Blockchain-Based Root of Trust Management in Security Credential Management System for Vehicular Communications

Arijet Sarker, SangHyun Byun, Wenjun Fan, Sang-Yoon Chang
{asarker,sbyun,wfan,schang2}@uccs.edu
University of Colorado Colorado Springs

ABSTRACT

Security Credential Management System (SCMS) provides the Public Key Infrastructure (PKI) for vehicular networking. SCMS builds the state-of-the-art distributed PKI to protect the vehicular networking privacy against an honest-but-curious authority (by the use of multiple PKI authorities) and to decentralize the PKI root of trust (by the Elector-Based Root Management or EBRM, having the distributed electors manage the Root Certificate Authority or RCA). We build on the EBRM architecture and construct a Blockchain-Based Root Management (BBRM) to provide even greater decentralization and security. More specifically, BBRM uses blockchain to i) replace the existing RCA and have the electors directly involved in the root certificate generation, ii) control the elector network membership including elector addition and revocation, and iii) provide greater accountability and transparency on the aforementioned functionalities. We implement BBRM on Hyperledger Fabric using smart contract for system experimentation and analyses. Our experiments show that BBRM is lightweight in processing, efficient in ledger size, and supports a bandwidth of multiple transactions per second. Our results show that the BBRM blockchain is appropriate for the root certificate generation and the elector membership control for EBRM within SCMS, which are significantly smaller in number and occurrences than the SCMS outputs of vehicle certificates. We also experiment to analyze how the BBRM distributed consensus protocol parameters, such as the number of electors and the number of required votes, affect the overall scheme's performances.

CCS CONCEPTS

• **Security and privacy** → **Domain-specific security and privacy architectures**; **Formal security models**; *Distributed systems security*; *Security protocols*; Access control;

KEYWORDS

Blockchain, PKI, SCMS, Vehicular Networking

ACM Reference Format:
Arijet Sarker, SangHyun Byun, Wenjun Fan, Sang-Yoon Chang. 2021. Blockchain-Based Root of Trust Management in Security Credential Management System for Vehicular Communications . In *The 36th ACM/SIGAPP Symposium on Applied Computing (SAC '21), March 22–26, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.1145/3412841.3441905

SAC '21, March 22–26, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8104-8/21/03…$15.00
https://doi.org/10.1145/3412841.3441905

1 INTRODUCTION

The vehicular networking technology enables greater information and intelligence on the vehicular control along with those provided from the sensing technologies such as cameras, RADAR, LIDAR. Vehicular networking, also known as vehicle-to-everything (V2X) or vehicular ad hoc networking (VANET), provide connectivity and information exchange between the vehicles, devices, and traffic infrastructures. For example, the vehicular networking can include the Basic Safety Messages (BSMs), traffic flow, road conditions, providing critical information for the vehicle operation and control.

Because of the application's critical nature, securing the integrity of the vehicular networking is important and requires a trustworthy key via a PKI. The vehicular networking additionally requires privacy because of the vehicle's operation and tracking can compromise the driver's privacy, similarly to the mobile phone tracking. The US Department of Transportation (USDOT) proposes an advanced PKI called Security Credential Management System (SCMS) to support these security requirements of vehicular networking. The privacy requirement drives the SCMS to divide the generation and provisioning process of those certificates among multiple autonomous authority organizations. SCMS therefore builds on the standard PKI, e.g., X.509 standard, but includes greater complexity to provide a state of the art distributed PKI [6]. SCMS provides its root of trust (the root certificate management) via the Elector-Based Root Management (EBRM). EBRM introduces multiple distributed set of electors to elect and govern the Root Certificate Authority (RCA) generating the root certificates for the other SCMS authorities. Our work focuses on the EBRM providing the root of trust for the PKI. While EBRM is designed to be resilient against compromises and provides the distributed management using electors [19], our work using blockchain provides an even greater resiliency and avoids the single point of failure in RCA by replacing RCA.

Our Contribution. Our work builds on the distributed PKI based on EBRM but introduces a blockchain to build a Blockchain-Based Root Management (BBRM). BBRM uses one blockchain but processes the two distinct certificates: i) *Root Certificate (RC)* to replace the RCA in the current SCMS and to provide the trust/certificates for other SCMS authorities and ii) *Elector Certificate (EC)* to control the memberships of the elector network. BBRM provides the following security benefits: greater decentralization in the PKI, greater resiliency against the authority failure, and greater transparency on the certificate transaction processing. BBRM also simplifies the SCMS architecture by both replacing the RCA with blockchain

in EBRM and offloading the SCMS manager's involvement in the elector voting to the blockchain protocol.

Paper Organization. Section 2 describes the state of the art SCMS for vehicular networking PKI including the EBRM. Our work is based on the EBRM scheme. Section 3 defines the contribution scope and threat model of our approach. Section 4 describes the design principle and system architecture of the proposed approach while Section 5 provides the actual design including transaction, function, network setup, and voting management in BBRM. Section 6 presents the implementation details of the proposed design while experimental result is analyzed in Section 7. Section 8 reviews the related work and Section 9 concludes the paper.

2 SCMS FOR VEHICULAR NETWORKING PKI

We provide an overview of the SCMS in Section 2.1 and then focus on EBRM, a part of SCMS, to describe it in greater details in Section 2.2, since our research focuses on EBRM to advance the SCMS design.

2.1 Security Credential Management System

Security Credential Management System (SCMS) [7] [21] is a PKI design for issuing digital certificates to the vehicles, infrastructures, and pedestrians/mobile nodes for V2X communications. SCMS adopts a strong threat model against the vehicular privacy. In addition to the standard adversary accessing the V2X channels at the time of the certificate use, SCMS considers an honest-but-curious adversary compromising the authority entities involved in the SCMS operations during the digital certificate generation [8, 9]. Such threat model motivates a decentralized PKI for the separation of the SCMS authority entities so that no single authority compromise breaches the vehicular privacy. SCMS therefore provides a distributed PKI involving multiple authorities, as opposed to the traditional centralized PKI (such as that used for public Internet) having one certificate authority for the certificate management including the certificate signing, generation, and issuance. Fig. 1 depicts the authority entities and their interactions where the line connections indicate interactions for the SCMS operations. Our work advances the operations of the electors and the RCA, which are highlighted with dotted box in Fig. 1 and described in greater details in Section 2.2.

