THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace

Co-tutelle internationale : Université Macquarie

Présentée et soutenue par :
Marina DEHEZ CLEMENTI
le jeudi 17 mars 2022

Titre :

L'utilisation des blockchains pour renforcer la sécurité et améliorer la confiance dans les réseaux distribués

Blockchain-enabled trust and security in distributed systems

École doctorale et discipline ou spécialité :
ED MITT : Informatique et Télécommunications

Unité de recherche :
ENAC-LAB - Laboratoire de Recherche ENAC

Directeur(s) de Thèse :

M. Emmanuel LOCHIN (directeur de thèse)
M. Dali KAAFAR (co-directeur de thèse)

Jury :
M. Mohamed Yacine GHAMRI-DOUDANE Professeur Université de La Rochelle - Président
M. Daniel AUGOT Directeur de Recherche INRIA Saclay - Rapporteur
Jean-Christophe DENEUVILLE Maître de conférences ENAC - Examinateur
M. Joaquin GARCIA-ALFARO Professeur Institut Polytechnique de Paris IMT - Rapporteur
M. Dali KAAFAR Professeur Université Macquarie Australie - Co-directeur de thèse
Mme Emmanuelle LACAN Cadre scientifique Airbus Defence & Space - Examinatrice
M. Cédric LAURADOUX Chargé de recherche INRIA Rennes - Examinateur
M. Emmanuel LOCHIN Professeur ENAC - Directeur de thèse

Typeset in $\LaTeX\,2_\varepsilon$.

This thesis is being submitted to Macquarie University and ISAE-SUPAERO, in accordance with the Cotutelle agreement dated 14 November 2018, in fulfilment of the requirement for the Degree of Doctor of Philosophy.

The work presented in this thesis is, to the best of my knowledge and belief, original except as acknowledged in the text. I hereby declare that I have not submitted this material, either in full or in part, for a degree at this or any other institution apart from Macquarie University and ISAE-SUPAERO.

Marina Dehez-Clementi

Acknowledgements

This is the end, and here I am. After three years of perpetual questioning, who would have known? It is difficult to find the right words to express my feelings and gratitude towards all the people that shared a moment of this great adventure, all the people that supported me along the way. Yet, first and foremost, I shall thank my supervisors, Prof Emmanuel Lochin and Prof Dali Kaafar, without who this journey would not have even started. I would not be here if it were not for their faith in me and the opportunity they gave me to prove them right. On the same level, I would like to thank Prof. Nicolas Larrieu, who started the adventure with me and encouraged me in my first moments as a researcher. Next, I would like to thank Dr Jean-Christophe Deneuville and Dr Hassan Asghar for their technical support, guidance and involvement in my research projects. I would be ungrateful if I did not thank Prof Jérôme Lacan for his invaluable advice and continuous support during my PhD study. My thesis would not be the same without our long afternoons of brainstorming. Of course, I would like to express my deepest thanks to the reviewers for dedicating time to review my thesis and the examiners for accepting to be part of my defense jury. In addition, I send a warm thanks to all the members of the ResCom team at ISAE-SUPAERO, the Optus Macquarie University Cyber Security Hub and the ResCo team at ENAC. Whichever lab I was working in these past three years, either here in Toulouse or overseas in Sydney, I always felt welcomed, supported and helped.

Mum, Dad, Grandma. This thesis, my successes and victories, I give it all to you. There is no word to describe how thankful I am to be born in your loving home. You sacrificed and endured difficult moments to provide everything so I could pursue my dreams. Your unconditional support and constant faith in me even when I believed I was lost, got me through every bad situation and disappointment. I wish to live up to you one day, and I hope this is proof enough that I started. There is another person I shall thank at this point because he showed me the true meaning of support and care, of reassurance when doubt was gnawing away my self-esteem. Thank you for sharing all my deceptions and victories, and for being the rock, I can hold on to in the strongest storm. To my friends, I hope you will recognize youselves, I thank you all for sharing my stress, my laughter, my pain, and most of all for listening to me complaining when that was necessary.

Oh, I almost forgot. There is one last person I swore to thank. And this person is me. I know how this sounds: pretentious, arrogant, egocentric, or maybe irrelevant for some. Yet, none of this would have been possible without me. So thank you old pal'. You deserve these kind words as much as all those persons that were here for you along the way. I am proud of you. I am looking forward to see what you will become and wish you the best of luck.

Quelle aventure! Merci.

List of Publications

Conferences and workshops.

- Marina Dehez-Clementi, Nicolas Larrieu, Emmanuel Lochin, Hassan Asghar, Dali Kaafar, *When Air Traffic Management meets Blockchain technology: a blockchain-based concept for securing the sharing of flight data*, at IEEE/AIAA 38th Digital Avionics Systems Conference (DASC2019), pp.1-10, 2019.

- Marina Dehez-Clementi, Jean-Christophe Deneuville, Jerôme Lacan, Hassan Asghar, Dali Kaafar, *Who let the $\mathcal{DOGS}$ out: anonymous-but-auditable communications using Group Signature schemes with Distributed Opening*, at the 4th International Workshop on Cryptocurrencies and Blockchain Technology (CBT2020), vol. 12484, 2020.

- Marina Dehez-Clementi, Jean-Christophe Deneuville, Jerôme Lacan, Hassan Asghar, Dali Kaafar, *A Blockchain-enabled Anonymous-yet-Traceable Distributed Key Generation*, at the 4th IEEE International Conference on Blockchain (Blockchain2021), vol. X, 2021.

Unpublished.

- Marina Dehez-Clementi, Jean-Christophe Deneuville, Jerôme Lacan, Hassan Asghar, Dali Kaafar, *A Blockchain-supported Threshold Anonymous-yet-Accountable Decryption service*, 2021.

Abstract

As of 2016, the number of deaths induced by road injuries reached 1.35 million, and is often due to human error. The technological expansion of the Internet and interconnected devices facilitate the exchange of information, sometimes vital. That is why a lot of work has been done towards the automatization of vehicles. Improving road safety is one of the motivational factors for research in this area, and the widespread adoption of Intelligent Transportation Systems (ITSs).

An ITS is defined as a particular ad-hoc network formed by vehicles with processing and wireless communication abilities, evolving in an urban environment (streets or highways). Vehicles can communicate either directly or through a intermediary node. The main focus of security in ITSs and vehicular communications is on providing *integrity* of the exchanged messages and *availability* of the services that support them, rather than the confidentiality of what they contain. Providing *accountability*, *i.e.* a way to identify the communicating entities and held them accountable for the messages they broadcast, in vehicular communications is essential. It ensures that any faulty or misbehaving node is identified, revoked, eventually punished for its actions and subsequent consequences. However, the presence of such an identifying mechanism poses a *privacy* risk to the users, even when they are behaving honestly.

This thesis focuses on the delicate *trade-off between anonymity and traceability* in distributed systems such as ITSs. We study the use of blockchains in the construction of privacy-preserving yet accountable threshold cryptographic primitives, and their application to the case of ITSs.

Our first contribution is a blockchain-based group signature scheme with distributed opening functionality called $\mathcal{DOGS}$. We will show that the scheme improves on a traditional group signature scheme and leverages a distributed key generation protocol to distribute the role of the opener over a set of nodes called the sub-openers.

Our second contribution is an anonymous-yet-traceable distributed key generation (DKG) protocol, called $\mathcal{BAT}-Key$, that utilizes a blockchain to provide trust among the participating distrusting entities. We will present how we augmented traditional DKG propositions with the anonymity property that protects the identities of the participants.

Our third contribution is a blockchain-based threshold encryption scheme with an anonymous-yet-accountable decryption service, called $\mathcal{TOAD}$. We will show that the scheme builds on threshold encryption and proposes a collaborative decryption process that protects the identity of the decryption servers.

Throughout the chapters, we will explain how the use of blockchain guarantees the traceability of the actions performed within the system by anonymous nodes and therefore ensures their accountability while preserving the privacy.

These schemes are of utmost importance in the era of digitization, even outside the field of ITS. Yet, we chose to exemplify their importance in the context of ITSs through our last contribution: the description of our construction of a blockchain-based privacy-preserving yet accountable Traffic Reporting system.

Résumé

En 2016, le nombre de décès dus aux accidents de la route atteignait 1,35 million, et ces accidents sont souvent imputables à l'erreur humaine. L'expansion technologique d'Internet et des réseaux interconnectés facilitent l'échange d'informations, parfois vitales. C'est pourquoi beaucoup de travaux ont été produits sur l'automatisation des véhicules. L'amélioration de la sécurité routière est l'un des facteurs qui motive la recherche dans ce domaine et pousse vers l'adoption de systèmes de transport intelligents (ITS).

Un ITS est défini comme un réseau ad-hoc particulier, formé de véhicules évoluant en milieu urbain capables de communiquer et traiter l'information reçue. Les véhicules peuvent communiquer directement, de pair à pair, ou via un nœud intermédiaire.

L'objectif principal de la sécurité des ITSs et des communications véhiculaires est de fournir l'intégrité des messages échangés et la disponibilité des services qui supportent ces échanges. La protection et la confidentialité de leur contenu est un objectif secondaire car non vital. Assurer la responsabilité, c'est-à-dire proposer un moyen d'identifier les entités communicantes et de les tenir responsables des messages qu'elles diffusent, est essentiel voire légalement obligatoire. Ce mécanisme doit garantir que tout nœud qui subit une faute, panne ou agit de façon malveillante, soit identifié, révoqué, finalement puni pour ses actions et leurs conséquences. Cependant, un tel mécanisme d'identification pose un problème et risque de compromettre la vie privée des utilisateurs, même lorsqu'ils sont honnêtes.

Cette thèse porte sur le délicat compromis entre anonymat et traçabilité dans

des systèmes distribués tels que les ITSs. Nous étudions l'utilisation des blockchains (chaînes de blocs) dans la construction de primitives cryptographiques à seuil. Ces primitives sont utilisées afin de préserver la vie privée, mais aussi la responsabilité des acteurs.

Notre première contribution, appelée $\mathcal{DOGS}$, est un schéma de signature de groupe basé sur la blockchain proposant la fonctionnalité d'ouverture distribuée. Nous montrons, dans cette thèse, que le système améliore un schéma de signature de groupe existant et exploite un protocole de génération de clé distribuée pour répartir le rôle de l'ouvreur (Opener) sur un ensemble de nœuds appelés les sous-ouvreurs (sub-openers).

Notre deuxième contribution est une génération de clé distribuée anonyme mais traçable, appelé $\mathcal{BAT} - Key$, qui utilise une blockchain pour assurer la confiance entre les différentes entités qui composent le système. Dans la suite de la thèse, nous expliquons comment nous améliorons les protocoles traditionnels avec la propriété d'anonymat qui protège l'identité des participants.

Notre troisième contribution, appelée $\mathcal{TOAD}$, est un schéma de chiffrement à seuil basé sur la blockchain avec un service de déchiffrement anonyme mais traçable. Ce schéma améliore grandement un schéma de chiffrement à seuil connu par un processus de déchiffrement collaboratif protégeant l'identité des serveurs de déchiffrement.

Tout au long des chapitres, nous expliquons comment l'utilisation des blockchains garantit la traçabilité des actions effectuées au sein du système par des nœuds anonymes et assure ainsi leur responsabilité tout en préservant leur vie privée. Ces schémas sont de la plus haute importance dans l'ère du numérique, même en dehors du domaine des ITSs. Pourtant, nous avons choisi d'illustrer leur importance dans le contexte des ITSs à travers notre dernière contribution : la description de notre construction d'un système de rapport de trafic routier basé sur la blockchain qui préserve l'anonymat des nœuds qui rapportent les informations, mais les tient pour responsables de leurs messages en cas de litige.

Contents

II Construction and analysis of the framework 69

List of Figures

List of Tables

1

Introduction

"Serons-nous capables de choisir les éléments de la technologie qui améliorent la qualité de vie et d'éviter ceux qui la détériorent ?"

— David Baltimore

With the ever-growing technological expansion of the Internet, distributed systems are becoming more and more widespread. An example of such growth is the advent of ITSs and Vehicular Ad-hoc NETworks (VANETs).

As of 2016, the number of deaths induced by road injuries reached 1.35 million. While this value remains unacceptably high, the 2018 Global Status Report on road safety of the World Health Organization (WHO) also reveals that the rate of deaths (number of deaths per global population size) is constant [3]. This suggests that existing road safety efforts may work in favor of safer and less-deadly road installations. Improved road safety is one of the motivational factors for research in this area, which will soon become a reality with increased intelligence in vehicles and the widespread adoption of ITSs. As of now, they are already becoming increasingly popular due to their capabilities to improve the vehicular network's efficiency (*e.g.,* reduction of traffic jams) and its safety (*e.g.,* use of intelligent collision avoidance systems). Hence, automobile manufacturers have started incorporating safety-enhancing instrumentation. The community is, in parallel, working on deploying and improving a communication infrastructure to contribute to increased safety, comfort of traveling, and users' entertainment.

ITS is defined as a particular ad-hoc network formed by vehicles with processing and wireless communication abilities, evolving in an urban environment (streets or highways) [4]. Vehicles can communicate either directly or through a RoadSide Unit (RSU). Consequently, vehicles can broadcast information to surrounding users, access network services, or obtain data from other networks, such as the Internet. ITSs involve intelligent components that are evolving in the physical world. As such, they are considered as Cyber Physical Systems (CPSs). Therefore, ITSs present a crosscutting nature:

- in the cyber world, ITS components are nodes connected via wired or wireless networks, and thus, they are vulnerable to cyber-attacks;
- in the physical world, the same components are physical 3-dimensional objects that can be potentially harmful to humans if taken over by malicious actors.

In ITSs, safety always prevails over all other considerations. That is why researchers have focused on developing safety safeguards through the rapid design of two types of messaging patterns, the Cooperative Awareness Messages (CAMs) [5] and the Decentralized Environmental Notification Messages (DENMs) [6], and associated services, often at the expense of security [7, 8]. Securing vehicular communications is a mandatory requirement for their full-scale deployment.

Unlike existing Information Technology infrastructures, the main focus of security in ITSs is on providing integrity of the vehicular communications and availability of the services that support them, rather than the confidentiality of what they contain. Systems must guarantee that a message comes from a trusted source and has not been modified during transmission. Indeed, let us take the example of Road Hazard Warning (RHW) applications [9]. The vehicle system will use information communicated from the infrastructure or from another vehicle to determine if a warning should be relayed. An incorrect transmission from an invalid or compromised node might jeopardize the safety of the vehicle and endanger others. In order to identify any misbehaving entity, ITSs traditionally implement a signature scheme. Hence, the messages are signed, via the authenticated public identity, or public key, attached to one vehicle by a Trusted Authority [10]. Providing *accountability* in vehicular communications is therefore essential as it ensures that any faulty or misbehaving node is identified, revoked, eventually punished, and can no longer participate to further communications. However, the presence of such a system poses a privacy risk to the users. Indeed, the network, or any passive eavesdropper, can be aware of the whereabouts of a specific user, at a specific time [11].

1.1 Approach

In the context of this thesis, we work on finding an acceptable trade-off between accountability and privacy in vehicular communications. In the following paragraph, we study how the blockchain technology can remedy the lack of traceability and accountability in vehicular communications. We present the inherent properties of blockchains and discuss how we can use them to this end in the context of ITSs.

1.1.1 Blockchain layer

In 2009, Satoshi Nakamoto published the white paper that revolutionized distributed systems. Until then, Byzantine Fault-Tolerant (BFT) consensus was notoriously known to be an arduous task to achieve. The term takes its name from the famous allegory of the“Byzantine Generals Problem” [12]. It refers to the situation in which the system’s actors must agree on a common strategy to avoid the failure of the system. Yet, in addition, some of these actors are unreliable. The presence of faulty entities is what makes agreement between parties within the same group highly difficult. While solving this problem, Nakamoto’s initial idea with Bitcoin [13] was to get rid of the intermediary in online monetary transactions. More specifically, he desired to reproduce the notion of cash and free the transacting users from their bank’s interaction and monitoring. The inherent property of cash is its anonymity. Real-world coins and banknotes are fungible; thus, no mechanism attaches one’s identity to a particular token, except the simple possession of it. For three years, the online community believed that Bitcoin was anonymous (even though it was never claimed in Nakamoto’s paper). Afterward, researchers and scientists proved that Bitcoin *is not* an anonymous digital currency [14]. This observation led to the massive development of anonymous blockchain plateforms. Among them, Monero [15] uses a combination of Ring Confidential Transactions [16], stealth addresses [17] and transactions over Tor and I2P [18] to build a secure, privacy-preserving decentralized blockchain system. Another example is Zcash [19]. It leverages the privacy-preserving Zerocoin protocol [20] which improves on Bitcoin by using Rivest–Shamir–Adleman (RSA) accumulators [21]. Other well known privacy-centric blockchains include DASH [22] previously known as Darkcoin [23], Horizen [24], Beam [25]. Then, where did Bitcoin fail?

Bitcoin is inspired by the implementation of real-life anonymity, which, unlike computers, partly relies on the deficiency of human memory. Indeed, let us take the following toy example: Alice goes to the bakery to buy a baguette, and she gives Bob the baker her coin to pay for the bread. Firstly, Bob can manually check the currency for authenticity or delegate the verification to a device (since the government creates the money, it is not easily counterfeit). Secondly, Bob can see Alice as a human being, holding this coin before paying. He can assert that the coin comes from Alice and is authentic. Yet, after an 8-hour-long shift, does Bob remember Alice at the end of the day? Probably not. The anonymity of Alice is *preserved* because: Bob does not know Alice’s personal information, and because their interaction is ordinary enough that Bob did not pay much attention to it. But now, let us change the media. Instead of Alice and Bob, we consider computers A and B communicating over an open (*e.g.,* Internet) communication channel. Because they are distinct machines, computers A and B need some identifiers (*e.g.,* account numbers) to reach out to one another. Before, Bob did not have to know Alice's name to interact with her, now it is mandatory. Moreover, while the human memory capacity depends on the individual, computers have standardized memory storage, of which the reliability is improving over the years. This improvement enables them to store a history of transactions. In real life, it would be like if Alice and Bob had little tags on their shirts saying their name, and Bob would write down on a paper-based ledger "Alice paid one coin for a baguette to Bob 29/10/2021 17:01:27". However, this is how Bitcoin works. The system is therefore far from being anonymous.

The blockchain technology is particularly attractive for its *inherent characteristics* of *decentralization* - fighting by design censorship and single points of failure, *transparency*, *open-sourceness*, and rises against proprietary software from its conception. Also, records are *immutable*. They cannot be changed unless the adversary takes control of more than 51% of the network's discriminating resource (which is highly unlikely when dealing with large networks). However, as illustrated by the aforementioned anonymity issue, there are still several challenges in blockchain-based applications, for example, in terms of: scalability [26], performance [27] and overall security [28].

Therefore, in the case of ITSs, data can be tracked via a blockchain, for instance by publishing integrity checks. Yet, it cannot be shared through the same media as the content of ITS messages would thwart the targeted anonymity requirement. Consequently, this leads to the question: how can users anonymously share relevant information within a ITS?

1.1.2 Network layer

A ITS operates on different Open Systems Interconnection (OSI) model layers: from the physical layer (which refers to signal and bits transmission via the vehicles' embedded equipment) to the application layer (which concerns the dissemination of CAMs and DENMs. The European Telecommunication Standards Institute (ETSI) is the European agency in charge of the definition of standards for better road safety. Among those specifications, the technical report [29] defines the Dedicated Short Range Communications (DSRC)/Wireless Access in Vehicular Environments (WAVE) protocol. It is used by vehicle and infrastructure nodes in Vehicle-to-Vehicle (V2V) and vehicle-to-infrastructure Vehicle-to-Infrastructure (V2I) communications. As such, vehicular communications essentially rely on a custom peer-to-peer (P2P) network facilitated with ITS-specific standards of communications. This P2P technology offers an alternative communication architecture to traditional client-server architecture. It perfectly matches the inherent characteristics of ITSs (such as their highly dynamic topology, unstable connectivity, adaptative propagation model). It enables file-sharing and content delivery among users, particularly useful when a vehicle needs to broadcast data to a large geographical area. The anonymity of participants is usually achieved by special routing overlay networks that hide the physical location of each node from other participants. Interest in anonymous P2P systems has increased in recent years for many reasons, ranging from the desire to share files without revealing one's network identity to distrust in governments, concerns over mass surveillance and data censorship. Methods used for anonymity in P2P networks can be categorized into different groups including broadcast encryption. This method exploits the broadcast feature by sending encrypted messages to all nodes in the network. All the messages have the same lengths and include accurate and dump messages. These messages are encrypted with the public key of the receiver. As a result, only the receiver with its private key can decrypt them.

Using broadcast encryption for data confidentiality and privacy-preservation, combined with a blockchain-based tracking system, seems like a good option to design anonymous-yet-accountable information-sharing services in ITSs.

