructure, buildings, to energy and environment.

On the cover

The photo on the cover shows an aerial view of freight train wagons. Freight trains usually cross several countries during a mission and therefore benefit from a European interoperable signalling system. ERTMS Level 3 introduces the concept of moving blocks, which requires the implementation of Onboard Train Integrity (OTI) systems. OTI systems monitor the integrity of a train, just as hash functions provide data integrity on a blockchain.

On the implementations

The ideas and designs presented in this thesis were implemented as proof of concepts using Javascript and Rust code. Code is not provided in this work but can be found online at:

`https://github.com/davidkester/pride`

Acknowledgments

This work would not have been possible if not because of Eelco Schrik vision and dare to push innovation in the railway sector and, at the same time, Mark van Hesse's critical and sober view. It was great to work on such a relevant and actual problem, but lifting on Eelco's and Mark's expertise on the subject really brought the project to life. A special acknowledgement goes to Ido, Joost and everybody at Mott Macdonald, where I was welcomed to IEN team right from the beginning and involved in all company activities. I am truly thankful I got the opportunity to graduate at company which gave me with a lot of freedom and support to work on my project.

I want to thank Zekeriya Erkin for being a remarkable supervisor. The comments and feedback obtained throughout the process made me grow in a professional but also personal way. I want to thank all master and Ph.D. students of the research group for their help and feedback. Special thanks to Tianyu Li, for his help during the writing of the BRAIN 2022 paper. In addition, I would like thank Alle-Jan van der Veen for his involvement in the project and sharp remarks.

During this project, I got the unique opportunity to talk and discuss ideas with relevant people in the field. In particular, a special mention goes to:

John Boss, ProRail
Ron van Kampen, Infraspeed
Rogier Meurs, NS
Jaco Schoonen, ProRail
Erik Spitter, Infraspeed
Marijn Verheul, NS
Sebastiaan Woertman, NS

Last but not least, a shoutout to my friends and family for their support and being with me during almost ten years of student life.

D. M. Kester
Rotterdam, December 2021

List of Tables

Contents

1

Introduction

Railway signalling systems are designed to control railway traffic alongside preventing collisions and unsafe situations. Passenger and freight trains are characterised by having a braking distance in the order of kilometres. Because of this large breaking distance, a train driver cannot solely rely on sight for a safe mission. The European signalling and speed control system, ERTMS, aims to replace different national legacy systems to increase the interoperability and capacity of the European rail network. Train and trackside equipment are assets maintained by different Key Management Center (KMC) across Europe. It is estimated that there are currently 100 KMC active in Europe. As of 2020, around 3 600 trains have been equipped with ERTMS in Europe and the goal is to have between 27 500 and 38 500 trains fitted with ERTMS by 2030. Approximately 6 120 km of tracks have been fitted with ERTMS, which accounts to 12% of the 2030 goal [47].

ERTMS is based on a wireless communication link between train and trackside entities known as Radio Block Centre (RBC) in charge of a geographic area or a specific railway line. A train and an RBC share a triple-DES key known as `KMAC`, used for mutual identification, authentication, and session establishment. The RBC provides speed boundaries and additional track information during a session, and the train sends periodic status updates, such as position. Each train can have 2000 `KMAC` loaded on board at most, which translates to 77 million possible train and trackside combinations. For any train and RBC combination in service, a `KMAC` needs to be shared beforehand between the respective KMC.

There are currently two procedures for key distribution between ERTMS entities [16, 17]. The first procedure, an offline method, exchanges `KMAC` using storage media such as compact discs and flash drives. The procedure is not scalable and prone to operational mistakes. The second procedure, an online method, was specified in 2015 to overcome the shortcomings of the offline method. The online method prescribes TLS-PKI for communication between KMC. In practice, the online method is currently not implemented and the offline implementation differs per KMC as the specifications only provide guidelines. `KMAC`'s are hardly refreshed once issued and on top of this, Liveri et al. [11] found out that around half of ERTMS KMC have not or have partially implemented cybersecurity measures, ranging from risk assessments and audits to upgrading outdated systems.

A single and scalable key management system would simplify key distribution between domains and improve cross-border operations. Nevertheless, a recent study from relevant stakeholders brought to light that a centralised approach is challenging to realise since it is difficult to place the responsibility of a PKI for the whole European railway area with one organisation [46]. Although previous work exists addressing ERTMS key management [10, 53], the work is either built around a centralised approach or aimed to reduce management overhead within KMC domains.

Conversations with KMC administrators pointed out that (freight) railway operators benefit the most from having as much KMAC as possible loaded into their trains. Each additional KMAC means that it can potentially serve more clients, possibly creating a competitive advantage over other operators. Without proper measures, a key management system could expose relationships that, this work assumes, operators might want to keep private, as these could reveal new commercial strategies or other sensitive information.

1.1. European railway landscape

Infrastructure Managers are in charge of trackside entities and it is common for a country to have one major Infrastructure Manager, for example, *Infrabel* in Belgium and *ProRail* in The Netherlands. Organisations in charge of trains are known as Railway Undertakings, Railway Operators or simply operators.

A KMC is in charge of the distribution, installation and deletion of cryptographic material from train or trackside entities owned by an organisation. In general, each organisation sets up its own KMC, but it can also delegate the task to a third party. A KMC must store keys in a way that they remain authentic and confidential. A K-KMC key is currently used for authentication and encryption between KMCs and the specifications do not prescribe how the K-KMC should be distributed besides stating "... the confidentially of K-KMC shall be guaranteed outside the involved KMCs" [16]. Usually only Infrastructure Manager KMC issues K-KMC; operators request K-KMC. The core functionalities of a KMC are summarised as follows:

1. KMC domain definition, i.e., defining which assets are managed by the KMC.
2. KMAC exchange with another KMC in a confidential, integer and authentic way.
3. Update and revocation of existing (already issued) KMAC.
4. Archiving of KMAC's and associated transaction data, such as a date and time.

1.2. Research objective

Based on interviews with operators, Infrastructure Managers and desk research, it became clear that ERTMS would benefit from a single, decentralised and scalable key management system on which KMAC's can shared in a secure way. This research project attempts to design such a system. The main research question for this work is formulated as follows:

"How can we design a scalable and decentralised key distribution system for the European railway signalling system?"

The system needs to facilitate mutual authentication between KMC's and every action must be recorded in order to facilitate audits or dispute resolutions. The research question is divided into the following sub-questions:

(i) How can we address key update and revocation?

(ii) How can we provide key auditability and traceability?

(iii) How can we anonymise relationships and transactions between KMC?

1.3. Our contribution

This work presents, to the best of our knowledge, the first privacy-preserving decentralised key management system tailored for railway KMCs. The system is built around a permissioned blockchain and allows KMCs to exchange symmetric keys for entity pairs in a confidential, integer and authentic way using transactions. A KMC is admitted to the system and afterwards able to 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 the asset owners or managers involved.

Without proper measures, anonymising users in a system opens the door to misbehaving activities such as denial of services attacks. The mitigate this risk, we enforce the use of authenticated but anonymous transactions. The tag is embedded in every transaction and each KMC can verify it is from the set of possible asset combinations 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, train, or trackside entities.

Identifying the asset combination in each transaction allows KMCs to trace back the KMAC history of their assets. The proposal also ensures that each pair can only have one valid KMAC at any point in time. Blockchain's append-only nature guarantees that all transaction data is appropriately recorded, as existing procedures require logging activities concerning key generation, distribution, and revocation [45]. In case of a node crash or failure, the latest state of any node can be reconstructed from the blockchain using only the private keys.

Altogether, this has resulted in:

Poster presentation
at the 7th Annual Cyber Security Next Generation Workshop

PRIDE: A Privacy-Preserving Decentralised Key Management System
David Kester, Tianyu Li, Zekeriya Erkin
Under review at the IEEE International Conference on Pervasive Computing and Communications - Workshops and Events - BRAIN 2022

A symposium was planned together with Mott Macdonald to present the design and ideas shown in this work to Dutch KMC administrators during the project. Unfortunately, due to the COVID-19 pandemic, the event has been postponed to March 10th, 2022.

1.4. Thesis outline

The rest of the thesis is structured as follows. Chapter 2 summarises relevant concepts used in this work and Chapter 3 reviews related work. Chapter 4 presents PRIDE, the proposed decentralised key management system, analysed and evaluated afterwards in Chapter 5. The paper ends with a discussion, future work and concluding remarks.

A digest of interviews and email conversations between *ProRail* and *NS* during this work can be found in Appendix A. During the project, a field trip took place at an RBC in operation, a brief summary of the experience is presented in Appendix B. The reader is encouraged to read these appendixes to get a better felling regarding the scope and possible end users of PRIDE.

2

Preliminaries

The first two sections introduce elliptic curve-based cryptography, the discrete logarithm, and associated hard problems. Next, cryptographic secure hash functions are presented, followed by encryption and authentication schemes. Zero-Knowledge Proofs (ZKP) are touched upon, and it is shown how the Fiat-Shamir heuristic creates a non-interactive ZKP. Section 2.6 gives an overview of the blockchain and related concepts such as consensus and anonymity. The last section is dedicated to ERTMS and ERTMS key management.

Notation The concatenation of strings α and β is denoted by $\alpha||\beta$. The expresion $x \in_R X$ means that x is chosen randomly from the set X according to the uniform distribution. The notation $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ is used for the floor and ceiling function, respectability returning an integer. The set of values produced by postfix operation $\mod N$ where N is a positive integer is

$$\mathbb{Z}_N = Z/NZ = \{0, 1, ..., N-1\}$$

and when N is a prime q, then $\mathbb{Z}_q$ yields a field $\mathbb{F}_q$

2.1. Elliptic-curve Cryptography

Cryptosystems based on elliptic curves achieve the same level of security as other types of cryptosystems using a smaller key size. For example, to achieve 128 bits of security, it is recommended to use 256-bit keys for elliptic curve-based cryptography and 3072-bit keys for factoring-based key cryptosystems [3].

An elliptic curve E is defined as the set of points $(x, y) \in \mathbb{F}_q^2$ satisfying Equation 2.1

$$y^2 = x^3 + ax + b \mod q \tag{2.1}$$

such that $4a^3 + 27b^2 \neq 0 \mod q$, where $a, b \in \mathbb{F}_q$ and q a large prime $q > 3$.

The group of elements generated by a curve E contain a base point or generator element (not necessarily unique) $G \in E(\mathbb{F}_q)$ of prime order n. The values q, a, b, G, n, h, where h is the cofactor of G, are the domain parameters of an elliptic-curve cryptosystem and are known to all participants. Standard bodies publish domain parameters of elliptic curves for several common field sizes resulting in *standard* curves.

Throughout this work, the additive notation is used and, without loss of generality, lower-case is used for elements in $\mathbb{Z}$ and upper-case for elements in $E(\mathbb{F}_q)$. For a given point $P = (x, y)$, the y coordinate

can be derived from x as there are two possible y coordinates for any x. To reduce space, a point can be represented in a compressed form, with x and bit to distinguish between the two y possibilities, i.e. $P = (x, \pm 1)$.

Point addition The addition of two points $P = (x_1, x_1)$ and $Q = (x_2, y_2)$ results in $R = (x_3, y_3)$ where x_3 and y_3 are computed using Equation 2.2 and Equation 2.3, respectively.

$$x_3 = \lambda^2 - x_1 - x_2 \mod q \tag{2.2}$$
$$y_3 = \lambda(x_1 - x_3) - y_1 \mod q \tag{2.3}$$

λ is obtained from Equation 2.4.

$$\lambda = \begin{cases} (y_2 - y_1)/(x_2 - x_1) \mod q & \text{if } P \neq Q \\ (3x_1^2 + a)/(2y_1) \mod q & \text{if } P = Q \end{cases} \tag{2.4}$$

Key-pair generation Given a generator G and security parameter κ, a key generation algorithm outputs a private key x and public key Y where $x \in_R \mathbb{Z}_q$ and $Y = xG$. Roughly speaking, the security parameter tells how many operations a (probabilistic polynomial time) adversary needs to perform in order to break a system.

2.1.1. Discrete Logarithm

Discrete logarithms are logarithms defined with regard to multiplicative cyclic groups. The discrete logarithm of an element Y to the base G is a unique integer $x \in \mathbb{Z}_q$ such that $Y = xG$. The discrete logarithm x is also referred to as the index of Y with respect to the base G.

Definition 2.1 (Discrete Log Problem (DLP))**.** Given G and xG, find x.

Definition 2.2 (Computational Diffie–Hellman Problem (DHP))**.** Given G, xG and yG, for some random x and y, find C such that $C = xyG$

Definition 2.3 (Decision Diffie–Hellman problem (DDH))**.** Given G, xG and yG, for some random x and y, it is not possible to distinguish xyG from some random value.

There are elliptic curves were solving the discrete logarithm problem is believed to be hard, i.e., there is no proven algorithm for solving it in a reasonable time. A problem is said to be hard if a polynomial-time adversary has a negligible advantage in solving the problem.
negligible.

2.2. Hash Functions

A hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$ is a function that, on a set of binary bit-strings of arbitrary size $\{0,1\}^*$ as input, outputs a fixed-length bit string known as a digest or hash value. A cryptographic secure hash function meets the following properties [51]:

- Collision Resistance: it is infeasible to find two different messages that output the same hash value.
- Preimage Resistant: given a hash value, it is hard to find a message that outputs such hash value.
- Second Preimage Resistant: given a message and corresponding hash value, it is hard to find a different message producing the same hash value.

2.3. Encryption Schemes

An encryption algorithm, or cipher, transforms plaintext into ciphertext using a secret key to achieve confidentiality. The process of converting a ciphertext into plaintext is known as decryption. Both algorithms are public, and the secrecy depends on the secret key. There are two types of encryption schemes: *symmetric* and *asymmetric* cryptosystems; the latter is usually referred to as public-key cryptosystems.

2.3.1. Symmetric setting

Symmetric key cryptography requires that each party have a copy of the same secret key. The same key is used for encryption and decryption, hence symmetric. Given a set of N parties, $N(N-1)/2$ different symmetric keys are needed to secure communication channels between all possible pairs.

Block ciphers operate on n bits blocks of plaintext, and the resulting ciphertext block is assumed to be of the same size. The most famous block cipher is the Data Encryption Standard (DES), with a block length of 64 bits. DES has been phased out, but it is still used as a component in triple DES, which is deemed secure until 2023. The Advanced Encryption Standard or AES is currently the standard symmetric encryption algorithm recommended by the US National Institute of Standards and Technology (NIST). AES works on 128-bit block sizes with key sizes of 128, 192, or 256 bits.

Block ciphers are used in a mode of operation to encrypt plaintext. These modes of operation divide the plaintext into a series of blocks of the cipher's block size and process the data following a defined procedure. For example, Cipher Block Chaining (CBC) Mode protects against deletion and insertion attacks.

2.3.2. Public key setting

The use of symmetric keys is replaced in public-key cryptography with a mathematically related *public* and *private* key pair. The public key can be safely published in a directory along with the holder's identity. The public key does not reveal anything about the private key. Anyone can encrypt a message with the public key; however, only the private key holder can decrypt the corresponding ciphertext.