Electors elect and govern the Root Certificate Authority (RCA) and operate based on the self-signed certificates. RCA issues the root certificates for the SCMS authority entities such as Policy Generator (PG), Misbehavior Authority (MA) and Intermediary Certificate Authority (ICA). PG maintains and signs the global configuration information and all the trusted certificate chains of SCMS. ICA shields the RCA from traffic and attacks by serving as a secondary certificate authority. MA identifies misbehavior or malfunction by devices by processing misbehavior report and, generate, sign and release certificate revocation list (CRL) to the SCMS authority entities. ICA provides certificates to Registration Authority (RA), Enrollment Certificate Authority (ECA), Pseudonym Certificate Authority (PCA), Linkage Authorities (LA_1 and LA_2). RA processes certificate request from End-Entity (EE) devices. An EE device, i.e., On-Board Equipment (OBE) or Road Side Equipment (RSE) sends

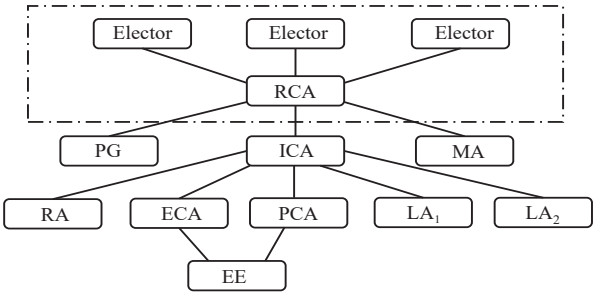


Fig. 1: SCMS Authorities and Interactions (the dotted box indicates the scope of our research contribution)

or receives messages. ECA provides enrollment certificates to EE devices to authenticate against RA. PCA issues short-term pseudonym certificates to EE devices. There are two Linkage Authorities (LA) in SCMS, LA1 and LA2 to generate pre-linkage values that used in the certificates to support efficient revocation.

2.2 Elector-Based Root Management (EBRM)

In SCMS, RCA, ICA, PG, MA, ECA, RA, PCA, LA1, LA2 and EE devices form standard PKI hierarchy which maintains the chain of trust (an ordered list of the certificates where the receiver can verify all the certificates upto the top of the certificate chain) among the authorities in SCMS and EE devices. There is EBRM scheme on top of the standard PKI hierarchy to manage the RCA and elector certificate such as establishing a new certificate or revoking an existing certificate. EBRM and standard PKI hierarchy can be implemented in parallel using different crypto-system since the electors are not part of the standard PKI hierarchy. This distributed management scheme of the electors follows a democratic voting system to add or revoke a RCA or elector certificate.

The electors vote for a RCA certificate or an elector certificate by signing an endorsement and all these endorsements are aggregated into a ballot. A ballot can be trusted by any authority in the system when there is a quorum of valid elector endorsements is included in the ballot. The value of quorum is defined in the Global Policy File (GPF). There are four endorsement types can be presented in a ballot: *Add RCA certificate*, *Revoke RCA certificate*, *Add elector certificate* and *Revoke elector certificate*. Only one type of endorsement can be contained in each ballot. In SCMS, the SCMS Manager (not shown in Fig. 1) conducts the RCA and elector voting by collecting all the elector endorsements/votes from the electors and delivering the certificate and the endorsements to PG. In case of RCA voting, it also presents the RCA certificate to electors.

3 CONTRIBUTION SCOPE & THREAT MODEL

We describe the problem statement and the contribution scope in Section 3.1 and the threat model in Section 3.2.

3.1 Contribution Scope

While there are multiple authorities for SCMS, our focus is on EBRM consisting of elector and RCA highlighted by dotted box in Fig. 1. Securing RCA is critical because it provides the root of

trust for SCMS, which in turn provides the certificates and the trust in keys to the vehicles for the vehicular networking. Our work therefore defends against the RCA compromise by having a blockchain to both replace the RCA (so that the electors are directly involved in the root certificate generation) and securely control the membership of the electors (add or revoke electors). Our work also reduces the SCMS Manager's role (whose primary role is to set and administer the SCMS policy) since the blockchain replaces the elector voting management which has previously been executed by the SCMS Manager.

3.2 Threat Model

In our threat model, we consider that adversaries cannot compromise the majority of the electors so that they can not control the majority votes in the voting process. We assume that standard cryptographic primitives such as forging digital signatures and finding hash collisions cannot be broken by the adversaries. In our blockchain-based voting framework, we further consider that an adversary can monitor the transactions sent by the electors to the blockchain. We also assume that the authority entities in BBRM comply to the regulations set up by the voting process such as providing votes if they support the RC or EC or only using the valid certificates containing the majority votes from the electors. This property is considered by implementing the accountability control on the authority entities. In other words, we consider that the authority entities involved in BBRM are trusted so they perform their roles correctly. We also consider that the integrity of the storage data, i.e. RC and EC is protected as the data are stored in a distributed manner using blockchain. Our proposed approach can handle m number of simultaneous electors compromise where the total number of electors are $2m + 1$. For example, if there are 3,5,7 and 9 electors, it can handle 1,2,3 and 4 electors compromise respectively. This means that as long as the majority of the electors are not compromised, the proposed system is not compromised. In SCMS, RCA provides the certificates to the respective authority entities in SCMS such as ICA, MA and PG. Thus, the attacker compromising the RCA can have control over the certificate issuance to the authority entities by RCA. In our case, the blockchain stores the RC with elector votes and the respective authority entities get the RC from blockchain. Thus, the RC is stored in a distributed manner rather than using a centralized authority like RCA in SCMS. An attacker compromising a single authority such as compromising RCA in SCMS can have control over the RC but using a distributed scheme to store RC can prevent that single point of compromise.

4 BBRM DESIGN PRINCIPLE AND ARCHITECTURE

In this section, we present the BBRM architecture and design principles by explaining its entities and by describing the advantages of applying blockchain for SCMS and the rationale for using one blockchain as opposed to using two blockchains to separate the two distinct functionalities of Root Certificate (RC) and Elector Certificate (EC). We present the acronyms with explanation for the terms we use in our BBRM scheme in Table 1.

