DECENT: Decentralized and Efficient Key Management to Secure Communication in Dense and Dynamic Environments

Marcus de Ree, *Member, IEEE,* Georgios Mantas, *Member, IEEE,*
Jonathan Rodriguez, *Senior Member, IEEE,* and Ifiok E. Otung

***Abstract*—Intelligent Transportation Systems (ITS), one aspect of the Smart City paradigm, aim to improve the efficiency, convenience, and safety of travelers. The integration of (vehicular) communication technologies allows communication between the on-board communication units (OBUs) of vehicles, roadside units (RSUs), and vulnerable road users (VRUs), and contribute to the efficacy of ITS applications. However, these additional sources of information must be reliable and accurate. Security primitives such as confidentiality, integrity, and authenticity are required, but only achievable when supported with a suitable cryptographic key management scheme. This paper presents the design of a decentralized and efficient key management scheme, abbreviated as the DECENT scheme. This scheme provides secure multi-hop communication in dense and dynamic network environments while functioning in a self-organized manner. Through threshold secret sharing techniques, network nodes act as a distributed trusted third party (TTP) such that a threshold number of nodes can collaborate to execute key management functions. These functions include decentralized node admission and key updating. Novelties include (i) the unique self-healing characteristic, meaning that DECENT is capable of independently recovering from network compromise, and (ii) guidelines for choosing an appropriate security threshold in any deployment scenario which maximizes the level of security while simultaneously guaranteeing that decentralized key management services can be provided.**

***Index Terms*—Ad Hoc Networks, Decentralized Systems, Key Management, Multi-hop Communication, Security.**

I. Introduction

SMART Cities are integrating advanced information and communication technologies (ICT) to improve the lives of its residents. Intelligent Transportation Systems (ITS) are one aspect of the Smart City paradigm, aiming to improve the efficiency, convenience, and safety of travelers (e.g., drivers, cyclists, pedestrians). Traditional systems gather data through sensors, cameras, and radar to monitor traffic conditions and provide travelers with information through visual signals. Nowadays, vehicles equipped with on-board communication units (OBUs) can utilize vehicular communication technologies which enable vehicle-to-vehicle (V2V) as well as vehicle-to-infrastructure (V2I) communication. These intelligent machines, able to process and share information, are therefore capable of enhancing the efficacy of ITS applications [1].

The utilization of vehicular communication technologies for ITS applications in Smart Cities faces both security and privacy challenges, hindering its practical implementation [2], [3]. The information exchanged between vehicles (i.e., OBUs), roadside units (RSUs), and vulnerable road users (VRUs) has to follow security primitives, such as data confidentiality, integrity, and authentication [4]. However, the integration of these security primitives relies on secure secret keys [5].

This paper describes a novel and decentralized key management scheme that can support cryptographic schemes (e.g., encryption and authentication schemes) to achieve the aforementioned security primitives. The deployment of the key management scheme is not limited to ITS applications or vehicular ad hoc networks; the key management serves other dense and dynamic network environments, such as mobile ad hoc networks, dynamic sensor networks [6], mobile small cells [7], or a combination of these, equally well.

A. Contributions

This paper describes a key management scheme that enables secure multi-hop communication and is flexible in terms of deployment scenarios. The key management scheme has been designed to function well in networks that are (i) deployed in an ad-hoc manner, (ii) independent of network infrastructure, (iii) dynamic in nature, (iv) scalable in terms of the number of participating nodes, and (v) capable of maintaining security long-term. The main contributions are as follows:

- The detailed description of our novel key management scheme, capable of supporting cryptographic schemes (e.g., encryption and authentication schemes) to secure multi-hop communication in dense and dynamic network environments. Details are given in Section IV.
- The key management scheme achieves the unique self-healing characteristic (i.e., capable of independently recovering from network compromise) due to our protocol design choices. Details are given in Section IV-F.
- The reclassification of trust levels for distributed trusted third party (TTP)-based schemes. Namely, the trust levels are directly related to the malicious capabilities of a TTP in case of compromise. Details are given in Section V-C.

Marcus de Ree and Jonathan Rodriguez are with the Instituto de Telecomunicações, 3810-193 Aveiro, Portugal, and also with the Faculty of Computing, Engineering and Science, University of South Wales, Pontypridd CF37 1DL, U.K. (e-mail: mderee@av.it.pt; jonathan@av.it.pt).

Georgios Mantas is with the Instituto de Telecomunicações, 3810-193 Aveiro, Portugal, and also with the Faculty of Engineering and Science, University of Greenwich, Chatham Maritime ME4 4TB, U.K. (e-mail: gimantas@av.it.pt).

Ifiok E. Otung is with the Faculty of Computing, Engineering and Science, University of South Wales, Pontypridd CF37 1DL, U.K. (e-mail: ifiok.otung@southwales.ac.uk).

- A comprehensive security and communication overhead comparison between our and related key management schemes. Details are given in Sections VI and VII.
- Improved guidelines for choosing security thresholds which maximizes the security and simultaneously guarantees the proper operation of the key management. Guidelines are provided for diverse deployment scenarios. Details are given in Section VIII.

B. Paper Outline

This paper is outlined as follows. The related works are covered in Section II. The preliminaries are covered in Section III. Our key management scheme is covered in Section IV. The security analysis of our key management scheme is covered in Section V. The security and overhead comparison between our and related schemes are covered in Sections VI and VII. Guidelines for practical implementations are covered in Section VIII. Future work is covered in Section IX. Finally, our conclusions are covered in Section X.

II. Related Work

The recent survey [8] explored a wide range of decentralized key management solutions (i.e., a collective of decentralized key management schemes that follow a similar approach) and evaluated these against requirements such as security, overhead, scalability, and sustainability. Their findings showed that the fully distributed TTP (FD-TTP)-based key management solution, illustrated in Figure 1, is favored to secure dense and dynamic network environments.

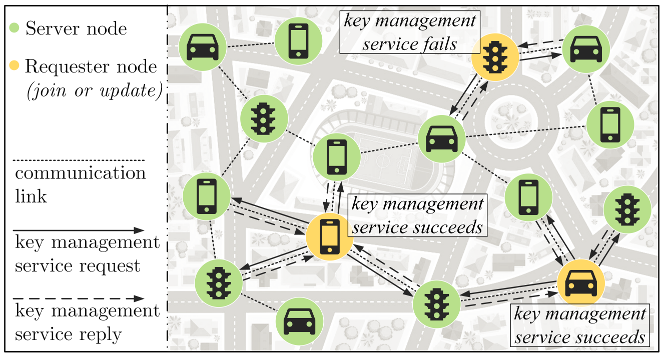

● Server node
● Requester node (join or update)
communication link
key management service request
key management service reply
key management service fails
key management service succeeds
key management service succeeds

Fig. 1. Illustration of a network containing 13 server nodes and 3 key management service-requesting nodes. Requester nodes require the assistance of at least (the threshold) 3 server nodes for the key management service to be successful.

The general idea behind the FD-TTP-based key management solution is to distribute the trust from a single centralized TTP to each network node such that a collective of network nodes can provide key management services. Technically speaking, a master private key that is usually held by the centralized TTP, used to provide a key management service (e.g., issue and sign a public key certificate or provide a node with its private key), is divided into shares using threshold secret sharing (TSS) techniques [9] and distributed among the network nodes. These (secret) shares enable network nodes to provide partial key management services such that a threshold number of these convert into the key management service as if this was directly provided by the centralized TTP. Each network node can therefore be considered to be a member of the FD-TTP. In this setting, network nodes are therefore no longer required to trust a centralized TTP to behave honestly, but instead trust that a collective of network nodes is not colluding. From a security standpoint, it is beneficial, in general, to distribute security tasks as much as possible and remove the reliance on a centralized TTP since this entity may be the target of malicious attacks which may disable or even compromise the entire system [10]. The key management services, illustrated in Figure 1, include:

1) providing a requesting node with its share of the master key (effectively joining the FD-TTP), and
2) issuing or updating the keying material of a network node (e.g., a signed public key certificate or an identity-based private key) that enables secure communication with other network nodes.