Public key infrastructure A public key infrastructure (PKI) facilitates the authentication and distribution of public keys. The most commonly employed approach consists of a certificate authority (CA) that issues digital certificates to verify that a particular public key belongs to a specific entity. The X.509 standard defines the digital certificate format used by most applications. Before a certificate is issued, a registration authority (RA) verifies the identity of entities requesting a digital certificate from the CA. A CA acts as a trusted third-party and is considered a single point of failure in a PKI. There can be many CAs and certificates can be chained as in Figure 2.1. In this case, clients only need to *trust* the Root CA.

2.4. Authentication Schemes

While encryption schemes, as described in Section 2.3, ensure confidentiality, authentication schemes are used to verify that a message is correct and that an entity is who it claims to be. Message Authentication Codes (MAC) provide authentication in the symmetric setting, and Digital signatures are used in a public-key environment.

2.4.1. Message Authentication Codes

Parties sharing a secret symmetric key can ensure that data transmitted between them has not been tampered with using a MAC. A MAC is the result of an algorithm sent alongside the data. Upon receiving a message and a MAC, a receiver computes the MAC again using the secret key and checks if the received MAC matches; in case it matches, it can be sure that the data has not been tampered with and sent from

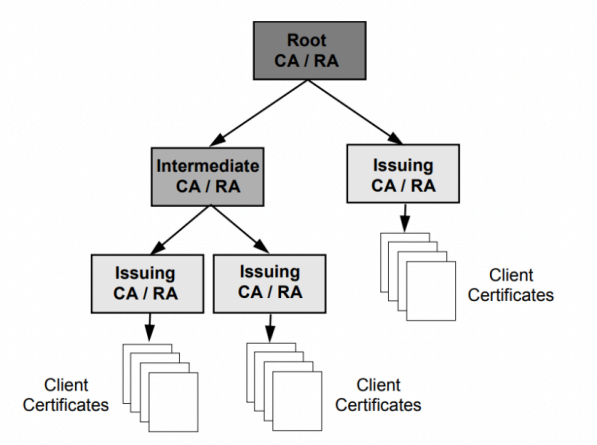


Figure 2.1: A PKI based on CAs where clients authenticate each other only trusting Root CA [46].

someone that also has the secret key. It should be hard for an adversary to forge a valid MAC without knowing the secret key.

2.4.2. Digital Signature Schemes

A signature algorithm is used to *sign* a message using a private key. The result can be verified by everybody using the corresponding public key and no one can realistically recover the private key from it. Because only the private key owner can create a valid signature, these schemes also provide non-repudiation: a message issuer providing a signature cannot deny having sent the message. The Digital Signature Algorithm (DSA) is a digital signature proposed by NIST based on modular exponentiation and the discrete logarithm problem. The EC variant is known as the Elliptic Curve Digital Signature Algorithm (ECDSA). A private key y owner generates a signature on message m using Algorithm 2.1. The signature is verified using algorithm 2.2 and the EC public key Y. The size of the signature is twice the size of the private key.

Algorithm 2.1 ECDSA Generation

G: Elliptic curve base point
function Generate(m, sk)
  $k \in_R \mathbb{Z}_q$
  $(x, y) = kG$
  $r = x \mod q$
  $s = k^{-1}(m + r \cdot y) \mod q$
  r, s
  return r, s
end function

Ring signatures A *ring* signature scheme consists of a set of public and private key pairs. A message m can be signed using one of the private keys and a set of public key and the result is verified using the public keys. It should be infeasible to tell which private key was used to created the signature. The construction can be extended to use more than one private key, creating threshold ring signatures. When a trusted group manager assigns the key pairs to, for example, users, to scheme is referred to as a group signature scheme [49].

Algorithm 2.2 ECDSA Verification

G: Elliptic curve base point
function Verify$(m, Y, (r, s))$
$u_1 = m \cdot s^{-1} \mod q$
$u_2 = r \cdot s^{-1} \mod q$
$(x, y) = u_1 G + u_2 Y$
$x = r \mod q$
return $r \stackrel{?}{=} x$
end function

2.5. Zero-Knowledge Proofs

Zero Knowledge Proofs (ZKP) are used by a prover to convince a verifier from a witness that he/she knows information satisfying a given relationship without revealing anything else [21]. A ZKP must satisfy the following properties:

- Completeness: If an honest prover knows a solution and follows the protocol correctly, it will always succeed in convincing a verifier.
- Soundness: A cheating prover that does not know a solution will fail with an overwhelming probability to convince a verifier.
- Zero-Knowledge: The result of the protocol does not reveal anything about the secret.

Sigma protocols are three-move interaction protocols used to create ZKP. The order of the messages is crucial for security. Schnorr's protocol, presented in Protocol 2.1, is an example of a Sigma protocol. The values T, c and r are called *commitment*, *challenge* and *response*, respectively. The protocol is used to prove knowledge of w such that $Y = wG$.

Prover (x, G)		Verifier (G)
$v \in_R \mathbb{Z}_q$ $T = vG$		
	$\xrightarrow{T}$	
		$c \in_R \mathbb{Z}_q$
	$\xleftarrow{c}$	
$r = v + cw \mod q$		
	$\xrightarrow{r}$	
		$rG \stackrel{?}{=} T + cY$

Protocol 2.1. Schnorr's protocol

Fiat and Shamir proposed a generic transformation to create non-interactive zero-knowledge proof from sigma protocols which are secure in the random oracle model [1]. In a non-interactive obtained from the Fiat-Shamir transformation, the challenge is chosen prover using a random oracle. In practice, the random oracle is replaced by a cryptographic hash function. The strong Fiat-Shamir transformation is achieved by including generator G and the statement to be proved Y in the hash function [4]. The non-interactive version of the Schnorr protocol is known as the Schnorr's signature and is presented in Protocol 2.2. The scheme is used to prove knowledge of a discrete logarithm w such that $Y = wG$ [6].

[1] It is not straightforward to prove the security of hash functions. Cryptographic hash functions are replaced with a utopian equivalent known as a random oracle to facilitate protocols proofs. There is still a debate regarding the security proofs obtained using the random oracle model [48].

By including a message m in the computation of the hash, the protocol can be used as a digital signature scheme using the discrete logarithm as the private key.

Prover (x, G)		Verifier (G)
$v \in_R \mathbb{Z}_q$		
$T = vG$		
$c = \mathcal{H}(G \parallel Y \parallel T)$		
$r = v - cw \mod q$		
	$\xrightarrow{(c,r)}$	
		$T' = rG + cY$
		$c \stackrel{?}{=} \mathcal{H}(G \parallel Y \parallel T')$

Protocol 2.2. Non interactive Schnorr's protocol, known as Schnorr signature

An honest prover will succeed in constructing a valid proof since the following holds:

$$T' = rG + cY = (v - cw)G + cwG = vG = T$$

Given generators G and H, Protocol 2.3 presents a 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 is known as a Chaum-Pedersen proof. We refer to (G, Y, H, Z) as a Diffie-Hellman (DH) tuple in case that $H = xG$ for $x \in \mathbb{Z}_q$.

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

Protocol 2.3. A non-interactive proof of knowledge to prove equality of a discrete logarithm.

Cramer et al. [10] 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. If it infesible to distinguish which of the possible relationships are being proven, a sigma protocol considered to be witness indistinguishable. Perfect special honest verifier implies witness-indistinguishability [10]. Protocol 2.4 is an example of a 1 out of 2 non-interactive witness indistinguishable protocol to prove knowledge of the discrete logarithm w such that $Y = wG$ or $K = wH$, assuming that the prover knows w such that $Y = wG$ is true. The result reveals nothing about which of the two witnesses corresponds to the statement to be proven.

Prover (w, G, H)		Verifier (G, H)
$v \in_R \mathbb{Z}_q$		
$T_1 = vG$		
$r_2, c_2 \in_R \mathbb{Z}_q$		
$T_2 = r_2 G + c_2 V$		
$c = \mathcal{H}(G \parallel H \parallel T_1 \parallel T_2 \parallel V \parallel K)$		
$c_1 = c - c_2 \mod q$		
$r_1 = v - c_1 w \mod q$		
	$\xrightarrow{(c_1, c_2, r_1, r_2)}$	
		$T_1 = c_1 U + r_1 G$
		$T_2 = c_2 K + r_2 G$
		$c = c_1 + c_2 \mod q$
		$c \stackrel{?}{=} \mathcal{H}(G \parallel H \parallel T_1 \parallel T_2 \parallel V \parallel K)$

Protocol 2.4. Non-interactive witness indistinguishable protocol

2.6. Blockchain

A blockchain is an append-only tamper-proof ledger that opened the door for the development of decentralised applications, such as cryptocurrencies, without needing a centralized authority [37]. Transactions are issued by clients and broadcasted through a peer-to-peer (P2P) network of nodes. New transactions are bundled together into *blocks*. The contents of a block are hashed together with the hash of the latest accepted block, creating a cryptographic chain of blocks: the blockchain.

Transactions In general, a cryptocurrency based on a blockchain consists of transactions transferring spending control from one entity to another. A transaction holds a certain amount of value in the respective currency, and a transaction that has not been *consumed* is known as an Unspent Transaction Output (UTXO). Each UTXO represents a chain of ownership and are used as input to create new transactions. Once consumed, i.e. used as a input for a new transcription, a former UTXO cannot be spent again. The sum of values contained in all UTXOs represent the total supply of a cryptocurrency at a given point in time.

Merkle root The blockchain append-only property implies that a size of the blockchain will only grow as more transactions are added to it. After a certain time, certain transactions could be removed to regain storage space. To do this, without breaking the blockchain, Nakamoto proposed to only include the Merkle root from a Merkle tree obtained from the transaction hashes in each block. Figure 2.2 illustrate a Bitcoin block, containing four transactions: Tx0, Tx1, Tx2 and Tx3. After certain condition is met, one or more transactions can safely be discarded.

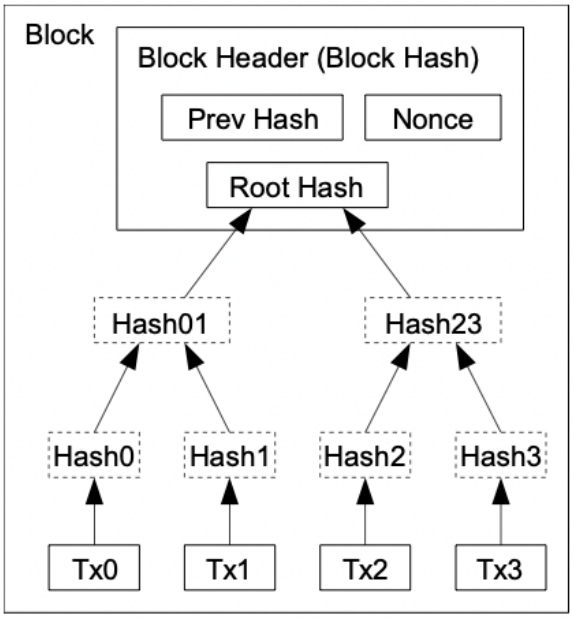


Figure 2.2: Example of a Bitcoin block containing four transactions. The block hash is made out of the hash of the previous block, a nonce used for the consensus algorithm, and a root hash, the result from the Merkle tree using the four transactions: Tx0, Tx1, Tx2 and Tx3. The block hash is included in the hash of the next block, and removing the transactions will not affect the block hash [37].

Depending and the application and design requirements, a blockchain can be either permissionless or permissioned:

Permissionless blockchain Bitcoin and Ethereum [56] are examples of permissionless blockchains, where anyone is allowed to read and write unto the system without a specific identity. These applications usually involve some type of cryptocurrency and are maintained by economic incentives [2]. A permissionless blockchain typically makes use of a consensus protocol where a new block is added to the blockchain by a node selected in proportion to a resource that is expected nobody can monopolise. In the case of Bitcoin, the protocol is based on computer power and is known as *proof of work* [39].

Permissioned blockchain When users need to be explicitly admitted to the system, it is considered a permissioned blockchain. Such a registration process prevents Sybil attacks[2] and allows the implementation of finite consensus protocols such as practical Byzantine Fault Tolerance (pBFT). Registration is generally done by a central authority assigning rights or attributes to users or organisations. Hyperledger Fabric and R3 Corda are two well-known platforms for the development and deployment of permissioned blockchains [57].

2.6.1. Consensus

A *distributed* system consists of a set of nodes spread-out across space improving scalability, resilience and possibly reduce latency when compared to non-distributed systems. A decentralised system is a distributed system where governance is spread over multiple entities [50]. A decentralised application relies on a consensus algorithm to ensure that nodes agree on some value generated by an honest node [39]. Consensus is, in general, a hard problem since nodes might crash or act maliciously. A consensus algorithm must guarantee the following two properties:

[2] An attacker creates a large number of users to obtain a disproportionately large influence on the system.

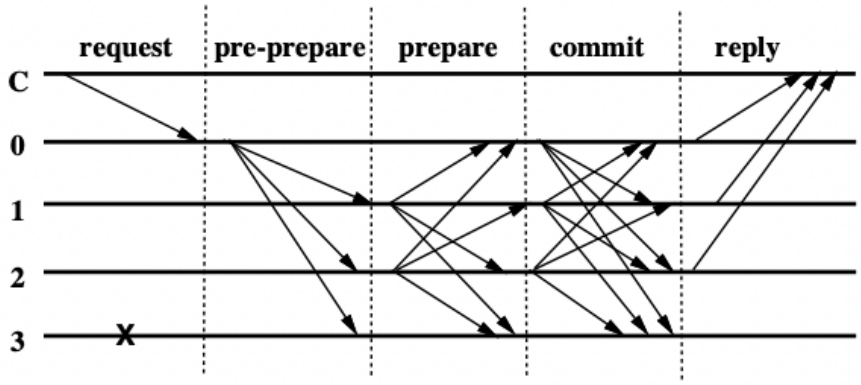


Figure 2.3: A 4 node pBFT setup where leader node 0 proposes a message received from a client C. Even though node 3 is faulty, the remaining nodes can reach consensus [8].

- Liveness: As long as the number of faulty nodes is below a certain threshold, it is not possible to indefinitely delay the acceptance of a correct message, in other words, consensus cannot stall.
- Safety: As long as the number of faulty nodes is below a certain threshold, a node cannot convince others to accept an incorrect or invalid message. The algorithm is also robust against network delays, partitions, and packet loss, duplication, and reordering.

Practical Byzantine Fault Tolerance In the late 90s, Barbara Liskov and Miguel Castro presented the pBFT algorithm to tolerate Byzantine faults [8], based on the the *Byzantine Generals Problem*. It is classical problem where generals, some of them traitors, of the Byzantine army are separated from each other and need to agree on a plan of action using messengers. Before pBFT, there was no practical way to solve this problem. An initial use case was a Byzantine-fault-tolerant network file system service running under different hardware and software. The number of faulty or malicious nodes f that a system with N nodes can withstand is given by Equation 2.5.

$$f = \left\lfloor \frac{N-1}{3} \right\rfloor \tag{2.5}$$

The algorithm starts by selecting a *leader* node, which will propose a message or request on request from a client. Leader election is done in a round-robin fashion, and a new leader is elected if the current leader crashes or acts suspiciously. We refer to the event that a new leader is elected as *view change*. The remaining $N-1$ nodes are referred to as *replicas*. Consensus is achieved in three phases:

- Pre-prepare: A node has received the message from the leader, and if valid, it will broadcast a pre-prepared message.
- Prepare: A node has received at least $\lceil (2N-1)/2 \rceil$ valid pre-prepared messages, including its own, and broadcasts a prepared message.
- Commit: A node has received at least $\lceil (2N-1)/2 \rceil$ valid prepared messages, including its own, and accepts the message proposed by the leader.