Acronym	Explanation
RCA	Root Certificate Authority
ICA	Intermediate Certificate Authority
PG	Policy Generator
MA	Misbehaviour Authority
EC	Elector Certificate
RC	Root Certificate
OSP	Ordering Service Provider
GCCF	Global Certificate Chain File
EBRM	Elector-Based Root Management
BBRM	Blockchain-Based Root Management
SCMS	Security Credential Management System

Table 1: Acronyms

4.1 Advantages of Using Blockchain for EBRM

BBRM applies blockchain for two functionalities: i) RC to replace the RCA to make the SCMS simpler and rid the corresponding vulnerabilities, e.g., the RCA being the single point of failure and ii) EC for elector network control within EBRM.

Blockchain is appropriate for such purposes since the transactions within the ledger are immutable, transparent, automatic, and distributed. The immutability (after the confirmation and the processing delay associated with it) yields the trust in the transactions, i.e., it makes the transactions of the RC/EC generation and revocation more trustworthy, since the transactions cannot be revoked for manipulations and has been voted/vouched by the other electors via the distributed consensus protocol. The transparency provides BBRM with greater accountability therefore informing the EC function of BBRM. As long as the nodes are alive and connected (or when they become connected to the SCMS network), the blockchain protocol additionally automatically synchronizes the RC states, including the currently active RC certificates. Lastly, the blockchain relies on a distributed consensus protocol to achieve the ledger synchronization making it appropriate for the EBRM (which already involves a distributed set of electors but for selecting the RCA only). BBRM in fact replaces the RCA to further decentralize the SCMS by avoiding the single authority processing and generating the RCs.

4.2 One Blockchain (vs Two Blockchains)

BBRM provides two functionalities of RC generation/revocation and EC generation/revocation. Despite these two distinct functionalities, we use one blockchain as opposed to two separate blockchains. Since the RC processing depends on the elector network (which can be controlled/changed by the EC certificates), using one blockchain enables the tighter synchronization and causality control without extra mechanisms beyond the blockchain protocol. For example, if the elector who is in the process of getting revoked votes/vouches for a RC transaction, using two blockchain require separate trustworthy timestamping between the two events. In contrast, BBRM using one blockchain enables a clearer ordering of which event occurred first thanks to the Ordering Service Provider in the permissioned blockchain (no forks in contrast to the permissionless blockchains such as blockchain-based digital currency).

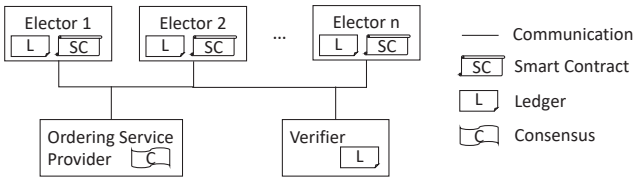


Fig. 2: Blockchain-Based Root Management (BBRM)

4.3 BBRM Structure and Entities

In this subsection, we describe the structure of BBRM and the entities used in BBRM. Our BBRM aims at managing the RC and EC by replacing the RCA and the function of SCMS Manager in elector voting management. Fig. 2 shows an overview of the proposed BBRM architecture as well as the entities used in BBRM. The following entities are part of the BBRM design.

Elector: It represents the core element of BBRM. Electors are involved in the voting process for the selection or revocation of an EC or RC. A quorum of electors can add or revoke an EC or a RC by signing and broadcasting certificates/votes in the blockchain network.

Verifier: It represents other entities in SCMS that need to know the EC and RC such as ICA, MA, and PG. BBRM issues certificates for the verifier. They have only read access in BBRM.

Ordering Service Provider: It provides the service of transaction ordering and atomic broadcast of the ordered transactions received from other entities, i.e., electors, verifiers in BBRM. It consists of a cluster of orderers. It provides bootstrapping with a genesis block on the system.

Every entity in the BBRM network communicates with each other using a protected, and reliable communication channel which is represented by a solid line in Fig. 2. In BBRM network, each SCMS entity has the distributed ledger. Ledger represents EC and RC with elector votes. It stores transaction log of functions such as adding RC, revoking RC, adding EC, and revoking EC etc. Transactions are collected inside blocks to be appended to the blockchain, enabling to know the history of generation or revocation of EC and RC. There are two types of roles to interact with the ledger - i) electors who are having both write and read access to the ledger and ii) verifiers who are having only the read access to the ledger. Any member who is allowed to join in the BBRM network receives new blocks of transactions from Ordering Service Provider. Smart Contract which consists of different functions (e.g., adding EC, revoking EC etc.) is used to write transactions in the BBRM ledger. This is shared by the different entities in BBRM. Consensus enables the distributed set of authorities to agree on the transaction content and order in BBRM blockchain (more specifically, in our work, the agreement among the authorized electors). We use the existing distributed consensus protocol for consortium blockchain and avoid the distributed consensus protocols for permissionless blockchains due to their inefficiencies.

Notation	Description
S	Set of Entities in BBRM Blockchain
E	Elector
V	Verifier
O	Orderer
OSP	Ordering Service Provider
χ	Consensus Algorithm
CT	Cipher Text
K_i	Public Key of the Entity $i \in S$
k_i	Private Key of the Entity $i \in S$
$CERT_i$	Certificate of the Entity $i \in S$
ID_i	ID of the Entity $i \in S$
Enc	Encryption Algorithm
Dec	Decryption Algorithm
m	Maximum Number of Simultaneous Electors Compromise can be Handled by BBRM or SCMS

Table 2: Notations and Variables

5 BBRM DESIGN

In this section, we describe about the design of the network setup, functionalities, and transactions used in BBRM. We also describe about the voting management in BBRM.