However, in addition to confidentiality, the data delivery service must ensure the authentication of the message. Developed in the 70's, digital signatures are one of the most important primitives in public-key cryptography and provide *authentication*, *integrity* and *non-repudiation* to various applications, including information sharing services. However, they do not provide the necessary privacy of the signer [30]. Therefore, we suggest to investigate towards group-oriented cryptographic primitives, such as ring [31] or group [32] signature schemes, if we want to find a way to hide the identity of the signer while guaranteeing its accountability.

1.2 Challenges, motivations and contributions

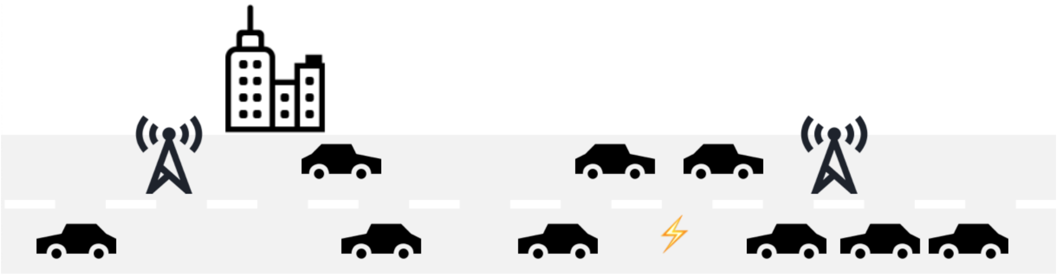
This thesis explores the use of blockchains in the construction of privacy-preserving yet accountable group-oriented cryptographic primitives. It focuses on the delicate trade-off between anonymity and traceability in distributed systems and how to balance these two properties with the help of distributed cryptography (notably Distributed Key Generation protocols and threshold encryption) and blockchain technology. Cryptography helps us address the issues of anonymity and privacy in communications and data exchanges. At the same time, blockchain improves traceability and contributes to the accountability of the nodes in the system. The thesis develops three primitives:

- $\mathcal{DOGS}$ is a blockchain-based group signature scheme with distributed opening functionality. This functionality is particularly important in ITSs as the nodes must remain accountable for their action. However, the power to identify the signing parties is no longer centralized in one trustful entity, which alleviates the risks related to this single point of failure (*e.g.,* compromising, censorship).

- $\mathcal{BAT} - Key$ is an anonymous-yet-traceable Distributed Key Generation (DKG) protocol that utilizes a blockchain to provide trust among the participating distrusting entities. The anonymity feature in DKG protocols is interesting to propose new services in the ITSs such as the anonymous (hence coercion-free) addition of new group members.

- $\mathcal{TOAD}$ is a blockchain-based threshold encryption scheme for an anonymous-yet-accountable decryption service. $\mathcal{TOAD}$ re-uses $\mathcal{BAT} - Key$ to effectively implement a distributed anonymous-yet-traceable issuing authority. In that case, the addition of group members is no longer centralized, which again reduces the risks related to having a single authority instead.

The constructions are then applied to the design of a Traffic Reporting system for ITSs. The vehicular environment and the corresponding description of its nodes is presented in Figure 1.1. We describe a blockchain-enhanced group-signature-based new-DEN messaging protocol that leverages the inherent characteristics of the three primitives we developed to provide privacy-preservation and accountability in the broadcasting of RHWs.



Legend of Figure 1.1

Iconography	Terminology
📡	Roadside units, RSUs, Infrastructure (nodes)
🏢	Trusted third party (TTP), trusted authority (TA), Infrastructure
🚗	Vehicles, on Board Units, OBUs, users
⚡	Road hazard, road event

FIGURE 1.1: Illustration of a simple ITS environment and its legend

1.2.1 $\mathcal{DOGS}$ for an Anonymous Authentication with Distributed Audit

Introduced by Chaum and van Heyst [32], group signature schemes enable members of a group to sign messages on behalf of the group without revealing their identity. Therefore, the recipient can check the validity of the signature but cannot identify the member who generated the signature. This is particularly important in ITSs in order to prevent tracking attacks [33]. However, signing members remain accountable for their messages as the *group manager*, a third-party managing the group, can open their signatures and hence identify them.

In [34], Bellare *et al.* presented some fundamental security notions for static and dynamic group signatures (in the latter case, the addition of group members can occur without resetting the scheme). A key feature of their scheme is that the group manager is split into two distinct entities: the *Issuer* which interacts with users to authenticate their credentials, and the *Opener* which is called when a signature needs to be opened, hence revealing the identity of the signer. The Opener acts as the *tracing authority* in charge of linking a signed message to its origin, namely the signer. This role is essential in the context of ITSs since all users should be accountable for their actions. By definition, the role of the Opener entails no privacy preservation. As shown by the analysis of Sakai *et al.* in [35], existing literature usually considers a unique entity to be playing the role of the Opener, inducing strong assumptions of the level of trust in such an entity as well as its resilience. For instance, in ITSs, vehicles use group signatures to sign the (location and time-dependent) road-safety messages they broadcast. The scheme protects a vehicle's identity unless the Opener is compromised, in which case signatures may be opened, which breaches user privacy. An alternative way to relax these strong assumptions is to implement a ***distributed tracing authority*** and distributed the role of the Opener.

Studies that propose a distributed traceability functionality for group signature schemes (*e.g.,* [36, 37]) often leverage Shamir Secret Sharing (SSS) scheme to generate shares of the Opener secret key and then distribute one share per user. This approach, however, presents an explicit limitation: the computation of these shares is again centralized, which still represents a single point of failure from an adversary perspective.

Our **first contribution**, called $\mathcal{DOGS}$ for *Group Signature scheme with Distributed Opening*, is a blockchain-based group signature scheme with distributed opening functionality. The new signature scheme improves on a traditional group signature scheme [34] and leverages a distributed key generation protocol [38] to split the role of the

Opener over a set of nodes called the sub-openers. The advantage of our DKG-enhanced group signature over SSS-based approaches lies in the fact that the key generation is not entrusted to a third party. Instead, it is the result of the collaboration of a group of users. Hence, none of them knows all the shares, making the scheme strong against a more powerful adversary. The use of blockchain guarantees the traceability of the actions performed within the system and, therefore, ensures the nodes' accountability.

1.2.2 $\mathcal{BAT}-Key$, the Privacy-preserving yet Accountable DKG

As explained previously, DKG protocols enable a set of n parties to obtain private chunks of a shared secret. However, the generation of these shares, and the computation of the common public key itself, are jointly performed. Each participant acts in turn as the dealer in the secret sharing scheme used to distribute shares of its private value. **The advantage of such protocols lies in that they never have to compute, reconstruct or store the secret key nor the secret shares in any single location** [39]. Pedersen was the first to introduce a DKG protocol in 1991 [40]. As explained by Kate *et al.* in [41], the field has been widely researched theoretically. However, some system requirements have often been neglected. Existing DKG protocols rely on heavy assumptions like synchronous communication, ask for a reliable broadcast channel, and do not efficiently detect invalid shares.

The idea that blockchains and smart contracts can augment DKG protocols has emerged over the past few years. They are usually used to dynamically define the set of participating entities, identify faulty parties and establish a common robust and trustworthy key within a network of untrustworthy peers. However, the challenges regarding the scalability and anonymity of blockchains are often overlooked. We argue that both properties are essential for new systems at the age of the widespread adoption of computing and general worry regarding user online privacy [42].

There are several application domains for DKG that justify the necessity of guaranteeing the anonymity and traceability properties for the participants during the execution of the DKG protocol, ranging from anonymous trading and the requirement of *fairness* to e-voting systems and their coercion-resistance property.

Our **second contribution**, named $\mathcal{BAT} - Key$ for *Blockchain-based Anonymous-yet-Traceable distributed Key generation*, is an anonymous-yet-traceable DKG protocol that utilizes a blockchain to provide trust among the participating distrusting entities. We improve traditional DKG propositions with the anonymity property, which guarantees to the peers wishing to participate in the protocol that their identity remains private. As such, this anonymous protocol can still generate a shared public/private key-pair, for a dlog-based encryption or signature scheme. Yet, thanks to the blockchain's immutable records, the protocol guarantees that misbehaving nodes can be identified and held responsible. Finally, we fork from standard blockchain applications by considering the scalability aspect as the main issue. That is why we developed our new technique on a sidechain [43]. A sidechain is a secondary blockchain connected to other blockchains via *two-way peg mechanism* by which assets are transferred from the mainchain to the sidechain and back at a fixed or pre-deterministic exchange rate. We show how it provides the scalability needed for mass-market applications from a theoretical and practical point of view.

1.2.3 $\mathcal{TOAD}$ and Threshold Anonymous-yet-Accountable Decryption

With the emergence of the Internet of Things (IoT), many low resources and constrained devices can communicate, and many security challenges unfold from their limited resources (*e.g.,* computational power, battery, and memory space). An axis of research consists in delegating computations, for instance, encryption, to an external third party. However, this method is risky as it requires the user that requests the decryption service to trust the server blindly. And this has two main consequences. Firstly, this Trusted Third Party (TTP) can deny its service to the requester, either maliciously - we call it *censorship*, or not. In both cases, the *availability* property, which guarantees that a service can be provided, is violated. The second issue is related to the *accountability* of the server: how can the requester know that the retrieved plaintext indeed corresponds to the decryption of the sent ciphertext?

One way to alleviate the decryption server's authority over regular users is to distribute its role among a set of parties. This is known as *distributed cryptography.* It was labeled in 1994 by Desmedt [44]. It consists of delegating a cryptosystem's operations in a fault-tolerant way among a group of *entities*, which can be processes or servers. Through distributed cryptography, the secret key is shared between several users such that they can recover it only if all of them collaborate. If only a determined number t of them is required, then the distributed cryptosystem is called *threshold cryptosystem* [40]. These cryptosystems leverage *secret sharing schemes* [45] to build distributed protocols that tolerate faulty parties. Implementing a threshold decryption scheme in place of a simple centralized decryption service and assuming that most of the decryption servers are honest solves the aforementioned problem of censorship, availability, and accountability at the expense of low-security assumptions. Yet, threshold cryptography does not solve one problem: who are the decryption servers? These nodes should be identified, authenticated, and communicate with others to promote their decryption service.

Our **third contribution**, titled $\mathcal{TOAD}$ for *ThredhOld Anonymous Decryption*, builds on threshold encryption and proposes a collaborative decryption process that protects the identity of the decryption servers. To the best of our knowledge, we present the first description of a *blockchain-based threshold encryption scheme* that supports an *anonymous-yet-accountable decryption service.* The blockchain has two purposes: it is used to advertise the service (*e.g.,* nodes that want to provide a decryption service declare their availability and generate a proof that they have the necessary resources to do so), and to trace the actions within the system (*e.g.,* logging each request, and tracking anonymous participation for further analysis if necessary). The threshold encryption alleviates the problems related to a central decryption server: namely censorship and availability. The anonymity during decryption is provided by a combination of a group signature scheme controlled by a distributed authority external to the main application and a mixing routine that contributes to constructing an anonymous DKG protocol.

1.3 Thesis Roadmap

In this last section, we summarize the outline of the thesis:

1. In Part I, we set the stage for our contributions.

    - In Chapter 2, we provide a general overview of ITSs and blockchains. We recall the definition of ITSs and the main characteristics of vehicular communications. We highlight the security threats and draw from a literature review the most popular mitigation techniques and their limitations. In parallel, we define the blockchain technology. We describe the main use cases that leverage blockchain, and analyze their limitations, inherent to the nature of the underlying technology. Finally, we explain that both systems present some similarities and how blockchains can be used to secure the vehicular communications.

    - Then in Chapter 3, we introduce three main primitives in distributed cryptography, namely: group signatures, distributed key generation protocols and threshold encryption schemes. For each scheme, we recall their most popular definition, and the different axes of improvement followed over the years. We are mostly interested in how these primitives can be distributed and how the trade-off between privacy and accountability in their distributed versions can be achieved.

2. In Part II, we present our blockchain-based anonymous-yet-traceable threshold cryptographic primitives and incrementally build the resulting Traffic Reporting framework.

    - In Chapter 4, we start exploring the trade-off between privacy and accountability in ITS-like distributed systems. In an ideal world, in which all the third authorities involved are considered honest, we present how a traditional group signature scheme alone can ensure this compromise and meet the stated objectives. We describe the use of group signatures in the context of ITS, from the setup to the signature opening step. And finally, we draw the limitations of this initial proposition *w.r.t.* the ITS environment and the privacy challenges by relaxing the trust assumptions (i.e. the third authorities are no longer entirely trusted). This relaxation led to the development of our first contribution, called $\mathcal{DOGS}$, as an improvement over the previous proposition when we consider that the third parties involved in the system are no longer trustable, and thus, the system assumptions are relaxed. We describe the new system and adversary models. We also recall the targeted objectives and present the chosen approach and method to meet the goals mentioned above. Then, we detail each step of the $\mathcal{DOGS}$ protocol in the context of ITS. Finally, we analyze this proposition from a security perspective and draw the limitations of the proposition *w.r.t.* the ITS environment and privacy challenges.

- In Chapter 5, we describe our second contribution, called $\mathcal{TOAD}$. We focus on the infrastructure nodes' censorship- and coercion-resistance. We explained why the presence of the Trusted Authority (TA) in Chapter 4 is a threat to users' privacy and accountability, as the centralization of its power induces availability risks. Then, we slightly modify the previous construction of the new-DEN messaging protocol to incorporate the new distributed Issuing authority.

- In Chapter 6, we present our third contribution, titled $\mathcal{BAT}-Key$. Indeed, the argumentation presented in the previous chapter and the construction of $\mathcal{TOAD}$ rely on the existence of a primitive that has yet to be proven. Therefore, we propose a construction of the said primitive, namely of a blockchain-enabled anonymous-yet-traceable DKG protocol which would complete the presented security framework.

3. In Part III, we conclude the thesis, list the achievements and draw the future work.

- In Chapter 7, we recall the context and summarize the challenges related to the securing of vehicular communications. Then, we combine our three primitives and map them onto the ITSs to build the final privacy-preserving and accountable Traffic Reporting protocol.

- Finally, in Chapter 8, we conclude and open on aspects that can be further improved in future work.
- In addition, we complete the current chapters with Appendix A, in which we detail the implementations of $\mathcal{DOGS}$ and $\mathcal{TOAD}$. We analyze the implementation choices *w.r.t.* two main metrics: the *gas consumption* and the *time of execution*. We draw the limitations of these initial propositions and suggest ways of improvements. Indeed, these are very naive proof-of-concept implementations (written in Python, no optimization work, testing on personal computer) which can be further refined. However, they are interesting in many ways, for instance, as they can illustrate the expected behavior of the protocol and help predicting bottlenecks.

Part I

About ITSs, Blockchains and Privacy

2 Background

"Relying on the government to protect your privacy is like asking a peeping tom to install your window blinds."

— John Perry Barlow

This chapter provides a general overview of ITSs (Section 2.1) and blockchains (Section 2.2). In Section 2.1, we recall the definition of ITSs, characterize the vehicular communications and explicit the security threats that constrain vehicular communications. We also draw from a literature review the most popular mitigation techniques and their limitations. In Section 2.2, we define the blockchain technology. We describe the main use cases that leverage blockchains, and analyze their limitations, inherent to the nature of the underlying technology. Then, we recall the definition of a recent type of blockchains, called sidechain, and explain how it goes beyond the aforementioned constraints. Finally, in Section 2.3, we map both ITS and blockchain systems by highlighting their similarities; and draw the thesis objectives.

2.1 Vehicular ad-hoc networks and communications

In this section, we mainly focus on ITSs and vehicular communications. We detail the system model for a traditional ITS, the different communication technologies and protocol stacks. Then, we highlight the application scenarios in which vehicular communications play the lead role. We recall the resulting security requirements and challenges in light of the adversary model and corresponding attacks. Finally, we present some mitigation techniques and their limitations.

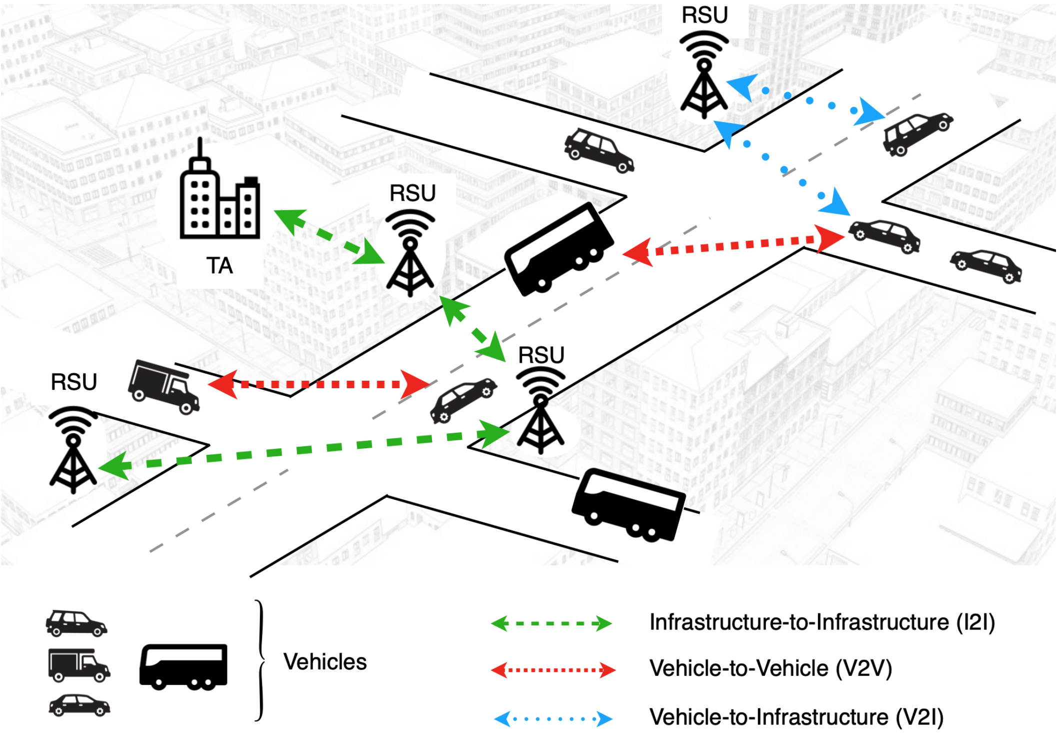

RSU
RSU
TA
RSU
RSU
Vehicles
Infrastructure-to-Infrastructure (I2I)
Vehicle-to-Vehicle (V2V)
Vehicle-to-Infrastructure (V2I)

FIGURE 2.1: Example of an Intelligent Transportation System.

2.1.1 System Model

Figure 2.1 illustrates an ITS [46]. It is composed of ad-hoc nodes, essentially the vehicles, equipped with sensors and radio access technologies that enable them to communicate, and infrastructure nodes, namely the RSUs, Base Stations and Wi-Fi hotspots. Vehicles are often loosely assimilated to the On-Board Unit (OBU) they embed [47]. Traditionally, the ITS architecture consists of four distinct components, the in-vehicle domain, the ad-hoc domain, the infrastructure domain, and the service domain, in which the vehicles can share and access data of various types and utilities [48].

In-vehicle domain. This domain relates to the communications inside the vehicle and depends on its inherent components. The system includes the Communications Control Unit (CCU), the OBU and the Human Machine Interface (HMI). The CCU handles communications from the physical layer (layer 1) to the network layer (layer 3) of the OSI model. It contains transceiver modules that enable interactions via different access technologies (such as Bluetooth or Wi-Fi). The OBU is equipped with hardware (with processing, storage, and communication capabilities, among others) and software to run the various applications that are partly responsible for the transmission of data. Finally, the HMI is a user interface that enables the drivers to exploit the OBU and CCU's capabilities in different use cases and provide relevant information about the road events (*e.g.,* traffic jams and accidents).

The ad-hoc domain. In this domain, a wireless network is formed dynamically between the vehicles to empower inter-vehicle communications. The main protagonists involved in its creation are vehicles and RSUs. Communications can be one-hop, *i.e.* going from a node to another and stopping, or multi-hop, *i.e.* hopping from car to car. The RSU is often used to expand the range of communications of a car.

The infrastructure domain. This domain refers to the backbone of the ITS. It includes roadside wireless infrastructure nodes, namely the RSUs, the Base Stations or the Wi-Fi hotspots, and the underlying wired network, which involves switches, routers and gateways. These infrastructure nodes are often deployed by ITS authorities or telecommunication/service providers.

The service domain. This domain is the top layer of the architecture. It supports the development of services, including traffic-related services (*e.g.,* information on road status provided by competent public authorities) and generic ones (*e.g.,* subscription-based proprietary services).

The goal of ITSs is to facilitate communication among nearby vehicles. Within this four-domain-based architecture, a vehicle can communicate with surrounding peers via two types of communication patterns:

- It uses V2V communications when addressing another vehicle in ad-hoc mode. In that case, it can receive, transmit and exchange valuable information on road and traffic conditions.

- Or, it leverages V2I communications when retrieving information from an infrastructure node. It is also the connection chosen to access the Internet.