A. Conventional FD-TTP-based Key Management

The first FD-TTP-based key management scheme was proposed by Luo et al. [11], [12] and was based on the traditional public key infrastructure (PKI). Therefore, the FD-TTP took the role of a certification authority (CA) that issues signed certificates to network nodes. Their key management scheme incorporated the extensions of verifiable secret sharing (VSS) [13] and proactive secret sharing (PSS) [14], [15] to prevent malicious adversaries from disabling or compromising the system (see Section III-B). More novel schemes, proposed by Deng et al. [16] and da Silva et al. [17] were based on traditional identity-based public key cryptography (ID-PKC) and the schemes by Zhang et al. [18], Li et al. [19], Lai et al. [20], Gharib et al. [21] and de Ree et al. [22] were based on certificateless public key cryptography (CL-PKC). Therefore, the FD-TTP took the role of a private key generator (PKG) and a key generation center (KGC), respectively, to provide network nodes with their (partial) private key.

Each of these conventional FD-TTP-based key management schemes therefore rely on the distributed shares as a *tool* to provide network nodes with their respective keying materials. Therefore, they suffer from a more complicated key management design since it requires separate establishment and updating protocols for a node's share as well as its keying material. Consequently, these additional protocols contribute to a higher computational and communication overhead.

B. Alternative FD-TTP-based Key Management

An alternative construction of a FD-TTP-based key management scheme was proposed by Saxena et al. [23] by basing their design on threshold-tolerant ID-PKC [24]. Threshold-tolerant ID-PKC can be considered a translation of Feldman's VSS scheme [13]. Namely, VSS allows nodes to validate the correctness of their secret share by checking the equivalence between (i) the public share computed from its secret share, and (ii) the public share computed from publicly available witness values (also known as commitment values). Actually, a node can use the public witness values to compute *any* node's public share. Saxena [24] proposed to have the secret shares act directly as private keys, turning the corresponding public shares into the public keys. The witness values and a node's identity remain publicly available information. This allows any node to non-interactively compute the public key of any other node without having to rely on the distribution of public keys, similar to traditional ID-PKC. The main difference with traditional ID-PKC is that the master private key only remains secret as long as no adversary is able to collect a threshold number of secret shares (i.e., private keys).

The equivalence between a node's secret share and its private key means that protocols for the establishment and update of secret shares and keying material are also equivalent. This leads to a much more simplistic key management design and a significant reduction in terms of computational overhead, communication overhead, and energy consumption. Unfortunately, Saxena's FD-TTP-based key management scheme [23] was only designed for short-lived networks, lacking the PSS extension to achieve long-term security. An adversary that manages to compromise a threshold number of devices, extract their secret shares and reconstruct the master private key would be able to eavesdrop on private communications and launch identity impersonation attacks, without any fear of being detected.

III. Preliminaries

A. Computational Assumptions

In this paper, we assume computational security based on the discrete logarithm (DL) and computational Diffie-Hellman (CDH) problems in the standard setting: p and q are large primes such that $q|p-1$ and g denotes a generator of cyclic subgroup $\mathbb{G} \subset \mathbb{Z}_p^*$ of order q. For convenience, we denote $DL(k)$ as any set of triples (p, q, g) which satisfy the above constraints and where q is a k-bit prime that is sufficiently large to fend off known attacks on the discrete logarithm.

- *DL Assumption:* For every probabilistic polynomial time algorithm I and every triple $(p, q, g) \in DL(k)$ with element $x \in \mathbb{Z}_q^*$ chosen at random, the probability $Pr[I(p, q, g, g^x) = x]$ is negligible.
- *CDH Assumption:* For every probabilistic polynomial time algorithm I and every triple $(p, q, g) \in DL(k)$ with elements $x, y \in \mathbb{Z}_q^*$ chosen at random, the probability $Pr[I(p, q, g, g^x, g^y) = g^{xy}]$ is negligible.

B. Adversarial Model

Due to the distribution of trust and the associated shares of a master secret, our key management scheme must consider attacks related to the misuse of these shares. We cover two types of attacks and an inherent characteristic of public key cryptographic infrastructures that impact the security of distributed TTP-based schemes.

- *Disruptive Adversary:* A disruptive adversary [22] is a malicious server that provides a false key management service. This false key management service can be the provisioning of a false subshare during (i) the admission of new nodes or (ii) the updating of every node's share. Providing a false subshare would lead to the creation of an incorrect share. This would cause honest servers to unknowingly provide false key management services in the future, crippling the key management.
- *Mobile Adversary:* A mobile adversary [25] is a malicious node that dynamically moves through the network and compromises network nodes, one at a time, with the goal to extract and collect a threshold number of shares. If the mobile adversary is successful, it is capable of reconstructing the master secret and impersonate the distributed TTP to launch further malicious attacks. The severity of these attacks depends on the capabilities of the master secret (i.e., the trust level of the distributed TTP).
- *Trust Level of the Distributed TTP:* Girault [26] defined three trust levels as an indication of the malicious capabilities of a compromised TTP:
  1) The TTP knows (or can easily compute) a node's private key and therefore, launch identity impersonation attacks without being detected.
  2) The TTP does not know (and cannot easily compute) a node's private key but is still able to launch identity impersonation attacks without being detected.
  3) The TTP does not know (and cannot easily compute) a node's private key nor is it able to launch identity impersonation attacks without being detected.

  This trust level can be considered as a additional layer of defense. In case a mobile adversary is successful, we wish to limit its malicious capabilities.

C. Threshold Cryptography

Secret sharing takes a piece of secret data and divides this into a multitude of shares that are distributed among a group of nodes. This secret data can be reconstructed by combining a number of shares. In our key management design, we use Shamir's TSS [9] as well as Feldman's VSS extension [13] and Jarecki et al.'s PSS extension [14], [15]. The incorporation of TSS specifies that any threshold number of shares are capable of reconstructing the master secret. The incorporation of VSS allows nodes to verify the correctness of obtained data, therefore countering disruptive adversaries. The incorporation of PSS restricts a mobile adversary in their pursuit of reconstructing the secret data. These schemes rely on polynomial interpolation and are therefore theoretically secure (i.e., fewer than a threshold number of shares does not reveal any information about the secret data). The associated protocols are as follows:

- *Setup:* A trusted dealer (TD) chooses large primes p and q such that $q|p-1$ and selects a generator g of cyclic subgroup $\mathbb{G} \in \mathbb{Z}_p^*$ of order q. The TD selects a threshold t and a random polynomial $f(x) = \sum_{i=0}^{t-1} a_i \cdot x^i \in \mathbb{Z}_q^*[x]$ where $f(0) = s$ and s represents the secret data.
- *Witness Generation:* The TD computes the set of witness values $W_i \equiv g^{a_i} \pmod{p}$ for $i \in [0, t-1]$ and publishes these in some public domain.

- *Share Distribution:* The TD computes the share $s_i = f(id_i)$ for node N_i with identity id_i and transmits this securely.
- *Share Verification:* Any node N_i with identity id_i can verify that its share s_i is correct by checking the following formula: $g^{s_i} \equiv \prod_{j=0}^{t-1} W_j{}^{(id_i{}^j)} \pmod p$.
- *Secret Reconstruction:* Any group of t nodes can reconstruct the polynomial f and determine secret data s using Lagrange interpolation: $f(x) = \sum_{i=1}^{t} s_i \cdot \lambda_i \pmod q$ where $\lambda_i(x) = \prod_{j=1, j\neq i}^{t} \frac{x-id_j}{id_i-id_j} \pmod q$.
- *Share Updating:* To update the shares of all the n nodes in the network, a cluster of t nodes are chosen. Each node $N_i (1 \leq i \leq t)$ selects a random update polynomial $\delta_i(x)$ of degree $t-1$ where $\delta_i(0) = 0$. Then, each node N_i computes and distributes a subshare $\delta_i(id_j)$ for every $N_j (1 \leq j \leq n)$. When node N_j obtained all t subshares, it can compute its new share $\bar{s}_i \equiv s_i + \sum_{i=1}^{t} \delta_i(id_j) \pmod q$. The new shares correspond to the updated polynomial $\bar{f}(x) \equiv f(x) + \sum_{i=1}^{t} \delta_i(x) \pmod q$ with the secret data s still intact since $\bar{f}(0) = f(0) = s$.

IV. The DECENT Scheme

This section presents the design of our DECENT (DECentralized and Efficient key managemeNT) scheme in detail. It follows the alternative construction of a FD-TTP-based key management scheme, similar to that of Saxena [23], to minimize the complexity and overheads. Furthermore, we extend their work with a share updating protocol to increase the level of security and gain the unique self-healing characteristic to achieve long-term security.