5.1 Notations and Variables

Notations and Variables are defined in Table 2. S is the set of entities used in BBRM. The set of the entities involved in BBRM is S ={OSP, E, V}. For any entity i, where $i \in S$, K_i is public key of the entity i and k_i is private key of the entity i. ID_i is the ID of the entity i, and $CERT_i$ defines the certificate of the entity i. χ is the consensus algorithm used in BBRM and CT defines the cipher text. We define Elector, Verifier, Ordering Service Provider, and Orderer as E, V, OSP, and O respectively. Here, E ={$E_1, E_2, \cdots, E_n$}, V ={$V_1, V_2, \cdots, V_n$}, and OSP ={$O_1, O_2, \cdots, O_n$}. Enc and Dec defines the encryption and decryption algorithm respectively. The maximum number of simultaneous electors compromise can be handled by BBRM is m where the total number of electors is $2m + 1$.

5.2 Network Setup

In this subsection, we present the design of BBRM network setup where two channels are created: i) System Channel (to store consortium configuration) and ii) Application Channel (to share ledger and chaincode among channel members). In this stage, the initial set of E, O and V are created and the credentials for E, O and V are generated in their respective machines.

1) System Channel: ID_E, K_E, k_E and ID_V, K_V, k_V of E and V are generated respectively and $CERT_E$, $CERT_V$ of E and V are calculated respectively according to Eq.1. After that, Cipher Text CT_E and CT_V using ID_E, $CERT_E$, K_E and ID_V, $CERT_V$, K_V are calculated respectively using asymmetric encryption, as described in Eq.2. CT_E, and CT_V are submitted to OSP.

$$CERT_i = Selfsign(k_i) \quad (1)$$

$$CT_i = Enc(ID_i, CERT_i, K_i) \quad (2)$$

$$(ID_i, CERT_i, K_i) = Dec(CT_i) \quad (3)$$

Name	Description
Version	This field includes the version number of the field. The certificate format may change over time, this version number keeps track of this change.
Serial Number	The value in this field differentiates one certificate from the another generated by the electors by providing unique identifier to each certificate.
Subject Name*	This field provides the information of the entities whose certificate is needed to add or revoke, i.e. an elector or who gets the certificate from the blockchain, i.e. ICA, MA or PG.
Issuer Name	This field contains the information of the elector who creates and signs the certificate.
Subject Public Key*	This field provides the information of the public key of the entities whose certificate is needed to add or revoke, i.e. an elector or who gets the certificate from the blockchain, i.e. ICA, MA or PG.
Subject Unique ID*	This field holds the value of the unique identifier of of the entities whose certificate is needed to add or revoke, i.e. an elector or who gets the certificate from the blockchain, i.e. ICA, MA or PG.
Issuer Unique ID	This field contains the value of the unique identifier of the elector, who creates and signs the certificate.
Validity Period	This field contains the two fields: i) valid from - determine the date when certificate becomes valid and ii) valid to - determines the date after which the certificate is no longer considered valid.
Digital Signature	This field contains the digital signature of the elector who creates and signs the certificate.
Algorithm	This field defines the hashing algorithm and the signing algorithm used in the certificate.
Function Type	We have four *Voting* functions: i) *AddRC*, ii) *RevokeRC*, iii) *AddEC* and iv) *RevokeEC*. This field contains the name of the specific function.

Table 3: Certificate Format. The Asterisks (*) are the fields which are dependent on the function type and, more specifically, on whether it is for SCMS-authoritiy certificate generation/revocation or for elector addition/deletion

CT_E and CT_V of E and V are decrypted using Eq.3 in *OSP* machine and the resulting plaintext are $(ID_E, CERT_E, K_E)$ and $(ID_V, CERT_V, K_V)$ respectively. Then, the genesis block of the system channel is generated using procedure Ordering Service Provider Registration and Consortium Member Registration. In Ordering Service Provider Registration procedure, the set up of Ordering Service is done by including $O(ID_O, CERT_O, K_O)$, *block size* and consensus algorithm χ, where χ is used by permissioned consensus algorithm. Consortium Member Registration procedure defines consortium member by including information of $E(ID_E, CERT_E, K_E)$ and $V(ID_V, CERT_V, K_V)$. The pseudocode to create system channel is shown in Algorithm 1.

2) Application Channel: Application channel is created using both Consortium Member Policy and Consortium Member Setting. In Consortium Member Policy, the policy for consortium member such as the privilege to read and write in consortium blockchain is defined. In Consortium Member Setting, the consortium member information are included. In BBRM, the consortium members are E and V. A genesis block of application channel is created using these information and the genesis block is broadcasted to E and V by *OSP*. The smart contract is deployed in the application channel by *OSP* which defines the functions can be used in consortium blockchain. The pseudocode to create application channel is shown in Algorithm 2.

Algorithm 1 System Channel

Input: $\{E_1\ (ID_{E_1}, CERT_{E_1}, K_{E_1}), \cdots, E_n\ (ID_{E_n}, CERT_{E_n}, K_{E_n})\}$, $\{O_1\ (ID_{O_1}, CERT_{O_1}, K_{O_1}), \cdots, O_n\ (ID_{O_n}, CERT_{O_n}, K_{O_n})\}$ $\{V_1\ (ID_{V_1}, CERT_{V_1}, K_{V_1}), \cdots, V_n\ (ID_{V_n}, CERT_{V_n}, K_{V_n})\}$

Output: Genesis Block of System Channel

1: **procedure** *Ordering Service Provider Registration*
2: *Ordering Service Provider* $\leftarrow \{O_1\ (ID_{O_1}, CERT_{O_1}, K_{O_1}), \cdots, O_n\ (ID_{O_n}, CERT_{O_n}, K_{O_n})\}$, *block size*, χ
3: **procedure** *Consortium Member Registration*
4: *Policy* $\leftarrow$ *Consortium Member Policy*
5: *Consortium Member* $\leftarrow \{E_1\ (ID_{E_1}, CERT_{E_1}, K_{E_1}), \cdots, E_n\ (ID_{E_n}, CERT_{E_n}, K_{E_n})\}$

Algorithm 2 Application Channel

Input: $\{E_1\ (ID_{E_1}, CERT_{E_1}, K_{E_1}), \cdots, E_n\ (ID_{E_n}, CERT_{E_n}, K_{E_n})\}$, $\{V_1\ (ID_{V_1}, CERT_{V_1}, K_{V_1}), \cdots, V_n\ (ID_{V_n}, CERT_{V_n}, K_{V_n})\}$