However, these communication means are subject to many constraints due to the inherent characteristics of the ITS environment and nodes. More specifically, they must comply with the *high mobility* of the vehicles. As a consequence, they must adapt to the *dynamic* and almost-unpredictable *topology* of these networks. As a consequence of these two first specificities, there are *frequent disconnections* in the network and *attenuations*, worsen by the open nature of ITSs (*i.e.* due to climate interference or density of traffic). Other restrictions affect the transmission medium, namely the air. Although the air is unlimited, the competent authorities (*e.g.,* the ETSI for Europe, the American National Standards Institute or ANSI in the U.S.) defined a standard of communications that restricts its use to a *limited bandwidth of frequencies* (subsection 2.1.2). Moreover, since the transmissions are wireless, anyone equipped with a transmitter can operate in the same frequency band as the actual users. Unlike other types of mobile networks, ITS nodes do not heavily suffer from energy, computing or storage capabilities problems, even though they remain limited due to the size of the systems. However, real-time processing is of utmost importance.

2.1.2 Vehicular communications

ETSI has designed a dedicated protocol stack to comply with the inherent characteristics of ITSs to provide the necessary media of communications, and facilitated the sharing of traffic-related information. It consists of four piled-up and two transverse layers (Figure 2.2).

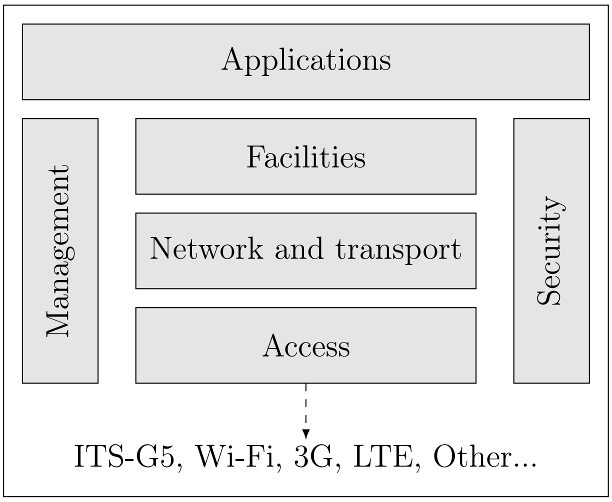


Figure 2.2: ETSI Layers

The *access layer* corresponds to layers 1 and 2 of the OSI model. The dedicated standard is ETSI-ES-202-663 [49]. It describes the standards related to the Physical and MAC layers. The *network & transport layer* replaces layers 3 and 4 of the OSI model. It is covered by the ETSI-EN-302-636 series of documents [50] referred to as GeoNetworking protocol standards. The *facilities layer* corresponds to OSI layers 5 and 6, and is presented in the standard ETSI-TS-102-894-1 [51]. It constitutes the intermediary layer between the application and the network & transport layers. It supports several services, among which the Cooperative Awareness (CA) [5] and the Decentralized Environment Notification (DEN) [6] services. They are two messaging standards that leverage vehicular communications to contribute to safety preservation within the system. The corresponding messages notify a vehicle of the state of nearby nodes (*e.g.,* speed, direction, location) and events on the road (*e.g.,* road hazard warning). These messages are used in the *application layer* by the Basic Set of Applications, that include Active Road Safety (cooperative awareness and road hazard warning) and Cooperative traffic efficiency (speed management and cooperative navigation), as detailed in the ETSI-TR-102-638 document [52]. In addition to these four primary layers, there are two additional sets of standards for *security* and *management.* The standard [53] specifies the security architecture and security management for vehicular communications. It goes through the various topics of certificate formats, trust and privacy management, authentication and authorization services and confidentiality of information. Similarly to the security layer, the *management layer* is transverse to the whole stack as management functions can be implemented at the access layer, the networking & transport layer or the facilities layer. It defines Decentralized Congestion Control (DCC) mechanisms to balance and optimize the use of the stations' resources.

2.1.3 Applications

This stack and corresponding communications standards are designed to provide some essential services to ITS nodes. There are several classifications of ITS applications [54–57]. Overall, there are two major categories: the *safety applications*, and the *non-safety applications.*

As indicated, the non-safety applications do not participate in safety preservation. Instead, they propose other services related to mobility, environment, infotainment. They can be sub-categorized into *comfort applications*, *interactive entertainment*, and *urban sensing.* For example, comfort applications enable a driver to get real-time information on weather, gas station, or restaurant location. Interactive entertainment aims to deliver relevant, entertaining information to the driver and passengers, including internet access, distributed games, or music download. Finally, urban sensing leverages the data collected by the vehicles' onboard sensors to monitor the environmental conditions and social activities in urban areas. They can be used to detect when the air is over-polluted for instance. These applications mainly rely on third parties and telecommunication operators to access the Internet and provide the requested services.

As such, vehicular communications involved in non-safety applications can be treated as any other Mobile Ad-hoc NETwork (MANET) communications.

The *safety applications* instead promote the sharing of information that directly helps preventing road accidents and are therefore closely linked to the ITS environment. They include services related to the safety of the users (*e.g.,* avoiding collisions) and the overall system's efficiency (*e.g.,* reducing road congestion). They rely on the aforementioned dedicated messaging patterns, namely the CAMs and DENMs, to avoid and decrease the number of casualties on road.

In this thesis, we are interested in securing the content and delivery of these messages *w.r.t.* the security requirements that we now introduce.

2.1.4 Security requirements and challenges

The previous sub-section highlights how the widespread adoption of ITSs can empower safety on the road. However, there is a famous quote from Blum and Eskandarian [58] saying:

> *"A wireless network of intelligent vehicles can make highway travel safer and faster. But can hackers use the system to cause accidents?"*

Safety in ITSs is critical because it affects the life of human beings. Therefore, it is essential that the data in the various aforementioned safety-related services is trusted and cannot be compromised. Over the years, there has been a consensus on the security requirements for an ITS system. They go as follows:

- **Requirement 1:***Authentication, identification, non-repudiation and accountability.* This requirement asks that messages are legitimate, *i.e.* generated by senders that are known and authorized to send them. A sender should not be able to deny the transmission of a message. Consequently, due to the critical nature of ITSs, any vehicle must be uniquely identifiable. When a node performs an action that opposes the safety of others or the well-functioning of the systems, it should be held accountable and accordingly be judged.

- **Requirement 2:** *Data consistency and integrity.* This property implies that the message content should be judged plausible (*w.r.t.* the ITS situation) by users evolving in tight space and time with its source. In addition, integrity mechanisms must also be implemented to detect any alteration of messages, performed on purpose, during transit between the source and the destination.
- **Requirement 3:** *Availability.* In case the network is attacked, the aforementioned safety applications should still be accessible to road users.
- **Requirement 4:** *Privacy and confidentiality.* On top of these mandatory security requirements that directly impact the trust one put in a transmitted message, users do not want to expose their identity or personal information while contributing to the system safety. Therefore, even in the case of safety applications, the system must provide a way to protect the data being shared from unauthorized eavesdroppers as it may contain personal identifiers.
- **Requirement 5:** *Real-time.* Finally, since the usefulness of safety messages are limited in time, so are the delivery and subsequent integrity, authentication checks.

Another parameter that complexifies the design of a security framework for vehicular communications is the variety of entities that composes the system, as shown in Figure 2.1. These include the aforementioned users (which equally refers to the driver, the vehicle or the OBU), RSUs, TAs (also called TTPs) but also the attacker.

In the context of ITSs, an attacker is one or more compromised entities working together to violate the security of honest users. There exist different types of attackers and scenarios of attacks which we detail in the following sub-section.

2.1.5 Adversaries and attacks

An attacker's model is usually defined by its location *w.r.t.* the system (inside or outside), its motivation (malicious or rational), its capabilities (active or passive), and its scope (local or extended) [54, 59, 60]. In the following paragraphs, we outline some examples of attacks against the aforementioned security requirements and detail the corresponding model of attackers.

Attacks on availability. They mainly focus on disabling the system by preventing communications between nodes. Denial of Service (DoS) attacks are prevalent and target the availability of the network services by, for instance, flooding the channel with high volumes of messages that cannot be processed. In that case, the attacker is either *inside or outside* the ITS and compromises the network *locally*. It is *malicious*, meaning that it gains no personal benefit out of the compromising of the system. Its goal is to harm users or the functionality of the network, and to this end, it is *active* and may employ unlimited resources. A similar attack performed at the physical layer is called a jamming attack. The attacker acts likewise by producing interference to disrupt the signal and prevent the communications. Other fatal attacks include greedy behaviour attack, blackhole, grayhole, sinkhole and wormhole attacks, broadcasting tampering... [55].

Attacks on authenticity. Sybil attacks, Global Positioning System (GPS) spoofing, tunnelling attacks are all examples of attacks that target the authenticity and identification requirement. For instance, in a Sybil attack, the attacker creates a large number of pseudonymous identities and uses them to gain a disproportionately large power within the system and influence it. If Sybil attacks were not mitigated in blockchains, one could rewrite the entire history of transactions and undermine the immutability property of this technology. In the tunnelling attack, the attacker creates a virtual channel, called *tunnel*, to connect distant network parts. This attack is often jointly performed with a GPS spoofing attack to deceive the vehicle from detecting the malicious node. Yet, the most scathing attack is the node impersonation attack. In that case, the attacker can obtain a valid identifier that traces back to another legitimate vehicle in the network. This attack goes against the uniqueness of the vehicle identifier. The attacker is either *inside* or *outside* the ITS and *actively* engages in the corruption of the system. It usually has a *rational* motivation as it seeks to hide behind another user to perform actions and gain (or at least not lose) from them. In a similar spirit, key and/or certificate replication attacks consist of duplicate keys and/or certificates commonly used for proof of identification. In that case, it introduces ambiguity and violates the accountability requirement.

Attacks on integrity. These attacks range from the simple re-transmission of outdated information (*e.g.,* replay attacks) to the fabrication of bogus messages (*e.g.,* illusion attacks). Among them, we find attacks that focus on tampering, suppressing or altering messages. The *active* attacker, either from *inside* or *outside* of the system, may falsify received data, for instance, by not indicating that a route is congested, to deceive users and increase traffic jams.

Attacks on accountability. The loss of events traceability is of significant concern in ITSs since the users may endanger others' lives. As such, the system should guarantee that they are held accountable for their actions and their messages. Non-repudiation attacks, for instance, focus on the erasure of actions traces and creating confusion for the auditing entity. They often go together with other availability attacks such as Sybil attacks.

Attacks on privacy. They represent an essential violation of drivers' privacy as ITS users. Among them, we find tracking attacks that concentrate on pursuing a vehicle during its journey by analyzing its actions traces. In this example, the *passive* attacker can be *inside* or *outside* the ITS and is *rational* to target identified nodes. It is considered an *extended* attacker as the impacts of the attack can lead to the exposure of the drivers' identities outside the ITS. Other attacks such as Man-in-the-middle attacks and brute force attacks can breach data confidentiality in addition to the privacy threat.

Timing attacks. The type of attacks that target the real-time dimension of ITSs is referred to as timing attacks. The active attacker delays the transmission of the messages to prevent further processing once released.

Over the years, several techniques have been developed to face the attacks mentioned above [4, 7, 55, 57, 61, 62]. They can be categorized into hardware-based and software-based mitigation techniques and will be detailed in the following subsection.

2.1.6 Mitigation techniques and limitations

There are many threats and attacks in ITSs, which will become even more serious due to the popularization of intelligent vehicles. In return, a variety of mitigation techniques that address the aforementioned challenges has been proposed. As mentioned in the Introduction (Section 1), we are mainly interested in the trade-off between privacy and accountability. As such, the following discussion focuses on existing privacy-enhancing technologies only specific to the ITS environment.

Anonymous communications based on Pseudonyms. In [63], Papadimitratos *et al.* describe a framework that targets the securing of vehicular communications. They are one of the first to establish that anonymity of the vehicular network entities is a requirement. In addition, they propose a pseudonym-based anonymous communication protocol that allows vehicles to broadcast messages with pseudonym identifiers instead of their original ones. This technique prevents an attacker from linking a specific car to their exchanged messages as pseudonyms are generated independently. Therefore, they do not contain any identifying information related to the unique identifier attached to the vehicle (by the manufacturer). Pseudonym-based schemes for anonymous vehicular communications have been widely researched as suggested by the numerous surveys on privacy schemes in ITSs [62, 64, 65]. To further enhance anonymity, the use of frequently changing pseudonyms is encouraged. There are four phases in the pseudonym cycle.

① The vehicle is issued a long-term identity by a Third Party.

② The vehicle authenticates itself towards the Certificate/Revocation Authority.

③ If ② is successful, the vehicle obtains a list of pseudonyms along with certificates (preventing the cars from forging pseudonyms) for signing the messages.

④ The vehicle can communicate using the pseudonyms given in ③.

Some schemes also complement the certificate issuance with public/private keypairs for data confidentiality. This method is known as Public Key Infrastructure-based Security and Privacy Schemes [62]. The advantage of such methods is their apparent simplicity (key management is also another field of research [66]). Indeed, the vehicle interacts with one authority to get authenticated pseudo-identities. However, the downside of this simplicity is centralization. Indeed, the Certificate (resp. Revocation) authority has total power over regular users. It can deny the service (breaking availability), issue non-unique pseudo-identities (threatening accountability), track the vehicles (breaching privacy), and many other attacks that involve a single trustful point of failure.

Mix-zone techniques. Inspired by the *mix network* [67] and the *dining cryptographers algorithm* [68], Beresford and Stajano propose in [69] the first description of a mix-zone technique that applies to anonymous communications in location-based applications. The underlying idea is to preserve the users' location privacy when requesting location-aware services and hide their identities from these applications. The mix zone is therefore defined as the spatial area of maximum size in which the users are indistinguishable from the application's point of view (the notion of anonymity set [70]). CARAVAN, proposed by Sampigethaya *et al.* in [71], is one of the first mix-zone-based location privacy scheme for VANETs. They consider the mobility patterns to group vehicles evolving nearby. This group feature allows them to augment the silent period and provide better unlinkability between locations. Other mix-zone techniques have been proposed since CARAVAN. Yet, there are several downsides. The first is the centralized model traditionally adopted to implement the mix zone [72]. Indeed in most cases, all messages exchanged within a mix-zone are encrypted by the public key of a nearby RSU [73]. The RSU becomes the single point of failure and, similarly to the Certificate authority in pseudonym-based anonymous vehicular communications, has too much power over the users. Moreover, existing mix-zone schemes are not flexible as vehicles have to change pseudonyms even when they do not need to. In addition, the effectiveness of mix-zones depends on the vehicles' density. Finally, within the mix-zone, the vehicle is supposedly anonymous from an external observer, but all nodes inside know each other.

Group communications. Another line of work considers forming groups and working on group communications. In an ITS, a set of vehicles can be seen as a group that share common interests. Traditionally in ITSs, group-oriented Privacy Enhancing Technogologies (PET) leverage a well-known cryptographic primitive called group signature scheme. These contributions are extensively studied in Chapter 7 which also provides the related literature review in the context of ITSs. Nonetheless, they have been widely researched outside the context of vehicular networks too and allow group members to sign messages on behalf of the group. This primitive has been privileged over other primitives, such as ring signatures [31] for instance, because it implements a way to track the signer (*i.e.* preserving accountability) while protecting the anonymity of the group member from external observers but also other group members.

Conclusion. While there are other techniques for privacy preservation in ITSs, we argue that a group signature-based privacy-preserving scheme can alone ensure authenticity, anonymity, and traceability for vehicular communications. **We will further explore this line of work in Chapter 3 and Chapter 7.** However, we are still concerned about accountability and related attacks and the absence of satisfying mitigation techniques against the deletion of actions traces. To this end, we suggest the use of a not-so-new-anymore technology called the blockchain. Indeed, we believe that, thanks to its inherent properties (distribution, immutability and transparency), the blockchains can tackle the non-repudiation attacks mentioned in previous paragraphs. In the following section, we present an overview of blockchain technology and recent advances.

2.2 Blockchains and recent advances

In this section, we define the blockchain technology. We describe the main use cases that leverage blockchains, and analyze their limitations. Then, we recall the definition of sidechains and explain how they can address the aforementioned limitations and expand the applicability of the blockchain technology beyond its current scope.

2.2.1 Description of a Blockchain

As the name suggests, a blockchain consists of a growing list of records, called blocks, linked together in a specific order. Indeed, data exchanged within the blockchain framework is organized into blocks to facilitate their propagation, verification and storage. Inside each new block, there is a mention of the latest block in the chain. As such, the blockchain is ordered. It can also be seen as a distributed database resulting from the combination of data exchange, processing and storage technologies inside a peer-to-peer network. It leverages modern cryptography, distributed protocols, point-to-point communication technologies and smart contract programming languages. The first and most popular blockchain technology is Bitcoin, introduced in 2009 [13]. It is a public proof-of-work-based cryptocurrency whose initial goal was to replicate the concept of cash.

2.2.2 A brief history of Blockchains

Blockchains have indeed gained in popularity since the advent of Bitcoin, the cryptocurrency developed by Satoshi Nakamoto [13]. In his working paper, the author defines the technology's essential concepts, including:

- the transactions, *i.e.* the way to distribute a piece of information,
- the timestamp server which ensures the synchronization of the network,
- the proof-of-work also known as the consensus algorithm,
- the P2P network which guarantees decentralization,

- and finally, the notion of incentive, under the form of a reward given to nodes that behave honestly and contribute to the network.

Blockchains are different from distributed databases and Distributed Ledger Technologies (DLT) in that there is no central authority that manages the authentication of the peers nor the verification of the records' validity. Instead, the nodes form the network and together execute a protocol known as the consensus algorithm. They also check whether the new transactions should be added to the previous ones. Blockchains are exceptionally famous because they are resilient in the presence of Sybil nodes (a feature known as fault tolerance) and prevent the deletion of the records (the immutability property). Nakamoto's blockchain network, of which a simple illustration is provided on Figure 2.2.2 A and B, runs as follows:

1. First, peers broadcast new transactions to all surrounding nodes that in turn broadcast them to their closest neighbours until all the network is reached.
2. Each node then individually collects incoming transactions and assembles them into blocks. Blocks are limited in size; therefore, once the block is full, received transactions go in the queue of pending transactions.
3. As soon as a node is able to propose a block, it will run some computations (under the form of hash functions) and finds a proof-of-work. In simple terms, the proof-of-work is the combination of the assembled block of transactions, a nonce (random number) and the resulting hash value such as it is less or equal to a target value called the difficulty. In other words, if $h()$ is the hashing function, the resulting hash value h_n is $h(b, n)$ where b is the block and n the nonce that validates the condition $h_n \leqslant h_0$, the difficulty.
4. When a node finds the combination $c_{b,n} = \{b, n, h_n\}$, it broadcasts it to all its peers.
5. Nodes, in turn, accept the block only if all transactions in it are valid and if $c_{b,n}$ respects the condition on the difficulty.