A. Assumptions

For our key management scheme, we assume the existence of a TTP (e.g., a network administrator) to bootstrap an initial set of nodes. We assume that each node carries a unique identity that is unchanged during its lifetime and non-transferable. Every node has the same transmission range and is capable of sending unicast, multicast or broadcast messages. We define a unicast message as a message which is cryptographically secured (e.g., by means of encryption) by the sender and only one receiver has the corresponding cryptographic keying material to extract the information from the message. We define a broadcast message as a message that is at least partially sent in plaintext such that every network node within transmission range of the sender receives and can extract the plaintext information from this message. Broadcast messages are invulnerable to malicious message modification attacks when both the sender and receiver are within each other's transmission range. Finally, we assume that each of the protocols which involves communication between multiple nodes remain within each other's transmission range during the execution of the protocol.

B. Overview

The DECENT scheme consists of the following protocols.

- *Network bootstrapping:* This protocol utilizes a TTP to bootstrap an initial set of nodes by providing each of them with a share of a pre-defined master secret. This master secret is defined as a *bivariate* polynomial; thus, we utilize a modified version of Shamir's TSS scheme [9] and Feldman's VSS scheme [13]. The use of secret sharing allows the distribution of trust and the establishment of the decentralized TTP (i.e., the FD-TTP), capable of providing key management services in a decentralized manner during network operation. This share also directly defines a node's public-private key pair.
- *Node admission:* The FD-TTP, collectively in possession of at least a threshold number of shares of the master secret, is capable of admitting new nodes to the network by providing them with subshares of the master secret. The new node effectively joins the network and becomes a member of the FD-TTP once it obtained a threshold number of subshares, allowing the joining node to compute its unique share of the master secret and define its public-private key pair.
- *Pairwise key establishment:* Each pair of network nodes can estimate their pairwise key, and therefore establish a secure communication channel in a non-interactive manner, limiting the communication overhead of our key management scheme.
- *Share updating:* This protocol is a modification of Jarecki et al.'s PSS scheme [14], [15]. This protocol is triggered periodically and updates both the master secret as well as every node's share of the master secret. This protocol can be interpreted as a self-organized network reboot which restores a potential network compromise caused by a mobile adversary [25].

C. Network Bootstrapping

The network bootstrapping protocol, covered in detail in Protocol 1, starts with the TTP defining and publishing public network parameters. These will inform participating network nodes how to perform the mathematical operations (step 1). This is followed by the generation of a symmetric bivariate polynomial which defines the master secret (step 2), the public witness values (step 3), and every node's share of the master secret (i.e., a *univariate* polynomial) (step 4). We assume that the TTP has access to a secure side channel (e.g., infrared, wire) to provide each node with the public witness values and their polynomial share. Upon retrieval, the nodes use the witnesses to verify that their polynomial share is correct (step 5). To clarify, the witnesses are directly distributed to the nodes instead of publishing these in a secure and public space because the witnesses will be periodically updated by the nodes themselves and would be unable to overwrite the values of the initial witnesses. Every node with a correct polynomial share essentially joins the distributed TTP as a member and is capable of providing partial key management services to others. These partial key management services cover the provisioning of subshares to a joining node or participation during the share updating protocol. Finally, the nodes use their polynomial share to define their public-private

Network Bootstrapping Protocol

1. First, the TTP generates primes p and q such that $q|p-1$, selects generator g of cyclic subgroup $\mathbb{G} \subset \mathbb{Z}_p^*$ with order q, selects threshold t, and publishes these in a secure, public space.
2. The TTP generates a random symmetric bivariate polynomial $f(x,y)$ of degree $t-1$ with secret coefficients $\alpha_{ij} = \alpha_{ji} \in \mathbb{Z}_q^*$:
$$f(x,y) = \sum_{i=0}^{t-1}\sum_{j=0}^{t-1} \alpha_{ij} x^i y^j \in \mathbb{Z}_q^*[x,y]. \quad (1)$$
The secret coefficients are defined as the master secret and denoted as symmetric matrix $A = [\alpha_{ij}]$.
3. The TTP computes the witness values W_{ij} corresponding to secret coefficients $\alpha_{ij} \in A$:
$$W_{ij} \equiv g^{\alpha_{ij}} \pmod p \text{ for } i,j \in [0, t-1]. \quad (2)$$
The witness values are denoted as symmetric matrix $W = [W_{ij}]$.
4. The TTP computes each node's polynomial share as $s_k(x) = f(x, id_k)$, where id_k represents the identity of node N_k. Then, the TTP securely transmits $\{s_k(x), W\}$ to node N_k.
5. Each node N_k verifies that the coefficients $\beta_i \in s_k(x)$ for $i \in [0, t-1]$ correspond to the witness values:
$$g^{\beta_i} \overset{?}{\equiv} \prod_{j=0}^{t-1} W_{ij}{}^{(id_k{}^j)} \pmod p. \quad (3)$$
6. Finally, each node N_k defines its private key SK_k and its corresponding public key PK_k:
$$SK_k = s_k(0), \quad (4)$$
$$PK_k \equiv g^{SK_k} \pmod p. \quad (5)$$

Network bootstrapping is complete once at least t nodes are initialized.

Protocol 1. The network bootstrapping protocol defines the network parameters and establishes a distributed TTP through secret sharing techniques.

key pair (step 6). With the initialization of at least a threshold number of nodes, the TTP is no longer required and leaves the network indefinitely.

D. Node Admission

With the establishment of the distributed TTP, it is possible for new nodes to join the network. A node that wishes to join may need to move around and broadcast multiple requests to reach enough members (i.e., servers) of the distributed TTP and be provided with enough subshares. The details are covered in Protocol 2. The node starts by generating a temporary public-private key pair and then requests to join the network by broadcasting its identity and its temporary public key (step 1). Any server within transmission range can compute a subshare and respond to the joining node by providing it with its subshare as well as the public witnesses. The witnesses are broadcasted in plaintext such that other nearby servers can verify that it has not provided false witnesses (step 2). The node can use these witnesses to verify whether the provided subshares are correct (step 3). Once the node received a threshold number of correct partial shares, it computes its polynomial share (step 4) and defines its public-private key pair (step 5).

Node Admission Protocol

1. First, node N_l computes its temporary public key PK_l from a random temporary private key $SK_l \in \mathbb{Z}_q^*$:
$$PK_l \equiv g^{SK_l} \pmod p. \quad (6)$$
The node N_l broadcasts $\{id_l, PK_l, \sigma_{SK_l}(id_l, PK_l)\}$, where $\sigma(\cdot)$ represents a signature. The broadcast is received by nearby server nodes, denoted as $N_k \in \Omega$.
2. Each server $N_k \in \Omega$ verifies the sigature and, upon verification, computes the subshare $s_{k\to l}$:
$$s_{k\to l} = s_k(id_l) \equiv \sum_{i=0}^{t-1} \beta_i id_l{}^i \pmod q. \quad (7)$$
The server N_k then broadcasts $\{id_k, W, E_{PK_l}(s_{k\to l}), \sigma_{SK_k}(id_k, W, E_{PK_l}(s_{k\to l}))\}$, where $E(\cdot)$ represents a ciphertext.
3. The node N_l computes the public key PK_k from identity id_k and witness values W:
$$PK_k \equiv \prod_{i=0}^{t-1} W_{i0}{}^{(id_k{}^i)} \pmod p, \quad (8)$$
such that node N_l can verify the signature. Upon verification, node N_l decrypts the ciphertext and verifies its extracted subshare $s_{k\to l}$:
$$g^{s_{k\to l}} \overset{?}{\equiv} \prod_{i=0}^{t-1}\prod_{j=0}^{t-1} W_{ij}{}^{(id_l{}^i \cdot id_k{}^j)} \pmod p. \quad (9)$$
4. Once node N_l obtained t subshares (potentially by broadcasting multiple requests over time), it uses Gaussian elimination [30] to compute the coefficients β_i for $i \in [0, t-1]$ of its polynomial share $s_l(x)$.
$$\left[\begin{array}{ccc|c} id_1{}^0 & \cdots & id_1{}^{t-1} & s_{1\to l} \\ \vdots & \ddots & \vdots & \vdots \\ id_t{}^0 & \cdots & id_t{}^{t-1} & s_{t\to l} \end{array}\right] \rightarrow \left[\begin{array}{c|c} & \beta_0 \\ I_{t\times t} & \vdots \\ & \beta_{t-1} \end{array}\right] \quad (10)$$
5. Finally, node N_l defines its private key SK_l and its corresponding public key PK_l as described in equations (4) and (5), respectively.