Output: Genesis Block of Application Channel

1: **procedure** *Consortium Member Policy*
2: *Policy* $\leftarrow$ *Consortium Member Policy*
3: **procedure** *Consortium Member Setting*
4: *Consortium Member* $\leftarrow \{E_1\ (ID_{E_1}, CERT_{E_1}, K_{E_1}), \cdots, E_n\ (ID_{E_n}, CERT_{E_n}, K_{E_n})\}$, $\{V_1\ (ID_{V_1}, CERT_{V_1}, K_{V_1}), \cdots, V_n\ (ID_{V_n}, CERT_{V_n}, K_{V_n})\}$

5.3 Functionalities and Transactions

There are total five functions - i) *AddRC*, ii) *RevokeRC*, iii) *AddEC*, iv) *RevokeEC*, and v) *Query*. The functions AddRC, RevokeRC, AddEC, and RevokeEC can be commonly termed as *Voting* function since these functions are related to the voting process of adding or revoking EC or RC certificate. AddRC belongs to the voting for the selection of a RC while RevokeRC corresponds to the voting for the revocation of a RC. AddEC belongs to the voting for the selection of an EC and RevokeEC corresponds to the voting for the revocation of an EC. Transactions submitted using the Voting function contain the same payload format meaning the same certificate format, however, the classification is made into four types to distinguish between the different objectives of the function, i.e. to specify whether it is for adding or revoking a certificate and whether it is for EC or RC. The format of the certificate for EC and RC follows X.509 standard with an extended field *Function Type*. The description of the certificate field is defined in Table 3. Since the pseucode for AddRC, RevokeRC, AddEC and RevokeEC functions

Algorithm 3 Voting Function

Input: $CERT_i[Arguments]$
Output: $BOOL$

1: **if** $Count.\ CERT_i[Arguments] \neq Len$ **then**
2: **return** $Error$
3: **else**
4: $CERT_i \leftarrow CERT_i[Arguments]$
5: $CERT_i.\ Putstate$
6: **return** $Success$

Algorithm 4 Query Function

Input: $CERT_i[Arguments[Key]]$
Output: $CERT_i$

1: **if** $CERT_i[Arguments[Key]].\ Getstate \neq CERT_i$ **then**
2: **return** $Error$
3: **else**
4: $print\ CERT_i$

are same, it is commonly provided in Algorithm 3. Those functions first check the number of arguments provided by the electors. If it does not match with the threshold, Len, then the program generates error. Otherwise, an index entry is generated for the certificate and the certificate with all the arguments is added in the ledger.

Query function is used to read transactions which are submitted using the Voting function for an EC or RC. The pseudocode of Query function is shown in Algorithm 4. Each transaction has a key value, the index entry for the transaction. A transaction with the specific index entry is searched in the ledger. If it does not found, then an error message is shown otherwise the certificate information of the specific transaction is shown.

All the transactions submitted using Voting function can be termed as *Voting* transaction. On a conceptual level, Voting transactions for a specific certificate can be classified into two categories - i) *Initiative* transaction and ii) *Vouching* transaction. Initiative transaction belongs to the start/leading of the specific certificate to be added or revoked while Vouching transaction corresponds to support for that specific certificate (the same subject and the public key as the Initiative transaction. In other words, Initiative transaction introduces a new subject-public key pair while Vouching transaction advocates that transaction (aside from the issuer and the hash/signature, the payload content of the certificate should be the same). The distinction between the Initiative and Vouching transaction is that Initiative transaction implies a vote so that it counts as the first vote; it's just a special vote and Vouching transaction is for vouching that Initiative transaction. The format is same in the certificate format regardless of whether it is Initiative transaction or Vouching transaction.

5.4 Voting Management

In this subsection, we describe the voting procedure in BBRM and how the EC and RC are managed differently than the current state of the art SCMS/EBRM. The electors-driven blockchain replaces the RCA and the centralized voting management role of SCMS Manager by constructing and distributing the certificates (EC and RC) in a distributed manner using blockchain. Each of the votes is in the forms of certificates/transactions where majority votes for certificates/transactions supporting the same certificate payload (the same subject-public key binding in the payload but with majority distinct issuers) mean that the corresponding certificate becomes effective. In other words, the electors issue certificates where there are majority votes for the certificates needed for it to either become a valid certificate or revoke an existing certificate. One voting round for a certificate which makes the certificate viable is called an event. If we define the total number of elector votes as $2m + 1$ then the range of the majority votes is defined by $\{ m + 1, m + 2, \cdots, 2m + 1 \}$ where $m + 1$ is the number of minimum votes and $2m + 1$ is the number of maximum votes needed to become majority. In the voting process, there is only positive vote means when an elector supports a certificate then only it provides vote for the certificate. Adding negative votes has additional complexities such as increasing the blockchain size, more complex counting process whereas adding positive votes makes it more simpler.

In SCMS, SCMS Manager conducts the voting management of electors to add or revoke elector or RCA. In our case, blockchain can replace the role of SCMS Manager for elector voting management to select EC or RC. In BBRM, any valid elector can propose for generation or revocation of EC or RC and broadcast the proposal to the other valid electors in BBRM for voting by submitting Initiative transaction. Valid electors can vote for the proposal by submitting Vouching transaction. When a quorum of votes are presented for a proposal then the certificate can be added or revoked. In SCMS, votes are collected and distributed by the centralized authority, SCMS Manager. However, it is handled using blockchain in BBRM. Thus, the elector voting management can be performed in a distributed manner in BBRM.

6 BBRM IMPLEMENTATION

In this section, we describe the implementation of BBRM. We use Hyperledger Fabric [3] (version 1.4) to build BBRM prototype varying the number of electors, i.e., 3,5,7, and 9 because three operational electors are needed for Proof of Concept (PoC) and nine operational electors are assumed for maximum electors deployment [4] in SCMS. The experiment is conducted on one physical machine with Windows 10 Home 64 bits, Intel(R) Core(TM) i7-4790K, 4.00GHz, 8 core CPU and 32768 MB, 1333 MHz RAM that runs virtual machines (VMs) with Ubuntu 16.04 64 bits. We run each entity on VM and the identical specification of all VMs is shown in Table 4. Our smart contract is implemented in Go language. There are three consensus algorithms in Hyperledger Fabric: Raft [16], Kafka [12] and Solo [18] of which Raft consensus algorithm is used in our implementation. Raft is a crash-fault-tolerant (CFT) ordering service based on leader and follower model. It is easier to set up, manage and provides more decentralization approach in managing consensus than Kafka and Solo. Moreover, Kafka has additional administrative overhead in managing Kafka cluster.