Consensus is reached when at least $\lceil (2N-1)/2 \rceil$ nodes have reached the commit phase. Figure 2.3 depicts the interaction between a leader node and three replica nodes, where one of the replicas is faulty. The drawback of pBFT is the exponential message increase as a function of the number of nodes. The protocol is considered to scale well up to 200 nodes [25].

Istanbul BFT Istanbul BFT (IBFT) is a blockchain consensus algorithm based on pBFT used in the *Quorum* blockchain. Like pBFT, each IBFT round achieves termination after three phases and has $\mathcal{O}(N^2)$ total communication complexity. The message proposed by the leader in IBFT is the proposed block to be added to the blockchain. If the block is accepted, IBFT requires each node to store at least $\lceil (2N-1)/2 \rceil$ of the commit messages in the corresponding block. The list of commit messages per block will most likely differ per node [35].

Raft Raft is a consensus algorithm designed as an alternative to Paxos, also distributed consensus algorithm. Paxos is considered to have two significant drawbacks: 1) it is difficult to understand, and 2) it does not provide a good foundation for creating practical implementations [42]. Paxos does not produces inconsistent results but can get stuk under certain rare conditions [39]. Raft is similar to pBFT, but the difference is that the replicas always trust the leader. While this construction reduces the communication complexity, it might not be acceptable in cases where nodes do not trust each other. Raft is known as a Crash Fault Tolerant algorithm.

2.6.2. Privacy and Anonymity

Without proper measures, transactions on a blockchain can leak more information than expected through *side channels*, which is especially a problem in cases where anonymity and privacy need to be preserved. For example, Bitcoin aimed to provide anonymity using pseudonyms, but in practice, information about the user could be inferred using network-layer deanonymization and graph analysis. These threats are also common in many other blockchain applications and should be taken into consideration.

Network-layer deanonymization When a client issues a transaction, it is propagated (gossiped) through the p2p network. Nodes in the (Bitcoin) network are connected via an unencrypted Transmission Control Protocol (TCP) channel and communicate using the Internet Protocol (IP). In certain situations, this construction makes it possible to match different accounts or transactions to one IP address. The use of a Virtual Private Network (VPN) or a Tor browser can reduce the risk of network-layer deanonymization.

Graph analysis and clustering In many blockchain implementations, a transaction is equal to receiving or transferring ownership from one entity to another. Therefore, a transaction is a chained series of transactions. All this information is available on the blockchain and can be used to deduce certain relationships or patters.

2.7. ERTMS

ERTMS is the standard for railway signalling systems set by the European Union. It aims to replace all different national systems by a single large interoperable European system. ERTMS has different levels of implementations: level 1, 2 and 3. Starting from level 2, trackside signals are no longer required, communication between train and trackside equipment is wireless and cryptographic primitives are introduced. ERTMS contains the *Euroradio* protocol, GSM-R and an application layer.

GSM-R Wireless communication between OBU and RBC takes place using a dedicated spectrum band through Global System for Mobile Communications – Railway (GSM-R), on which ERTMS relies for data encryption. GSM-R is built on top of standard GSM and is considered to be outdated, and obsolete [52]. The expectation is that GSM-R will remain in service until 2030 and be replaced by Future Railway Mobile Communication System (FRMCS)

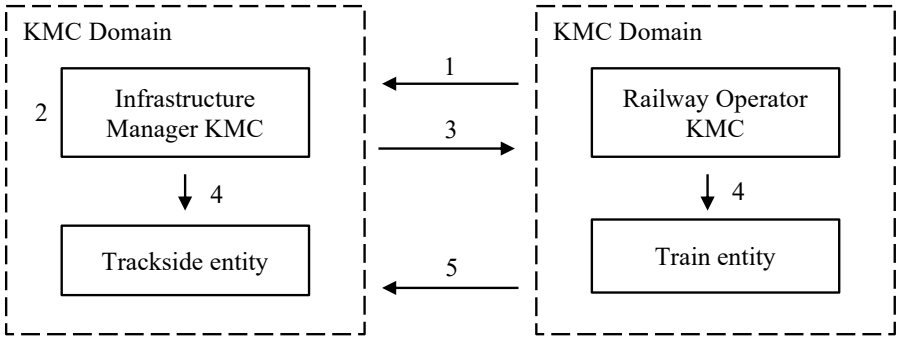


Figure 2.4: KMAC exchange between two KMC domains: 1) An operator requests a KMAC for a specific On Board Unit (OBU)-Radio Block Center (RBC) pair. 2) The Infrastructure Manager (IM) acknowledges the request and generates a unique KMAC. 3) The KMAC is *securely* shared with the operator 4) Both KMC load the KMAC into their respective assets, i.e., KMAC entities. 5) The OBU performs an operation check with the RBC via GSM-R.

Table 2.1: Interactions between ERTMS entities

Interactions	Description
Between KMCs	KMAC exchange, update or revocation of KMAC between KMC domains
Between KMC and train or RBC	KMAC installation or deletion in an RBC or train by the home KMC.
Between train and RBC	Secure communication during a mission. A session is established using the KMAC for the corresponding combination.

Euroradio The Euroradio protocol sits in between the communication layer GSM-R and the application layer protocol. The protocol authenticates messages between a train and an RBC using a session key KSMAC derived from KMAC and nonces[3] exchanged between the parties [15]. Each message is authenticated using CBC-MAC and DES with an additional triple-DES round at the end. An attack against this protocol is presented in [9]

2.7.1. Key Management

The generalized key distribution procedure between two KMC has summarized in Figure 2.4. A train and an RBC combination is *created* when a Railway Operator KMC requests a KMAC for a respective pair for the first time. The request is made to an Infrastructure Manager KMC in charge of a trackside entity, which upon acceptance following internal procedures, generates a KMAC for the requested combination. The key is subsequently shared with the Railway Operator KMC. Both KMC will load the KMAC into their respective entities, followed by an operation test, where the train establishes a session with the trackside entity using the corresponding KMAC for the first time [16]. Three types of interactions can be distinguished between ERTMS entities; these are described in Table 2.1. The first and third step in the key management procedure correspond to a KMC - KMC interaction. The ERTMS specifications prescribes two methods of interaction: offline and online.

Offline KM Secure KMAC exchange is achieved by exchanging beforehand a long-term symmetric key K-KMC between two KMC. The K-KMC is used to mutually authenticate both parties and to encrypt the shared KMAC when a request is approved. The encrypted KMAC is shared using offline storage devices such as optical compact discs or USB flash drives.

Online KM Specified in 2015 to circumvent the shortcomings of the offline method. The Transport Layer Security (TLS) protocol using a PKI is prescribed to provide confidentiality, authenticity and integrity of the distributed KMAC between KMC's. A TLS-Public Key Infrastructure (PKI) structure relies on digital certificates managed and distributed by a Certificate Authority (CA). Each KMC is responsible for setting up its own PKI and establish how other KMC's must interact with it [46]. No KMC supports online key management to date.

NIST will deprecate the triple-DES algorithm from 2023 onwards. It is expected that the Euroradio protocol will still use triple-DES keys after 2023, as the cost of replacing equipment plays a major role when deciding to migrate to newer technology. Updating the KMAC more regularly will likely become more critical to compensate for the security vulnerability. This update interval is still a subject of debate. NIST recommends not processing more than 2^{20} 64 bits of data using a single triple-DES key; therefore, a KMAC should be replaced before 349 523 sessions.

3
Previous work

This chapter reviews previous work and concepts related to key management. Section 3.1 summarises specific work addressing ERTMS key management and discusses their shortcomings. Section 3.2 gives an outline of authentication methods used nowadays, a crucial requirement for ERTMS, with a focus on decentralised approaches. Section 3.3 presents techniques used in Blockchain based applications to achieve anonymity.

3.1. Existing ERTMS key management proposals

An universal key management scheme for ERTMS is proposed by Thomas et al. [53] under the name TRAKS (Train and RBC Authenticated Key Scheme). TRAKS introduces a key generation algorithm and a new layer between the Infrastructure Manager KMC and the RBCs, see Figure 3.1. The new layer, NID_C, represents a specific region or line and is assigned a number of RBCs to it. A secret is generated for each NID_C by the Infrastructure Manager, which is used to generate a key loaded into each RBC under the corresponding NIC_C. A `KMAC` for an RBC and train pair is derived from the RBC key together with the train identifier. The `KMAC` is loaded into the train using the standard offline or online methods. When a train requests a session with a corresponding RBC, the RBC is able to derive the `KMAC` loaded into the train at runtime using the train identifier and its RBC secret. A Railway Operator does not gain a operational benefit from implementing TRAKS, as the advantage is mainly found on the RBC.

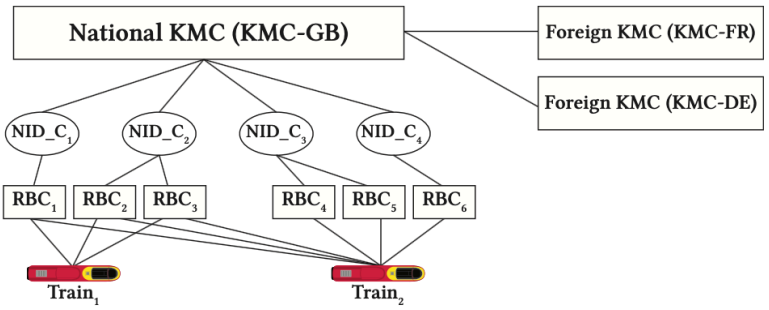


Figure 3.1: TRAKS introduces a layer NIC_C between the (National) Infrastructure Manager KMC and the RBC [53]

Franeková et al. [19] published in 2015 an online key management system based on asymmetric cryptography specifically for ERTMS. The work does not provide clear details about the actual implementation and seem to imply a centralised approach.

3.2. Decentralised Authentication Techniques

A core requirement of a key management system is to provide authentication. Authentication refers to providing assurance of an entity's identity and verifying that data has not been modified [3]. Decentralised approaches are reviewed and discussed for the design of a single ERTMS key distribution and management system, as centralised methods are not feasible within the European Union [46]. Three decentralised authentication frameworks stand out: Log-based PKI's, the Web of Trust (WoT) and blockchain-based approaches.

A popular and common authentication technique consists of a PKI using a certificate authority (CA). Users trust[4] one or more (root) certificates, and identities are bound to a public key using X.509 certificates that can be traced back to a CA. These CAs are considered to be a single point of failure. An additional drawback of traditional CA is the lack of transparency, as it is not possible to tell which entities have obtained a certificate from a given CA. This problem became evident when certificates were issued on behalf of a compromised Dutch CA DigiNotar in 2011. Log-based PKI aims to create public logs to monitor CA activities overcome the vulnerabilities and limitations of the traditional PKI. However, these log-based proposals have not gained widespread adoption as they still have certain shortcomings during the deployment process [33] and are vulnerable to split-world attacks. A split world attack consists of presenting a different view of the correct logs to a user [29]. Google's Certificate Transparency is an example of a Log-based PKI [58].

The Web of Trust has proven to be a clever decentralised solution to the authentication problem but has some awkward drawbacks [20, 58]. For example, a new entity can only join the network after being invited by another entity already on the system, creating a high entry barrier.

The tamper-proof property of a blockchain has made it an appealing foundation for various use cases, such as notarisation, accounting and auditing. Several authentications have also been proposed using a blockchain as a basis. For example, CertLedger is a blockchain-based authentication system that prevents split-world attacks and ensures transparency [29]. Authentication systems compatible with traditional X.509 certificates and relying on a blockchain are proposed in [55] and [58]. Hammi et al. [24] present a blockchain-based PKI architecture using bloom filters. These works address the authentication problem in a decentralised and transparent way but do not provide any additional key management functionalities.

Hyperledger Indy provides a distributed-ledger-based foundation to facilitate Self-Sovereign Identities (SSI) in combination with a blockchain. SSI is a proposal to allow users to manage their identities and how they disclose personal information to others. A User or identity holder can request certain credentials from an organisation (Issuer), which can be used with other credentials to verify itself to other organisations (Verifier) as needed. This User, Issuer, and Verifier combination is known as *the trust triangle* [44]. SSI's main advantage is to allow a user to have several different types of identities and/or credentials and to choose exactly which information to disclose to whom.

Blockchain popularity gave rise to the development of platforms to create and deploy industry-grade blockchain-based applications and services. Corda from R3 [5] and Hyperledger Fabric from the Linux Foundation [2] are two well-known open-source platforms. The platforms often use the term distributed ledger to refer to a blockchain. Both platforms allow the execution of arbitrary and programmable logic embedded in transactions, known as *smart contracts*. Being a platform, Corda nor Fabric provide any *plug-and-play* decentralised key management system fulfilling Section 1.1 requirements.

[4] For example, a list of trusted root certificates used by Apple devices can be found here: https://support.apple.com/en-bh/HT212140

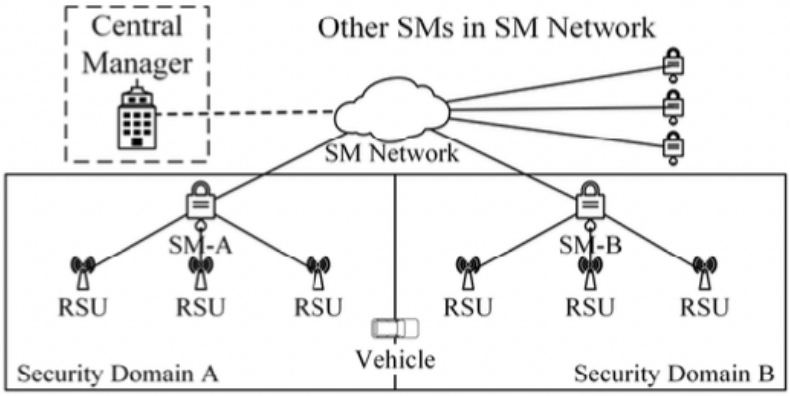


Figure 3.2: A decentralised secure vehicular communications key distribution system based on blockchain. A secure vehicular communications system consists of vehicles that communicate with Road Side Units (RSU). RSU's belong to a security domain. Key distribution procedures are triggered when there is a vehicle handover between RSU's. Special handover procedures take place when the handover takes place between RSU's from different security domains 3.2.

A secure vehicular communications system that resembles the ERTMS architecture is presented in [43]. This system consists of vehicles, just like trains, that communicate with Road-Side infrastructure Units (RSU) build along the road. As set of RSU belong to the domain of a Security Manager (SM). When a vehicle transitions from one security domain to another other, a key exchange procedure with several handshakes takes place involving a central authority. To make the key exchange procedure from the above structure more efficient, [30] proposes a blockchain system (SM Network) to replace the key management tasks of the central authority, see 3.2 for a graphical representation. A Central Manager is appointed for registration tasks. The design make use of proof-of-work as consensus mechanism and activities between the security managers are explicitly stored on the blockchain. The results using the blockchain approach showed that the handover procedure is more efficient under certain circumstances, i.e., take less time.

Decentralised key management systems have been proposed for different Internet-of-Things (IoT) applications [23, 38]. Uses cases for these systems approximate the ERTMS structure, but do not explicitly hide relationships between actors.