Protocol 2. The node admission protocol allows new nodes to join the network and become a new member of the distributed TTP.

E. Pairwise Key Establishment

The use of bivariate polynomials in our scheme allows network nodes to establish a secure communication channel in a non-interactive manner. The details are covered in Protocol

Pairwise Key Establishment Protocol

1. Suppose that network nodes N_k and N_l wish to securely communicate. They compute keys $K_{k,l}$ and $K_{l,k}$, respectively, through polynomial evaluation:
$$K_{k,l} = s_k(id_l) = f(id_l, id_k) \quad (11)$$
$$K_{l,k} = s_l(id_k) = f(id_k, id_l) \quad (12)$$
The keys $K_{k,l}$ and $K_{l,k}$ are equal due to the symmetric property $f(x,y) = f(y,x)$.

For secure communication, it is good practice to employ key separation. The resulting encryption & decryption key $K_{k,l}^{enc}$ and signing & signature verification key $K_{k,l}^{mac}$ can be used to perform authenticated encryption.

Protocol 3. The pairwise key establishment protocol allows network nodes to determine their shared symmetric key.

3. Each network node can evaluate their polynomial share to estimate their pairwise symmetric key (step 1). The computed key is symmetric since the master polynomial has the symmetric property that $f(x,y) = f(y,x)$. This pairwise symmetric key allows multi-hop (i.e., end-to-end) secure communication since a message encrypted at the transmitting end can only be decrypted at the receiving end by the node that the pairwise key is shared with [27]. To prevent any security vulnerabilities, we advise the use of key separation [28] through an appropriate *key derivation function* (KDF) to establish two unique symmetric keys of appropriate bit-length. These keys can then be used to perform authenticated encryption.

Our pairwise key establishment is similar to that of Saxena [23], [29]. However, Saxena claims that this pairwise key establishment provides unconditional security, i.e., not based on any complexity assumption. We can prove that, unlike Saxena's claim, the pairwise key establishment is secure under the DL assumption. Utilizing the public witness values, any node would be able to compute the following:
$$g^{K_{k,l}} \equiv \prod_{i=0}^{t-1}\prod_{j=0}^{t-1} W_{ij}{}^{(id_l{}^i \cdot id_k{}^j)} \pmod p. \quad (13)$$
The secrecy of the shared pairwise key between any two network nodes is therefore not unconditionally secure, but secure under the DL assumption. The shared pairwise keys would only provide unconditional security if the scheme did not disclose the public witnesses.

Share Updating Protocol

1. First, each node in a cluster of network nodes, denoted $N_k \in \Omega$ where $|\Omega| = t$, generates a random symmetric bivariate update polynomial $f_k(x,y)$ of degree $t-1$ with random coefficients $\alpha_{ij} = \alpha_{ji} \in \mathbb{Z}_q^*$:
$$f_k(x,y) = \sum_{i=0}^{t-1}\sum_{j=0}^{t-1} \alpha_{ij} x^i y^j \in \mathbb{Z}_q^*[x,y]. \quad (14)$$
The random coefficients generated by node N_k are defined as its update secret and denoted as symmetric matrix $A_k = [\alpha_{ij}]$.
2. Each node $N_k \in \Omega$ computes their update witnesses W_{ij}^k corresponding to random coefficients $\alpha_{ij} \in A_k$:
$$W_{ij}^k \equiv g^{\alpha_{ij}} \pmod p \text{ for } i,j \in [0, t-1]. \quad (15)$$
The update witnesses of node N_k are denoted as symmetric matrix $W_k = [W_{ij}^k]$.
3. Each node $N_k \in \Omega$ computes for every node $N_l \in \Omega$ (including itself), an update polynomial share:
$$s_{k \to l}(x) = f_k(x, id_l). \quad (16)$$
The node N_k then broadcasts $\{W_k\}$ and unicasts $\{s_{k\to l}\}$ to each node $N_l \in \Omega$ using authenticated encryption.
4. Each node $N_k \in \Omega$ verifies that the coefficients $\beta_i \in s_{l\to k}(x)$ for $i \in [0, t-1]$ correspond to the update witnesses W_{ij}^l:
$$g^{\beta_i} \overset{?}{\equiv} \prod_{j=0}^{t-1} W_{ij}^{l\ (id_k{}^j)} \pmod p. \quad (17)$$
5. Finally, each node $N_k \in \Omega$ computes its updated polynomial share $\bar{s}_k(x)$ and the symmetric matrix $\bar{W}$ of updated witness values $\bar{W}_{ij}$ for $i,j \in [0, t-1]$:
$$\bar{s}_k(x) \equiv s_k(x) + \sum_{N_l \in \Omega} s_{l \to k}(x) \pmod q, \quad (18)$$
$$\bar{W} = [\bar{W}_{ij} \equiv W_{ij} \cdot \prod_{N_l \in \Omega} W_{ij}^l \pmod p]. \quad (19)$$

Network nodes that are not part of Ω will have an outdated polynomial share at the end of this protocol. These nodes would have to execute the node admission protocol (i.e., Protocol 2) to obtain their updated polynomial share.

Protocol 4. The technical details of the share updating protocol.

F. Share Updating

The share updating protocol, covered in detail in Protocol 4, is executed on a periodic basis by a threshold number of server nodes within each other's transmission range. This requirement places an upper bound on the height of the threshold for a particular network (see section VIII). Each server in this cluster generates a random symmetric bivariate update polynomial which defines its update secret (step 1), its update witnesses (step 2), and update polynomial shares (step 3). Each pair of server nodes securely exchanges update polynomial shares and broadcasts its update witnesses. The update witnesses are broadcasted in plaintext to prevent a malicious server from cheating (e.g., by providing different servers with update polynomial shares created from different update secrets). Every server node then verifies the received update polynomial shares (step 4). Once the server nodes received all the update polynomial shares, they combine these to obtain their updated polynomial share and the updated set of witnesses (step 5). After the successful execution of the share updating protocol, the network nodes with an updated polynomial share can broadcast a notification that other nearby network nodes can request them to have their polynomial share updated by executing the node (re)admission protocol.

In contrast to the closely related key management schemes with a share updating protocol [11], [12], [19], [22], our protocol is unique in the fact that it allows the periodic updating of *all* the coefficients of the update polynomial[1]. Meaning that the compromise of the master secret would be negated after the execution of our share updating protocol. We call this unique characteristic *self-healing*, and DECENT is the first FD-TTP-based key management scheme to benefit from this property. Therefore, the incorporation of this share updating protocol provides significant security benefits.

V. Security Analysis

In this section, we evaluate the level of resiliency that our DECENT scheme achieves against disruptive adversaries (V-A) and mobile adversaries (V-B). Furthermore, we explore the security guarantees of DECENT in case of network compromise based on the achieved trust level (V-C).

A. Security Evaluation against Disruptive Adversaries

In the following two theorems, we prove that our DECENT scheme is resilient against disruptive adversaries that attempt to provide a false key management service.