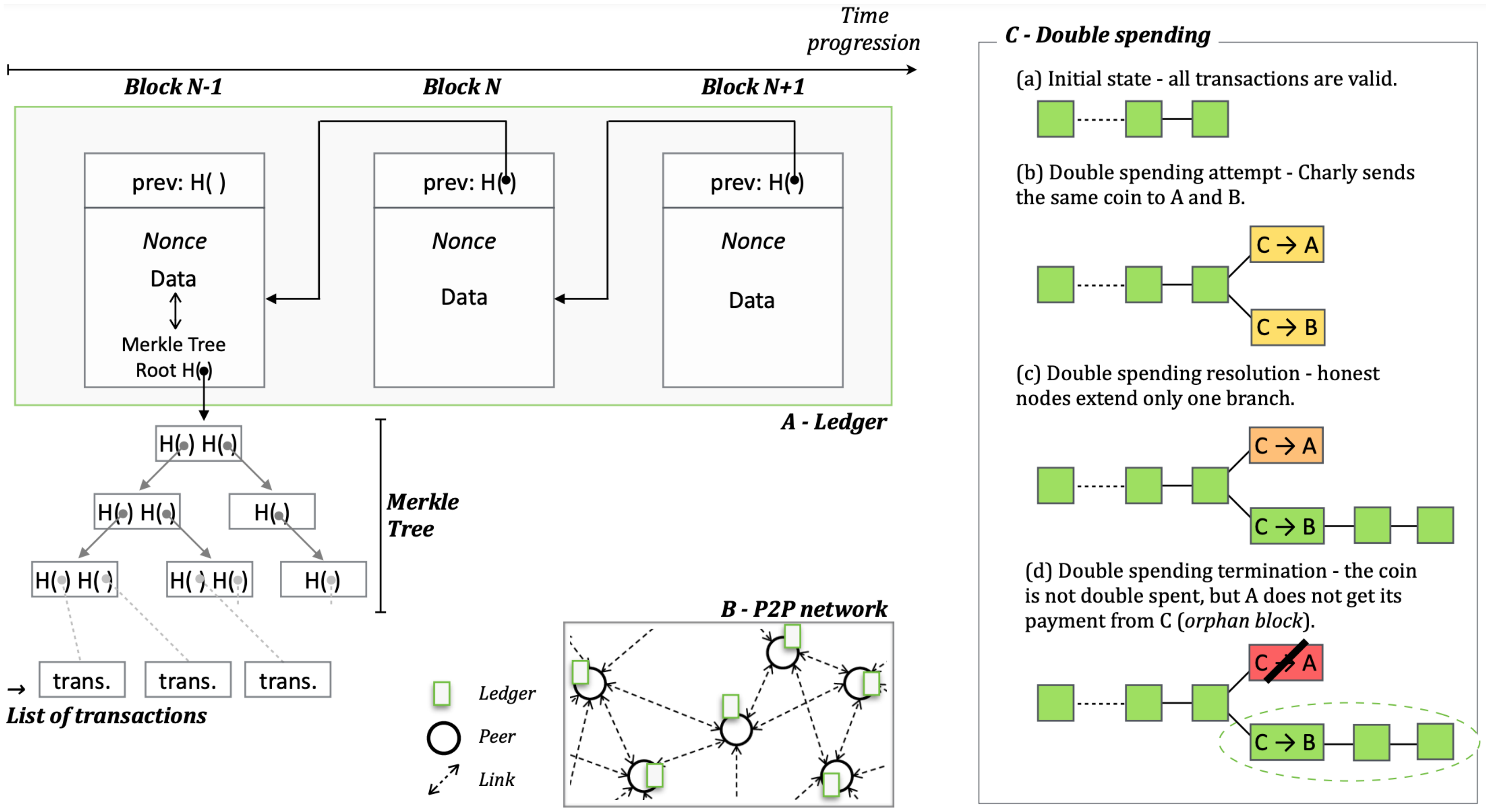

Time progression
Block N-1
Block N
Block N+1
prev: H( )
Nonce
Data
Merkle Tree Root H( )
prev: H( )
Nonce
Data
prev: H( )
Nonce
Data
A - Ledger
H( ) H( )
H( ) H( )
H( )
H( ) H( )
H( ) H( )
H( )
Merkle Tree
trans.
trans.
trans.
List of transactions
Ledger
Peer
Link
B - P2P network
C - Double spending
(a) Initial state - all transactions are valid.
(b) Double spending attempt - Charly sends the same coin to A and B.
C → A
C → B
(c) Double spending resolution - honest nodes extend only one branch.
C → A
C → B
(d) Double spending termination - the coin is not double spent, but A does not get its payment from C (orphan block).
C → A
C → B

FIGURE 2.3: A simple illustration of blockchain components and basic notions, including: the ledger (Sub-figure A) with transactions, blocks, hash pointers, nonces and Merkle Tree; the peer-to-peer network (Sub-figure B) with the ledger replication and nodes' connectivity; and the double spending issue and the longest-chain mitigation technique (Sub-figure C).

6. Finally, nodes notify their acceptance of the block by working on creating the next block in the chain that would contain all the transactions in their pending queue that have not already been inserted in the newly validated block. Moreover, to acknowledge its creation and make it immutable, they add the hash of the accepted block as the previous hash in the new block.

Double-spending attack is performed by one peer trying to transfer the same coin to two different nodes simultaneously. The bitcoin network addresses the double-spending problem by encouraging the peers to consider the longest chain as the correct one and extend it (Figure 2.2.2). In other words, the Bitcoin system does not rely on a single trusted third party to validate and store the transactions. Instead, the correctness of the information stored in the distributed ledger is ensured by the majority's agreement. However, the biggest issue of this principle occurs when 51% of the nodes are malicious. Some argue that controlling more than half the computational resources of the Bitcoin network is either unfeasible (too many participants) or useless (no point in destroying a system you own). Yet, the attack is still theoretically viable and led to the development of alternative solutions as part of Blockchain 2.0 project. Some very complex taxonomies have tried to categorize the different blockchain technologies depending on parameters such as the permission to use and maintain the ledger [74, 75], or the following set of components action, consensus, DL, token [76]. More generally, blockchains can be divided into four main categories:

1. *public and permissionless* - anyone can access the data stored in the ledger and contribute to the network (e.g. propose new blocks, validate the ones received);
2. *public and permissioned* - anyone can join and access the data but only authorized members can maintain the ledger, *i.e.* validate transactions and add new blocks to the previous records;
3. *private and permissionless* - the access to the network is subject to control, but within the network of authorized participants, the permissions are the same for all the nodes (e.g. propose new transactions and validate blocks);

4. *private and permissioned* - the access is restricted to authorized nodes, and only a few of them have the right to manipulate data (e.g. validation, addition functions).

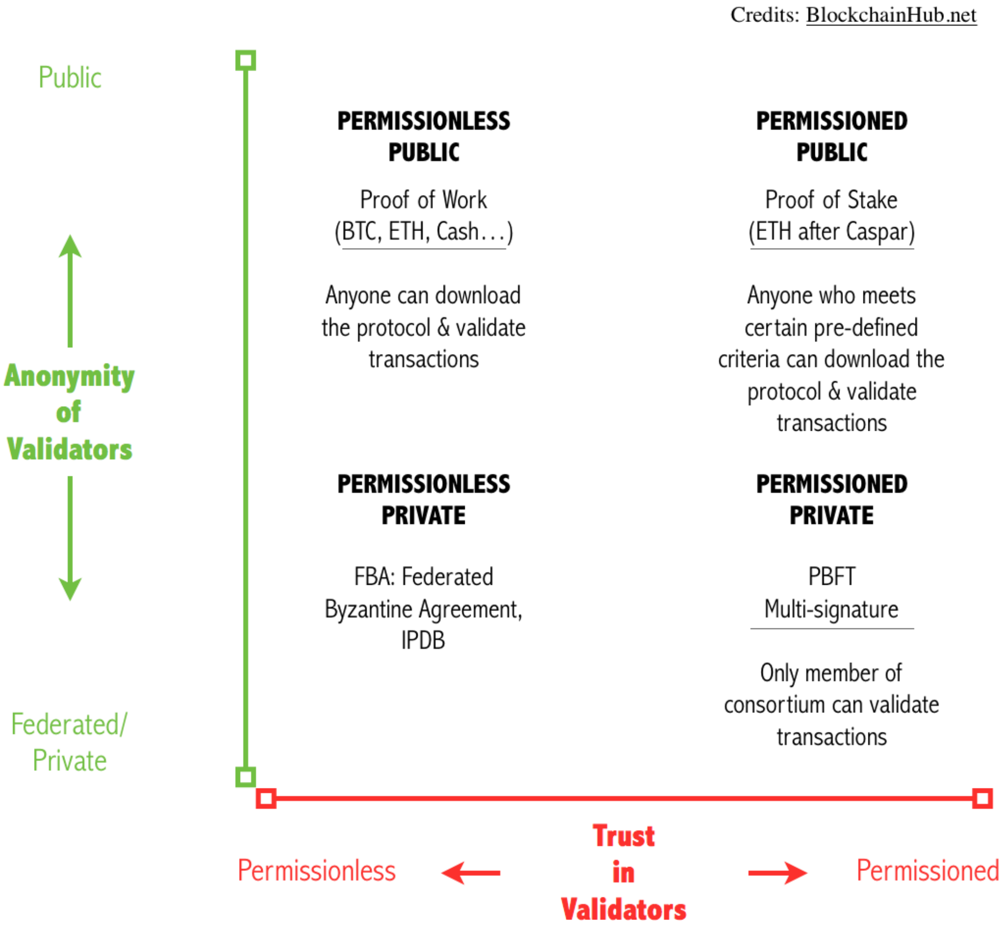

Credits: BlockchainHub.net
Public
Anonymity of Validators
Federated/ Private
PERMISSIONLESS PUBLIC
Proof of Work (BTC, ETH, Cash...)
Anyone can download the protocol & validate transactions
PERMISSIONED PUBLIC
Proof of Stake (ETH after Caspar)
Anyone who meets certain pre-defined criteria can download the protocol & validate transactions
PERMISSIONLESS PRIVATE
FBA: Federated Byzantine Agreement, IPDB
PERMISSIONED PRIVATE
PBFT Multi-signature
Only member of consortium can validate transactions
Permissionless
Trust in Validators
Permissioned

FIGURE 2.4: Classification of blockchain technologies

The consensus algorithm is the core mechanism that deals with selfish, faulty or malicious nodes. It ensures the system's resilience to node failures, network partitioning, delayed, dropped or compromised messages, among others. The design of this algorithm is inherently tied to the type and the application context of the blockchain solution. For instance, Bitcoin is widely adopted by a large community and completely open (no central entity monitors the participants). Thus, the consensus algorithm should rely on a resource that is common to all participants and fairly distributed, *i.e.*

the computational power of their devices (at least that was true before the release of specific machines such as Field Programmable Gate Arrays (FPGA) and Application Specific Integrated Circuits (ASICS)). On the other hand, private blockchain implementations often rely on a smaller P2P network of which authorization policies and authentication mechanisms control the access. Hence, in these cases, using computational power is meaningless and wasteful. The consensus algorithm would privilege a reputation-based mechanism that weights the votes of the peers according to their reputation in the network.

Since 2009, more than 1600 cryptocurrencies have been created. Garriga et al. in [77] study the top 11. About 30 consensus algorithms have been identified by the Web3 Foundation [1], and among them, 18 have been analyzed by Xiao *and al.* in [78]. Not all these propositions are equally popular, nor can they all compete with the original blockchain implementation. However, two names are gaining interest from the community: Ethereum and Hyperledger Fabric, especially since, unlike Bitcoin, they both enable users to run pieces of code on-chain. They are respectively known as smart contracts and chain codes. Another comprehensive taxonomy of blockchains is presented by Bellini *et al.* in [79]. They go beyond existing classifications of blockchains by using the Formal Concept Analysis method [80] *w.r.t.* nine properties (e.g. openness, access management, consensus).

2.2.3 Blockchains and Applications

While originally blockchains served as a methodology to record cryptocurrency transactions, their functionalities have evolved into a number of applications. For instance, in [81], Dai investigates how blockchains could enable a real-time, verifiable and transparent accounting system. The author explains how the technology can securely store accounting data and instantly share relevant information between interested parties. The author suggests that the use of blockchain would improve the verifiability of the business data and facilitate reporting. Since then, subsequent works have explored the feasibility of such a system [82–84], all presenting the blockchain technology as a

[1] https://web3.foundation/

legitimate response to replace trusted intermediaries in a system and strengthening accountability.

That is why other fields of research have started investigating the possibility to use blockchains [85]. From blockchain-based supply chain to blockchain-based IoT, the logging of data inside a public distributed ledger reinforces the trust of the users in the system. Two prominent examples of such phenomena are blockchain-based e-voting and health systems. However, due to the critical nature of data these systems handle, new privacy preservation and data confidentiality concerns emerge in both cases.

Blockchain-based e-voting systems. When it comes to online electoral systems, also referred to as e-voting systems, citizens are concerned with the censorship an organization can operate on the system. Most of all, how to prevent a powerful third entity from tampering with the votes cast by users? One solution that has been found is to leverage the decentralized nature of blockchains and immutability of their records [86, 87]. The distribution of the votes database reduces the risk of database manipulation as faulty records would be detected and cheaters identified. In [88], the blockchain is used as a data recording system. The nodes are used to collect votes from the users. They are connected via the blockchain network. Each of them participates in turn in the generation of a block after an election process. The blockchain model used is similar to Bitcoin, yet instead of storing transactions, votes are. One apparent limitation is the absence of data confidentiality; the votes are transmitted in plaintext. Subsequent works have tried to remedy the issue. For instance, in [89], Hjálmarsson and Hreiõarsson propose to use an Ethereum-based private blockchain instead. The smart contract definition of the election emphasizes its security and cost efficiency while guaranteeing voters' privacy. The proposed scheme brings the novelty of using smart contracts deployed on a private blockchain and defining roles in the system. However, it is unclear how or if voter privacy is preserved, which is a key element for public adoption. Fusco *et al.* propose *crypto-voting*, a research proposal to design an e-voting system using blockchain. While the paper is not detailed, it presents the novel idea of using joint blockchains to separate the collection of voters and their votes from counting votes and issuing final results. Recently, in [90], Goundar *et al.* suggest addressing the privacy and data confidentiality related issues by using permissioned blockchains as it is more flexible and because access control can be defined *w.r.t.* the application's needs.

Blockchain-based Health Systems. One primary advantage of using blockchains in the healthcare industry is that it facilitates the interoperability between healthcare actors and systems (from healthcare databases, providing increased access to patient medical records, to prescription databases and hospital assets) [91–93]. MedRec is one of the first blockchain-based record management systems for Electronic Health Records (EHRs). They use a blockchain that supports the use of smart contracts to log patient-provider relationships associated with a medical record. They can also define the viewing permissions and data retrieval instructions on external databases. The blockchain does not store the data but hash values for data integrity checks. Instead, they use a data exchange off-chain channel established between a patient and a provider database. The blockchain is used to track the data and its evolution throughout the life of the patient. Subsequent studies have focused on facilitating access to the medical data by patients, providers and authorized third parties while still keeping the patient information private. In [94], Vora *et al.* propose a blockchain-based framework for efficient storage and maintenance of the EHRs. They consider a larger system, including a database manager, which maintains patient health records, and a cipher manager, which ensures the encryption and decryption of these records. They define five contracts ranging from the classification contract that manages the composition of the network to the permission contract that defines the access rights to data. Once again, the blockchain is used to give the patient complete control over their data and enable them to monitor the actions undertaken over it.

2.2.4 Inherent limitations of Blockchain

Blockchain technology is desirable for its key properties. First, it is decentralized, fighting by design censorship and single points of failure. Then, it is transparent and open-source and rises against proprietary software from its conception. As an example, the strength of Bitcoin lies in its community, who kept checking the source code of the technology and suggested improvements over the years. Also, records are immutable and cannot be changed unless the adversary takes control of more than 51% of the network's discriminating resource (which is highly unlikely).

Open issues and current drawbacks. There are still a number of challenges in blockchain-based applications, for example in terms of scalability [26], performance [27], privacy [95] and security [28]. This led to a huge surge in the interest toward blockchain and the development of blockchain-based systems in recent years, as shown by the increase in the number of blockchain-based cryptocurrencies. A recent proposition for tackling these challenges while still promoting interoperability was formulated by Back *et al.* in 2014 [43]. They named their technology *sidechains*.

Sidechain. Back *et al.* first presented the concept of sidechains in [43] to address some of the Bitcoin blockchain shortages [13, 96, 97]. The application primarily focuses on providing flexibility in trade-offs (scalability vs decentralization, security vs cost) to adapt the Bitcoin blockchain to the considered use case. Also, it is supposed to contribute to enlarging its scope of application to smart contracts, for instance, and developing new features such as privacy-preservation or censorship-resistance (e.g. use of ring signatures [98, 99]). Lastly, the authors observe that the technology is a closed environment in which the *same set of algorithms* safeguards assets. Extending the Bitcoin blockchain with sidechains would reinforce its security as low-value or high-risk transactions could be made outside the main chain. A sidechain is a secondary blockchain connected to other blockchains through a *two-way peg mechanism*. A two-way peg is a mechanism by which assets are transferred from the mainchain to the sidechain and back at a fixed or pre-deterministic exchange rate. The sidechain may have its own protocol and implementation, which can be completely different from the main blockchain. This versatility enables the design of other functionalities. Users that already own assets on the main blockchain can access those features by transferring some on the sidechain. Besides, the sidechain is isolated from the mainchain. Therefore, if it is subject to an attack or a cryptographic fault, the resulting damage is entirely circumscribed to the sidechain environment itself.

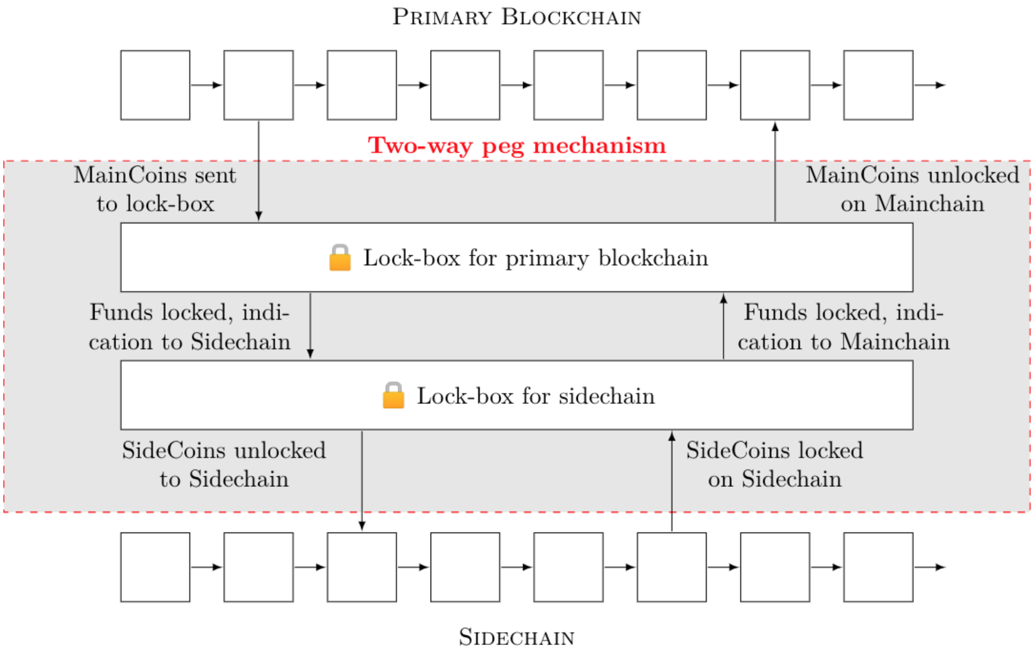

Primary Blockchain
Two-way peg mechanism
MainCoins sent to lock-box
MainCoins unlocked on Mainchain
Lock-box for primary blockchain
Funds locked, indication to Sidechain
Funds locked, indication to Mainchain
Lock-box for sidechain
SideCoins unlocked to Sidechain
SideCoins locked on Sidechain
Sidechain

Figure 2.5: Illustration of a two-way peg [1]

Two-way peg. Sidechains rely on a two-way peg mechanism for locking and unlocking coins from and to the mainchain. According to Singh *et al.* 2020 state-of-the-art review [1], there are three major design choices proposed in the literature to implement two-way pegs. The first one, labelled *centralized two-way peg*, is also the simplest and relies on a trusted third party to lock the assets when transferred back and forth the mainchain (Figure 2.5). The main drawback with the latter proposition lies in the presence of the trusted third entity that comes with its flaws: it is a centralized authority in a decentralized system that leads to a conflicting design policy. Moreover, it introduces a single point of failure and facilitates censorship. An improvement over the previous design is the *federated* (or *multi-signature*) *two-way peg* model. The difference sits in the authority that holds the assets in custody. Instead of relying on the power of a single node, the authorization process is delegated to a majority of federated entities.

These are designated beforehand to control the lock-box and must agree for this transaction to be processed. While this method improves upon the previous one, it is not optimal in practice as the group of regulators is often small. Moreover, the assets may be stolen from the lock-box if the majority of the federated nodes is compromised [100]. That is the reason why the third model was developed. Simplified Payment Verification (SPV) enables *"a lightweight client to prove that a given transaction is included in a legitimate block of the longest Proof-of-Work blockchain without downloading the entire chain"*. The idea behind the SPV proofs is to remove the middlemen and empower a node to prove directly to another one the claim that it owns some specific assets. The disadvantage of an SPV-based design is the slowness of the chain (the user needs to wait for confirmation and reorganization periods before having access to its assets).

Smart Contracts. The most popular sidechain platforms were designed to provide flexibility, scalability, and interoperability between existing blockchain networks. Hence, it is no surprise to notice that most handle smart contract programming.

According to the current literature, a smart contract is *"a computerized transaction protocol that implements the terms of the contract"*. The term was first introduced in 1994 by Szabo, who initially defined a smart contract as

> *"a set of promises, specified in digital form, including protocols within which the parties perform on these promises"* [101]

In the context of blockchains, we adopt Wang *et al.* 's definition:

> *"a smart contract is an executable code that runs automatically on the blockchain by consensus nodes without any trusted third party"* [102]

Some examples of platforms that support smart contract development include the POA Network [103], and RSK [104]. The POA Network is said to facilitate the development of decentralized applications from Ethereum to its native environment. Famous use cases focus on dealing with the scalability and high gas consumption of the Ethereum network. However, the architecture of the POA Network and the restricted number of validator nodes lead to centralization and generate governance issues. RootStock is also an attractive candidate: it is an open-source sidechain pegged to the Bitcoin main-net and supports the execution of smart contracts. It is mainly used in Decentralized Finance (DeFi) for retail payment systems, supply chain traceability, and digital identity management. However, the major drawback of this solution lies in its non-openness.

2.3 Thesis' objectives

As we have seen over the previous paragraphs, vehicular communications contribute to better safety and efficiency in the ITSs by facilitating the sharing of traffic-related information. However, safety often comes at the expense of poor security and privacy. Indeed, vehicular communications disclose rich details on road users and their mobility habits. It is mainly the case with DENMs that are relayed through the DEN basic service.