3.3. Blockchain Anonymisation Techniques

This section reviews state of the art anonymisation techniques used in blockchain applications where privacy is required. For some techniques, the protocol is explicitly presented to illustrate the interaction between the involved entities.

Private channels To exchange sensitive and private information between a group, Hyperledger Fabric offers private channels. A private channel is, in essence, a separate blockchain only available to specific members, completely separated from other blockchains (or channels). From the railway perspective, this would mean setting up a private channel for each KMC pair, i.e., a group of two. Private channels are meant for larger groups, and from Fabric's documentation, each separate channel introduces administrative overhead. Fabric also offers *private data collections* where a hash of the private data is distributed across all peers on the blockchain as evidence of the transaction and is used for validation and audit purposes. The actual confidential data is only shared with authorised organisations but before setting up so-called anchor peers. The drawback of these two concepts in Fabric is that the actual data is only stored in the authorised peers, eliminating redundancy in case of node failure or crash. Corda does not offer standalone privacy features and a choice needs to be made between privacy

and security [26, 34].

Coin mixers and tumblers The relationship between the issuer and the receiver of a transaction can be hidden by first sending the transaction to a third party known as a mixer or tumbler. The mixer will subsequently send the amount to the receiver in exchange for a fee. The mixer is aware of the relationship, and the users have to trust the mixer will not expose them. As more transactions are sent through the mixer, the less likely a relationship can be deduced. It is common in cryptocurrencies with construction to only allow transactions with fixed-value denominations. Cryptocurrency Dash relies on a set of mixers to achieve anonymity, where every node on the Dash blockchain can become a mixer after having stored a certain amount of value. These special nodes are called *master* nodes [12]. The idea is that a set of master nodes reduces centralisation; however, the number of master nodes in practice is much smaller than the coin users.

Blind signatures The Okamoto-Schnorr's blind signature is a modified version of Schnorr's signature scheme proposed by Okamoto et al. [41]. The scheme provides a signature from an authority for a message requested by a certain user, blinding the user from the other users. The scheme can be used in blockchain applications to hide the identity of the transaction issuers. In the case of a private blockchain, the blinding process can be performed by the same authority in charge of the enrolment process, such as in the electronic voting system presented in [27]. The drawback is the centralisation introduced by the authority. The scheme where Bob obtains a signature Bob for a message m from an Authority is summarised in Protocol 3.1, where the Authority has a private key s and corresponding public key $Y = sG$. The resulting signature for message m is (ρ, ϵ) and can be verified by other users computing $\alpha' = \rho G + \epsilon Y$ and checking if $\epsilon = \mathcal{H}(m||\alpha')$ holds. An honest prover will succeed in constructing a valid proof since to following holds:

$$\begin{aligned}\alpha' = \rho G + \epsilon Y = (r - \beta)G + \epsilon Y &= (v - es - \beta)G + \epsilon Y \\ &= vG - esG - \beta G + \epsilon Y \\ &= A - \beta G + (\epsilon - e)Y \\ &= A - \beta G + \delta Y\end{aligned}$$

Authority (Y, s)		**Bob** (Y)
$v \in_R \mod q$		
$A = vG$		
	$\xrightarrow{A}$	
		$(\beta, \delta) \in_R \mod q$
		$\alpha = A - \beta G + \delta Y$
		$\epsilon = \mathcal{H}(m\|\|\alpha)$
		$e = \epsilon - \delta \mod q$
	$\xleftarrow{e}$	
$r = v + es \mod q$		
	$\xrightarrow{r}$	
		$\rho = r - \beta \mod q$

Protocol 3.1. Okamoto-Schnorr Blind Signature

Hierarchical Deterministic (HD) address A private key k is extended using another random number c known as chain code [1]. From this extended private key, multiple child public/private key pairs can be generated that are not linkable to each other using Equation 3.1, where k_i is the ith private child key and kG parent public key. Multiple grandchild keys can be generated from each child key, again not linkable to each other. A child key cannot be used to derive the parent private/public key, hence hierarchical. This process can be repeated almost indefinitely, creating a tree structure. This allows users to use a different public/private key for each transaction. A disadvantage is that a user needs a secure channel to share the key with each sender.

$$k_i = k + \mathcal{H}(c \parallel kG \parallel i) \tag{3.1}$$

Linkable ring signatures A linkable spontaneously anonymous group (LSAG) signature scheme achieves anonymity, linkability, and spontaneity [32]. 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. The RingCT protocol, used by cryptocurrency Monero to achieve anonymous transactions, is based on the variant of LSAG. RingCT uses a mixin[5] count to hide the actual input being spent in a transaction. An empirical study concluded that 63% of Monero transactions with a mixin count of one or more could be traced back to the issuer [36]. The issue is primarily due to inputs used that were generated before the RingCT protocol became effective; transactions using the RingCT protocol are considered untraceable. Currently, Monero enforces a minimum mixin count of 4.

Stealth address Stealth addresses are used to anonymise both transaction issuer and receiver [54]. 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 incoming 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 $x = \mathcal{H}(aR) + b$. These scheme achieves *untraceability*, i.e., for each incoming transaction, all possible senders are equiprobable; and provides *unlinkability*, making impossible to prove for any two outgoing transactions they were sent to the same person. Protocol 3.2 illustrates a procedure where Alice constructs a stealth address with Bob's public spend and view key. Bob check the result and can be certain that he is the receiver. Bob is not able to deduce from the result alone that Alice generated the transaction.

Alice (A, B, G)		**Bob** (a, b, A, B, G)
$r \in_R \mod q$		
$R = rG$		
$P = \mathcal{H}(rA) + B$		
	$\xrightarrow{(R,P)}$	
		$P \stackrel{?}{=} \mathcal{H}(aR)G + B$
		$x = \mathcal{H}(aR) + b$

Protocol 3.2. Stealth address system

[5] Not to be confused with a mixer or tumblers. There are discussions in the Monero community about using another term for this parameter.

Set Membership Different types of uses cases requiring anonymity, ranging from cryptocurrencies to credential systems, can be described using the same generic statement: given a set y, prove that a certain element belongs to the set without revealing the actual element, i.e., a proof of membership. Commitment schemes based on ZKP are used to solve these kinds of problems. The size of these schemes usually increases linearly as a function of the set size y, which can become a problem in certain use cases. Camenisch, Chaabouni and Shela [7] have proposed an efficient protocol to prove set membership resulting in a constant size proof regardless of the set size. The protocol assumes that the set y is known to all participants. The scheme does not take linkability into account.

3.4. Discussion

Previous work addressing ERTMS key management either proposed a centralised solution, or focused on improving key distribution at the Infrastructure Manager side [19, 53]. These works do not propose a single nor decentralised system. A review of existing decentralised systems from similar fields, such as vehicular communication systems and IoT, concluded that there is no single decentralised system addressing the structure needed by ERTMS.

Platforms on which industry-grade decentralised applications are build on, such as Corda and Fabric, do not provide a key management system directly fulfilling the requirements summarised in Section 1.1 and are not suitable for private relationships between several small groups of participants. These platforms do provide a solid foundation on which a decentralised system can be implemented once designed.

The study of existing key management and authentication systems showed that blockchain technology is a suitable solution for building a single and decentralised system. Moreover, the append-only property allows the system to comply with the requirement of archiving `KMAC`'s and associated transaction data. Techniques to achieve anonymity in blockchain applications have been reviewed and summarized in Table **??** using three criteria: anonymity, authority, traceability. Partial anonymity means that anonymity is achieved from a subset, authority indicates the need of a trusted third party and linkability indicates if two or more transactions can be traced back to the same entity (not necessarily revealing the identity of the entity).

The study of anonymity techniques raised awareness that achieving privacy and security at the same time is not trivial. Systems that address privacy and security simultaneously usually achieved this at the cost of, for example, computational or storage complexity. Security, loosely speaking, aims to prevent system misuse or make it inaccessible from attacks such as Denial of Service [28]. The definition of anonymity and privacy is not so straightforward. Krishnamurthy [28], state that privacy is about keeping certain *information* from leaving a system inadvertently while anonymity is about performing some action without revealing some identity. Without loss of generality, this work will use both terms interchangeable to address both requirements.

The ownership and registration of assets, i.e. trains and RBCs, is generally public information. The purchase or lease of these objects often goes through open and public tenders. Moreover, upon inquiring a railway operator, it might be desirable to have a single asset directory where assets are listed together with their home KMC. Currently, it is sometimes *a bit of a puzzle* to figure out under which KMC domain an asset belongs. Therefore, asset ownership is not considered not private information, but the relationships between these entities will be considered private information during the design of a single decentralised system.

4

PRIDE: Privacy-Preserving Decentralised Key Management System

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 KMC (see Figure 2.4). 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. KMC use transactions to distribute KMACs for a particular train and RBC combination.

The design of PRIDE is presented in six parts: 1) Initialisation. 2) Hiding confidential information using AES. 3) Anonymous relationships using *tags* which at the same time authenticates a KMC. 4) Traceable relationships from the tag that ensures a tag can only have one valid KMAC at any time. 5) Transaction creation and verification. 6) Block and blockchain creation and verification. The relevant notation set for this work is summarised in Table 5.1.

Only registered actors are able to issue valid transactions and all system actors verify its authenticity without deducing any underlying relationships. Note that the tag allows all system actors to link transactions for a given combination but not all actors are not able to deduce the train or RBC that the tag identifies.

4.1. Initialisation

RC are established beforehand and grant KMC (administrators) access to the system. More than one RC can be appointed to avoid centralisation; for example, each Member State could designate a national RC. The reasoning behind this is based on the current specifications, which dictate that Member States are responsible for assigning unique KMC identifiers [14]. Each KMC has a unique identifier and generates a private and public key pair. The RC signs the public key creating a certificate as proof of admittance.

Table 4.1: Notation

Symbol	Description
$\mathcal{H}(\cdot)$	Secure hash function
q	A prime number
G	Elliptic curve base point, $G \in E(\mathbb{F}_q)$
$\mathcal{Y}$	Set of train and 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 entity a, $x_a \in \mathbb{Z}_q$
Y_a	Public key for entity a, $Y_a = x_a G$, $Y_a \in \mathcal{Y}$
K_{ab}	Tag for combination a and b
π	Index for combination a and b
k	Mixin count
u	Transaction one-time private key
sk_{KMC}	Private key of a KMC, $sk_{KMC} \in \mathbb{Z}_q$
pk_{KMC}	Public key of a KMC, $pk_{KMC} = sk_{KMC} \cdot G$

KMC Domain Once registered, a KMC creates a domain by registering assets into the system. A KMC generates a private and public key for each asset in its domain. Similar to the previous process, the public key and asset identifier are signed by the KMC, creating a certificate. All registered KMC establish a peer-to-peer network of nodes, and each node has a list of registered KMC and assets with their corresponding identifier, public key, and certificate. $\mathcal{Y}$ denotes the set of public keys of all registered asset available to all KMCs. Each KMC has one KMC private key and a set of asset private keys, one private key for each asset in its domain. All private keys must be backed up and kept safe.

The registration of trains and RBCs into the system will create an European ERTMS entity directory available to all registered KMC.

4.2. Hiding Confidential Information

Symmetric encryption allows both KMC to retrieve the KMAC from the same transaction. KMACs are encrypted using AES-GCM, and the resulting ciphertext, MAC and IV are included in the transaction payload. The ciphertext ensures confidentiality, whereas the MAC is used to check message 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. AES-GCM requires an Initialisation vector (IV) as an input, which must not be reused [13] as the ciphertexts of two different messages obtained from the same secret key and IV would reveal the secret key.

4.3. Anonymous Relationships

Let the set of trains $\mathcal{U} \subset \mathcal{Y}$ and the set of trackside entities $\mathcal{V} \subset \mathcal{Y}$, 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 combination is uniquely mapped to a tag K using a one way function, $f : (Y_i, Y_j) \rightarrow K$. The tag results from a Diffie-Hellman key exchange using Equation 4.1. The result, used as an identifier rather than a secret, is included in each transaction and has 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.

$$K_{ij} = x_i Y_j = x_j Y_b = K_{ji} \tag{4.1}$$

Apart from being an identifier, the tag is used to authenticate a transaction. A transaction is considered authentic if it can be proved to be issued by a registered KMC and for a combination of registered assets. The tag forms a DH tuple (G, Y_i, Y_j, K) for which a NIZKP proof as presented in Protocol 2.1 can be created. The result would convince all system actors that the tag is computed by a registered KMC using f but reveals the public keys involved, instantly revealing the relationship between the two KMC. To prove authenticity while remaining anonymous, Protocol 2.1 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 $\{(G, Y_i, Y_j, K) \mid Y_i \in \mathcal{U} \wedge Y_j \in \mathcal{V}\}$ satisfy Equation 4.1. The tag and proof generation for trackside entity a and train b is created using Algorithm 4.1. Any system actor can verify the authenticity of the tag by running Algorithm 4.2. When including a message m in the hash function, Algorithm 4.1 and 4.2 can be considered to be the generation and verification algorithms of a digital signature scheme for message m using one of the assets private keys. In fact, it can be argued that the result is already a signature for the tag.

Algorithm 4.1 Generation

G: Elliptic curve base point
function generateTag($a, b, \mathcal{U}, \mathcal{V}$)
  $K = x_a Y_b$ ▷ Tag computation
  $v \in_R \mathbb{Z}_q$
  $R, S, c, r = \{\}$
  $n = 0$
  $\pi = 0$
  for each $i : Y_i \in \mathcal{U}$ **do**
    for each $j : Y_j \in \mathcal{V}$ **do**
      $n = n + 1$
      if $i = a \wedge j = b$ **then** ▷ Same as Protocol 2.3
        $\pi = n$ ▷ Combination index
        $R_\pi = vG$
        $S_\pi = vY_b$
      else ▷ skipped if $k = 0$
        $c_n, r_n \in_R \mathbb{Z}_q$
        $R_n = r_n G + c_n Y_i$
        $S_n = r_n Y_j + c_n K$
      end if
    end for
  end for
  $s = \mathcal{H}(G \parallel R_1 \parallel S_1 \parallel ... \parallel R_n \parallel S_n \parallel K)$
  $c_\pi = s - \sum c \mod q$
  $r_\pi = v - x_a c_\pi$
  return $c, r, \mathcal{U}, \mathcal{V}, K$
end function

Algorithm 4.2 Verification

G: Elliptic curve base point
function verifyTag($c, r, \mathcal{U}, \mathcal{V}, K$)
  $R, S = \{\}$
  $n = 0$
  for each $i : Y_i \in \mathcal{U}$ **do**
    for each $j : Y_j \in \mathcal{V}$ **do**
      $n = n + 1$
      $R_n = r_n G + c_n Y_i$
      $S_n = r_n Y_j + c_n K$
    end for
  end for
  $s = \mathcal{H}(G \parallel R_1 \parallel S_1 \parallel ... \parallel R_n \parallel S_n \parallel K)$
  return $s \stackrel{?}{=} \sum c \mod q$
end function

Mixin Count The length 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 $k < |\mathcal{U} \times \mathcal{V}|$ and selecting k combinations from $\{(Y_i, Y_j) \mid Y_i \in \mathcal{U} \wedge Y_j \in \mathcal{V}\} \backslash (Y_a, Y_b)$. Let $\mathcal{S}$ be the set of k elements from a uniform random sample without replacement from $\{1, ..., |\mathcal{U} \times \mathcal{V}|\} \backslash \{\pi\}$, where π is the index of combination (Y_a, Y_b). The resulting ordered multi sets fed into Algorithm 4.1 and 4.2 are

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

and

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

The mixin count can be seen 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. $k = 0$ results in Protocol 2.3, instantly exposing the relationships.