Theorem 1. *Any individual joining node N_l can detect which server(s) $N_k \in \Omega$ provided a malicious key management service during the execution of the node admission protocol, i.e., the joining node N_l can verify that provided subshare $s_{k\to l}$ from server N_k is correct.*

Proof. We prove this theorem by showing that the public witness values $W_{ij} \in W$ can be utilized to pre-compute the public equivalent of subshare $s_{k\to l}$ that joining node N_l should received from server $N_k \in \Omega$. We prove that the key management service from server $N_k \in \Omega$ has been trustworthy if and only if server $N_k \in \Omega$ honestly evaluated its polynomial share $s_k(x)$ in computing the subshare $s_{k\to l}$ through the following series of mathematical equivalences:

$$\prod_{i=0}^{t-1}\prod_{j=0}^{t-1} W_{ij}{}^{(id_l{}^i\cdot id_k{}^j)} \equiv \prod_{i=0}^{t-1}\prod_{j=0}^{t-1} g^{(\alpha_{ij}\cdot id_l{}^i\cdot id_k{}^j)} \pmod p \quad (20)$$

$$\equiv g^{(\sum_{i=0}^{t-1}\sum_{j=0}^{t-1}\alpha_{ij}\cdot id_l{}^i\cdot id_k{}^j)} \pmod p \quad (21)$$

$$\equiv g^{f(id_l,id_k)} \pmod p \quad (22)$$

$$\equiv g^{s_k(id_l)} \pmod p \quad (23)$$

$$\equiv g^{s_{k\to l}} \pmod p \quad (24)$$

□

Theorem 2. *Any individual server node N_l can detect which server(s) $N_k \in \Omega$ provided a malicious key management service during the execution of the share updating protocol, i.e., the server node N_l can verify that provided update polynomial share $s_{k\to l}(x)$ from server N_k is correct.*

Proof. We prove this theorem by showing that the update secret A_k, corresponding to the broadcasted witnesses W_k, of server N_k were used in the generation of the update polynomial share $s_{k\to l}(x)$. First, we rewrite the update polynomial share $s_{k\to l}(x)$ as follows:

$$s_{k\to l}(x) \equiv f_k(x, id_l) \pmod q \quad (25)$$

$$\equiv \sum_{i=0}^{t-1}\sum_{j=0}^{t-1}\alpha_{ij}\cdot x^i\cdot id_l{}^j \pmod q \quad (26)$$

$$\equiv \sum_{i=0}^{t-1}(\sum_{j=0}^{t-1}\alpha_{ij}\cdot id_l{}^j)\cdot x^i \pmod q \quad (27)$$

$$\equiv \sum_{i=0}^{t-1}\beta_i\cdot x^i \pmod q \quad (28)$$

In the above series of mathematical equivalences, we found the following relationship between the coefficients of the obtained update polynomial share $\beta_i \in s_{k\to l}(x)$ and the update secret A_k:

$$\beta_i \equiv \sum_{j=0}^{t-1}\alpha_{ij}\cdot id_l{}^j \pmod q \text{ for } i \in [0, t-1] \quad (29)$$

The server node N_l can use the witnesses W_k to verify that the coefficients $\beta_i \in s_{k\to l}(x)$ are created from the update secret A_k, thereby verifying that the update polynomial share $s_{k\to l}(x)$ has been generated honestly.

$$g^{\beta_i} \equiv g^{\sum_{j=0}^{t-1}\alpha_{ij}\cdot id_l{}^j} \pmod p \quad (30)$$

$$\equiv \prod_{j=0}^{t-1}(g^{\alpha_{ij}})^{id_l{}^j} \pmod p \quad (31)$$

$$\equiv \prod_{j=0}^{t-1} W_{ij}^{k\,id_l{}^j} \pmod p \quad (32)$$

□

B. Security Evaluation against Mobile Adversaries

In the following theorem, we prove that our DECENT scheme is resilient against mobile adversaries under the assumption that share updating phases are executed prior to any mobile adversary compromising and extracting a threshold number of polynomial shares. We prove that a mobile adversary is unable to uncover the master secret $A = [\alpha_{ij}]$.

Theorem 3. *An adversary who knows fewer than the threshold number of polynomial shares, collected in between the execution of two consecutive share updating protocols, cannot determine the master secret $A = [\alpha_{ij}]$.*

Proof. We prove this theorem by contradiction. We denote that the adversary gathered $m < t$ polynomial shares in between

[1]The related schemes, all following the conventional FD-TTP-based key management structure, chose not to update the leading coefficient (i.e., the master private key) since this updates the master key pair and causes every node's keying material to become outdated. These schemes therefore prevented every node from having to update its keying material following the share updating protocol. Thus, they had to make a trade-off between overhead and achieving self-healing.

the execution of two consecutive share updating protocols. We denote the polynomial share of a network node N_k as follows:

$$s_k(x) \equiv \sum_{i=0}^{t-1} \beta_i^{(k)} \cdot x^i \pmod q \text{ where } \beta_i^{(k)} \equiv \sum_{j=0}^{t-1} \alpha_{ij} \cdot {id_k}^j \tag{33}$$

For any fixed value of i (i.e., selecting the i^{th} coefficient of each collected polynomial share), the mobile adversary obtains the following system of linear equations:

$$\begin{cases} \beta_i^{(1)} = \alpha_{i0} + \alpha_{i1} \cdot id_1 + \alpha_{i2} \cdot {id_1}^2 + \cdots + \alpha_{i|t-1} \cdot {id_1}^{t-1} \\ \beta_i^{(2)} = \alpha_{i0} + \alpha_{i1} \cdot id_2 + \alpha_{i2} \cdot {id_2}^2 + \cdots + \alpha_{i|t-1} \cdot {id_2}^{t-1} \\ \quad \vdots \\ \beta_i^{(m)} = \alpha_{i0} + \alpha_{i1} \cdot id_m + \alpha_{i2} \cdot {id_m}^2 + \cdots + \alpha_{i|t-1} \cdot {id_m}^{t-1} \end{cases}$$

Based on the fundamental theorem of linear algebra [30], it is not possible to solve a system of m linearly independent equations with t unknowns where $m < t$. Therefore, the adversary is unable to determine any element of the master secret $A = [\alpha_{ij}]$ with fewer than t polynomial shares. □

C. Security Evaluation regarding DECENT's Trust Level

Recall that conventional FD-TTP-based key management schemes establish a distributed TTP (e.g., a distributed CA, PKG, or KGC) of which its members use their shares of a master private key to perform a key management service (e.g., creating a signature for a public key certificate or compute a node's identity-based (partial) private key). The honesty of the key management service can be verified with the master public key. To limit overheads associated with key updating, those schemes chose to keep the master public-private key pair intact throughout the entire network lifetime. This can have severe consequences in practical scenarios where an adversary managed to uncover the master private key. In contrast, our DECENT scheme does not require the virtual reconstruction of a master private key to perform some kind of key management service, meaning that our scheme can update all the coefficients of the master secret polynomial without it impacting key management overheads.

The ability to periodically renew the entire master secret provides significant benefits in terms of security. This periodic renewal of the master secret effectively reboots the network on a periodic basis without any intervention from an outside TTP. Consider the scenario in which a mobile adversary successfully reconstructed the master secret: this allows the adversary to successfully impersonate a malicious FD-TTP. In DECENT, a mobile adversary will only be able to impersonate the FD-TTP until the master secret is updated. This limits the window of opportunity for malicious attacks and significantly reduces the payoff for launching such attacks. By incorporating this window of opportunity, or the amount of compromising effort required by an attacker, we can redefine the trust levels of FD-TTP-based key management schemes as follows:

- The TTP knows (or can easily compute) a node's private key and launch identity impersonation attacks for an indefinite amount of time (i.e., Girault's trust level 1).
- The TTP knows (or can easily compute) a node's private key and launch identity impersonation attacks for a limited amount of time (i.e., DECENT's trust level).
- The TTP does not know (and cannot easily compute) a node's private key but is still able to launch identity impersonation attacks for an indefinite amount of time (i.e., Girault's trust level 2).
- The TTP does not know (and cannot easily compute) a node's private key, but is still able to launch identity impersonation attacks for a limited amount of time (i.e., Girault's trust level 3).

TABLE I
THE PROPOSED RECLASSIFICATION OF TRUST LEVELS FOR DISTRIBUTED TTP-BASED KEY MANAGEMENT SCHEMES.

Malicious capabilities of a compromised distributed TTP	*Indefinite compromised schemes*	*Limited-time compromised schemes*
Complete compromise[1]	Deng et al. [16] da Silva et al. [17] Saxena et al. [23]	DECENT
Partial compromise[2]	Zhang et al. [18] Li et al. [19] Gharib et al. [21]	Luo et al. [11], [12] Lai et al. [20] de Ree et al. [22]

1: Complete compromise allows the malicious TTP to compute a node's private key, compromising all communications.
2: Partial compromise limits the abilities of the malicious TTP to identity impersonation attacks.

Based on these definitions, we argue that the trust level of our DECENT scheme is higher than trust level 1 and worse than trust level 3, but not necessarily better nor worse than trust level 2. Therefore, we propose to reclassify the trust levels of distributed TTP-based schemes on the definitions above instead of Girault's original trust hierarchy [26], previously covered in the adversarial model (see Section III-B). Table I summarizes the achieved trust level of our DECENT scheme and related FD-TTP-based key management schemes based on our reclassification.