About the DENMs. The RHW application comprises multiple use cases with the common objective to improve safety on the road and traffic efficiency by using available communications, namely V2V and V2I. The ETSI-TC-ITC is a document that defines the decentralized DEN basic service towards the support of the RHW application. The DEN basic service is an application support facility provided by the facilities layer (Figure 2.2). It constructs, manages and processes the DENMs via an Intelligent Transportation System Station (ITS-S) application.

The structure of a DENM is presented on Figure 2.6. It consists of two main blocks of data: the header, and the body. Road users' privacy is threatened due to the field `ActionID`, in the `body` block. The `ActionID` value indeed contains the `StationID` data, which is unique for each vehicle.

Requirements. In this thesis, we present an alternative approach based on blockchains and distributed cryptography that meets the three following requirements:

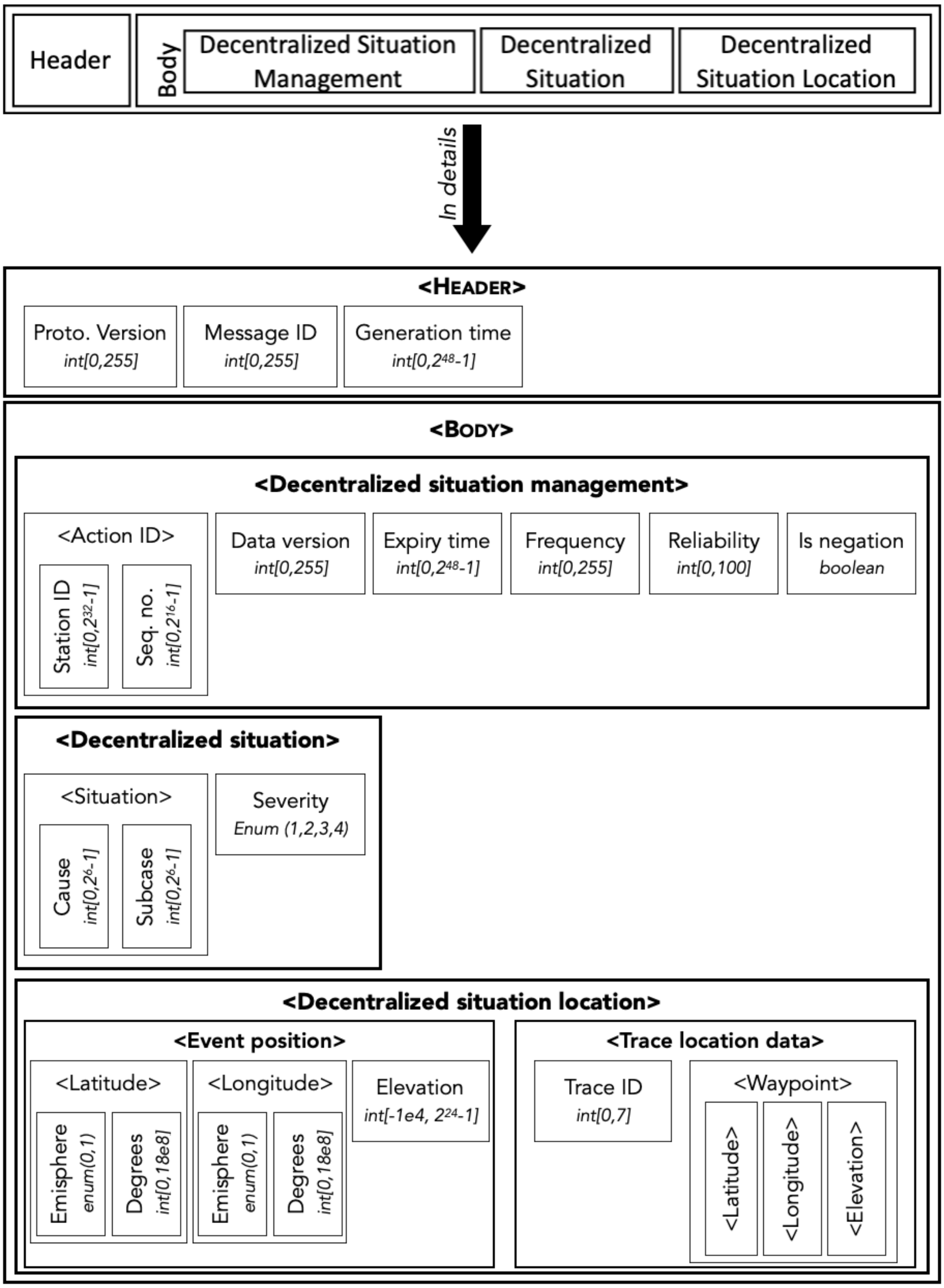


FIGURE 2.6: Detailed overview of the structure and data contained in a Decentralized Environment Notification Message (DENM)

1. Vehicular communications are anonymous;
2. Vehicular communications are traceable;
3. No TTP is required during the setup of the infrastructure.

Indeed, the ITS and the blockchain network present similarities: a decentralized nature, transient components and communication links, untrustworthy peers. The mapping between the two layers, illustrate in Figure 2.7, can be explicit as follows. It seems fair to assume that full nodes in the blockchain layer can handle the role of an infrastructure node and provide the same services as an RSU, as they possess the required amount of resources. Similarly, light nodes can be played by the vehicles as they both lack computational and storage resources. Moreover, their connection with their respective network is transient either because of their physical or, in time, mobility.

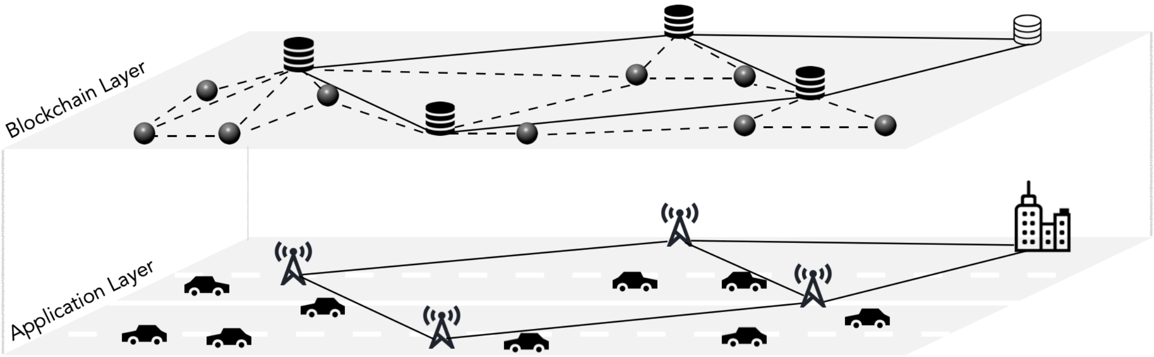


Figure 2.7: A layered overview of the proposed framework

Objectives. We suggest to replace the `ActionID` field with a new anonymous-yet-traceable parameter. To be more specific, this parameter is a signature obtained via the execution of a new group signature scheme of our design. Hence, the objective of the thesis is to build the blockchain-based cryptographic framework that will enable this replacement while still enforcing the stated requirements:

1. "Vehicular communications are anonymous": The new parameter should not reveal the user/station identity nor be linkable to past values of this parameter.

2. “Vehicular communications are traceable”: The new parameter should be:

    - **authentic**: a valid value implies that the message sender deliberately generated it for the associated message;
    - **unforgeable**: only the sender can give a valid value for the associated message;
    - **non-reusable**: this value cannot be attached to another message;
    - **non-repudiable**: the sender cannot deny having generated this value for the associated message;
    - **integrity**: the value ensures the content of the associated message has not been modified.

3. The absence of the TTP implicitly means that anyone can be eligible to be an RSU. The collection of safety and non-safety messages sent by many vehicles requires the deployment of a considerable number of RSU stations in different locations. This can be very costly, and it is not clear who will fund this installation. The idea here is to develop the concept of RSU-as-a-service, which should present with the following characteristics:

    - distribution: the group of nodes that play the role of an RSU is decentralized and not controlled by a TTP;
    - the addition/deletion of a node in this group or any other action on or by the group is traceable, and this historical data is public and immutable;
    - when non-safety RSU services are requested, the decision to provide the service or not is subject to a group consultation and requires a broad consensus.

Organisation. Stemming from the previous analysis, it appears that using a blockchain may naturally answer some challenges. In addition, we would like to suggest that the `ActionID` value be replaced by a blockchain-based anonymous-yet-traceable group signature, called $\mathcal{DOGS}$. In the remaining of the thesis, we will therefore analyze the privacy risks incurred by vehicular communications (Chapter 3). Then, we propose a blockchain-based cryptographic framework, built up around $\mathcal{DOGS}$, to address the identified threats (Chapters 4,5 and 6).

3

Literature review

"I have not failed. I just found 10000 ways that won't work."

— Thomas Edison

In this chapter, we introduce three main primitives from distributed cryptography which are building blocks of our protocols in the later sections. Namely, we review: group signatures in Section 3.1, distributed key generation protocols in Section 3.2 and threshold encryption schemes in Section 3.3. For each scheme, we recall their definition, and the different axes of improvement followed over the years. We present some use case literature that illustrates their prevalence but also their limitations in terms of centralization and privacy. Then, we focus on two main features lacking in the traditional constructions of these primitives: the distribution of these primitives via the use of blockchains; and the trade-off between privacy and accountability in distributed versions of these primitives.

3.1 About Group Signatures and Decentralization

We start with group signature schemes. We recall their standard definition [34], and discuss several improvements that were proposed over the years. We discuss the literature that focuses on distributing the opening functionality and related challenges. Finally, we analyze the most recent works in blockchain-based group signature schemes.

3.1.1 Definition and history

Group signature schemes were first introduced in by Chaum and van Heyst [32] as a generalization of the notion of membership authentication. The new primitive enables a user, also referred to as *group member*, to convince a verifier that he belongs to a certain group, without revealing his identity. This construction presents three features:

1. only members of the group can sign messages;
2. the receiver can check the validity of a group signature but cannot discover the identity of the signer;
3. and if necessary, the signature can be opened and reveal which group member signed the message.

Their paper details four constructions, from the most naive version, in which each group member gets a list of identities to use, to more intricate ones, as, for instance, the fourth proposition that leverages zero-knowledge proofs [105]. Each construction relies on a TA and defines two protocols: the confirmation protocol on one side, which enables group members to claim a signature, and the disavowal protocol on the other, which allows them to deny a signature.

From 1991 to approximately 2003, group signatures have been extensively studied. Several schemes have been proposed either to improve the performance [106–108] of the original proposition; or to provide additional functionalities compatible with the dynamics of the groups [107, 109, 110], such as the revocation of users or the addition of group members with time.In [107], Camenisch and Stadler focus on improving the efficiency of previous group signature schemes. The first improvement results in a public key of constant size instead of linear in the size of the group for previous constructions. The second concerns the dynamic addition of new group members. In previous works, it was necessary to modify the public key of the group to allow new members in. To address both flaws, they use efficient proofs (also called signatures) of knowledge of Double Discrete Logarithms (DDL). The security of the scheme unfolds from the intractability of the Double Discrete Logarithms Problem (DDLP), which asks that an opponent should solve the Discrete Logarithm problem twice. The resulting group signature is an encryption of the membership key of the signer and proof that they know a valid membership certificate. Another line of work consists in addressing the revocation issue. Bresson and Stern propose the first group signature scheme targeting this open problem [109]. More specifically, they improve on [107] and propose to use a Certificate Revocation List (CRL) maintained by the group manager to advertise recently revoked public membership keys. In that case, signers must prove, in a zero-knowledge fashion described in their paper, that their membership key contained in the signature is not present in the revocation list. The presence of a CRL addresses the previous issue of practicality, as it does not require the intervention of the group manager during signature verification. Moreover, it respects the requirement of *forward secrecy* as the group manager does not have to reveal extra information about the revoked member — other than its revoked status — that may link its identity to past group-signed messages. Indeed, in this case, the forward secrecy property asks that all signatures are generate independently, hence, that opening one of them does not reveal additional information on other non-opened signatures. However, the signature size increases linearly *w.r.t.* the number of revoked members. Moreover, the method requires that the verifier possesses an up-to-date version of the CRL before verifying new signatures, otherwise it may accept signatures from revoked members. Providing a constant-size revocation mechanism remained an open challenge up until 2014 [111].

In 2003, Bellare, Micciancio, and Warinschi gave a formal treatment of the security properties of group signature schemes [112]. The paper captures security ideas from previous works regarding unlinkability, unforgeability, collusion resistance, exculpability and framing resistance. They propose formal definitions for the notions of full-anonymity and full-traceability. They explain how these two security properties entail the security requirements considered to that point. In addition, the paper formalizes the group signature scheme $\mathcal{GS}$ as the combination of a digital signature scheme, a public-key encryption scheme and a non-interactive proof system. They extend their static definition of group signatures in [34] in which they present a new system model composed of two authorities (the Issuer and the Opener) instead of one (the Group Manager). This is so far the first step undertaken towards decentralization in group signature schemes. Following the suggestions of previous studies, the group manager becomes the Issuer and the Opener. The first authority is in charge of issuing a membership certificate to new group members, while the second has the power to open a signature and reveal the signer's identity. This separation is proved to provide more security in case the authorities are dishonest. In addition, the paper presents a new model with strong formal definitions of security and construction of a dynamic group signature scheme proven secure under the general assumption of the existence of trapdoor permutations [113].

In the following paragraph, we will briefly describe Bellare, Shi and Zhang's original work [34], denoted later on as the BSZ group signature scheme, as it will be the basis for later improvements (developed in Chapter 4).Assuming the existence of a digital signature scheme $\mathcal{DS} = (\texttt{KeyGen}, \texttt{Sign}, \texttt{Verify})$ Existentially UnForgeable under Chosen Message Attacks (EUF-CMA) [113], a public-key encryption scheme $\mathcal{AE} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ INDistinguishable under Adaptative Chosen-Ciphetext Attacks (IND-CCA2) [114] and a simulation-sound non interactive zero knowledge proofs of membership in Nondeterministic Polynomial time (NP) languages [115], Bellare *et al.* describe the following algorithms for the construction of their group signature scheme $\mathcal{GS}$ (the complete construction is recalled on Fig. 3.1):

- $\texttt{GKg}$ is the initialization algorithm. A trusted third party TA must execute this group-key generation algorithm to produce the triplet (gpk, ok, ik), where gpk is the group public key, ok the Opener's private key and ik the Issuer's private key (of which the corresponding public keys are broadcast to all).

- `UKg` corresponds to the user-key generation algorithm. One user can execute it to obtain a personal public/private key pair $(upk[i], usk[i])$ which corresponds to a temporary identity. After its execution, the user has a public identity now authenticated by the local authority, namely the Issuer.
- `Join`, `Iss` are two joint algorithms respectively referring to the user requesting to join the group and the Issuer answering. These are interactive algorithms that result if the request is approved, in the addition of the user's public key in the registration table reg. At this point, the user has an authenticated personal public identity (under the form of a certificate) and obtains from the Issuer a signing key $gsk[i]$. This step is what enables the traceability of users by linking their personal public and private keys to a certified signing key. The Issuer is the only entity able to associate both back together.
- `GSig` and `GVf` are respectively the group signing and group signature verification algorithms.
- `Open` can be executed by the Opener only to open a signature and therefore identify a signer.
- Finally, `Judge` is the algorithm used to determine if the proof generated after the opening algorithm, claiming that a certain signature is associated with a specific user, is correct or not.

While presenting the foundations of dynamic group signature schemes, Bellare, Shi and Zhang overlooked the issues of group member revocation and length of the signatures. Therefore, from 2003 to 2010, the research on group signature schemes kept trying to improve its flaws. When it comes to efficiency, Boneh, Boyen and Shacham focus on reducing the size of the signatures, to 200 bytes and offer approximately the same level of security as a regular RSA signature of the same length, through the use of bilinear pairings [116]. Regarding the revocation problem, Boneh and Shacham improve on the previous schemes by formalizing the concept of Verifier-local revocation (VRL) and augmenting the scheme proposed in [116] with this new feature. The concept is

```
Algorithm $\mathsf{GKg}(1^k)$
    $R_1 \xleftarrow{\$} \{0,1\}^{p_1(k)}$ ; $R_2 \xleftarrow{\$} \{0,1\}^{p_2(k)}$
    $r_e \xleftarrow{\$} \{0,1\}^{r(k)}$ ; $(pk_e, sk_e) \leftarrow \mathsf{K_e}(1^k; r_e)$
    $(pk_s, sk_s) \xleftarrow{\$} \mathsf{K_s}(1^k)$
    $gpk \leftarrow (1^k, R_1, R_2, pk_e, pk_s)$
    $ok \leftarrow (sk_e, r_e)$ ; $ik \leftarrow sk_s$
    Return $(gpk, ok, ik)$
```

```
Algorithm $\mathsf{UKg}(1^k)$
    $(upk, usk) \xleftarrow{\$} \mathsf{K_s}(1^k)$
    Return $(upk, usk)$
```

```
Algorithm $\mathsf{GVf}(gpk, (m, \sigma))$
    Parse $gpk$ as $(1^k, R_1, R_2, pk_e, pk_s)$
    Parse $\sigma$ as $(C, \pi_1)$
    Return $V_1(1^k, (pk_e, pk_s, m, C), \pi_1, R_1)$
```

```
Algorithm $\mathsf{GSig}(gpk, \mathbf{gsk}[i], m)$
    Parse $gpk$ as $(1^k, R_1, R_2, pk_e, pk_s)$
    Parse $\mathbf{gsk}[i]$ as $(i, pk_i, sk_i, \mathrm{cert}_i)$
    $s \leftarrow \mathsf{Sig}(sk_i, m)$ ; $r \xleftarrow{\$} \{0,1\}^k$
    $C \leftarrow \mathsf{Enc}(pk_e, \langle i, pk_i, \mathrm{cert}_i, s \rangle; r)$
    $\pi_1 \xleftarrow{\$} P_1(1^k, (pk_e, pk_s, m, C),$
            $(i, pk_i, \mathrm{cert}_i, s, r), R_1)$
    $\sigma \leftarrow (C, \pi_1)$
    Return $\sigma$
```

```
Algorithm $\mathsf{Open}(gpk, ok, \mathbf{reg}[], m, \sigma)$
    Parse $gpk$ as $(1^k, R_1, R_2, pk_e, pk_s)$
    Parse $ok$ as $(sk_e, r_e)$; Parse $\sigma$ as $(C, \pi_1)$
    $M \leftarrow \mathsf{Dec}(sk_e, C)$; Parse $M$ as $\langle i, pk, \mathrm{cert}, s \rangle$
    If $\mathbf{reg}[i] \neq \varepsilon$ then
        Parse $\mathbf{reg}[i]$ as $(pk_i, sig_i)$
    Else $pk_i \leftarrow \varepsilon$ ; $sig_i \leftarrow \varepsilon$
    $\pi_2 \leftarrow P_2(1^k, (pk_e, C, i, pk, \mathrm{cert}, s), (sk_e, r_e), R_2)$
    If $V_1(1^k, (pk_e, pk_s, m, C), \pi_1, R_1) = 0$ then
        Return $(0, \varepsilon)$
    If $pk \neq pk_i$ or $\mathbf{reg}[i] = \varepsilon$ then return $(0, \varepsilon)$
    $\tau \leftarrow (pk_i, sig_i, i, pk, \mathrm{cert}, s, \pi_2)$
    Return $(i, \tau)$
```

```
Algorithm $\mathsf{Judge}(gpk, i, \mathbf{upk}[i], m, \sigma, \tau)$
    Parse $gpk$ as $(1^k, R_1, R_2, pk_e, pk_s)$
    Parse $\sigma$ as $(C, \pi_1)$
    If $(i, \tau) = (0, \varepsilon)$ then
        Return $V_1(1^k, (pk_e, pk_s, m, C), \pi_1, R_1) = 0$
    Parse $\tau$ as $(\overline{pk}, \overline{sig}, i', pk, \mathrm{cert}, s, \pi_2)$
    If $V_2(1^k, (C, i', pk, \mathrm{cert}, s), \pi_2, R_2) = 0$ then
        Return 0
    If all of the following are true then return 1
    Else return 0:
        - $i = i'$;
        - $\mathsf{Vf}(\mathbf{upk}[i], \overline{pk}, \overline{sig}) = 1$;
        - $\overline{pk} = pk$
```

```
Algorithm $\mathsf{Join}(\mathrm{St}_{join}, M_{in})$
    If $M_{in} = \varepsilon$ then
        Parse $\mathrm{St}_{join}$ as $(gpk, i, upk_i, usk_i)$
        $(pk_i, sk_i) \xleftarrow{\$} \mathsf{K_s}(1^k)$ ; $sig_i \leftarrow \mathsf{Sig}(usk_i, pk_i)$
        $\mathrm{St}'_{join} \leftarrow (i, pk_i, sk_i)$ ; $M_{out} \leftarrow (pk_i, sig_i)$
        Return $(\mathrm{St}'_{join}, M_{out}, \mathsf{cont})$
    Else
        Parse $\mathrm{St}_{join}$ as $(i, pk_i, sk_i)$
        Parse $M_{in}$ as $\mathrm{cert}_i$
        $\mathrm{St}'_{join} \leftarrow (i, pk_i, sk_i, \mathrm{cert}_i)$
        Return $(\mathrm{St}'_{join}, \varepsilon, \mathsf{accept})$
```

```
Algorithm $\mathsf{Iss}(\mathrm{St}_{issue}, M_{in}, dec)$
    $M_{out} \leftarrow \varepsilon$ ; $dec' \leftarrow \mathsf{reject}$
    If $dec = \mathsf{cont}$ then
        Parse $\mathrm{St}_{issue}$ as $(gpk, ik, i, upk_i)$
        Parse $M_{in}$ as $(pk_i, sig_i)$
        Parse $ik$ as $sk_s$
        If $\mathsf{Vf}(upk_i, pk_i, sig_i) = 1$ then
            $\mathrm{cert}_i \leftarrow \mathsf{Sig}(sk_s, \langle i, pk_i \rangle)$
            $\mathrm{St}_{issue} \leftarrow (pk_i, sig_i)$
            $M_{out} \leftarrow \mathrm{cert}_i$ ; $dec' \leftarrow \mathsf{accept}$
    Return $(\mathrm{St}_{issue}, M_{out}, dec')$
```

FIGURE 3.1: Bellare, Shi and Zhang's dynamic group signature scheme (later on referred to as BSZ).

similar to CRL, however only a certain number of verifier nodes should have an up-to-date list. The VRL system ensures, unlike in [109], that signatures are verified in constant time despite the number of revoked entities.