4.4. Traceable Relationships

A train and RBC combination can only have one valid KMAC at any point in time. A modified stealth address scheme is used to achieve this while keeping the issuer KMC and receiver KMC of a KMAC transaction anonymous. For every transaction, an issuer looks up the public key pk_{KMC} of the receiver, generates $r, s \in_R \mathbb{Z}_q$ and computes $u = \mathcal{H}(r \cdot pk_{KMC}) + s$, see Algorithm 4.3. The resulting stealth address in the transaction is the tuple $(R, P = uG, S = sG)$. The receiver checks for every incoming transaction if $P = \mathcal{H}(sk_{KMC} \cdot R)G + S$ using Algorithm 4.4, where sk_{KMC} is his/her private key. If it holds, then he/she knows that the transaction is meant to him/her. The transaction private key u is only known to the issuer, which is used to update the KMAC from the corresponding transaction.

To update or renew a KMAC, a new KMAC transaction is issued disclosing the private key u of the KMAC to be replaced. KMACs from transactions with disclosed private key are considered to be revoked, ensuring that every asset combination has only one valid KMAC at any time. Transactions with undisclosed private keys are known as unspent transactions and the number of valid KMACs, valid tags and unspent transactions in the system is equal at any point in time. All peers in the network maintain a list of unspent transactions with the associated tag.

Algorithm 4.3 Stealth address generation

G: Elliptic curve base point
function generateStealth(pk_{KMC})
  $r, s \in_R \mathbb{Z}_q$
  $R = rG$
  $S = sG$
  $u = \mathcal{H}(r \cdot pk_{KMC}) + s$
  $P = uG$
  return (R, P, S)
end function

Algorithm 4.4 Stealth address check

G: Elliptic curve base point
sk_{KMC}: Private key
function checkStealth(R, P, S)
  return $P \stackrel{?}{=} \mathcal{H}(sk_{KMC} \cdot R)G + S$
end function

4.5. Transaction Creation & Validation

A transaction is a tuple containing the elements of Table 4.2. Date and time are included to check freshness and for bookkeeping. The hash of the transaction uniquely identifies a transaction in the system and ensures that the contents have not been altered. There are two types of transactions, original and update transactions:

Original Transaction The result from Algorithm 4.1 is attached 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. The transaction is validated using Algorithm 4.2. Even though two KMC can compute a tag for the same combination and create a valid proof, it is assumes that only the Infrastructure Manager KMC will issue transactions (see Figure 2.4).

Update Transaction KMAC transactions corresponding to known tags in the system are referred to as update transactions. An update transaction for a tag is considered authentic 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. Every node verifies its correctness and validity before adding it to its pool of unconfirmed transactions.

Table 4.2: Transaction structure

Item	Description
timestamp	Date and time of transaction creation
destination	Stealth address (see Section 4.4)
tag	Train and trackside combination identifier (see Section 4.3)
payload	Encrypted KMAC, MAC and IV (see Section 4.2)
proof	Result from tag Algorithm 4.1 or transaction private key u
hash	The digest of the transaction from a hash function

4.6. Blocks

Unconfirmed transactions are grouped together in a block by a *leader* node and proposed for acceptance by broadcasting it to the network. Besides a list of transactions, the block contains a block number, the hash of the latest accepted block, Merkle root from the transactions, issuer identifier and signature over the block's digest. Each node will verify the correctness and validity of the block and the included transactions. The 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.

Blockchain The first block is known as the *genesis* block, and it is agreed upon during the Initialisation phase. Leader election and block acceptance depend on the consensus protocol (see Section 2.6.1). PRIDE assumes that actors in the system may act maliciously, therefore IBPF as consensus algorithm, which is byzantine fault tolerant.

4.7. Discussion

Once PRIDE is setup, KMC are registered on it and have up their domains, `KMAC` exchange can take place. Figure 4.1 provides a sequence diagram involving two actors, Infrastructure Manager (IM KMC) and a Railway Operator (RO KMC); and the peer-to-peer network (P2P). The sequence illustrates the procedure for a `KMAC` of a new asset combination, where RO KMC requests a `KMAC` for the first time for train b under its domain and RBC a under the domain of IM KMC, whose identities are found on the asset directory.

RO KM computes the tag K_{ab} using Algorithm 4.1 (or just Equation 4.1) and sends this *securely off-chain* to the IM KMC. Upon receiving the request, verifying that the tag is correct and accepting the request following internal procedures, IM KMC creates *the* proof using Algorithm 4.1 followed by a stealth address using Algorithm 4.3 and RO KMC public key pk_{RO}. Both results are put together in a transaction with the additional items presented in Table 4.2. The `KMAC` in the transaction is encrypted using AES and the key resulting from a Diffie-Hellman key exchange between the actors using RO KMC public key pk_{RO} and IM KMC private key sk_{IM}. The unconfirmed transaction is send to the P2P network by IM KMC.

During the consensus protocol, all nodes in the P2P network execute Algorithm 4.2 to verify the transaction authenticity. If valid, the transactions, together with other transactions, are added to the blockchain in a new block. RO KMC (just as all other KMC) will scan all the transactions in the just accepted block using Algorithm 4.4 and its secret key sk_{RO} to check if one or more transactions belong to it. Algorithm 4.4 will result in `true` for a transaction with tag K_{ab}. RO KMC can decrypt the `KMAC` in the transaction using the resulting key from a Diffie-Hellman key exchange using its private key sk_{RO} and IM KMC public key pk_{IM}.

Both KMC subsequently load the `KMAC` into their assets using internal procedures followed by a an operation test, where the train establishes a session with the trackside entity using the corresponding `KMAC` for the first time.

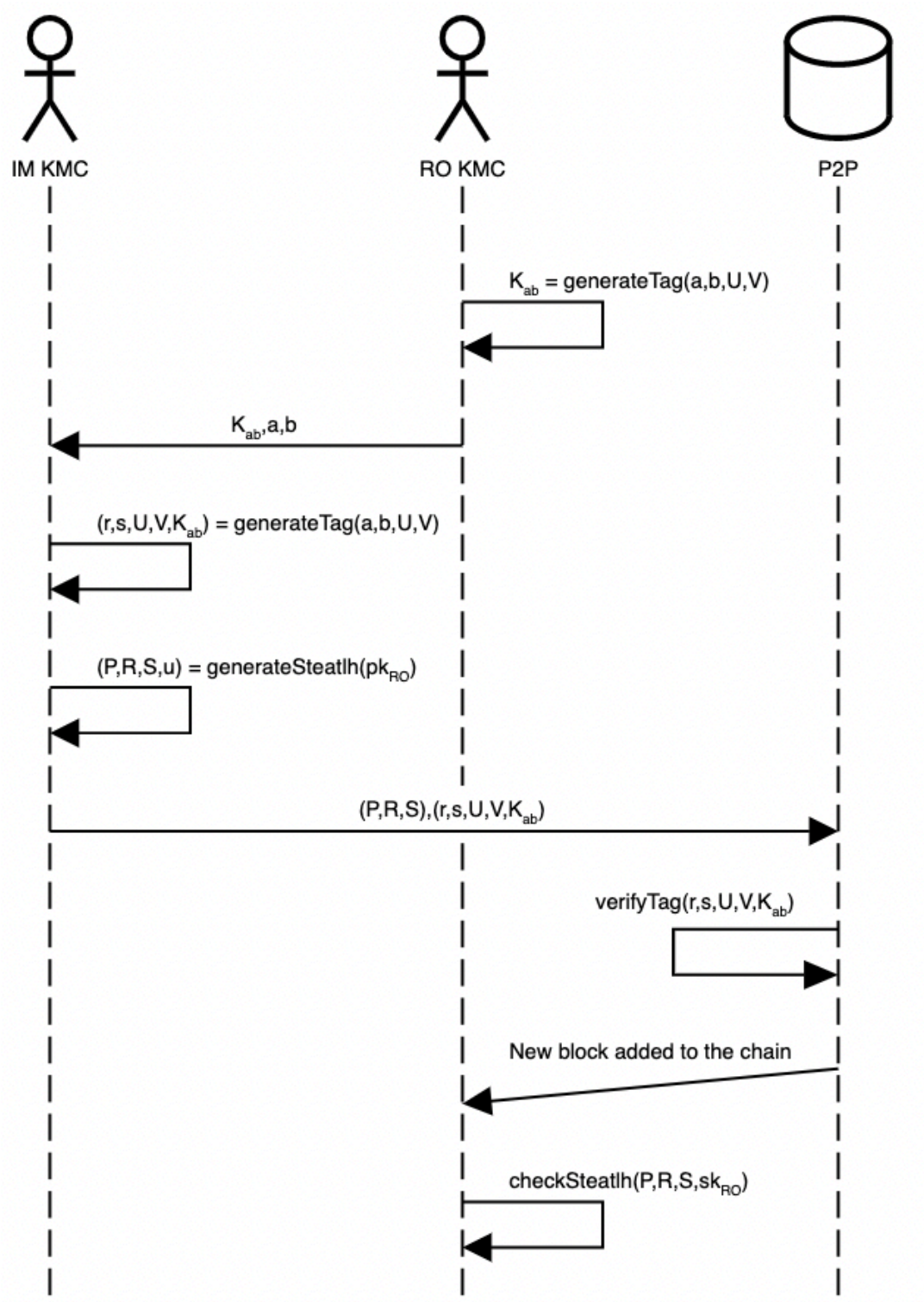


Figure 4.1: Sequence illustrating the procedure for a `KMAC` of a new asset combination,

5
Evaluation

This chapter evaluates PRIDE. Section 5.1 shows how PRIDE is analysed against the *semi honest* adversary model. Section **??** analyses the complexity of PRIDE from a computational, communication and space point of view. A proof of concept is built using Rust and Javascript to evaluate PRIDE's performance using settings close to the expected railway landscape in 2030. The results are presented in Section 5.3 and are used to discuss scalability.

5.1. Security

PRIDE is believed to be secure under the *semi honest* adversary model where an adversary controls one of the parties and follows the protocol precisely as specified. In the semi-honest model, an adversary tries to learn more information than it should form the system. A protocol secure under the presence of semi-honest adversaries is guaranteed not to leak inadvertent information [31], such as private keys. PRIDE relies on three protocols for authentication: 1) Anonymous relationships 2) Stealth addresses 3) ECDSA. The first two are on transactions, and the third one blocks.

5.1.1. Anonymous relationships

The tag that uniquely identifies a train and RBC combination is the result of a Diffie-Hellman key exchange. From Definition 2.3, given the public keys associated with the combination, the result is indistinguishable from a random value. Therefore, the tag alone does not revealing the entities involved nor their private keys.

The tag is verified by presenting the result of a witness indistinguishable protocol, generated by Algorithm 4.1 and verified using Algorithm 4.2. An honest prover using Algorithm 4.1 will always succeed in constructing a valid proof satisfying Algorithm 4.2, since correctness is trivial for R_i and S_i for $i \neq \pi$

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

and

$$\begin{aligned} S_\pi &= r_\pi Y_b + c_\pi K \\ &= (v - x_a c_\pi) Y_b + c_\pi K \\ &= v Y_b - c_\pi x_a Y_b + c_\pi K \\ &= v Y_b - c_\pi K + c_\pi K = v Y_b \end{aligned}$$

Without knowing the private key, a cheating prover will fail to convince the verifier. This can be proven by contradiction.

Lemma 1. *A probabilistic polynomial-time adversary cannot compute a valid proof accepted by Algorithm 4.2 without knowing one of the two asset's private key x.*

Proof. Assume that a cheating prover does not know one of the private keys x was able to compute a proof accepted by Algorithm 4.2. 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 $r_\pi G = R_\pi - c_\pi Y_a$, which would be essentially solving the discrete logarithm problem. □

A verifier cannot infer which index corresponds to π; therefore, our protocol is witness indistinguishable and no adversary can distinguish the asset combination involved with non-negligible better than $1/(k+1)$ probability.

5.1.2. Stealth address

A stealth address creates anonymous and unlinkable transactions. 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 herself and gives the actual receiver only *view* control. We refer the reader to [40, 54] for the formal proofs regarding anonymity and unlinkability. The security of the protocol relies on the transaction private key u.

Lemma 2. *A probabilistic polynomial-time adversary cannot derive the transaction private key u.*

Proof. Assume that an adversary obtains a stealth address (R, P, S) and the receiver's corresponding private key b. The adversary needs to find s such that $sG = P - \mathcal{H}(bR)G = S$, by doing so, it solves the discrete logarithm problem. □

5.1.3. ECDSA

ECDSA is used to authenticate a KMC proposing and accepting blocks. The hash of a block is signed using Algorithm 2.1. The result is verified using Algorithm 2.2 and ensures that only a registered KMC proposes a block. There is no formal security proof for DSA, and its elliptic curve variant [18], but it has proven to be secure empirically in applications such as Bitcoin. However, it is crucial to use a cryptographically secure pseudorandom number generator and properly implement the algorithm.

5.2. Complexity Analysis

The complexity analysis is based on 256-bit EC cryptography which provides $\kappa = 128$ bits of security. A security strength of 128 bits is believed to provide acceptable security for protection and processing information until 2030 and beyond [3]. Furthermore, Koblitz curve secp256k1 is used together with hash function SHA-256 and cipher block AES-256 using a 96-bit IV.

Table 5.1: Notation used in Evaluation chapter

Symbol	Description
κ	security parameter in bits
n	EC private key size in bits, 2κ
P	Compressed EC point size in bits, $n+1$
$\mathcal{H}(\cdot)$	Secure hash function
G	Elliptic curve base point
IV	Initialisation vector
MAC	Message authentication code
N	Number of nodes
f	Acceptable faulty nodes
k	Mixin count
l	Transactions per block

5.2.1. Computational Complexity

The operations involved in transaction creation and verification are summarized in Table 5.2. The generation and verification time is linearly bounded by the mixin count k. Each k introduces 4 EC point multiplications and 2 EC point additions. The verification protocol has two additional EC point multiplications and two additional EC point additions, regardless of k.

Table 5.2: Number of operations of Algorithms

Operation	Algorithm 4.1	Algorithm 4.2	Algorithm 4.3	Algorithm 4.4
EC point multiplication	$2+4k$	$4+4k$	4	2
EC point addition	$2k$	$2+2k$	0	1
Modular addition	$k+1$	k	0	0
Modular multiplication	k	0	1	0

5.2.2. Communication Complexity

The communication between the nodes depends on the consensus protocol. IBFT is used in PRIDE, introducing a communication complexity of $\mathcal{O}(N^2)$. With 100 nodes, the system can withstand $f = 33$ faulty nodes (see Equation 2.5). During each round, the total maximum and the minimum number of messages per phase are shown in Table 5.3.

Table 5.3: Number of messages during a consensus round

Phase	Maximum	Minimum
Pre-prepare	$3f$	$3f$
Prepare	$(3f)^2$	$(3f)(3f-f)$
Commit	$3f(3f-1)$	$(3f-f+1)(3f+1)$

5.2.3. Space Complexity

A transaction consists of the items presented in Table 4.2 and the corresponding size in bits is shown in 5.4. We assume that identities are stored in 32-bit variables.