VI. SECURITY COMPARISON

In this section, we discuss the security strength of the related FD-TTP-based key management schemes and compare it to the security strength of our proposed DECENT scheme. We define the security strength as the extent of security features that are integrated into the protocols of each FD-TTP-based key management scheme. Table II summarizes our findings.

A. Resiliency against Disruptive Adversaries

Recall that we defined a disruptive adversary as a malicious server that provides a false key management service to a requesting node. The requester node may request server nodes for key management services to *establish* its share of the master secret (i.e., share establishment protocol) or its public-private key pair (i.e., key establishment protocol). The requester node may also request server nodes for key management services to *update* its share of the master secret

TABLE II
COMPARISON OF SECURITY FEATURES THAT ARE INTEGRATED INTO THE PROTOCOLS OF FD-TTP-BASED KEY MANAGEMENT SCHEMES.

FD-TTP-based Key Management Scheme	Resistance against ...		Trust Level	Self-Healing
	Disruptive Adversary	Mobile Adversary		
DECENT	4/4	✓	1-3	✓
de Ree et al. [22]	5/6	✓	3	✗
Luo et al. [11], [12]	3/6	✓	3	✗
Li et al. [19]	2/6	✓	2	✗
Saxena et al. [23]	2/2	✗	1	✗
Lai et al. [20]	0/4	✗	3	✗
Zhang et al. [18]	0/4	✗	2	✗
Gharib et al. [21]	0/4	✗	2	✗
Deng et al. [16]	0/4	✗	1	✗
da Silva et al. [17]	0/6	✗	1	✗

(i.e., share updating protocol) or its public-private key pair (i.e., key updating protocol). To provide resiliency against a disruptive adversary, requester nodes must be able to verify the honesty of a provided key management service.

- *Combined Key Management Service Verifiability*: The protocol allows a requester node to verify the correctness of the combination of a threshold number of partial key management services.
- *Partial Key Management Service Verifiability:* The protocol allows a requester node to verify the correctness of a key management service provided by an individual server node. Partial key management service verifiability induces the verifiability of the combined key management service. This level of verifiability also allows the identification of the disruptive adversaries.

Our DECENT scheme consists of two key management service protocols, the node admission and share updating protocol. Section V-A showed that the node admission protocol allows partial (and hence also combined) key management service verifiability (2 out of 2) and also that the share updating protocol allows partial (and hence also combined) key management service verifiability (2 out of 2). Therefore, we denote DECENT's resiliency against disruptive adversaries as 4 out of 4 in Table II. The resiliency against disruptive adversaries of related FD-TTP-based key management schemes were evaluated in a similar manner.

Luo et al. [11], [12] incorporated verifiability into its three distinct key management service protocols. However, nodes are only capable of verifying the combined key management service [31]. Li et al. [19] only incorporated verifiability into one of three considered key management service protocols. This protocol does achieve partial key management service verifiability. De Ree et al. [22] incorporated verifiability into its three key management service protocols but lacks partial key management service verifiability in its share establishment protocol. Saxena et al. [23] incorporated verifiability into its one key management service protocol and achieves partial key management service verifiability. The FD-TTP-based key management schemes [16]–[18], [20], [21] did not incorporate verifiability into any of its key management service protocols.

B. Resiliency against Mobile Adversaries

Recall that we defined a mobile adversary as a malicious node that compromises network nodes, extract their secret shares, with the goal to reconstruct the master secret key. A share updating protocol provides resiliency against a mobile adversary since secret shares from different updating phases are incompatible in the reconstruction of the master secret key. Therefore, resiliency against mobile adversaries is achieved by DECENT, Luo et al. [11], [12], Li et al. [19] and de Ree et al. [22] while not achieved by [16]–[18], [20], [21], [23].

C. Trust Level

The trust level of a FD-TTP suggests the amount of trust that network nodes must have in this entity and is based on its malicious capabilities in the case of compromise. These capabilities of a FD-TTP are determined by the underlying public key cryptographic infrastructure. The scheme from Luo et al. [11], [12] is based on traditional PKI and thus reaches trust level 3 [26]. The schemes from Zhang et al. [18], Li et al. [19], Lai et al. [20], Gharib et al. [21] and de Ree et al. [22] are based on CL-PKC. Depending on the key generation technique, the FD-TTP reaches either trust level 2 or 3 [32]. Namely, if the key generation technique allows a node to create only one valid key pair, as in [20], [22], the detection of multiple valid key pairs would indicate malicious behavior from the FD-TTP and thus reaches trust level 3. If the key generation technique allows a node to create multiple valid key pairs, as in [18], [19], [21], there is no way to detect any malicious behavior from the FD-TTP and thus only reaches trust level 2. The schemes from Deng et al. [16], da Silva et al. [17] and Saxena et al. [23] are based on ID-PKC and thus only reach trust level 1 since a compromised FD-TTP would be capable of computing every node's private key and launch attacks for an indefinite amount of time [26]. Even though DECENT is also based on ID-PKC, our trust level reclassification indicates that DECENT reaches a trust level between 1 and 3 since a compromised FD-TTP can only launch attacks for a limited amount of time (see Section V-C).

D. Self-Healing

We define self-healing as a property of a key management scheme that is capable of independently recovering from network compromise. We consider a network as compromised when a malicious entity uncovers the master secret key (e.g., when a mobile adversary is successful). The DECENT scheme achieves self-healing by periodically updating the master secret key through the execution of the share updating protocol. None of the related key management schemes consider the potential compromise of the master secret key. Instead, [11], [12], [19], [22] assume that a mobile adversary is unable to compromise sufficient shares in between two consecutive share updating phases whereas [16]–[18], [20], [21], [23] do not consider the possibility of network compromise entirely.

This self-healing property achieved by DECENT is also beneficial in comparison to other limited-time compromised schemes [11], [12], [20], [22] (see Table I) since they require a TTP to execute a potentially expensive manual network reboot.

TABLE III
COMPARISON OF THE COMMUNICATION OVERHEAD (I.E., THE NUMBER OF MESSAGE EXCHANGES REQUIRED TO COMPLETE EACH PROTOCOL) PER PROTOCOL OF FD-TTP-BASED KEY MANAGEMENT SCHEMES.

FD-TTP-based Key Management Scheme	Initial Share Establishm. Protocol	Initial Key Establishm. Protocol	Distr. Share Establishm. Protocol	Distr. Key Establishm. Protocol	Share Updating Protocol	Key Updating Protocol	Secure Ch. Establishm. Protocol	Key Revocation Protocol
Luo et al. [11], [12]	$t+1$	t	$2t+2$	$t+1$	$3nt+4n+t$	nt	$2d$	$(m-1)^2t^2+t$
Deng et al. [16]	t^2	t^2	$t+1$	$n+t+1$	–	–	0	–
da Silva et al. [17]	t^2	t^2	$t+1$	$t+1$	–	$n+t$	0	$nt+n+t^2-2t-1$
Zhang et al. [18]	t	t	$t+1$	$t+1$	–	–	$2d$	–
Li et al. [19]	t^2	t^2	$t+1$	$t+1$	$dnt+dn-dt$	–	$2d$	–
Lai et al. [20]	t^2	$2t^2$	$t+1$	$2t+2$	–	0	$2d$	0
Gharib et al. [21]	$t+1$	t	$t+1$	$5t+1$	–	0	$2d$	0
de Ree et al. [22]	t	$3t$	$2t+2$	$t+3$	$2nt+3n-t^2-2t$	0	$2d$	0
Saxena et al. [23]	t		$t+1$		–		0	–
DECENT	t		$t+1$		$nt+2n-t$		0	–

n: number of network nodes, t: security threshold, d: average distance in hops between two network nodes, m: number of hops to flood a local area

VII. OVERHEAD COMPARISON

In this overhead comparison, we focus our attention to the communication overhead of the alternative FD-TTP-based key management schemes. For a comprehensive evaluation of the communication overhead of conventional FD-TTP-based key management schemes [11], [12], [16]–[22], we refer the reader to [22]. A unicast, multicast and broadcast message are considered to contribute one message to the communication overhead since these are single transmissions even though the number of receivers vary. The communication overheads, summarized per protocol per scheme in Table III, are clearly lower for alternative FD-TTP-based key management schemes.