Since 2010, group signature schemes have also benefited from lattice-based [117] and quantum-based [118] cryptography in terms of efficiency (thanks to their parallelizable structure), post-quantum security and security from worst case assumptions.

Yet, in comparison, little work has been done towards a distributed opening functionality *i.e.* in designing a group signature scheme that would not require the presence of a trusted Opener.

3.1.2 Toward a Distributed Opening functionality

Several studies have focused on enhancing a generic group signature scheme with a distributed traceability functionality [119]. In particular, Ghadafi proposes a formalization of the notion of dynamic group signature with distributed traceability [37]. In his paper, the author proposes to distribute the tracing authority (or also called the *opening* authority) among a certain number κ of *"tracing managers"*. The scheme relies on a modified version of a distributed tag-based encryption scheme with public verification. The signature scheme is additionally said to offer strong security requirements, namely: correctness, anonymity, non-frameability, traceability and tracing soundness. The scheme is proven secure in the standard model, it relies on bilinear groups and related complexity assumptions.

A year later, Blömer, Juhnke and Löken propose a variant of the generic static group signatures with distributed traceability [36] and claim it to be more efficient than Ghadafi's construction [37]. They use a threshold public-key encryption scheme to distribute the opening functionality by splitting the Opener's secret key into shares. As opposed to the previous proposition, Blömer *et al.*'s scheme is secure in the random oracle model and builds on Boneh, Boyen and Shacham's proposition [120]. Therefore, the number of participants here is fixed, while it was variable in Ghadafi's paper.

Both propositions present the same drawback. The main problem of centralization lingers: all the shares required to reconstruct the secret are stored in a single place. Therefore, the opponent that compromises this entity gains the capability of de-anonymizing any user. Indeed, while the *opening* authority is distributed (*i.e.* there no longer is a single Opener secret key but multiple shares and each one is given to one entity named *tracing manager*), the computation of these shares is centralized and executed by a "trusted third party" during the setup of the system (via the execution of `GKg` algorithm [37] or `Setup` [36]).

3.1.3 Applications of group signature schemes

The improvements of group signatures made them progressively more suitable to the design of privacy-preserving systems and protocols.

The idea to use group signatures in voting systems started to emerge in 1999 with Nakanishi *et al.*'s paper [121]. They propose to use a linkable group signature scheme, in which anyone can determine whether two signatures were made by the same person, combined with a secret voting protocol. In this particular case, the group signature is supposed to be used only once as the linkability is obtained by committing to the underlying secret key of the group member. Since the secret key is unique and attached to a user during the entire execution of the voting protocol, then so is the commitment. In the same line of work, Alshammari *et al.* [122] propose to combine a quantum-entangled state and group signature scheme to detect cheating during the vote.

Another application of group signature schemes is illustrated by Bringer *et al.*'s work on a new biometric-based remote authentication scheme [123]. The group signature scheme is used during the authentication phase and takes as inputs the biometric data. It contributes to the privacy of the scheme despite the use of personal identifiers.

However, the real playground in which group signature schemes reveal all their capabilities relates to ITSs. The most recent works in the field relate to group signature-based authentication [124**?** , 125] and associated topics such as the development of efficient, scalable and secure key distribution techniques [126, 127], or the design of attack detection systems [128].

3.1.4 Blockchain-based Group Signature schemes

There are in general two categories of works on blockchain-based Group Signature schemes. The first demonstrates how blockchain benefits from the use of traditional group signature schemes. The second shows how using blockchain can improve group signatures and address the issue of centralization.

In [129], Zhang *et al.* propose a group signature scheme used for the validation of blockchain blocks to address the inherent issues of blockchains, including double-spend attacks or selfish mining (Chapter 2). In this scheme, they use BLS signature scheme [116] for their aggregation property. Therefore, they consider that a block is valid if and only if it is signed with a valid group aggregate signature. This signature is the result of the aggregation of group signatures produced by the members of the group to which the block creator belongs. The idea is quite interesting when it comes to blockchain-based mobile-edge computing as the consensus relies on the computational puzzle. Indeed, since mobile devices are light nodes, it is easy to obtain enough computational power to rewrite the public ledger. Zhang *et al.*'s scheme may be considered as an interesting mitigation technique. Another example on how blockchains can benefit from group signatures is Gong *et al.*'s work. In [130], they focus on the practicality of group signature-based techniques that target privacy protection. The authors propose a new scheme based on threshold group signature, with reduced computational complexity. The threshold nature of the scheme implies that the resulting signature is valid if formed by a threshold number of correct partial signatures. The end goal is to protect user privacy in a mobile blockchain node network based on edge computing. Here, the blockchain guarantees data security and helps establishing integrity assurance for the edge computing system. It also ensures that storage resources allocated on edge devices are balanced and efficient. The use of the threshold group signature guarantees the property of traceability and coalition resistance. Wang also works on improving some aspects of edge computing in [131]. He presents an edge computing data storage protocol that leverages the blockchain and a group signature scheme to provide data protection without relying on a trusted third party. However, since the edge devices have limited computational and communication resources, alike [130], the author's goal is to reduce the complexity of group signatures while preserving their security features. The paper describes a relaxed definition of Clarisse *et al.*'s short group signature scheme [132] by removing the traceability property. In the protocol, the blockchain acts as a platform through which users can request data storage and access services. The Simplified Group signature (SGS) is used to authorize or not the users' requests. The traceability property is not ensured via the group signature scheme but captured at the network level and recording inside the blockchain.

Other studies instead use blockchains to improve group signature constructions. Indeed, group signature schemes are widely used for efficient anonymity and traceability. However, in an untrusted environment, previous propositions suffer from collusion attacks between the two main authorities, called management center or Issuer, and revocation center or Opener, which reduces the security of the scheme. That is why Cao *et al.* propose to replace the Group Manager, or any third authority, with a smart contract [133]. They leverage homomorphic encryption [134], distributed ElGamal proxy re-encryption [135] and group signatures to design a decentralized group signature scheme under the form of four smart contracts. They use the Quorum consensus mechanism [136] which is an algorithm that uses a voting consensus technique on a private blockchain. While the idea of replacing the group manager with smart contracts is interesting in many ways, we are concerned about the feasibility of the solution or the enhancement it proposes. In terms of feasibility, smart contracts, and all the data it contains, are public, which therefore includes private keys for certificate issuance. But once this key is publicly available, how do we prevent counterfeiting? In the same line of work, Devidas *et al.* propose to decentralize the role of the group manager and leverage the blockchain to support their Decentralized Group Signature Scheme (DGSS) [137]. They improve on Lee *et al.* discrete logarithm (dlog)-based group signature scheme and augment it by considering that a user no longer interacts with a single group manager but with a set of managers. During the initialization of the protocol, each manager issues a portion of certificate to the user to be used afterwards during the signing algorithm. They are leveraging the Lagrange Interpolation theorem inside the initialization algorithm for certificate computation and the identification algorithm, where managers pool their shares together to identify a signer. They adopt a similar approach as the one pursued in $\mathcal{DOGS}$. However, they are not interested in the distributed setup phase that identifies and distributively certifies these group managers.

Conclusion In our first contribution $\mathcal{DOGS}$, we develop a blockchain-based group signature scheme with distributed opening functionality. Unlike other presented works, we leverage DKG protocol, instead of traditional secret sharing protocols, to compute the shares of the tracing authority's secret key. The blockchain is used to support the distribution of the Opener. We believe that, while the combination of DKG and blockchain may add computational heaviness to the scheme, it frees the system from any third authority (hence single point of failure) and improves the protocol's security in terms of anonymity while still providing traceability functionality.

3.2 Distributed Key Generation protocols related to $\mathcal{BAT} - Key$

In this section, we focus on DKG protocols. We recall the definition that we will used in future chapters and the different ways they have been enhanced. We discuss the literature that focuses on anonymizing the participation to the protocol and related challenges. And finally, we analyze the most recent works in blockchain-based DKG protocols.

3.2.1 Definition and history

A DKG protocol is a cryptographic primitive in which multiple parties contribute to the calculation of a shared public and private key set. For instance, it is extensively used in threshold cryptosystems to perform secret sharing. It enables a set of n parties to generate a pair of public/private keys jointly according to the decentralization defined by the underlying cryptosystem. The advantage of such a protocol is that they never have to compute, reconstruct or store the secret key in any single location. Moreover, compared to traditional secret sharing schemes, a DKG protocol does not require the presence of a trusted third party (usually called the dealer). The public key is the public output of the protocol, while the private key is maintained as a virtual (since no single party has full knowledge about it) secret that is shared through a threshold scheme. For dlog-based schemes, the DKG protocol is in charge of generating a secret sharing of a random, uniformly distributed value x and revealing the value $y = g^x$ publicly. The first formal description of such DKG was provided in 1999, and revisited in 2007, by Gennaro *et al.* [39].

In the following, we recall Gennaro *et al.*'s secure DKG protocol, denoted GJKR-DKG protocol, and illustrated in Figure 3.2.

The generation part of the protocol starts with each party P_i performing a Verifiable Secret Sharing (VSS), Pedersen-VSS [138], of a random value z_i as a dealer and sends two values to the other P_j's namely $s_{i\to j}$ and $s^{'}_{i\to j}$. Each P_j verifies the shares it receives from other parties. If the k^{th} verification fails, P_j broadcasts a complaint against P_k which is then checked by other P_is. A party is qualified if and only if there are less than t valid complaints against them. Finally, each party can construct the set of qualified parties denoted QUAL and its share of the final common shared secret value as the sum of the shares received from other $P_j \in$ QUAL. Formally the secret value is:

$$x = \sum_{i s.t. P_i \in QUAL} z_i$$

In the reconstruction of the secret value x, qualified parties are first asked to broadcast their shares, which are checked for correctness, and which can then be combined, for instance, via Lagrange interpolation to reconstruct x.

Generating x:

1. Each player P_i performs a Pedersen-VSS of a random value z_i as a dealer:
   (a) P_i chooses two random polynomials $f_i(z), f'_i(z)$ over $\mathbb{Z}_q$ of degree t:

   $$f_i(z) = a_{i0} + a_{i1}z + \ldots + a_{it}z^t \qquad f'_i(z) = b_{i0} + b_{i1}z + \ldots + b_{it}z^t$$

   Let $z_i = a_{i0} = f_i(0)$. P_i broadcasts $C_{ik} = g^{a_{ik}}h^{b_{ik}} \bmod p$ for $k = 0, \ldots, t$. P_i computes the shares $s_{ij} = f_i(j), s'_{ij} = f'_i(j) \bmod q$ for $j = 1, \ldots, n$ and sends s_{ij}, s'_{ij} to player P_j.

   (b) Each player P_j verifies the shares he received from the other players. For each $i = 1, \ldots, n$, P_j checks if

   $$g^{s_{ij}}h^{s'_{ij}} = \prod_{k=0}^{t} (C_{ik})^{j^k} \bmod p \qquad (2)$$

   If the check fails for an index i, P_j broadcasts a *complaint* against P_i.

   (c) Each player P_i who, as a dealer, received a complaint from player P_j broadcasts the values s_{ij}, s'_{ij} that satisfy Eq. 2.

   (d) Each player marks as *disqualified* any player that either
   – received more than t complaints in Step 1b, or
   – answered to a complaint in Step 1c with values that falsify Eq. 2.
2. Each player then builds the set of non-disqualified players *QUAL*. (We show in the analysis that all honest players build the same set *QUAL* and hence, for simplicity, we denote it with a unique global name.)
3. The distributed secret value x is not explicitly computed by any party, but it equals $x = \sum_{i \in QUAL} z_i \bmod q$. Each player P_i sets his share of the secret as $x_i = \sum_{j \in QUAL} s_{ji} \bmod q$ and the value $x'_i = \sum_{j \in QUAL} s'_{ji} \bmod q$.

Extracting $y = g^x \bmod p$:

4. Each player $i \in QUAL$ exposes $y_i = g^{z_i} \bmod p$ via Feldman VSS:
   (a) Each player P_i, $i \in QUAL$, broadcasts $A_{ik} = g^{a_{ik}} \bmod p$ for $k = 0, ..., t$.

   (b) Each player P_j verifies the values broadcast by the other players in *QUAL*, namely, for each $i \in QUAL$, P_j checks if

   $$g^{s_{ij}} = \prod_{k=0}^{t} (A_{ik})^{j^k} \bmod p \qquad (3)$$

   If the check fails for an index i, P_j *complains* against P_i by broadcasting the values s_{ij}, s'_{ij} that satisfy Eq. 2 but do not satisfy Eq. 3.

   (c) For players P_i who receive at least one valid complaint, i.e. values which satisfy Eq. 2 and not Eq. 3, the other players run the reconstruction phase of Pedersen-VSS to compute z_i, $f_i(z)$, A_{ik} for $k = 0, \ldots, t$ in the clear. For all players in *QUAL*, set $y_i = A_{i0} = g^{z_i} \bmod p$. Compute $y = \prod_{i \in QUAL} y_i \bmod p$.

Figure 3.2: Gennaro *et al.*'s secure DKG protocol (GJKR-DKG)

3.2.2 Main results and enhancements of DKG

Over the years, DKG protocols have been improved to address their main limitations. Earlier papers focus on the impracticability of the initial protocols when considering large groups, *e.g.,* Canny and Sorkin's paper [139]. The authors present a sparse matrix DKG for dlog-based cryptosystems that reduce both communication complexity and the running time for the users participating in the protocol.

Recently, researchers essentially focused on addressing strong synchronization requirements [41, 140, 141]. While [41] describes a partially-synchronous DKG, [140] and [141] goes a step deeper by presenting asynchronous DKG algorithms.

Other works instead target the high communication cost of the protocol. For instance, in [142], Gurkan *et al.* introduce a DKG protocol with aggregatable and publicly-verifiable transcripts. Instead of considering braodcast communications, they leverage a gossip-based dissemination protocol which reduces the verification and communication complexities.

To the best of our knowledge, anonymous DKG has not been studied in the literature. The anonymity property applies to the participation to the protocol. We focus on protecting the anonymity of the participating peers when they broadcast their secret during the VSS part of the protocol.

3.2.3 Applications of DKG

DKG protocols have been used to tackle real-world problems. For instance, in [41], Kate *et al.* propose the first DKG protocol for use over the Internet. They formalize a practical model for the Internet and adapt VSS scheme to improve its efficiency. The improvement of DKG comes from the new VSS technique developed and contributes to design a partially asynchronous DKG. In cloud storage, systems need to reduce the amount of redundant data. In parallel, the users demand their data be encrypted to ensure privacy. In [143], Duan tries to find a trade-off between the two aforementioned conflicting requirements. To this end, the author leverages threshold signatures, distributed oblivious key generation and the developed security framework. The resulting protocol leverages DKG to remove the need for a centralized key server, reducing the cost of infrastructure and security protections.

More recently, industrial control systems (ICS) have gained great attention, and their security is highly important due to the impact of industrial system failures on citizens. In [144], Kilinc *et al.* focus on the communication protocols of industrial systems. Indeed, it is challenging to design secure protocols due to their limited resources. In this paper, DKG is used in the context of an ICS environment to improve its robustness and the quality of the data it extracts.

Also, it is worth noting that Beullens *et al.* recently developed a DKG protocol in the Quantum Random Oracle Model [145]. They focus on addressing the open issue of robustly and securely performing a DKG in the very hard homogeneous space setting without needing a trusted dealer.

Another recent area of vivid research is the design of blockchain-based DKG protocols as we will explain in the following subsection.

3.2.4 Blockchain-based DKG scheme

One of the first blockchain-based DKG protocols, called ETHDKG, was introduced by Schindler *et al.* in 2019 [38]. It is an open-source smart-contract-based implementation of a DKG protocol for dlog-based cryptosystems. It leverages Ethereum's smart contracts as a medium of communication. It implements the traditional Joint-Feldman DKG protocol enhanced with Neji *et al.*'s suggestions [146] to tackle biasing attacks. It uses the Ethereum blockchain and smart contracts to dynamically define the set of participating entities, incentivizes participation and penalizes adversarial behaviour. However, since ETHDKG is built on top of Ethereum, the solution is no longer financially viable (in 2019, 1 ETH was worth around 200 USD; it costs 2500 USD as per June 2021).

Arising from the difficulty of developing and deploying smart contracts on Ethereum and the huge cost of execution, Lei *et al.* opted for a blockchain-based DKG protocol that leverages Tendermint instead [147]. In the context of blockchains and cryptocurrencies, it is mainly used to provide a secure and more efficient consensus algorithm compared to the traditional proof-of-work. Tendermint is an easy-to-use software that facilitates the secure and consistent replication of an application on several machines and can replace any consensus engine from other blockchain software. Compared to ETHDKG, the proposition is more scalable.

Another contribution in the field of blockchain-based DKG is BADGER [148]. It is an RSA [134] threshold signature system. Participants collaborate in a distributed setup for a $(t, n)-$threshold signature scheme. The system leverages a distributed ledger to enable communications between these parties. Similarly to ETHDKG, the authors also tie communications and subsequent actions to a single distributed ledger that facilitates (by its inherent characteristics, including its public nature and immutability properties) the auditability of the system. BADGER combines the threshold key generation scheme of Yung, Frankel and Mackenzie [149] with Damgård and Koprowski's RSA-based signature scheme [150].

Conclusion To the best of our knowledge, our second contribution $\mathcal{BAT} - key$, is the first description of a DKG protocol augmented with a blockchain for traceability purposes, that provides anonymity to the parties during the establishment of a common group key. Compared to previous works in which participants are not anonymous from an external observer nor the other participating entities, our protocol is designed to address this flaw. We insist on the necessity of providing a trade-off between the anonymity of the participants and the traceability of their action to prevent coercion, censorship, and cheating. Yet, like them, the protocol fits into the same line of works where scalability is a major feature to consider.

3.3 Threshold Encryption with Anonymous Decryption

Finally, we conclude this literature review with a state-of-the-art on threshold encryption schemes. We recall the main definition we used in subsequent chapters, and the different ways they have been enhanced. We discuss the literature that focuses on anonymizing the participation to the decryption process and related challenges. Finally, we analyze the most recent works in blockchain-based threshold encryption schemes.

3.3.1 Definition and history

Threshold cryptosystems are first introduced by Desmedt in [151] as a continuation of the independent works of Boyd [152] and Croft-Harris [153]. They are presented as a mean towards fighting abuse of power by the government or any organization. Initial schemes were meant to compute joint or threshold signatures such that the recipient of the jointly signed message could only determine that the signature is correct. Later on, Frankel proposes a scheme allowing t-out-of-n shareholders, *i.e.* entities that possess a share of the secret key, to decrypt incoming ciphertexts. The basic scheme considers a cryptosystem with two 'clerks' A and B, *i.e.* the decryption server and a source. The idea is to split the message m to encrypt into $\{m_1, m_2, \ldots\}$. All the m_i with an odd i will be encrypted with A's encryption key, and all the m_i with an even i with B's encryption key, before being transferred. In that way, neither A nor B is able to reconstruct the message without the other's collaboration. The scheme leverages an ElGamal [154] public key encryption scheme and is further extended to a t-out-of-n version.

One of the most popular definition of threshold encryption scheme is given by Shoup and Gennaro in [155]. A t-out-of-n threshold cryptosystem consists of four algorithms $\mathcal{TE} = (\texttt{KeyGen}, \texttt{Encrypt}, \texttt{PArtialDecrypt}, \texttt{Combine})$ defined as follows:

- $(ek, dk_1, \ldots, dk_n) \leftarrow \mathcal{TE}.\texttt{Keygen}(1^k, n, t)$: The *key generation algorithm* takes as inputs the security parameter k, the number of decryption severs n and the threshold parameter t. It outputs the public key ek and a list $dk_1, \ldots, dk_n$ of private key dk.