In Fig.3, the testbed of the implementation of BBRM is presented. The client sends transaction proposal or query request to the peers. The signature of the client is verified by each peer. If the client is verified then the inputs of the transaction proposal or query request is taken as arguments by the peers and the respective function of the smart contract is executed. The function for transaction proposal and query request are Voting and Query function respectively. The

Resourse	Detail
OS	Ubuntu 16.04, 64-bit, Kernel 4.15.0-113-generic
RAM	DDR3, 2048 MB
CPU	Intel(R) Core(TM) i7-4790K, 1 Core, 4 GHz
Adapter	NAT Network

Table 4: Machine Specification

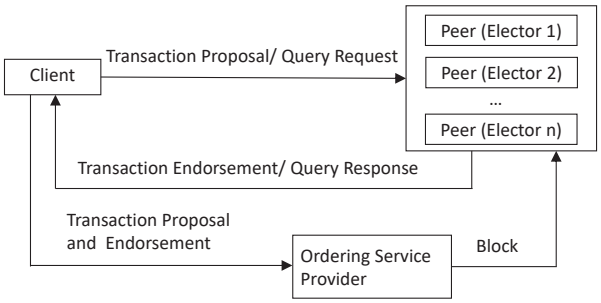


Fig. 3: Implementation of BBRM

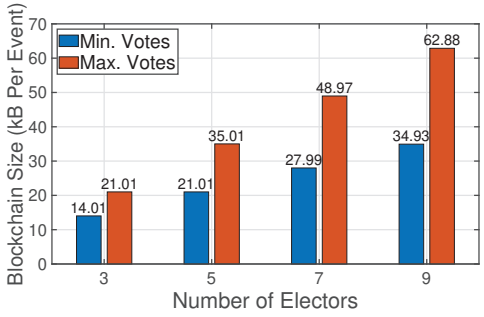


Fig. 4: Blockchain Size

peers send the query response to the client for the query request and the client verifies the signature of the peers and the query responses to determine whether the query responses are same. The client does not need to send the query response to OSP. The peers send the transaction endorsement for the transaction proposal to the client. The endorsements are verified by the client and the client sends the transaction proposal and endorsements within a single transaction message to OSP. OSP orders the transactions and create block. The block is delivered to all the peers on the channel.

7 EXPERIMENTAL RESULT

In this section, we present the analyses of BBRM and our system to show that it is appropriate for the SCMS application. We analyze the transaction scalability in relation to the SCMS application, blockchain size, resource consumption, throughput, latency, and round trip time (RTT) of the system.

7.1 Transaction Scalability and Blockchain Size Rate

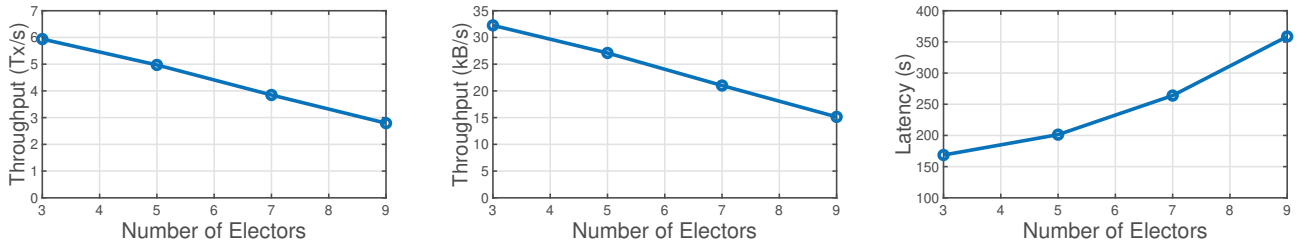
BBRM is for the EBRM within SCMS and not for the outputs of SCMS (vehicle certificates). Therefore, the application does not require high transaction scalability. SCMS is designed and recommends the update of the RCA every three years. The validity period of the RCA certificate is actually 40 years but SCMS recommends 3 years, which is shorter than the validity period of the RCA certificate [5].

In this section, we study the transacation scalability and the blockchain size rate in kB per event. *Min. Votes* and *Max. Votes* define the minimum and maximum number of votes in an event needed for a certificate to be valid or invalid. In each scenario (Min. Votes and Max. Votes for 3,5,7 and 9 electors) the average of 10 events has been taken. It can be said that the blockchain size grows very slowly over time considering the blockchain size for four different scenarios (3, 5, 7 and 9 electors) with Min. Votes and Max. Votes in Fig. 4. From our analysis of 3 to 9 electors for Min. Votes and Max. Votes it can be said that if the increase of the number of elector follows the pattern $\{p, p+2, (p+2)+2, (p+4)+2 \cdots\}$ then the blockchain size increases ≈ 7 kB per increment and follows the pattern of $\approx \{7 \times p, 7 \times p + 2, 7 \times (p+2) + 2, 7 \times (p+4) + 2, \cdots\}$ kB respectively where $p = 3$.

BBRM focusing on the root management of SCMS has the transaction scalability below most other blockchain implementations and applications and its scalability requirement is supported by the default Hyperledger Fabric. Although the performance of Hyperledger Fabric blockchain depends on many variables such as block size, transaction size, network size, hardware configuration etc, the Hyperledger Fabric ledger growth rate can scale up to 20,000 transactions per second with 2.9 kB per transaction size [10], which corresponds to 58,000 kB/s or 1.829×10^{12} kB/year. In BBRM, for each RC generation/revocation event and the corresponding EC events (required for the RC generation/revocation), the ledger growth rate ranges from 5.9×10^{-7} kB/s to 6.65×10^{-6} kB/s or 18.68 kB/year to 209.6 kB/year, depending on the number of electors (3 vs. 9) and the number of votes (Min vs. Max). These transaction scalability grows linearly with the number of RC generation and revocation events. If focusing only on the RC event without the EC events, then the ledger growth rate ranges from 1.5×10^{-7} to 6.6×10^{-7} or 4.7304 kB/year to 20.81376 kB/year. The per-event scalability for BBRM is significantly less (i.e., ten orders of magnitude smaller) than the capacity supported by Hyperledger Fabric.