A. Initial Share & Key Establishment Protocols

For these protocols, the communication overhead is defined as the least number of transmissions required to initialize t nodes. Due to the correspondence between a node's secret share and its public-private key pair, these protocols are combined into one network bootstrapping protocol. The overhead is in part determined by assumptions on the TTP. DECENT and [23] assume that a centralized TTP can bootstrap the network, causing an overhead of t. Decentralized bootstrapping would cause an overhead of at least t^2.

B. Distributed Share & Key Establishment Protocols

For these protocols, the communication overhead is defined as the least number of transmissions required to provide one joining node with its secret share and its public-private key pair, respectively. Due to the correspondence between a node's secret share and its public-private key pair, these protocols are combined into one node admission protocol. DECENT and [23] can assume that the broadcast request from a joining node reaches at least t servers which, in turn, reply with a unicast transmission, causing an overhead of $t+1$.

C. Share & Key Updating Protocols

For these protocols, the communication overhead is defined as the least number of transmissions required for every network node to update its secret share and its public-private key pair, respectively. Due to the correspondence between a node's secret share and its public-private key pair, these protocols are combined into one share updating protocol. In DECENT, t servers transmit one broadcast and $t-1$ unicast messages to update the master secret, one broadcast is transmitted per node to inform its nearby nodes that it can assist in updating their secret shares, and $n-t$ nodes transmit one broadcast and receive t unicast messages to be readmitted. This causes a total overhead of $nt+2n-t$.

D. Secure Channel Establishment Protocol

For the secure channel establishment protocol, the communication overhead is defined as the least number of transmissions required for two arbitrary nodes to establish a secure channel. DECENT and [23] allow non-interactive secure channel establishment and thus have an overhead of 0.

VIII. PRACTICAL GUIDELINES FOR DEPLOYMENT

The deployment of the DECENT scheme requires the selection of appropriate network parameters. In this section, we provide guidelines for the selection of appropriate security thresholds for a variety of deployment scenarios.

A. Restrictions Imposed by Protocols

Each key management service protocol requires a certain network topology to be correctly executed. We identified the following three levels of requirements that a key management service protocol can impose on the network topology.

1) The protocol allows a requester node to make multiple requests over time to reach t server nodes.
2) The protocol requires that a requester node is within transmission range of at least t server nodes.
3) The protocol requires a cluster of t server nodes that are all within each other's transmission range.

The DECENT scheme has two key management service protocols, the node admission protocol and the share updating protocol. In the node admission protocol, a requester node is

TABLE IV
THE IMPLEMENTATION DETAILS FOR DETERMINING THE IMPROVED SECURITY THRESHOLD GUIDELINES.

Implementation details	
Simulation platform	Python 3.8.5
Node distribution	Random
Node densities (nodes/km^2)	$\{100, 200, 500, 1000, 2000, 4000\}$
Transmission ranges (m)	$\{10, 25, 50, 100, 150, 250, 400\}$
Network estimation	Maximum clique

TABLE V
GUIDELINES FOR SELECTING AN APPROPRIATE SECURITY THRESHOLD FOR A VARIETY OF DEPLOYMENT SCENARIOS.

Thresholds achieving 95% key management service availability							
Node Density	Transmission Range (meters)						
(nodes/km^2)	10	25	50	100	150	250	400
100	1	2	3	4	6	11	20
200	2	3	4	6	10	19	36
500	2	4	6	12	19	39	80
1,000	3	5	9	19	32	67	148
2,000	4	7	13	31	55	123	281
4,000	5	10	21	53	98	231	537

allowed to roam the network and periodically request nearby server nodes for subshares to establish its own public-private key pair (i.e., requirement level 1). However, the share updating protocol requires a cluster of t server nodes to be within each other's transmission range to prevent any malicious server from disabling the key management (i.e., requirement level 3).

A guideline for the selection of appropriate security thresholds was previously given in [22]. However, their guidelines were based on whether a requester node is within transmission range of at least t server nodes (i.e., requirement level 2). Therefore, they accept security thresholds which are unable to execute the share updating protocol. Furthermore, they only considered a transmission range of 150 meters.

B. Simulation Results

We used Python to simulate networks and estimated their maximum clique to determine the maximum threshold t which satisfies network topology requirement level 3. Implementation details are shown in Table IV; the full implementation is available at [33]. The simulation results, i.e., the upper bounds on the security thresholds for which there is a 95% probability that a cluster of t nodes exists, are shown in Table V.

The implementation results are graphed in Figure 2. The graph uses a *log*-scale on the y-axis to represent all the data in a readable manner. Furthermore, we included the security threshold guidelines previously provided by [22]. It can be observed that the suggested security thresholds by [22] are about twice as high compared to our results and could have caused the inability of executing the share updating protocol, leading to a malfunctioning key management scheme.

C. Discussion

It is important to consider the effect that the security threshold has on overheads. For a maximum security threshold,

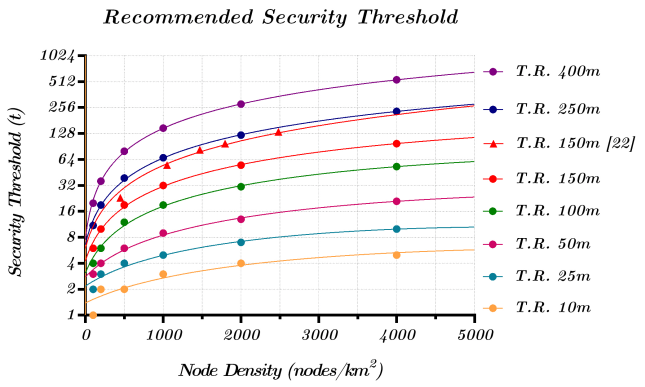


Fig. 2. Guidelines for selecting the security threshold t for a deployment scenario with a particular node density and transmission range (T.R.).

key management service protocols can impose a significant communication overhead due to the necessary message exchanges, computational overhead due to combining the large number of partial key management services, and memory storage overhead from storing the k-bit long polynomial share coefficients. It may therefore be beneficial to reduce the security threshold (e.g., by half) and increase (e.g., double) the frequency of executing the share updating protocol.

IX. FUTURE WORK

As a future work, we intend to extend or modify the design of our DECENT scheme in the following ways: (i) include a node accusation and revocation mechanism that ensures that malicious nodes are removed and prevented from rejoining the network, (ii) include a decentralized method for network bootstrapping, (iii) modify the protocol design to be based on discrete logarithms in the elliptic curve group to reduce the key sizes and improve performance, and (iv) implement and measure the performance of our DECENT scheme.

X. CONCLUSION

In this paper, we proposed the design of our novel decentralized key management scheme entitled DECENT that is suitable for a wide range of deployment scenarios. This key management scheme deviates from the conventional structure of FD-TTP-based key management by having a direct correspondence between a node's secret share and its public-private key pair. The establishment or update of a node's secret share is therefore equivalent to the establishment or update of that node's public-private key pair. This means that the share establishment protocol is equivalent to the key establishment protocol and the share updating protocol is equivalent to the key updating protocol. These equivalences have the following benefits: (i) it simplifies the key management design, (ii) it significantly reduces overheads, and (iii) it allows our key management to benefit from the self-healing property by allowing the master secret to be periodically updated. With self-healing, we mean that a potential network compromise is restored through the execution of the share updating protocol (i.e., a self-organized network reboot). We have shown that

DECENT is very competitive in terms of security and low in terms of overheads compared to closely related schemes. Finally, our simulation results provide network administrators with insight into the selection of appropriate security thresholds for practical deployments.

References

[1] A. Laouiti, A. Qayyum, M. Naufal, and M. Saad, "Vehicular Ad-Hoc Networks for Smart Cities," Springer, 2016.

[2] V. Sucasas, G. Mantas, F. B. Saghezchi, A. Radwan, and J. Rodriguez, "An Autonomous Privacy-Preserving Authentication Scheme for Intelligent Transportation Systems," *Comput. Secur.*, vol. 60, pp. 193-205, 2016.

[3] B. Ji *et al.*, "Secrecy Performance Analysis of UAV Assisted Relay Transmission for Cognitive Network with Energy Harvesting," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7404-7415, 2020.