Table 5.4: Transaction size per item

Item	size (bits)
timestamp	32
destination	$3 \cdot P = 3 \cdot (n + 1)$
tag	$P = n + 1$
payload	$n + IV + MAC = n + 96 + 128$
proof	$2 \cdot k \cdot (n + 32) \vee n$
hash	n

Table 5.5: Block size per item

Item	size (bits)
timestamp	32
issuer	32
signature	$2n$
hash	n
Merkle root	n
Transaction list	$l \cdot (1765 + k \cdot 576)$
Commit list	$\lceil (2N - 1)/2 \rceil \cdot (2n + 32)$

With $n = 256$ and summing the elements together results in original transaction size of $1796 + k \cdot 578$ bits, where the mixin count linearly bounds the transaction size. For an update transaction, the tag signature is replaced by a n-bit private key u, resulting in constant transaction size of 2052-bit.

A block consists of the items presented in Table 5.5 with corresponding sizes in bits. The identity of the issuer always accompanies each signature. Additionally, IBFT requires that each block has a fixed number of $\lceil (2N - 1)/2 \rceil$ commit signatures, where N is the number of nodes. A block with no transactions nor commitments results in 1376 bits. 77 commitments is about 4.5 kilobytes[6]. In practice, we expect trains to have 200 KMAC loaded on-board on average. Figure 5.1 depicts the blockchain size in gigabytes[7] maintained by $N = 100$ nodes after 7.7 million original transactions for different numbers of transactions per block l and transaction size k.

[6] 1 kilobyte = 1000 bytes, 1 byte = 8 bits
[7] 1 gigabyte = 1000^3 bytes

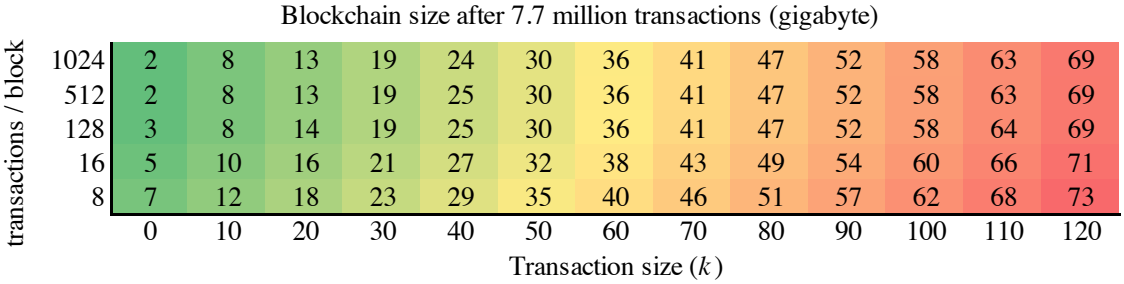


Figure 5.1: Blockchain size in gigabytes after 7.7 million original transactions as a function of transactions per block and mixin count.

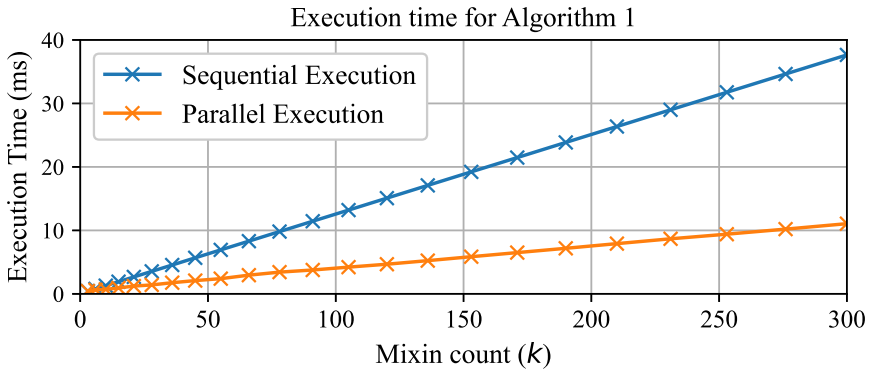


Figure 5.2: Sequential and parallel execution times for Algorithm 4.1. The execution time increases linearly as a function of the mixin count, as expected.

5.3. Performance Analysis

Performance is measured by implementing PRIDE as a proof of concept. The results are presented in the section.

5.3.1. Transactions

The complexity of creating and verifying a transaction is dominated by Algorithm 4.1 and 4.2. These two algorithms were implemented in the Rust programming language. The 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 5.2. Both protocols are suited to be executed in parallel. Using the same machine, the execution time improved a factor of 4.2; results are shown in orange in the same figure. As expected, the execution time increased linearly a function of the mixin count. The average generation time for a tag signature with $k = 120$ resulted in 4674μ seconds.

5.3.2. Blockchain

The whole design of PRIDE was implemented in Node.js, simulating up to 100 fully connected nodes spread over five different machines geographically separated over three Dutch cities: Delft, Rotterdam, and Gouda. Each machine is assigned a Fully Qualified Domain Name (FQDN); see Table 5.6. One RC was established by generating a private key and root certificate under the name `RC NL Root CA` using OpenSSL and installed in each machine. Certificates are generated for each FQDN simulating the registration process. The nodes communicate using WebSockets over TLS, ensuring confidential data transfer between nodes. Each node is identified by a number $i \in 1, ..., N$ and listens to TCP port $8000 + i$.

Table 5.6: Proof-of-concept network setup

IP	FQDN	Location	CPU	Operating system
86.93.162.125	gd.wlkn.nl	Gouda	Intel Core i5-4570	Ubuntu 20.04.3 LTS
178.84.070.143	rtd.wlkn.nl	Rotterdam	Intel Core i5-4570	Ubuntu 20.04.3 LTS
136.144.208.161	dt.wlkn.nl	Delft	Intel Xeon[8]	Debian 10
178.84.70.143	ut.wlkn.nl	Rotterdam	Intel Core i5-4570	Ubuntu 20.04.3 LTS
178.84.70.143	rtd.wlkn.nl	Rotterdam	Apple M1	macOS 11.6

[8] BladeVPS PureSSD X4 from TransIP

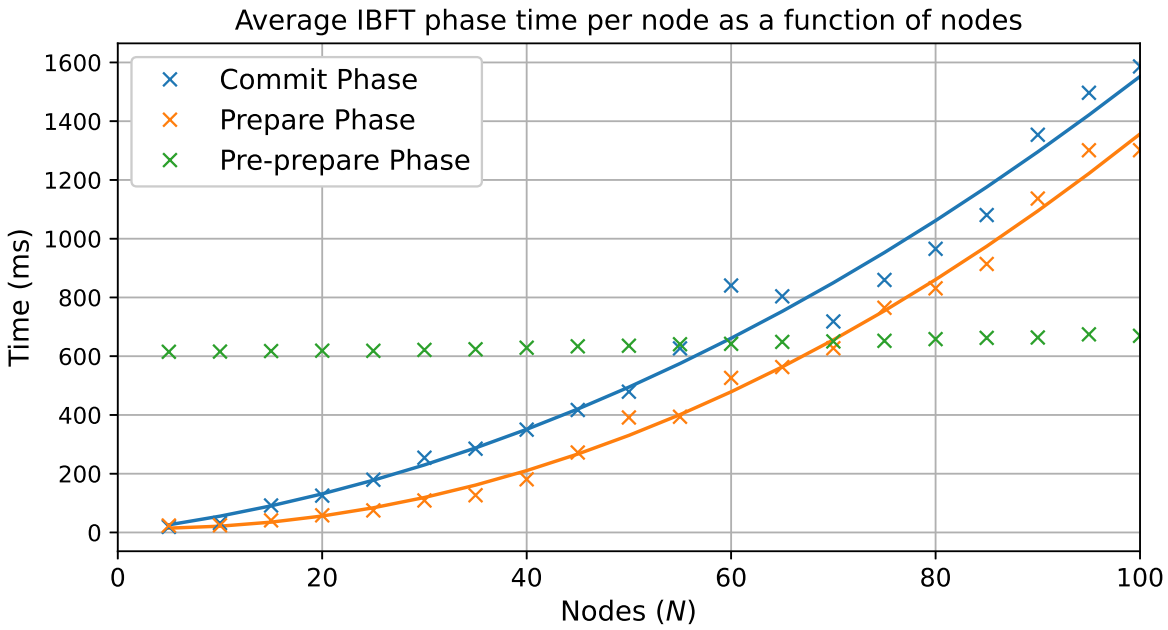


Figure 5.3: Average pre-prepare, prepare and commit time as a function nodes N. The solid line is the result of a quadratic regression.

The proof-of-concept is aimed to measure throughout as a function of communication overhead. Leader election and view change are not considered in the implementation. For each measurement, a fixed leader generates 120 transactions per block and proposes 20 blocks. The number of nodes was increased from 5, in steps of 5, until 100 nodes. Because the number of available processing units was smaller than the simulated nodes, the processing time for Algorithm 4.1 and 4.2 were simulated to avoid biased results from CPU time sharing. Using a timer and the results from Part 5.3.1, a time-out of $5ms$ per transaction was programmed simulating the generation and verification of transactions with $k = 120$.

Each replica node starts measuring time as soon as it receives a pre-prepared message; that is, time measurements are relative to each node. The results are presented in Figure 5.3. The pre-prepare phase showed a constant pre-prepare time of $120 \cdot 5ms \approx 600ms$ as a function of the number of nodes, as expected. Both prepared and commit time showed an exponential increase as a function of nodes, as expected from the number of messages exchanged from Table 5.3. With 100 nodes, on average after 20 blocks with each containing 120 transactions, all replica nodes reached the *prepared* phase after 1,3 seconds and the *commmited* phase after 1,6 seconds. The distribution of the prepared and commit time of the replicas averaged over 20 blocks is presented in Figure C.1 and Figure C.2, respectively.

The time between the leader node starts the execution and *all* nodes are committed is presented in Figure 5.4. The time includes transaction generation, and waiting for all nodes to commit. The result as function of nodes does not show a *nice* quadratic behaviour as the replicas. This could be attributed to network delays, but this needs further investigation. Overall, the maximum time is 8 seconds, which equal to 15 transactions per second. Note that time was measured until the last node committed. In practice, it suffices to wait for $N - f$ to start a new round.

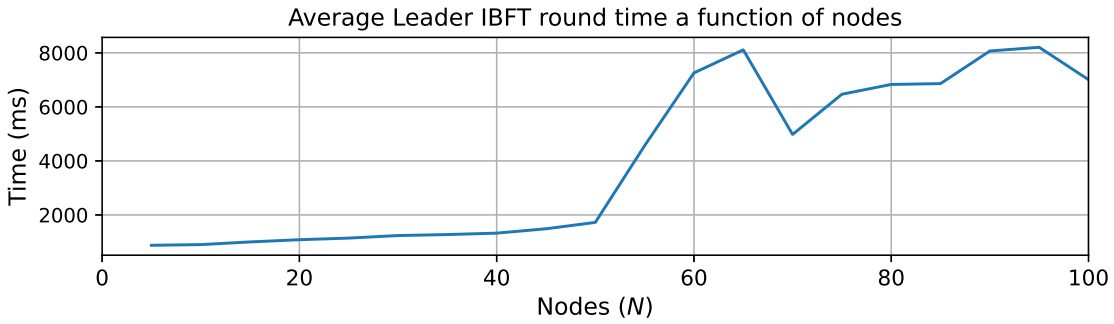


Figure 5.4: Average leader round time as a function of nodes N

5.3.3. Scalability analysis

PRIDE running on a P2P network of one hundred nodes achieves a throughput of at least 15 transactions per second using a mixin count of 120. From each original transaction, the chance of correctly guessing the involved train and RBC is roughly 0.8%. The throughput presents a significant improvement over the current off-line method, where the exchange of a KMAC can takes days, or even weeks. The storage need for all 7.7 million KMAC original transactions is about 70 gigabytes. The price for professional Solid State Drives is around 1 Euro per gigabyte at the moment of writing; storage should therefore not be a, considering that update transactions take significantly less space, and at some point in time, transactions can be discarded. The KMAC update interval is still a subject of debate.

From the result presented in Figure 5.3, it is interesting to see that, from starting from 40 nodes, the prepare and commit phase times start to dominate the consensus time. Therefore, it makes sense to bundle as many transactions as possible into one block, reducing the size of the blockchain due to the additional information each block contains. A larger block size equals a smaller blockchain but comes at the price of a longer average confirmation time. Specifying a maximum block time puts abound the maximum confirmation time. The pre-prepare phase time can be reduced by using a more powerful computer and with more processing logic units, as Algorithm 4.1 and 4.2 are suited to be executed in parallel.

6

Discussion and Future Work

The European railway signalling system consists of a wireless communication channel between a train and an RBC. A session between the two of these entities is established using a pre-installed symmetric key KMAC, unique per combination and used for mutual authentication. The train provides periodic location reports during a session, and the RBC provides Movement Authorities. Trains and RBC are generally under the responsibility of different KMCs, creating a KMAC distribution problem between them. There is currently no single system to be used by all KMC, and studies have concluded that a centralised approach is not feasible since it is difficult to place the responsibility for such a single system with one organisation [46]. This thesis reviews previous work addressing the ERTMS key distribution problem and the current key distribution and management approaches used in similar fields such as automotive and IoT applications. This showed that, to the best of our knowledge, there is no single system that can be directly adopted by all ERTMS KMC for key distribution and management. Moreover, most work either makes use of central entity or makes a trade-off between security and anonymity.

During the design of a single system aimed to be used by all KMC, privacy turned out to be an underexposed aspect. At the moment, the existing implementations and procedure are generally only between two KMC; therefore, privacy has not been an issue. Railway operators, especially freight operators, benefit the most from having as much KMAC as possible loaded into their trains and every additional KMAC installed in their assets can be translated to a potential competitive advantage over other operators. State-of-the-art anonymity techniques in blockchain applications have been studied.

The main research question on which this work is based is as follows:

"How can we design a scalable and decentralised key distribution system for the European railway signalling system?"

This chapter discusses how PRIDE achieves the research goal, examines its limitations, and presents how these can be addressed in future work.

6.1. Discussion

This work presents PRIDE, a decentralised key management system for railway KMCs. The system is built around a permissioned blockchain and allows KMCs to exchange KMAC for entity pairs in a confidential, integer, and authentic way using transactions. Besides a registration centre, no other third parties are involved and KMC can spontaneously establish a relationship. KMCs maintain a p2p network of nodes on which the blockchain is hosted and are jointly responsible for the system. The advantages of a decentralised system become more noticeable as more entities start making of the system.

The design has been analysed from a security and performance point of view. The security of PRIDE relies on the assumption that solving the discrete logarithm problem is hard as long as the private keys are not compromised. Our security analysis shows that PRIDE is secure on the semi-honest model. The performance analysis presented in Section 5.2 show that the generation and verification of original transactions have linear communication complexity as a function of the mixin count. PRIDE is practical and scalable at the same time, which is proved from results obtained from a proof of concept simulating up to one hundred KMCs achieving a throughput of at least 15 transactions per second.