- $c \leftarrow \mathcal{TE}.\texttt{Encrypt}(ek, m, L)$: the *encryption algorithm* takes a plaintext m, the public key ek and a label L as inputs and outputs a ciphertext c.

- $c_i \leftarrow \mathcal{TE}.\mathtt{PartialDecrypt}(ek, i, dk_i, c, L)$: the *decryption algorithm* takes as inputs the public key pk, an index i, the private key dk_i, a ciphertext c and a label L. Then it outputs a *decryption share* c_i.

- $m' \leftarrow \mathcal{TE}.\mathtt{Combine}(ek, c, L, (c_1, \ldots, c_n))$: On inputs the public key ek, a ciphertext c, a label L and a list of decryption shares $c_1, \ldots, c_n$, the *recovery algorithm*, also called *combining algorithm*, outputs a cleartext m.

From 2000, further work has been done to improve threshold encryption in terms of efficiency [156, 157] (providing constant-size ciphertexts) and security (without random oracles [158]) in the standard model [159, 160]. Other works focused on the dynamic dimension of threshold encryption schemes, such as [161]. In this paper, Delerablée and Pointcheval focus on enabling any user to dynamically join the system as a possible recipient. They build on [156] to generalize the notion of threshold public-key encryption to the dynamic setting. The underlying encryption leverages a hybrid scheme based on the combination of a Key Encapsulation Mechanism (KEM) and a Data Encryption Mechanism (DEM), also referred to as the KEM-DEM methodology [162]. The paper introduces a new security model and proves the scheme to be secure in this model. However, it is still limited as it does not handle adaptative adversaries nor chosen-ciphertext attacks.

Over the years, various underlying encryption schemes have been tried to support threshold encryption: fully homomorphic [163–165], lattice-based [166]. More recently, the research community has focused on using threshold encryption in different use cases.

3.3.2 Applications of Threshold Encryption

In [167], Baraani-Dastjerdi *et al.* present a secret voting scheme that leverages threshold encryption to preserve the privacy of participants, namely the voters, and ensure the accuracy of the votes against any dishonest entity inside the system which can be the voters themselves, the candidates, or the administrator. The threshold encryption scheme used is a generic construction that leverages SSS [45] to distribute N partial encryption functions and keys to N candidates. The voting scheme assumes that each ballot is encrypted and sent to the N candidates. Each candidate decrypts the ballot and sends the result to the counter. Then, the counter computes and publishes the final result. The scheme is proven complete, privacy-preserving and sound. Even though the scheme may not be as practical as it is presented, it provides verifiability, fairness and soundness of the voting process.

Threshold encryption is also popular in addressing the privacy issue of decentralized cloud storage systems. For instance, in [168], Yao *et al.* propose to combine a threshold encryption scheme with a secure decentralized erasure code to design a secure cloud storage system. In this context, the use of threshold encryption prevents the collusion attack (among storage servers and key servers), the stealing of partially-decrypted ciphertexts (from more than a certain threshold number of servers), and even man-in-the-middle attacks.

Overall, threshold encryption is used in many applications requiring additional data security [169, 170]. For instance, Chen *et al.* propose a threshold hybrid encryption method to verify the integrity of cloud data without a trusted center. The underlying encryption combines the Advanced Encryption Standard (AES) and Elliptic Curve Cryptography (ECC) to promote encryption efficiency and guarantee the confidentiality of data and keys. SSS is used to distribute the partial keys to a set of managers, which replaces the trusted center during auditing. In case a manager fails to participate in decryption, the threshold feature guarantees that others will be available to restore the requested data, ensuring the robustness of the system.

3.3.3 Blockchain-based Threshold Encryption scheme

Threshold encryption is largely used in IoT-based environments, for instance, in the context of data collection along different trajectories to build a system capable of superseding the GPS in urban areas. However, for the commercialization of such systems, the designers must guarantee the trustworthiness of the location-dependent input data and the privacy of the participants. This is the trade-off targeted by Kong *et al.* in [171]. They leverage threshold encryption to develop a privacy-preserving location proof acquisition scheme. The primitive is used to hide the location proof from owners such that the proofs can only be successfully verified with the collaboration of at least a threshold number of users.

In [172], Yakira *et al.* present Helix, a blockchain-based consensus protocol for ordering transactions among nodes in a distributed manner. It mainly focuses on the fairness aspect of the ordering. Therefore, it leverages threshold encryption to obfuscate the transactions from the nodes, by design fighting censorship from other greedy participants and ensuring fairness during the election of the validation committee.

Conclusion None of these works have proposed blockchain-based implementation of threshold encryption, nor do they provide anonymity of the decrypting while peforming collaborative decryption. Our third contribution $\mathcal{TOAD}$ is, to the best of our knowledge, the first description of a blockchain-enabled threshold encryption scheme that supports an anonymous decryption service.

Part II

Construction and analysis of the framework

4

Privacy and Accountability from an ideal world to a practical setting

"Every great advance in science has issued from a new audacity of imagination."

— John Dewey

In this chapter, we start exploring the trade-offs between privacy and accountability in ITSs. We present how, in a ideal setting all the involved third authorities are trusted, a traditional group signature scheme can ensure this balance and meet the stated objectives of anonymity and traceability (Section 4.1). Then, by reducing the security assumptions *i.e.* by relaxing the trust model, we draw the limitations of the proposed solution. In Section 4.2, we propose a new group signature scheme with a distributed opening functionality. The construction leverages a distributed key generation protocol and a public bulletin board to meet the same security requirements in a stronger security model.

4.1 Group Signatures for the ideal setting

In this section, we first propose to use a basic group-signature-based authentication scheme to control the DENMs' validity in a privacy-preserving and accountability-enhancing manner. We describe this ideal setup, *i.e.* the system and communications models are perfect, and describe the adversary considered. We present our objectives and detail our approach to meet them. We detail the construction of the new-DEN messaging pattern and describe how the group signature scheme fits inside these ITS messages. We analyze the proposed protocol and draw its limitations when the security assumptions are relaxed.

4.1.1 The setup

Figure 4.1 illustrates the system model. It is delimited by a geographical area also referred to as Region of Interest (RoI). The system is composed of one TA, an RSU, and a group of OBUs. We assume that both the TA and the RSU are *fully trusted*. The OBUs are also trusted as sensors. If an OBU is deviating from the protocol, we consider it as compromised by the adversary.

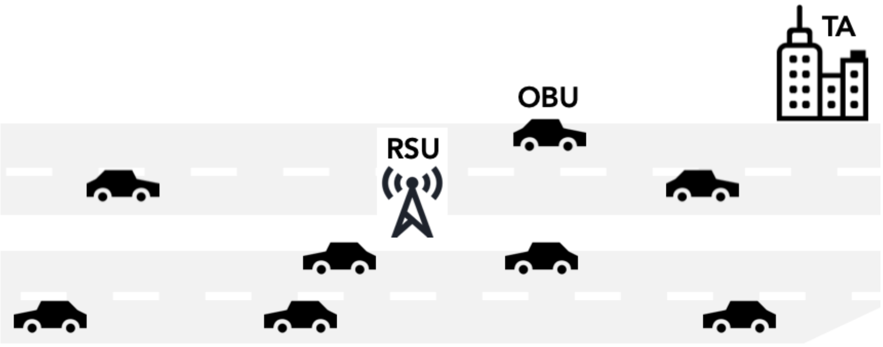


FIGURE 4.1: Illustration of an ideal system model with one Trusted Authority, one Roadside unit and a set of On-Board Units.

Communication model. We focus on the Decentralized Environmental Notification messaging pattern and assume that all broadcast messages follow the DENM construction, as shown in Figure 2.6. We consider that all parties have access to a shared public and authenticated communication channel. We assume that the RSU provisions a view of the broadcast messages in their order of arrival. Doing so provides synchrony in the sense that any message broadcast during one of the protocol phases can be accessed through the RSU database by all other parties before the next phase starts. Each vehicle is denoted V_i, with i an integer that varies from 1 to N the total number of vehicles in the system. The V_is are identified by a unique authenticated identifier Id_i (*e.g.,* given by the manufacturer and certified by a trusted authority). For the sake of simplicity, we assume that $Id_i = i$. V_i possesses a long-term public/private keypair denoted (pk_i, sk_i) (this keypair is associated to i and can be certified by the automotive authorities).

Adversary Model. We consider an adversary that is inside the ITS. As such, it is an authenticated member of the network that can communicate with other members. Moreover, the adversary is active, *i.e.* it can generate packets and send them over other ITS nodes. We consider that the adversary is rational, and this characteristic implies that its attacks are directed. This type of adversary usually does not aim to harm the network members but rather seek personal profit. In our case, for instance, the adversary's capabilities can express as two profiles of attackers.

> **Attacker 1.** The first category will try to breach the anonymity property of the system, *i.e.* to infer the identity of the participating ITS nodes. Therefore, the attacker, denoted A_1, can perform de-anonymization attacks (*i.e.* intercept a message and identify the signer). Such attacks usually consist in eavesdropping and recording all communications in the system for further analysis.
>
> **Attacker 2.** The second category will try to escape the traceability property of the system. To this end, the attacker A_2 can perform key recovery attacks or forgery attacks (*i.e.* create a signature that would defeat the opening procedure

or frame a legitimate user). Such attacks consist in not using the proposed messaging pattern to escape the tracing authorities.

4.1.2 Objectives and approach

Our objectives in this chapter are twofold. Firstly, we want to improve the privacy of the ITS nodes. More specifically, we focus on anonymizing vehicular communications to prevent the adversary from inferring the identity of the communicating parties. Secondly, we want to guarantee the accountability of the ITS users. To this end, we concentrate on tracking the OBUs actions and messages. In the following sub-sections, we explain how we approach this challenge.

Approach.

The definition of *privacy* depends on the context. Traditionally, it represents *someone's right to keep their personal matters and relationships secret*. It essentially relates to the information generated by (e.g. conversations, purchases) and from (e.g. medical records, credit card trails) a human being. In the age of digitization, this translates into *information privacy* also called *data privacy*. In [173], Turn and Ware explain that:

> *"Privacy is an issue that concerns the computer community in connection with maintaining personal information on individual citizens in computerized record-keeping systems. It deals with the rights of the individual regarding the collection of information in a record-keeping system about his personal activities, and the processing, dissemination, storage, and use of this information in making determinations about him."*

Information that could identify individuals is referred to as Personally identifiable information (PII). In the case of ITSs, the vehicle's unique identifier is considered as a PII.

With the massive development of technological innovation, information privacy is becoming increasingly complex as data is being collected and exchanged by so-called smart objects (*e.g.,* smartphones, smart watches, smart cars). The technology gets more sophisticated and invasive, leading to extremely intricate data use and an ever-increasing difficulty to protect them (efficiently). The notion of privacy is closely related to another important property, known as anonymity.

Anonymity. The first attempt to formally define *anonymity* dates back to 1998, with Reiter and Rubin's *Crowds* system [174]. Their new approach for increasing the privacy of web transactions was based on hiding one's actions within the actions of many others. Consequently, they define the degree of anonymity as the probability that an intruder can successfully map a player to its original action. Therefore, to ensure anonymity, their system requires an appropriate set of subjects with the same attributes. That is, the anonymity property can be viewed as *un-identifiability* of a subject within a set of subjects, also referred to as the *anonymity set.*

Accountability. Yet, most distributed systems must continue working even in case of failure or attack. This is known as the *robustness* property. For instance, an ITS should demonstrate robustness to comply with the safety requirements. In such systems, accountability is often considered as an improvement of robustness. Indeed, the accountability property ensures that malicious entities are identified and removed from the system, *i.e.* it incarnates the robustness of the traceability mechanism even in the presence of faulty/malicious entities. Accountability relies on the auditability of a system, *i.e.* on the production of secure, tamper-proof, transparent records of actions pertained by identified nodes inside the system. To provide these logs, distributed systems traditionally implement a tracing mechanism.

Method.

We suggest using group signatures to provide both privacy preservation (under the form of anonymity within a group) and accountability of the anonymous nodes. In Figure 4.2, we present the new DENM structure. We propose to modify the field `ActionID`, in the `body` block. Indeed, the original `ActionID` value contains the `StationID` data, which is unique for each vehicle (as shown on Figure 2.6) and constitutes a privacy threat. In the new-DENM structure, the field has been replaced by a group signature, applied on the header value, and issued by the corresponding station. This replacement adds a communication overhead that corresponds to the difference in size between the `ActionID` value and the group signature scheme chosen (explored in more details in Chapter 7).

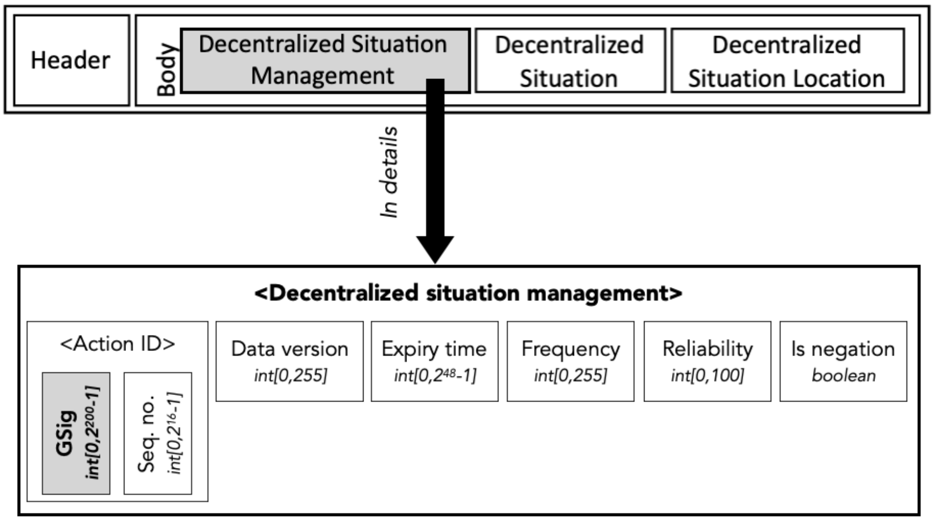


Figure 4.2: Overview of the proposed new Decentralized Environmental Notification Message strucutre.

Definitions.

Consequently, the aforementioned objectives imply that the chosen group signature scheme presents the following four desirable features: correctness (of the group-signature-based scheme), and its anonymity, traceability and non-frameability properties. They are defined as follows. Let λ denote the security parameter.

Definition 1 (Correctness)**.** *The correctness property guarantees that signatures issued by honest users:*

i) should pass the verification,

ii) should trace to the correct issuer if opened with the Opening key, and

iii) the proof output by the `Open` *process should verify the Judge algorithm.*

Definition 2 (Anonymity)**.** *Let $\mathcal{U}_0$ and $\mathcal{U}_1$ be two honest registered users, and σ a valid signature issued by $\mathcal{U}_b$ for some $b \in \{0,1\}$. The* anonymity property *requires that no Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$ can guess b with non negligible (in λ) advantage.*

Definition 3 (Traceability)**.** *An adversary breaks the* traceability property *if she succeeds in creating a valid signature σ such that either:*

i) no registered user can be identified when σ is legitimately opened,

ii) the proof, produced by an honest opening, revealing that σ belongs to user $\mathcal{U}$, does not convince the `Judge` *algorithm.*

Definition 4 (Non-frameability)**.** *Finally, the* non-frameability property *requires that no PPT adversary $\mathcal{A}$ can create a valid signature that would trace to an honest user if opened unless this user has effectively issued it.*

In the following sub-sections, we illustrate the proposed $\mathcal{GS}$-based new-DEN messaging protocol and how it provides the targeted properties. We will analyze it and draw its limitations to justify the construction of a new primitive, called $\mathcal{DOGS}$, that will be presented in the next section.

4.1.3 Description of the new-DEN messaging pattern

Group signature schemes and corresponding literature review are presented in Chapter 3. In this section, we choose a traditional dynamic (meaning that the addition and revocation of users can be done during the execution of the protocol) group signature scheme for dlog-based cryptosystems as defined in [34]. We note here that the choice of BSZ construction in place of any other has been made based on the explicit descriptions of each algorithm constituting the group signature scheme, and of the surrounding security framework that matched our greater applicative context. From our standpoint, we see no objection to using another discrete-logarithm-based group signature scheme. Yet, a thorough analysis of the said construction would be necessary to further conclude.

BSZ construction consists in the following tuple of algorithms: (GKg, UKg, Join, Iss, GSig, GVf, Open, Judge). The RSU plays the role of the Group Manager (GM) *i.e.* it embodies both the Issuer and the Opener authorities. Therefore, it is in charge of the registration of the group members and issues them certificates allowing them to use the signature scheme. Moreover, it ensures the accountability property as it can de-anonymize the group signatures - to open them. In the following subsections, we describe the use of a group signature scheme within an ITS from the setup of the scheme to its actual usage.

① **Setup.** The RSU executes the GKg function of the group signature scheme $\mathcal{GS}$. It generates two pairs of keys: (ipk, isk) for the issuing feature; and (opk, osk) for the opening feature. The RSU forms the group public key $gpk = (pp, ipk, opk)$ which contains the public parameters pp (*i.e.* the parameters of the $\mathcal{GS}$'s underlying primitives) and its public keys for certification and opening. Then, it advertises (broadcasts) gpk to the OBUs in its RoI and publishes the same values onto its public database for free access.

② **OBU Registration.** Each OBU needs to generate a personal and temporary public/private keypair (upk_i, usk_i) and get it certified for further use during the subsequent steps of the protocol. Indeed, instead of using their long-term identifiers (i, pk_i, sk_i), the OBUs create some temporary credentials that are linked to their true identity only inside the RSU's registration table reg. The OBUs execute the UKg function and obtain (upk_i, usk_i). Then, they engage in the Join/Iss joint protocols with the RSU. Each vehicle V_i uses its long-term keys to sign the temporary ones: $sig_i = \mathtt{Sig}_{pk_i}(usk_i)$ where Sig is the underlying signature scheme. Then, V_i sends (i, pk_i, sk_i) and (pk_i, sig_i) to the RSU as a request for registration. The RSU checks that sig_i is correct with the corresponding verification algorithm $\texttt{Vf}$. If so, it computes the certification $cert_i = \texttt{Sig}_{sk_s}(\langle i, pk_i \rangle)$ and sends it back to V_i. In parallel, the RSU adds $(i, pk_i, sk_i, sig_i, cert_i)$ to its private registration table reg.

③ **OBU Signing.** Upon successful registration, the registered OBUs can use the group signature scheme. Consider the hazard defined by some characteristics related to the decentralized situation and its management. When V_i needs to inform the network about this hazard, it will form a new-DEN message. The DEN basic service will fill the new-DEN template similar to what it was doing previously. Namely, the only thing that changes is the ActionID field. Instead, it will replace the StationID value by the group signature gs_i of the OBU. This signature is obtained by applying the group signature algorithm $\texttt{GSig}$ to the header of the new-DEN message.

④ **OBU Verifying.** Other OBUs and the RSU can verify that the signature gs_i is correct by employing the $\texttt{GVf}$ routine. To this end, the proof contained in $gs_i = (C, \pi_1)$ is checked via the verification algorithm of the underlying zero-knowledge proof algorithm as $V_1(gpk, m, gs_i)$ where m is the header of the broadcast new-DEN message, and (V_1, P_1) is one of the two Non-Interactive Zero Knowledge (NIZK) proofs systems used in the BSZ group signature scheme (as defined in Chapter 3, Figure 3.1).

⑤ **Opening.** In case of dispute, *i.e.* if there is a contention between several ITS entities about a specific message, the RSU can open the signature with the $\texttt{Open}$ algorithm. In that case, the RSU parses the group signature as $gs_i = (C, \pi_1)$. It decrypts C into $\langle i, pk, cert, s \rangle$ and checks if the identity i is in the reg table. Then it checks the proof π_1, and if all are correct, it reveals i and a proof τ that the verification process was correct. The proof τ contains now public information such as usk_i and sig_i but also the proof π_2 generated with the second set of NIZK proofs system (P_2, V_2) used in the BSZ group signature scheme (as defined in Chapter 3, Figure 3.1).

⑥ **Judging.** This proof τ is generated to prevent any colluding RSU from opening a signature onto another identity than the signer's one. The OBUs can check the result of the `Open` function with the `Judge` function and all the other available data.

4.1.4 Analysis of the $\mathcal{GS}$-based new-DEN messaging protocol

In the following paragraph, we will demonstrate that the replacement of the `StationID` value by the `GSig` value improves the privacy and accountability properties of the DEN basic service. Hence, we show that the aforementioned attacks 1 and 2 can no longer be performed, and then we extend the proof to a more general setting.

Attacker 1. We already ascertain that the `StationID` field was the one responsible for the privacy breaches during the broadcast of the DENMs. Since the new-DEN message template replaces the `StationID` by the `GSig` value, the privacy-preservation is ensured by the anonymity of the underlying group signature scheme.