[4] W. Duan *et al.*, "Emerging Technologies for 5G-IoV Networks: Applications, Trends and Opportunities," *IEEE Netw.*, vol. 34, no. 5, pp. 283-289, 2020.

[5] A. Munir, and F. Koushanfar, "Design and Analysis of Secure and Dependable Automotive CPS: A Steer-By-Wire Case Study," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 4, 2020.

[6] S. Goudarzi, N. Kama, M. H. Anisi, S. Zeadally, and S. Mumtaz, "Data Collection Using Unmanned Aerial Vehicles for Internet of Things Platforms," *Comput. Electr. Eng.*, vol. 75, pp. 1-15, 2019.

[7] J. Rodriguez *et al.*, "Secure Virtual Mobile Small Cells: A Stepping Stone Towards 6G," *IEEE Commun. Standards Mag.*, vol. 5, no. 2, 2021.

[8] M. de Ree *et al.*, "Key Management for Beyond 5G Mobile Small Cells: A Survey," *IEEE Access*, vol. 7, pp. 59200-59236, 2019.

[9] A. Shamir, "How to Share a Secret," *Commun. ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[10] J. Li *et al.*, "Decentralized On-Demand Energy Supply for Blockchain in Internet of Things: A Microgrid Approach," *IEEE Trans. Comput. Soc. Syst.*, vol. 6, no. 6, pp. 1395-1406, 2019.

[11] H. Luo and S. Lu, "Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks," Univ. California, Los Angeles, CA, USA, Tech. Rep. UCLA-CSD-TR-200030, 2000.

[12] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang, "URSA: Ubiquitous and Robust Access Control for Mobile Ad Hoc Networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 6, pp. 1049-1063, 2004.

[13] P. Feldman, "A Practical Scheme for Non-Interactive Verifiable Secret Sharing," *Proc. 28th Annu. Symp. Found. Comput. Sci. (SFCS)*, Los Angeles, CA, USA, pp. 427-437, 1987.

[14] S. Jarecki, "Proactive Secret Sharing Public Key Cryptosystems," M.S. thesis, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 1995.

[15] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive Secret Sharing OR: How to Cope with Perpetual Leakage," *Proc. CRYPTO*, Santa Barbara, CA, USA, pp. 339-352, 1995.

[16] H. Deng and D. P. Agrawal, "TIDS: Threshold and Identity-based Security Scheme for Wireless Ad Hoc Networks," *Ad Hoc Netw.*, vol. 2, no. 3, pp. 291-307, 2004.

[17] E. da Silva and L. C. P. Albini, "Towards a Fully Self-Organized Identity-Based Key Management System for MANETs," *Proc. 9th IEEE Int. Conf. Wireless and Mobile Computing, Networking and Communications (WiMob)*, Lyon, France, pp. 717-723, 2013.

[18] Z. Zhang, W. Susilo, and R. Raad, "Mobile Ad-Hoc Network Key Management with Certificateless Cryptography," *Proc. 2nd Int. Conf. Signal Process. Commun. Syst. (ICSPCS)*, Gold Coast, QLD, Australia, pp. 1-10, 2008.

[19] F. Li, M. Shirase, and T. Takagi, "Key Management using Certificateless Public Key Cryptography in Ad Hoc Networks," *Proc. 5th IFIP Int. Conf. Network and Parallel Computing (NPC)*, Shanghai, China, pp. 116-126, 2008.

[20] J. Lai, W. Kou, and K. Chen, "Self-Generated-Certificate Public Key Encryption without Pairing and its Application," *Inf. Sci.*, vol. 181, no. 11, pp. 2422-2435, 2011.

[21] M. Gharib, Z. Moradlou, M. A. Doostari, and A. Movaghar, "Fully Distributed ECC-based Key Management for Mobile Ad Hoc Networks," *Comput. Networks*, vol. 113, pp. 269-283, 2017.

[22] M. de Ree, G. Mantas, J. Rodriguez, I. E. Otung, and C. Verikoukis, "DISTANT: Distributed Trusted Authority-based Key Management for Beyond 5G Wireless Mobile Small Cells," *Comput. Commun.*, vol. 176, pp. 218-233, 2021.

[23] N. Saxena, G. Tsudik, and J. H. Yi, "Efficient Node Admission and Certificateless Secure Communication in Short-Lived MANETs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 2, pp. 158-170, 2009.

[24] N. Saxena, "Public Key Cryptography Sans Certificates in Ad Hoc Networks," *Proc. 4th Int. Conf. Applied Cryptography and Network Security (ACNS)*, Singapore, Singapore, pp. 375-389, 2006.

[25] R. Ostrovsky and M. Yung, "How to Withstand Mobile Virus Attacks," *Proc. 10th ACM Symp. Princ. Distribu. Comput. (PODC)*, Montreal, QC, Canada, pp. 51-59, 1991.

[26] M. Girault, "Self-Certified Public Keys," *Proc. EUROCRYPT*, Brighton, U.K., pp. 490-497, 1991.

[27] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Proc. EUROCRYPT*, Paris, France, pp. 335-338, 1984.

[28] A. J. Menezes, P. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, 5th ed., Boca Raton, FL, USA: CRC Press, 2001.

[29] N. Saxena, G. Tsudik, and J. H. Yi, "Efficient Node Admission for Short-lived Mobile Ad Hoc Networks," *Proc. 13th IEEE Int. Conf. Network Protocols (ICNP)*, Boston, MA, USA, pp. 269-278, 2005.

[30] D. Poole, *Linear Algebra: A Modern Introduction*, 2nd ed., Canada: Thomson Brooks/Cole, 2006.

[31] M. Narasimha, G. Tsudik, and J. H. Yi, "On the Utility of Distributed Cryptography in P2P and MANETs: the Case of Membership Control," *Proc. 11th IEEE Int. Conf. Network Protocols (ICNP)*, Atlanta, GA, USA, pp. 336-345, 2003.

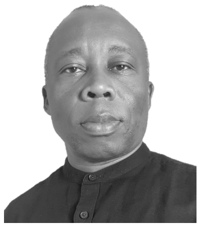
[32] S. S. Al-Riyami, and K. G. Paterson, "Certificateless Public Key Cryptography," *Proc. ASIACRYPT*, Taipei, Taiwan, pp. 452-473, 2003.

[33] M. de Ree, *DECENT Simulation*, GitHub, 2022. [Online]. Available: https://github.com/mderee/Public-Scripts/blob/main/DECENT-sim.

Marcus de Ree (Member, IEEE) received his M.Sc. degree in applied mathematics, focused in the mathematical theory of communication systems, from San Diego State University, USA, in 2017, and Ph.D. degree in electronic engineering from the University of South Wales, UK, in 2021. He is a postdoctoral researcher in secure wireless communication at the Instituto de Telecomunicações, Aveiro, Portugal, contributing to the NATO-funded PHYSEC project as technical manager and the EU-funded Moore4Medical project.

Georgios Mantas (Member, IEEE) received his M.Sc. degree in information networking from Carnegie Mellon University, USA, in 2008, and Ph.D. degree in electrical and computer engineering from the University of Patras, Greece, in 2012. In 2014, he became a postdoctoral researcher at the Instituto de Telecomunicações, Aveiro, Portugal, where he has been involved in research projects, such as ECSEL-SemI40, CATRENE-MobiTrust and FP7-SEC-SALUS. Since 2020, he has been a senior lecturer with the University of Greenwich, UK.

Jonathan Rodriguez (Senior Member, IEEE) received his M.Sc. degree in electronic and electrical engineering and Ph.D. degree in mobile communications from the University of Surrey, UK, in 1998 and 2004, respectively. He is a senior researcher at the Instituto de Telecomunicações, Aveiro, Portugal, and a full professor at the University of South Wales, UK. His professional affilliations include Chartered Engineer, IET Fellow, HEA Senior Fellow, and Associated Editor of *IET Communications*.



Ifiok E. Otung received his M.Sc. degree in electrical engineering from the University of Ife, Nigeria, and Ph.D. degree in satellite communications from the University of Surrey, UK, in 1995. He is a chartered engineer and professor of satellite communications with the University of South Wales (USW). He founded the M.Sc. program in mobile and satellite communications, has supervised around 150 postgraduate projects, is an associate editor of *IET Journal of Engineering* and co-editor of *IET Advances in Communications Satellite Systems*.