Key update and revocation To address KMAC update and revocation, PRIDE uses the unspent transaction output paradigm found in cryptocurrencies. For every train and RBC pair, there can only be one valid KMAC at any point in time. A KMAC is updated or revoked when a new transaction is issued revealing the private key of the previous KMAC, creating a chain of KMAC transactions for a given pair where only the *newest* transaction is considered to contain a valid KMAC. This construction is built upon stealth addresses that ensure a transaction does not disclose issuer and receiver to the system. This answers the first research sub-question *how can we address key update and revocation?*

Auditability All transaction data is recorded on the blockchain, and each transaction corresponding to an asset combination includes a tag linking all transactions that have been issued for the combination. Even a bit flip anywhere on the blockchain will be noticeable, ensuring that a transaction cannot be changed or deleted once published on the blockchain. Together with the digital signatures, the append-only property and the tags, make PRIDE an excellent tool for audits and dispute resolutions. Using Protocol 2.3, a KMC can prove ownership of a specific tag without revealing any other relationship to an auditor. This answers the second research sub-question *how can we provide key auditability and traceability?*

Anonymity The stealth address approach used to link transactions does not reveal the sender and the receiver. The tag in each transaction identifying a train and RBC pair does not disclose the entities involved under the decisional Diffie-Hellman assumption. Nevertheless, the stealth address and tag alone do not ensure that transactions are issued by a valid KMC nor provide any security against, for example, denial-of-service attacks. A signature scheme is built to prove the tag's validity to address this issue. The home KMC assigns each asset a private and public key pair, and the tag results from a Diffie-Hellam key exchange. Based on a witness indistinguishable proof, it can be shown that the tag is the result from 1-out-of-k train and RBC combinations, without revealing the actual combination. The proof result is attached on the first transactions for a given train and RBC combination. The space and computational complexity of the proof increase linearly with k, creating a trade-off between anonymity and performance. This approach does not provide perfect anonymity. The scheme does not give *exculpability*; that is, as other KMC disclose their private keys, it is possible to determine the entities involved by means of elimination. This addresses the third research sub-question *how can we anonymise relationships and transactions between KMC?*

Side-channel attacks It is important to note that anonymity can be compromised even if the protocols are followed correctly as designed. For example, Infrastructure Managers updating KMAC at peculiar times or using obvious refresh intervals could reveal the identify of certain organization. Specifying an equal update interval and KMAC lifespans would reduce this risk. Also, the same mixin count used to achieve anonymity can backfire it, as using a disproportionately large value compared to other transactions can give away the entities involved, as addressed by Möser et al. [36] in their empirical study of Monero transactions. This flaw is can be addressed by enforcing the same mixin count for all transactions.

6.2. Future Work

Asset ownership The current design of PRIDE assumes that KMC will honestly register RBCs and trains. PRIDE can be extended to support ERTMS equipment vendors to account for this and prevent misuse. Vendors, such as Siemens, Alstom, and CAF, would register assets and transfer ownership to KMC using transactions recorded on the PRIDE blockchain. Transferring ownership is also desirable if an asset is sold or leased; in this case, an asset will be assigned a new private and public key. The addition of ownership transfer would turn PRIDE into a full-fledged ERTMS registry involving all stakeholders.

Non repudiation PRIDE assigns each train and RBC pair a unique identifier that can only be computed by their respective home KMC and assumes that only Infrastructure Managers issue transactions. However, the current construction allows both KMC to issue an original transaction for the same pair without revealing which of the two KMC issued the transaction, lacking non-repudiation between KMC. This might be a problem during a dispute resolution as both KMC could deny having issued a specific transaction. Attaching a signature obtained from a ring signature scheme using the public keys of all registered Infrastructure Managers would ensure that only Infrastructure Managers issue transactions without telling which one.

Private key loss or corruption PRIDE relies heavily on the assumption that private keys are adequately backed up and kept safe. Infrastructure Managers are required to maintain a list of private keys of all unspent transactions, which in case of loss or corruption would inhibit `KMAC` update or revocation. To mitigate the risk, a hierarchical deterministic key approach can be implemented where nonces are derived from a master private key rather than randomly. In case of a crash or system failure, a KMC can derive the transaction private key from the master key. The inherent drawback is that loss or corruption of the master key has far more implications than loss or corruption of a single regular private key. A second issue is that the same KMC private key is used with every other KMC when deriving the encryption key from the Diffie-Hellman key exchange (see Section 4.2). The resulting key is used to encrypt and decrypt `KMAC`. In case that a KMC private key is compromised, all `KMAC` related to the KMC will also be compromised. PRIDE does not provide forward secrecy, but in order to reduce the consequence of a compromised KMC private key, each train and RBC can be assigned a second private and public key pair. One pair is used for the computation of the tag, and a second one to derive the encryption key from the Diffie-Hellman key exchange. In this case, a compromised key will only affect a single train and RBC pair.

Anonymity More advanced cryptographic techniques can be used to proof a 1-out-of-many statement yielding logarithmic complexity without the necessity of a mixin count [22].

6.3. Migration strategy

ERTMS specifies an offline and online method for key management methods and requires all ERTMS entities to support both methods. The online method is preferred over the offline method. The latter is intended to be used as a fallback system; however, the distribution of `KMAC` between KMCs is still done using the offline method. KMC are still working on the actual design and implementation of the online method, as the specifications do not prescribe exactly how the interaction between KMC should take place except prescribing TLS-PKI. PRIDE is not an alternative to the online method but rather a solution that fits the online method with the advantage that KMC will only need to support and maintain a single system. The blockchain nodes simulated in the proof-of-concept used X.509 certificates for authentication and TLS 1.3 for confidential communication, meeting the TLS-PKI requirement. The offline method can still be used as a fallback method if PRIDE becomes unavailable. However, as KMC join and maintain the system and due to the decentralised nature of PRIDE, such an event is highly unlikely. Even if there are not enough nodes available for consensus, KMC can still query the blockchain

from the active nodes.

According to the online method, loading of KMAC into a train or RBC (step 4 of Figure 2.4), can also be done using TLS-PKI. Future work might include studying the possibility that trains and RBC directly query PRIDE for the latest KMAC, allowing train and RBC to be synchronised as soon as a KMAC is issued and accepted on the blockchain.

6.4. Conclusion

We have presented PRIDE, a decentralised key distribution and management system aimed at railway KMCs across Europe. PRIDE hides relationships between KMC by assigning each train and RBC combination an identifier meaningful only to their respective KMC, but verifiable by all KMC using witness indistinguishable proofs. By introducing a mixin count, it is possible to weigh anonymity against performance. A blockchain maintained by KMC spread geographically across Europe creates a redundant system eliminating single points of failure. In case of node crash or failure, any KMC can reconstruct its latest state from the blockchain using only its private keys. Also, with a blockchain as the system's backbone, all transaction data is appropriately recorded as required by exiting procedures. The security analysis shows that the design is secure against semi-honest participants. Experimental results showed that PRIDE is a feasible key management and distribution system that fits the European Union's railway vision for 2030.

A

ERTMS Key Management Centres

During the course of this project, conversations and email exchange took place with relevant stakeholders of the ERTMS programme in The Netherlands. The most important findings are summarised in this appendix and should give the reader a general idea and feeling about the past, present and future of ERTMS in The Netherlands.

A.1. Infrastructure Manager KMC

Based on conversations and mail exchange with Jaco Schoonen, Asset Management – ERTMS Centrale Systemen at ProRail.

ProRail is the designated Infrastructure Manager in The Netherlands and started the development of its KMC around 2007. The KMC was established during the construction of the *Betuweroute* en the Schiphol–Antwerp high-speed railway (HSL), both fitted with ERTMS Level 2. The implementation, based on the offline method, consisted of a laptop and a set of procedures [45], both still in operation today, no conceptual changes have been made so far. Database backups are made after each session and stored in compact discs, which can be used in restore the system in case of laptop failure. ProRail generates and manages the cryptographic keys using `KM4E`, a special tool developed by the Swiss Federal Railways (SBB). The tool has a check for weak triple-DES keys, but the manual does not reveal anything about the key derivation function. According to SUBSET-038, the technical implementations for key generation performed by the KMC do not need harmonisation.

Generated `KMAC`s are mostly destined to new trains or requests for new trajectories for existing trains, as `KMAC`s are hardly ever refreshed. The majority of the requests are made for freight trains. Freight trains are usually owned by lease companies, which benefit from having their equipment loaded with as many valid `KMAC` as possible to be deployed on different countries and track lines.

Even though refreshing cryptographic keying material is a standard safety procedure, the current procedures discourage this practice. For a rolling stock owner, refreshing a `KMAC` from an operation point of view can cost up to EUR 1000,- per key. On the other side, it is not feasible for ProRail to periodically refresh the keys due to the manual and operational hassle involved with it and therefore issues keys with a long lifespan. Because of the current implementation, Infrastructure Managers and operators are not interested in periodically refreshing the cryptographic material and there is no defined refresh interval. In Germany triple-DES keys are required by the Federal Office for Information Security (BSI) to be refreshed at least every 5 years.

ProRail is currently working on the development of a new KMC that will support Online Key Management. The development is done together with NS and other operators. The PKI design or choice needed for Online Key Management is not part of the interoperability specification, but it is relevant. A KMC will have to be flexible in order to deal with possible different choices from other KMC's. There is an idea to develop and offer a Key Management platform to operators, requiring the development and maintenance of only one system. An operator will have the choice to set up a traditional KMC or to purchase a KMC domain as a service from ProRail. Nothing will change in terms of responsibilities, and operators will be responsible for having the right keys loaded in their equipment. ProRail expects to release a tender for its new KMC in the spring of 2022.

A.2. Railway Operator KMC

Based on conversations and mail exchange with Marijn Verheul, Rolling Stock Software Desk at NS.

The *Nederlandse Spoorwegen* (NS) is the largest railway operator in The Netherlands. Until December 2020, NS had their key management outsourced to a third party. Starting of January 2021, NS moved away from that construction and started to manage their own KMC. During the transition, it appeared that agreements made with this third party and other stakeholders were not formally established, and operational administration was not properly organised nor recorded. For example, communication between other KMC's was done using personal email addresses instead of using a dedicated KMC mailbox. Also, it came to light that different types of agreements were made between different KMC.

NS manages a fleet with different train series. The ERTMS fleet of NS consists of TRAXX locomotives, the Eurostar, the SNG, and the ICNG series. For each of these series, there are different key management procedures and ProRail and NS have agreed to use a different `K-KMC` for each train series. According to the specifications, only one `K-KMC` should be used between KMCs. InfraBel, the Belgium Infrastructure Manager, does follow the specifications and only share one `K-KMC` creating an awkward situation between the neighbouring countries. ERTMS equipment manufacturers are also known to not strictly follow the specification as some suppliers come up with proprietary solutions to load `KMAC`s into the trains. For example, the `KTRANS` specified for verification and integrity during `KMAC` transport is not used for on the SNG and ICNG series. This practice is allowed under certain circumstance, but shows that there is a lot of variation for the same procedure.

The specifications offer plenty of room for own interpretation, which could be one of the reasons organisations come up with different solutions for the same problems. The specification documents are technically in order, but a lot is being overlooked from the operational point of view. Some items are discussed in more detail than others. For example, in Subset 038, it is stated that each KMC shall be able to manage three types of users: an administrator, an operator, and a maintainer. But the roles or differences between the roles are not further explained. Some argue that the *lack* of specification allows KMC to set up their implementation optimally for their specific situation.

Offline Key Management is done using basic and outdated software, see Figure A.1. However, the main security threat in the process is considered to be related to human procedures. `KMAC` are loaded using USB flash drives and computers, which can easily become compromised when left unattended at the workshops. Online Key management aims to solve this issue but the specifications still mandate the use of the GSM-R network to load `KMAC` into trains. Because of this, NS is worried that it will not be possible to load or refresh cryptographic material at a busy train yard due to the limited capacity and bandwidth of GSM-R. For this reason, NS has stalled the adoption of Online Key Management to is still fully committed to implement it because the offline method is simply not scalable for a large fleet. NS and Prorail are working together in a project to arrive at an online key management solution. At the moment, new trains are currently being delivered that do not support Online Key Management. Once the online method becomes available, trains will have be to modified to support online key management.

On Board Information Services (OBIS) OBIS is an Information Technology (IT) platform introduced by NS in 2010. At the beginning, the main goal was to provide passengers Internet connection through Wifi and to present real-time travel information on displays. But quickly a lot of development was put into it, becoming a mature platform where many (internal) applications and services come together. Currently, OBIS makes Operational Technology (OT) systems information available to IT systems, but not the other way around. Work is being done in order to make this a two-way path, facilitating software updates and possibly even cryptographic key management within the NS KMC domain. OBIS relies on a PKI deployed by NS.

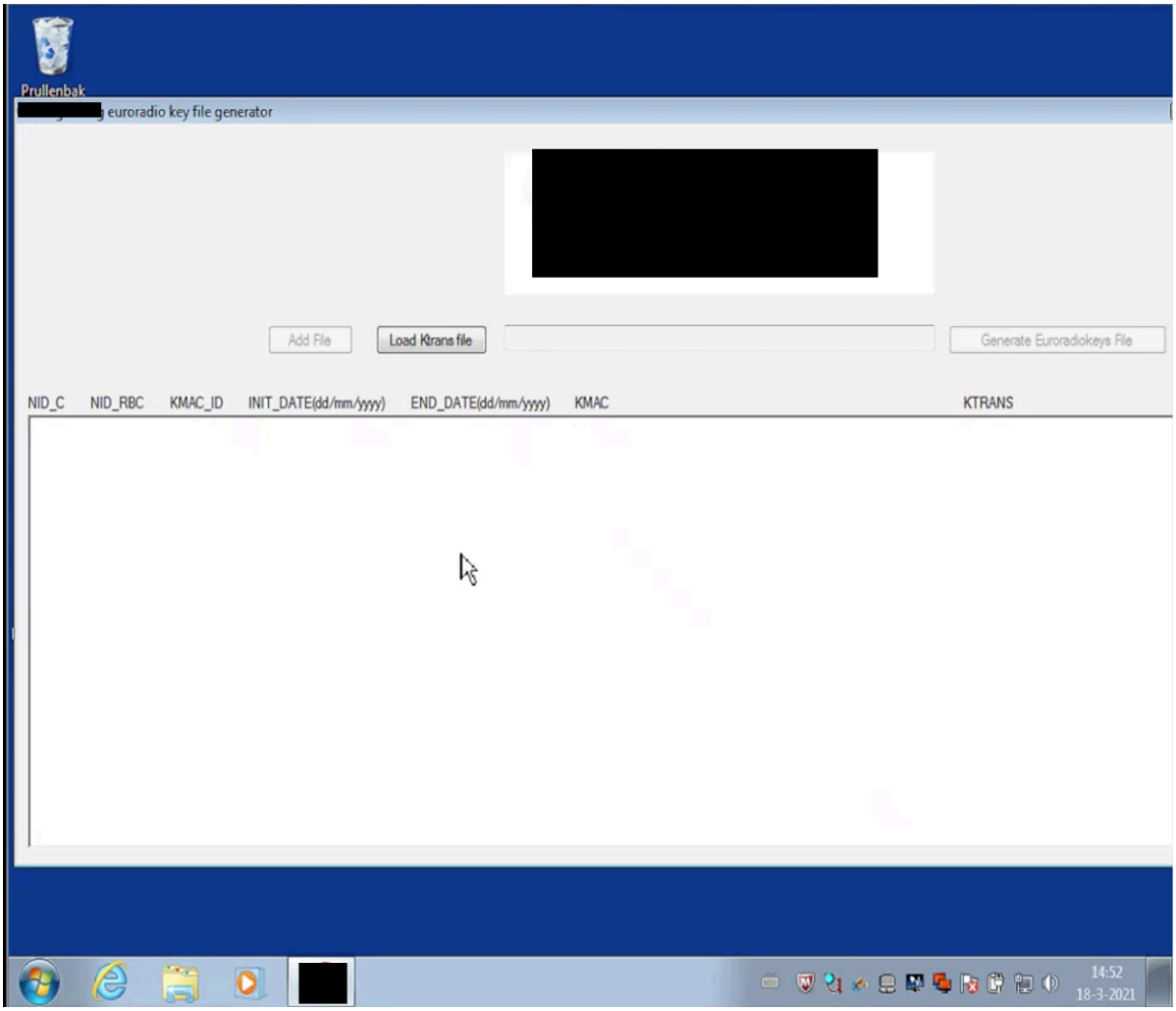


Figure A.1: Software developed by an ERTMS equipment vendor to prepare a KMAC for transportation. Loading the KMAC into the KMAC entity is done using similar software. The software does not work on newer operating systems such as Windows 10 (Between the time the interview took place and the publication of this work, the supplier updated the software, and it is now supported on Windows 10).