7.2 Transaction Throughput

Transaction throughput can measure how fast the blockchain system is, in case of blockchain, it is the rate at which valid transactions are committed by the blockchain network for a defined period of time. In our case, we measure the transaction throughput of the Voting function since the valid transactions are sent using the Voting function can change the blockchain state. The transaction throughput is calculated in both transactions per second (Tx/s) and kilobytes per second (kB/s). We have calculated the transaction throughput taking the average of five iterations. In each iteration, 1000 transactions are generated. From Fig. 5a and Fig. 5b, it can be seen that the transaction throughput for 3,5,7 and 9 electors are $\approx$ 6,5,4,3 Tx/s and 32, 27, 21, 16 kB/s respectively. It can be derived from the analysis that when the number of electors increases to 3x, the transaction throughput rate reduces to approximately half. Considering the blockchain size of 3,5,7 and 9 electors for Min. Votes and Max.



(a) Transaction Throughput (Tx/s) **(b) Transaction Throughput (kB/s)** **(c) Transaction Latency (s)**

Fig. 5: Transaction Performances: Throughput and Latency

Votes in Fig. 4, it can be said that the transaction throughput is fast enough to support the functionalities of BBRM blockchain. For example, if we consider the 9 elector case, which has the slowest transaction throughput rate and maximum blockchain size for one event among the 3,5,7 an 9 electors, the transaction throughput is ≈ 16 kB/s. In this case, for one event, the blockchain size for Max. Votes and Min. Votes are 62.88 kB and 34.93 kB respectively. Thus, it is needed only few seconds to complete one event.

7.3 Transaction Latency

In blockchain, transaction latency is the amount of time needed for a transaction to be accepted in the network. This includes the time from the point of submitting a transaction by a client to the point the transaction is widely available in the network. More specifically, transaction latency includes the time of submitting a transaction by a client to peers, executing the transaction by peers, sending the response by peers to the client, broadcasting the transaction and response to the OSP by client and deliver the transaction and response to all the peers by OSP. The transaction latency is calculated taking the average of five iterations. In each iteration, 1000 transactions are generated. From Fig. 5c, it can be seen that the transaction latency for 3,5,7 and 9 electors are 168.606s, 201.276s, 263.974s and 358.658s respectively. It can be derived from the analysis of 3 to 9 electors that the transaction latency increases to more than half when the number of electors increases to 3x and the difference in transaction latency between two consecutive set of electors, i.e., 3 and 5 electors, increases when the number of elector increases.

7.4 Resource Consumption

We have also calculated the resource consumption (CPU usage, RAM) for running the blockchain in each elector node for four different scenarios. Electors can have different applications for running all the functions of an elector where the resource is consumed. This blockchain application will also consume resources. Thus the measurement can give an idea of the configuration needed for running the elector. We have used 2048 MB RAM for each of our virtual machines. We have analyzed the percentage of RAM used in each machine for four different scenarios (3,5,7, and 9 electors). It can be observed that when the number of electors increases to 3x, the percentage of RAM usage also increases to more than 3x, but still it is 31.8% of 2048 MB RAM. However, the percentage of CPU usage is almost simliar in all the four scenarios. The results are shown in Table 5.

Number of Electors	RAM (%)	CPU (%)
3	10	4.20
5	18	4.40
7	21.90	4.40
9	31.80	4.20

Table 5: Resource Consumption

8 RELATED WORK

In this section, we describe about the related work of blockchain for decentralized PKI and blockchain for vehicular networking.

Blockchain for Decentralized PKI Blockchain is used to build a decentralized PKI which builds resilience against an authority failure or compromise. In [22], a blockchain-based PKI framework is proposed which introduces X.509 hybrid certificate to manage PKI and smart contract to validate the chain of trust for a given certificate. A smart contract based web of trust model is proposed in [1] to verify or vouch for the attributes of identity and to detect rogue certificates by publishing all the certificates to the network immediately. In [20], the subject (i.e., the web server) of the certificate publishes the CA-signed certificate as a form of transaction and miners add the transaction to the append-only blockchain by verifying the chain of the certificate. In [2], the authors propose Blockstack, a naming and storage system which uses underlying Bitcoin Blockchain to bind the names with the associated public keys. In [15], Instant Karma PKI (IKP) is proposed to respond to CA misbehaviour using an incentive based approach and report unauthorized certificates using smart-contract based blockchain system. These related works consider a single authority for constructing a certificate whereas, in our approach, we consider the authority compromise. We use a distributed mechanism so that the integrity is still preserved even if one authority is compromised.

Blockchain for Vehicular Networking In [23], a blockchain-based rating system is proposed based on the messages exchanged by the vehicles where the credibility of the received messages from neighbouring vehicles is measured using the trust score provided by vehicles. The authors in [11] have utilized consortium blockchain and smart contract to prevent data sharing without authorization by achieving secure data sharing and data storage in vehicular networks. In [13], The authors proposed blockchain-based privacy

preserving announcement network, *CreditCoin* to motivate users with incentives to share traffic information while maintaining privacy. To address the privacy and security concerns in carpooling such as leakage of users' privacy sensitive information, uploading false location by malicious users the authors in [14] approached a blockchain-based privacy-preserving carpooling scheme on vehicular fog computing to support data auditability, one-to-many matching, destination matching and conditional privacy. In [17], an incentive-punishment mechanism is proposed to motivate vehicles to report authenticate traffic information where the reward and punishment records are stored in the blockchain. In [24], data security sharing and storage system based on the consortium blockchain (DSSCB) is proposed to address the problem of uploading data to a trusted centralized database which is vulnerable to malicious tampering and data leakage.