B

Radio Block Centre

The Dutch part of the Schiphol–Antwerp high-speed railway (HSL-Zuid) is fitted with ERTMS Level 2 and consists of two RBCs: one for the North part, and one for the South part. The division occurs at *Rotterdam Centraal* station, where the trains need to switch to the legacy Dutch signalling system. The HSL-Zuid is used by passenger and maintenance trains, the list of which is presented in B.1 with corresponding `NID_ENGINE`. A `NID_ENGINE` is a unique number assigned by the European Union Agency for Railways to each On Board Unit (OBU) inside a train. The OBU establishes a session on behalf of the train with the RBC using a `KMAC`. A train has usually two OBUs, one at the front and one at the back.

The RBCs on the the HSL-Zuid are maintained by *Infraspeed Maintenance BV*, which has a contract with the State of The Netherlands and needs to guarantee, at least, a 99% availability of the HSL-Zuid. During this project, a visit was done at the Infraspeed monitoring centre. Figure B.1 gives a general impression of the *control* room.

Figure B.1: Impression of Infraspeed control room where all systems are actively being monitored.

Table B.1: Train series making use of the HSL-Zuid with `NID_ENGINE` range.

Series	`NID_ENGINE`
Windhoff	6118-6119
Eurostar	12900-12999
Thalys	2154500-2154699
E186	18601-18645, 111-125, 300-340, 142, 144, 148, 149, 195-199

The RBC corresponding to the South part of the HSL-Zuid is shown in Figure B.2. An RBC has three processing units performing the same computations independently from each other. A result is accepted if all three outputs are the same. A closer look at Figure B.2 show three secure digital (SD) cards where the `KMAC`s are stored. 152 different OBUs

Figure B.2: One of the two RBCs maintained by Infraspeed. An RBC has three processing units performing the same computations independently from each other.

C

Prepare and commit times

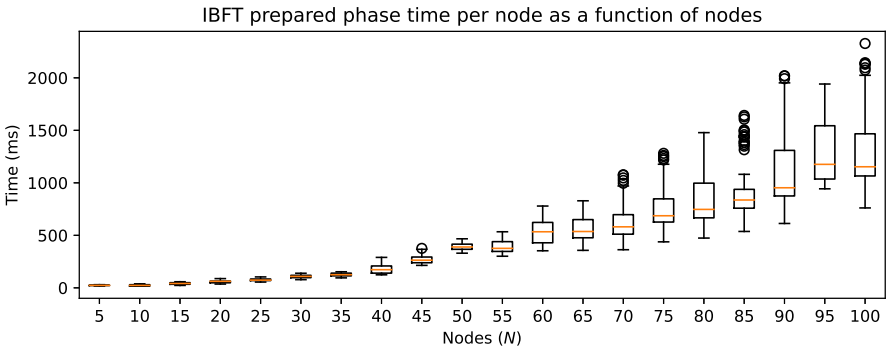


Figure C.1: Average replica prepared phase time as a function of nodes

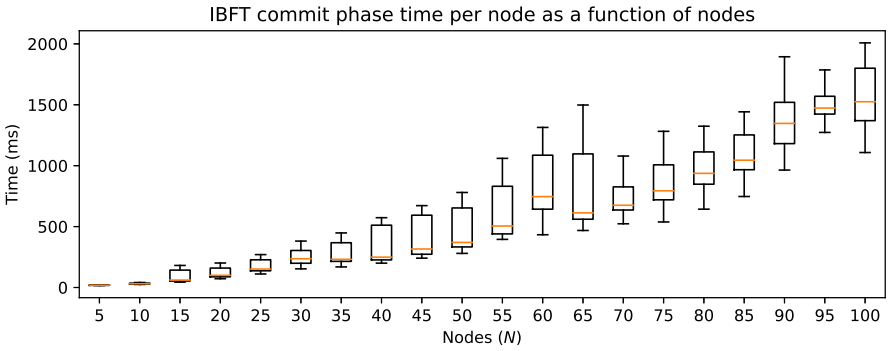


Figure C.2: Average replica prepared phase time as a function of nodes

Bibliography

[1] Bip 0032, 2017. URL https://en.bitcoin.it/wiki/BIP_0032.

[2] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, and et al. Hyperledger fabric. *Proceedings of the Thirteenth EuroSys Conference*, Apr 2018. doi: 10.1145/3190508.3190538. URL http://dx.doi.org/10.1145/3190508.3190538.

[3] E. Barker. Recommendation for key management: Part 1 - general. 2020. URL https://doi.org/10.6028/NIST.SP.800-57pt1r5.

[4] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, pages 626–643, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-34961-4.

[5] R. Brown. The corda platform : An introduction. 2018. URL https://www.r3.com/wp-content/uploads/2019/06/corda-platform-whitepaper.pdf.

[6] J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. 1997.

[7] Jan Camenisch, Rafik Chaabouni, and abhi shelat. Efficient protocols for set membership and range proofs. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, pages 234–252, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-89255-7.

[8] Miguel Castro. Practical byzantine fault tolerance. 04 2001.

[9] Tom Chothia, Mihai Ordean, Joeri Ruiter, and Richard Thomas. An attack against message authentication in the ertms train to trackside communication protocols. pages 743–756, 04 2017. doi: 10.1145/3052973.3053027.

[10] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, pages 174–187, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[11] Rossen Naydenov Dimitra Liveri, Marianthi Theocharidou. Railway cybersecurity: Security measures in the railway transport sector. 2020. URL https://www.enisa.europa.eu/publications/railway-cybersecurity/.

[12] Evan Duffield and Daniel Diaz. Dash: A payments-focused cryptocurrency. 2015. URL https://github.com/dashpay/dash/wiki/Whitepaper.

[13] Morris Dworkin. Sp 800-38d. recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac. 2007.

[14] *Responsibilities and rules for the assignment of values to ETCS variables*. European Union Agency for Railways, 12 2011. Set of specifications 3 (ETCS B3 R2 GSM-R B1).

[15] *EuroRadio FIS*. European Union Agency for Railways, 12 2015. Set of specifications 3 (ETCS B3 R2 GSM-R B1).

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

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

[18] Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the provable security of (ec)dsa signatures. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1651–1662, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978413. URL https://doi.org/10.1145/2976749.2978413.

[19] Mária Franeková, Peter Lüley, Karol Rástočný, and Juraj Ždánsky. Proposal of on-line key management system solutions for railway applications based on asymmetric cryptography. In Jerzy Mikulski, editor, *Tools of Transport Telematics*, pages 188–197, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24577-5.

[20] Conner Fromknecht and D. Velicanu. Certcoin : A namecoin based decentralized authentication system 6 . 857 class project. 2014.

[21] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, page 291–304, New York, NY, USA, 1985. Association for Computing Machinery. ISBN 0897911512. doi: 10.1145/22145.22178. URL https://doi.org/10.1145/22145.22178.

[22] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. Cryptology ePrint Archive, Report 2014/764, 2014. https://ia.cr/2014/764.

[23] Sufian Hameed, Syed Attique Shah, Qazi Sarmad Saeed, Shahbaz Siddiqui, Ihsan Ali, Anton Vedeshin, and Dirk Draheim. A scalable key and trust management solution for iot sensors using sdn and blockchain technology. *IEEE Sensors Journal*, 21(6):8716–8733, 2021. doi: 10.1109/JSEN.2021.3052009.

[24] Badis Hammi, Ahmed Serhrouchni, Sherali Zeadally, and Adja Elloh Yves Christian. A blockchain-based certificate revocation management and status verification system. *Computers & Security*, 104:102209, 01 2021. doi: 10.1016/j.cose.2021.102209.

[25] Vincent Wijdeveld Erik Sennema Isitan Görkey, Chakir El Moussaoui. comparative-study-bft. 04 2020. URL https://repository.tudelft.nl/islandora/object/uuid:01083a4a-900b-4cf9-9746-cb9258c11d9e?collection=education.

[26] Tommy Koens, Scott King, Matthijs van den Bos, Cees van Wijk, and Aleksei Koren. Solutions for the corda security and privacy trade-off: Having your cake and eating it. ING, 2019. URL https://mondovisione.com/_assets/files/Corda_DoSt_v1.6.pdf.

[27] Lafourcade P. Manset D. Koscina, M. and D. Naccache. Blindcons: A consensus algorithm for privacy preserving private blockchains. 2018. URL https://sancy.iut-clermont.uca.fr/~lafourcade/BlindCons.pdf.

[28] Balachander Krishnamurthy. Lecture 2: Privacy, security, anonymity. URL https://www.cs.columbia.edu/~bala/s14/pub_lec2.pdf.

[29] Murat Yasin Kubilay, Mehmet Sabir Kiraz, and Haci Ali Mantar. Certledger: A new pki model with certificate transparency based on blockchain, 2018.

[30] Ao Lei, Haitham Cruickshank, Yue Cao, Philip Asuquo, Chibueze P. Anyigor Ogah, and Zhili Sun. Blockchain-based dynamic key management for heterogeneous intelligent transportation systems. *IEEE Internet of Things Journal*, 4(6):1832–1843, 2017. doi: 10.1109/JIOT.2017.2740569.

[31] Yehuda Lindell. How to simulate it - a tutorial on the simulation proof technique. Cryptology ePrint Archive, Report 2016/046, 2016. https://ia.cr/2016/046.

[32] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy*, pages 325–335, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-27800-9.

[33] Stephanos Matsumoto, Pawel Szalachowski, and Adrian Perrig. Deployment challenges in log-based pki enhancements. *Proceedings of the Eighth European Workshop on System Security*, 2015.

[34] Richard Gendal Brown Mike Hearn. Corda: A distributed ledger. R3, 2019. URL https://www.corda.net/wp-content/uploads/2019/08/corda-technical-whitepaper-August-29-2019.pdf.

[35] Henrique Moniz. The istanbul BFT consensus algorithm. *CoRR*, abs/2002.03613, 2020. URL https://arxiv.org/abs/2002.03613.

[36] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. An empirical analysis of traceability in the monero blockchain. *Proceedings on Privacy Enhancing Technologies*, 2018:143–163, 06 2018. doi: 10.1515/popets-2018-0025.

[37] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. URL http://www.bitcoin.org/bitcoin.pdf.

[38] Sarra Naoui, Mohamed Elhoucine Elhdhili, and Leila Azouz Saidane. Security analysis of existing iot key management protocols. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–7, 2016. doi: 10.1109/AICCSA.2016.7945806.

[39] Arvind Narayanan, Joseph Bonneau, Edward W. Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies - A Comprehensive Introduction*. Princeton University Press, 2016. ISBN 978-0-691-17169-2.

[40] Shen Noether. Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098, 2015. https://ia.cr/2015/1098.

[41] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, pages 31–53, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg. ISBN 978-3-540-48071-6.

[42] Diego Ongaro and John K. Ousterhout. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference*, 2014.

[43] Panagiotis Papadimitratos, Levente Buttyan, Tamas Holczer, Elmar Schoch, Julien Freudiger, Maxim Raya, Zhendong Ma, Frank Kargl, Antonio Kung, and Jean-Pierre Hubaux. Secure vehicular communication systems: design and architecture. *IEEE Communications Magazine*, 46(11):100–109, 2008. doi: 10.1109/MCOM.2008.4689252.

[44] Alex Preukschat. *Self-Sovereign Identity decentralized digital identity and verifiable credentials*. O'Reilly Media, S.l, 2021. ISBN 9781617296598.

[45] *Gebruiksvoorschrift ETCS Key Management*. ProRail, 8 2012. GVS60560-1.

[46] Meurs Rogier. Visie op ertms key management van vervoerders en infrabeheerders, 2021.

[47] Matthias Ruete. Work plan 2020 of the european coordinator for ertms. 2020. URL https://ec.europa.eu/transport/sites/default/files/work_plan_ertms_2020.pdf.

[48] Bassam El Khoury Seguias. Monero's building blocks part 1 of 10 – prerequisites, 2018. URL https://delfr.com/wp-content/uploads/2018/04/Monero_Building_Blocks_Part1.pdf.

[49] Bassam El Khoury Seguias. Monero's building blocks part 3 of 10 – introduction to ring signatures, 2018. URL https://delfr.com/wp-content/uploads/2018/04/Monero_Building_Blocks_Part3.pdf.

[50] Bassam El Khoury Seguias. Bitcoin consensus in distributed systems, 2020. URL https://delfr.com/wp-content/uploads/2020/11/Bitcoin_General_Consensus.pdf.

[51] Nigel P. Smart. *Cryptography Made Simple*. Springer Publishing Company, Incorporated, 1st edition, 2015. ISBN 3319219359.

[52] Aleksander Sniady and Jose Soler. An overview of gsm-r technology and its shortcomings. In *2012 12th International Conference on ITS Telecommunications*, pages 626–629, 2012. doi: 10.1109/ITST.2012.6425256.

[53] Richard J. Thomas, Mihai Ordean, Tom Chothia, and Joeri de Ruiter. Traks: A universal key management scheme for ertms. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACSAC 2017, page 327–338, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450353458. doi: 10.1145/3134600.3134631. URL https://doi.org/10.1145/3134600.3134631.

[54] Nicolas van Saberhagen. Cryptonote v 2.0. 2013. URL https://bytecoin.org/old/whitepaper.pdf.

[55] Ze Wang, Jingqiang Lin, Quanwei Cai, Qiongxiao Wang, Jiwu Jing, and Daren Zha. Blockchain-based certificate transparency and revocation transparency. In *Financial Cryptography Workshops*, 2018.

[56] Daniel Davis Wood. Ethereum: A secure decentralised generalised transaction ledger. 2014.

[57] Karl Wüst and Arthur Gervais. Do you need a blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54, 2018. doi: 10.1109/CVCBT.2018.00011.

[58] Alexander Yakubov, Wazen M. Shbair, Anders Wallbom, David Sanda, and Radu State. A blockchain-based pki management framework. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6, 2018. doi: 10.1109/NOMS.2018.8406325.