9 CONCLUSION

Security Credential Management System (SMCS) is a highly advanced PKI for vehicular networking involving multiple authorities and distributed operations. SCMS uses Elector-Based Root Management (EBRM) which provides the initial root of trust for the SCMS PKI. Due to its importance to the rest of the SCMS operations and eventually to the vehicular communications (where vehicles are the clients of the SCMS PKI), we advance the EBRM design and propose and build Blockchain-Based Root Management (BBRM). BBRM advances the two existing functionalities of SCMS EBRM (the root certificate generation and the elector membership control) but increases the security and integrity by utilizing the benefits provided by blockchain (high transaction integrity and distributed operations for greater compromise and failure resiliency). Our experiments based on Hyperledger Fabric show that BBRM's transaction performances and the execution overheads are appropriate for the SCMS root management. In this work, we conduct an application-specific study building on the current SCMS design for vehicular networking PKI. Future directions include generalizing the BBRM design for a generalized framework and constructing BBRM adaptations to apply it to other advanced PKI designs relying on multiple authorities for distributed operations, e.g., for privacy or compromise-resiliency purposes.

ACKNOWLEDGEMENT

This research is supported in part by Colorado State Bill 18-086. We would also like to thank the anonymous reviewers for their helpful feedback.

REFERENCES

[1] Mustafa Al-Bassam. 2017. SCPKI: a smart contract-based PKI and identity system. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. ACM, 35–40.

[2] Muneeb Ali, Jude Nelson, Ryan Shea, and Michael J Freedman. 2016. Blockstack: A global naming and storage system secured by blockchains. In *2016 {USENIX} Annual Technical Conference ({USENIX}{ATC} 16)*. 181–194.

[3] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. 2018. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*. 1–15.

[4] Benedikt Brecht. 2018. Impact on EE Storage. https://wiki.campllc.org/display/SCP/Impact+on+EE+Storage. [Online; accessed 16-June-2020].

[5] Benedikt Brecht. 2020. SCMS Proof–of–Concept Implementation. https://www.its.dot.gov/pilots/pdf/SCMS_POC_EE_Requirements20160111_1655.pdf. [Online; accessed 16-June-2020].

[6] Benedikt Brecht and Thorsten Hehn. 2019. A security credential management system for V2X communications. In *Connected Vehicles*. Springer, 83–115.

[7] Benedikt Brecht, Dean Therriault, André Weimerskirch, William Whyte, Virendra Kumar, Thorsten Hehn, and Roy Goudy. 2018. A security credential management system for V2X communications. *IEEE Transactions on Intelligent Transportation Systems* 19, 12 (2018), 3850–3871.

[8] Joakim Brorsson, Paul Stankovski Wagner, and Martin Hell. 2018. Guarding the Guards: Accountable Authorities in VANETs. In *2018 IEEE Vehicular Networking Conference (VNC)*. IEEE, 1–4.

[9] Chang-Wu Chen, Sang-Yoon Chang, Yih-Chun Hu, and Yen-Wen Chen. 2017. Protecting vehicular networks privacy in the presence of a single adversarial authority. In *2017 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 1–9.

[10] Christian Gorenflo, Stephen Lee, Lukasz Golab, and Srinivasan Keshav. 2019. Fastfabric: Scaling hyperledger fabric to 20,000 transactions per second. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 455–463.

[11] Jiawen Kang, Rong Yu, Xumin Huang, Maoqiang Wu, Sabita Maharjan, Shengli Xie, and Yan Zhang. 2018. Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Internet of Things Journal* 6, 3 (2018), 4660–4670.

[12] Jay Kreps, Neha Narkhede, Jun Rao, et al. 2011. Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, Vol. 11. 1–7.

[13] Lun Li, Jiqiang Liu, Lichen Cheng, Shuo Qiu, Wei Wang, Xiangliang Zhang, and Zonghua Zhang. 2018. Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles. *IEEE Transactions on Intelligent Transportation Systems* 19, 7 (2018), 2204–2220.

[14] Meng Li, Liehuang Zhu, and Xiaodong Lin. 2018. Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing. *IEEE Internet of Things Journal* 6, 3 (2018), 4573–4584.

[15] Stephanos Matsumoto and Raphael M Reischuk. 2017. IKP: Turning a PKI around with decentralized automated incentives. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 410–426.

[16] Diego Ongaro and John Ousterhout. 2016. In search of an understandable consensus algorithm (extended version). *Retrieved July* 20 (2016), 2018.

[17] Yuwen Pu, Tao Xiang, Chunqiang Hu, Arwa Alrawais, and Hongyang Yan. 2020. An efficient blockchain-based privacy preserving scheme for vehicular social networks. *Information Sciences* (2020).

[18] Md Sadek Ferdous, Mohammad Jabed Morshed Chowdhury, Mohammad A Hoque, and Alan Colman. 2020. Blockchain Consensus Algorithms: A Survey. *arXiv* (2020), arXiv–2001.

[19] Virendra Kumar and Benedikt Brecht. 2018. Elector-based Root Management. https://wiki.campllc.org/display/SCP/Elector-based+Root+Management. [Online; accessed 16-June-2020].

[20] Ze Wang, Jingqiang Lin, Quanwei Cai, Qiongxiao Wang, Jiwu Jing, and Daren Zha. 2018. Blockchain-based certificate transparency and revocation transparency. In *International Conference on Financial Cryptography and Data Security*. Springer, 144–162.

[21] William Whyte, André Weimerskirch, Virendra Kumar, and Thorsten Hehn. 2013. A security credential management system for V2V communications. In *2013 IEEE Vehicular Networking Conference*. IEEE, 1–8.

[22] Alexander Yakubov, Wazen Shbair, Anders Wallbom, David Sanda, et al. 2018. A blockchain-based pki management framework. In *The First IEEE/IFIP International Workshop on Managing and Managed by Blockchain (Man2Block) colocated with IEEE/IFIP NOMS 2018, Tapei, Tawain 23-27 April 2018*.

[23] Zhe Yang, Kan Yang, Lei Lei, Kan Zheng, and Victor CM Leung. 2018. Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet of Things Journal* 6, 2 (2018), 1495–1505.

[24] Xiaohong Zhang and Xiaofeng Chen. 2019. Data security sharing and storage based on a consortium blockchain in a vehicular ad-hoc network. *IEEE Access* 7 (2019), 58241–58254.