Attacker 2. The `GSig` value is a signature. Since the group signature scheme chosen is non-frameable, traceable, and anonymous, the probability that the adversary successfully generates a correct group signature that does not link back to them is negligible. Therefore, the only way to escape traceability would be not to sign messages. However, messages that are not signed would simply be discarded.

More generally, let us assume the existence of a dynamic group signature scheme that is anonymous, traceable and non-frameable [34], denoted $\mathcal{GS}$ = (`GKg`, `UKg`, `Join`, `Iss`, `GSig`, `GVf`, `Open`, `Judge`). In the following, we define more formally the notion of privacy preservation and accountability that were presented in previous sections. Also, we consider that the privacy of users is protected if a stronger security notion is ensured, namely the *unlinkability* property defined as follows.

Definition 5 (Unlinkability). *Let $\mathcal{M} = (m_1, \ldots, m_k)$ be a list of k broadcast* new-DEN *messages such that exactly* 2 *of them have been sent by the same node. The* privacy-preservation property *requires that no PPT adversary $\mathcal{A}$, with the knowledge that the* *same user broadcast exactly two messages, can guess which messages with non-negligible (in λ) advantage.*

Definition 6 (Valid new-DEN message.)**.** *A new-DEN message is said to be* valid *if and only if all the fields inherited from the initial DENM satisfy the original requirements AND the $\mathtt{GSig}$ value is correct.*

Definition 7 (Accountability)**.** *Let m_0 be a broadcast new-DEN message. The* accountability property *requires that no PPT adversary $\mathcal{A}$ can generate a valid m_0 which does not link to its identity with non negligible (in λ) advantage.*

Theorem 1. *If $\mathcal{GS}$ is anonymous, traceable and non-frameable, then the proposed new-DEN messaging protocol is correct, privacy-preserving, and accountable.*

Proof. The adversary is the only malicious node in the network; by assumption, it cannot compromise other nodes. Thus, it is not able to reduce the size of the set $\mathcal{M}$ of broadcast new-DEN messages. The GM is honest; therefore, the opening secret key osk is secure and kept private.

Correctness. The structure of the DENM has not changed except for the group signature field. Yet, the chosen scheme is provent correct. Thus, the new-DEN messages issued by honest users are valid.

Privacy-preservation. By the assumption that the $\mathcal{GS}$ is anonymous, it directly unfolds that two group signatures generated by the same user are not linkable to each other nor to the signer. The proposed new-DEN messaging pattern is privacy-preserving.

Accountability. Since $\mathcal{GS}$ is traceable and non-frameable, the adversary cannot generate a group signature that opens onto another legitimate user. Therefore, it directly unfolds that the adversary cannot generate a new-DEN message that links back to another legitimate user. The proposed new-DEN messaging pattern provides accountability. □

4.1.5 Limits

The major limitation of the proposed messaging protocol lies in the strong security assumptions used to build it (including the full trust in the GM, the limited capabilities of the adversary). When we relax these assumptions to obtain a more practical setting, for instance, by considering that the GM is honest-but-curious, or that the adversary can compromise a certain number of nodes, the security requirements do not hold anymore. Indeed, let us consider that the adversary can compromise the RSU, and therefore the GM of the group. This would lead to catastrophic damages in terms of availability (*e.g.*, the RSU can apply censorship and perform denial of service attacks), privacy (*e.g.*, the RSU can arbitrarily open signatures and reveal the identity of the signers), and accountability (e.g. the RSU can collude with greedy users and change the group signature into one of another legitimate node).

In the following section, we propose a new group signature construction to address the aforementioned flaws.

4.2 A new group signature construction: $\mathcal{DOGS}$

In this section, we present our first contribution. It builds upon a traditional dynamic group signature scheme combined with threshold cryptography to decentralize the GM role while ensuring the targeted accountability and privacy requirements. More specifically, it augments the original group signature construction with a distributed key generation mechanism for dlog-based cryptosystems and a public bulletin board. Below, we describe the new system and adversary model. We present the approach and the method chosen to construct this primitive. Then, we detail each step of the $\mathcal{DOGS}$-based new-DEN messaging pattern and analyze this proposition *w.r.t.* the privacy-preservation and accountability security requirements.

4.2.1 The new setup

Figure 4.3 illustrates the new system model. It is composed of one TA, a group of RSUs, and a group of OBUs. It is not viable for the adversary to focus on compromising the OBUs because the ITS is transient, *i.e.* the topology is always evolving. Instead, the adversary targets the only stationary points: the RSUs.

Adversary Model. We consider the same type of adversary as described in Section 4.1. The adversary is an active insider node, and it is still rational as its goals are to escape traceability or breach users' anonymity. However, the adversary can now corrupt a limited number t of RSUs with $t < p/2$, where p is the **total number of RSUs deployed**. Yet, the adversary is static, meaning that it cannot corrupt any additional the opening parties during the opening phase.

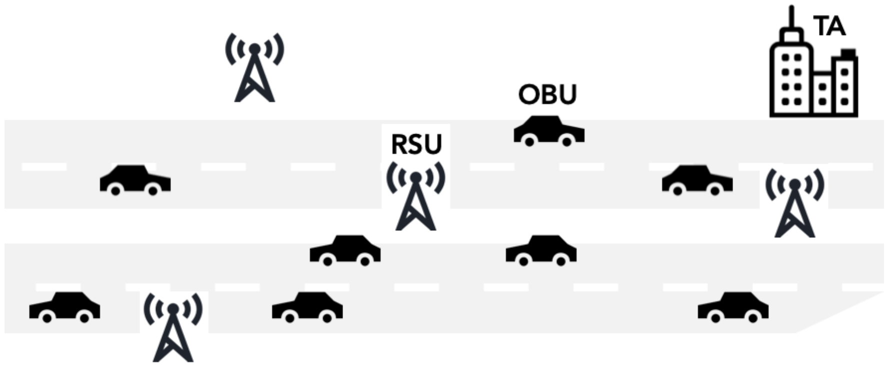


FIGURE 4.3: Illustration of the new system model with one Trusted Authority (TA), a group of Roadside units (RSUs) and a group of vehicles (OBU).

Communication Model. Instead of relying on the RSU for synchrony and logging functionality, we assume the availability of an **append-only web bulletin board** as it is defined by Heather and Lundin in [175]. A web bulletin board system typically involves the online board, reader and writer entities. Its aim is to allow various parties, called the writers, to publish some information so that other nodes, the readers, can access them. The board is usually said *append-only* in the sense that once something is published, it can never be removed or altered. In our case, a node can be both a writer and a reader. In our proposition, we assume our append-only web bulletin board to satisfy the following properties. The web bulletin board has:

i) *unalterable consistent history*, which implies that whenever a reader retrieves the contents of the board at two different times, it is able to check that the content of the board at time t_0 is the same at time t_1 expect for possibly newly appended messages.

ii) *certified publishing*, meaning that whenever a reader retrieves the contents of the board, he can either detect corruption of the board, or have proof, for each message on the board of the origin of the message and the origin's intention of publishing its message at this point in the board's sequence of messages.

iii) *timely publication*, which ensures that newest messages are appended with timestamps that are greater than the timestamp of the last message inserted in the board.

The messages that transit within the network are also published onto this bulletin board under the form of *transactions* and have the same structure: [label, payload, G-sig]. The label field informs the users in the public network to which stage of the protocol the transaction belongs to. The payload contains utility information and its structure depends on the label. The G-sig contains the group signature value issued by the algorithm `GSig` on the original identity of the publishing node. This field is used to filter, authenticate and trace the transactions (filtering: an incorrect group signature should lead to the rejection of the transaction; and tracing: a correct group signature always traces back to an identified user).

4.2.2 Objectives and approach

In the following, our objectives are essentially the same as previously, namely we still focus on providing privacy and accountability in vehicular communications. Yet, in addition, we want to alleviate the power of the RSU. To this end, we propose a new construction of a group signature scheme and demonstrate that the new primitive is also anonymous, traceable and non-frameable.

Approach.

We aim at censorship resistance through distribution. Similarly to [34], we suggest splitting the role of the GM, played by the RSU in our group signature scheme. We, therefore, consider one Issuer, role ensured by TA, and a set of sub-Openers, played by the RSUs, to replace the Opener entity.

Method.

We base our construction on the BSZ group signature scheme [34]. To distribute the role of the Opener onto a set of RSUs, we use GJKR-DKG protocol [39] to generate the opening public/private key pair (as introduced in Chapter 2).

4.2.3 Description of $\mathcal{DOGS}$

There are three phases in $\mathcal{DOGS}$:

> **Phase 1** *"Distributed Generation of the Opening Keys"*,
>
> **Phase 2** *"Inter Communications and App-related event logging"* and
>
> **Phase 3** *"Auditing and Identification"*.

We transpose them into five distinct modules (Fig. 4.4) namely **Bootstrapping**, **Opening Keys (OK) Generation** and **Registration** (phase 1), **Application** (phase 2) and **Audit** (phase 3).

Algorithms and their usage. Users evolving in the system become group members by engaging in the $Join/Iss$ joint protocols with the Issuer. They can become sub-opener authorities and participate to the opening process if they own a share of an Opening secret key. The scheme is specified as a tuple $\mathcal{DOGS}$ = ($\texttt{GKg}$, $\texttt{dOKg}$, $\texttt{UKg}$, $\texttt{Join}$, $\texttt{Iss}$, $\texttt{GSig}$, $\texttt{GVf}$, $\texttt{Request}$, $\texttt{Collaborate}$, $\texttt{dOpen}$, $\texttt{Judge}$) of PPT algorithms

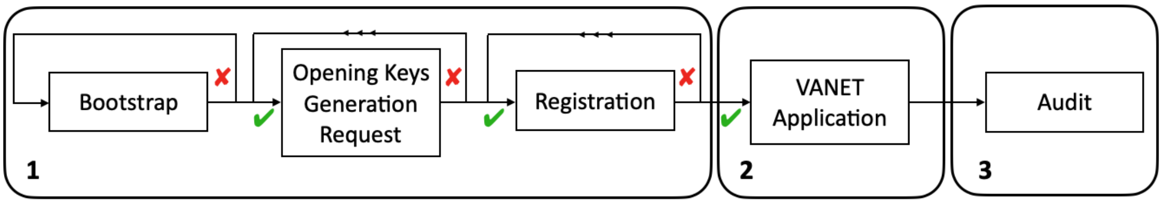


Figure 4.4: $\mathcal{DOGS}$ Workflow diagram

(Algorithms 1, 2, 3, 4, 5), whose intended usage and functionalities are presented in this section.

Phase	Module	Algorithm		Type	Comments
①	Bootstrap	GKg		Bootstrap	publication of the group public key *gpk*, and the Issuer public key *ipk*
	Opening Keys Generation	dOKg	sharing	Opening Keys (OK) Generation Request	U_i generates and distributes commitments (C_{ij}), encrypted shares $\overline{s_{ij}}$ and value h^{s_i}, for $k = 0..t$, $j = 1..n$
			share_verification	Claim	a user U issues a claim $=< status, data, key, proof >$
			claim_verification	Claim Verification	all U check the validity of the claim; 2 issues: accept/reject
			key_derivation	Key Publication	the Issuer determines and publishes the resulting Opening public key *opk*
	Registration	UKg		none	U_i generates local identity $\mathbf{upk}[i], \mathbf{usk}[i]$
		Join		Joining Request	by U_i, sends personal data to Issuer
		Iss		Issuing Reply	2 issues: accept/reject. If accept, the Issuer records the link between $\mathbf{upk}[i], \mathbf{usk}[i]$ and (i, pk_i, sig_i)
②	Application	GSig		Signature	user U produces a group signature σ under *opk*
		GVf		Verification	user U verifies a group signature σ under *opk*. 2 issues: accept/reject.
③	Audit	Request		Open Request	user U requests the opening of a group signature σ under *opk*
		Collaborate		Secret Key Publication	one of the authorized users publishes the re-constructed Opening secret key *osk*
		dOpen		Open	the signature σ is opened and the result is published by the requester
		Judge		Judge	a user - not the requester - asserts and publishes the validity of the requester's opening

Table 4.1: Description of the transaction (Tx) types and corresponding published data.

Algorithm 1: Bootstrap module.

```
Function $GKg(1^k)$
begin $GKg$
    $R_1 \xleftarrow{\$} \{0,1\}^{p_1(k)}$; $R_2 \xleftarrow{\$} \{0,1\}^{p_2(k)}$ ; $(pk_s, sk_s) \xleftarrow{\$} K_s(1^k)$
    $gpk \leftarrow (1^k, R_1, R_2)$; $ik \leftarrow sk_s$
    Posts $BC.post(gpk, ipk \equiv g^{ik})$ ; Triggers tx: Bootstrap
```

Algorithm 2: Opening Keys Generation (dOKg) module.

```
Function $sharing(t)$
begin $sharing$ by $U_i$
    $\mathcal{R} \leftarrow \{U_j$ for $j = 1..n$, $j \neq i\}$ i.e. all the $U_j$ users in RoI
    User $U_i$ performs the first step of [? ]'s DKG protocol (Fig.2)
    Posts $BC.post(C_{ik}, (\overline{s_{ij}})_{k_{ij}}, h^{s_i})$ for $k = 0..t$ and $j = 1..n$; triggers tx: Opener
      Generation Request

Function $share\_verification((\overline{s_{ij}})_{k_{ij}}, pk_i)$
begin $share\_verification$ by $U_j$
    $BC.read(U_i$'s transaction$) \equiv < C_{ik}, (\overline{s_{ij}})_{k_{ij}} >$ for $k = 0..t$ and $j = 1..n$
    Checks that:
      1. $(\overline{s_{ij}})_{posted} == (\overline{s_{ij}})_{received}$
      2. $g^{s_{ij}} h^{s'_{ij}} = \Pi_{k=0}^{t} (C_{ij})^{j^k} \mod p$
    if $check\ fails$ then
        Posts $BC.post($claim$, < C_{ik}, (\overline{s_{ij}})_{k_{ij}} >, k_{ij}, \pi(k_{ij}))$; triggers tx: Claim
    else
        Posts $BC.post($no_claim$, < C_{ik}, (\overline{s_{ij}})_{k_{ij}} >, \epsilon, \epsilon)$; triggers tx: Claim

Function $claim\_verification(record \equiv < status,\ data,\ key,\ proof >)$
begin $claim\_verification$ by $U_k$
    User $U_k$ parses record as $< status, data, key, proof >$.
    if $status == claim$ then
        $claim_{ij} \leftarrow ($claim$, < C_{ik}, (\overline{s_{ij}})_{k_{ij}} >, k_{ij}, \pi(k_{ij}))$
        $BC.read(U_i$'s transaction$) \equiv < C_{ik}, (\overline{s_{ij}})_{k_{ij}} >$ for $k = 0..t$ and $j = 1..n$
        Checks $claim_{ij}$ by:
          1. $(\overline{s_{ij}})_{posted} == (\overline{s_{ij}})_{received}$
          2. $DLEQ - verify(g, pk_j, pk_i, k_{ij}, \pi(k_{ij})) == True$
        if $1.\ or\ 2.\ fails$ then
            $\mathcal{R} \leftarrow \mathcal{R} \setminus U_j$
            Posts $BC.post(claim_{ij}, invalid)$; triggers tx: Claim Verification
        else
            $(s_{ij})_{received} = Dec_{k_{ij}}(s_{ij})$
            if $(g^{s_{ij}} \neq F_i(j))$ then
                Posts $BC.post(claim_{ij}, valid)$; triggers tx: Claim Verification
            else
                $\mathcal{R} \leftarrow \mathcal{R} \setminus U_j$
                Posts $BC.post(claim_{ij}, invalid)$; triggers tx: Claim Verification

Function $key\_derivation(\{h^{s_i}\}_{U_i \in \mathcal{R}})$
begin $key\_derivation$ by Iss
    Reads BC and, for all $U_i$ and $claim_{ik}$:
    if $k \in N\ where\ |N| \geqslant \frac{n}{2}$ /* That is, there is a majority of participants that
      issued a valid claim against $U_i$.*/ then
        $\mathcal{R} \leftarrow \mathcal{R} \setminus U_i$; posts BC.post(< $cert_i$, revoked>); triggers tx: Revocation
    For all $U_j \in \mathcal{R}$, the Issuer retrieves $h^{s_j}$
    $opk \equiv \Pi(h^{s_j})_{U_j \in \mathcal{R}}$; posts $BC.post(opk, \mathcal{R})$; triggers tx: Key Publication
```

Algorithm 3: Registration module.

```
Function $Join(St_{join}, M_{in})$
begin $Join$ by user $U_i$
    if $M_{in} = \epsilon$ then
        $St_{join} \equiv (gpk, i, \mathbf{upk}[i], \mathbf{usk}[i])$; $(pk_, sk_i) \xleftarrow{\$} \mathrm{K}_s(1^k)$; $sig_i \leftarrow \mathrm{Sig}(\mathbf{usk}[i], pk_i)$
        $St'_{join} \leftarrow (i, pk_i, sk_i)$; $M_{out} \leftarrow (pk_i, sig_i)$; Sends $(\overline{(St'_{join})}_{ipk}, M_{out}$, cont) to Iss
        Posts $BC.post(H(St'_{join}), M_{out}$, cont); triggers tx: Joining Request
    else
        $St_{join} \equiv (i, pk_i, sk_i)$; $M_{in} \equiv cert_i$
        $St'_{join} \leftarrow (i, pk_i, sk_i, cert_i)$; Sends < $(\overline{St'_{join})}_{ipk}, \epsilon$, accept> to Iss
        Posts $BC.post(H(St'_{join}), \epsilon$, accept>); triggers tx: Joining Request

Function $Iss(St_{issue}, M_{in})$
begin $Iss$ by Iss
    $M_{out} \leftarrow \epsilon$; $dec' \leftarrow$ reject
    if $dec = cont$ then
        $St_{issue} \equiv (gpk, ik, i, \mathbf{upk}[i])$; $M_{in} \equiv (pk_i, sig_i)$; $ik \equiv sk_s$
        BC.read(< $H(St'_{join}), M_{out}$, cont >)
        if $H(St'_{join})_{posted} \neq (St'_{join})_{received}$ then
            Registration process is aborted
        if $Vf(\boldsymbol{upk}[i], pk_i, sig_i) = 1$ then
            $cert_i \leftarrow \mathrm{Sig}(sk_s, < i, pk_i >)$; $St_{issue} \leftarrow (pk_i, sig_i)$
            $M_{out} \leftarrow cert_i$; $dec' \leftarrow$ accept
        Posts $BC.post(St'_{issue}, M_{out}, dec')$; triggers tx: Issuing Reply
```

Algorithm 4: Application module with the $GSig$ and and GVf algorithms of the new dynamic group signature scheme with distributed opening and automated recording in public ledger.

```
Function $GSig(gpk, ipk, \boldsymbol{gsk}[i], m)$ by $U_i$
begin $GSig$
    $gpk \equiv (1^k, R_1, R_2)$; $ipk \equiv pk_s$; $\mathbf{gsk}[i] \equiv (i, pk_i, sk_i, cert_i)$
    $s \leftarrow \mathrm{Sig}(sk_i, m)$; $r \xleftarrow{\$} \{0,1\}^k$; $C \leftarrow \mathrm{Enc}(opk, < i, pk_i, cert_i, s >; r)$
    $\pi_1 \xleftarrow{\$} P_1(1^k, (opk, ipk, m, C), (i, pk_i, cert_i, s, r), R_1)$; $\sigma \leftarrow (C, \pi_1)$
    Posts $BC.post(m, opk, \sigma)$; triggers tx: Signature

Function $GVf(gpk, ipk, (m, opk, \sigma))$
begin $GVf$ by $U_j$
    $gpk \equiv (1^k, R_1, R_2)$; $ipk \equiv pk_s$; $\sigma$ as $(C, \pi_1)$
    Posts $BC.post(m, opk, \sigma, V_1(1^k, (opk, ipk, m, C), \pi_1, R_1))$; triggers tx: Verification
```

Algorithm 5: *Audit* module.

```
Function $Request(<m, opk, \sigma>, cert_i)$
begin $Request$
    Posts $BC.post(audit, <m, opk, \sigma>, cert_i)$; triggers tx: Open Request
Function $Collaborate(<open\ request,\ opk>)$
begin $Collaborate$
    Listens to transactions
    if $transaction{=}{=}open\ request$ then
        For all $P_j$ in $\mathcal{R}$: Sends $s_j$ to $P_{j+1}$
        User $P_{t+1}$ computes $osk = \Sigma_{k=1}^{t+1} s_k$; posts $BC.post(<m, opk, \sigma>, \mathcal{R}, osk)$
        triggers tx: Secret Key Publication
Function $dOpen(gpk, ipk, osk, \mathbf{reg}[], m, \sigma)$
begin $dOpen$
    $gpk \equiv (1^k, R_1, R_2)$; $ipk \equiv pk_s$; $\sigma \equiv (C, \pi_1)$; $M \leftarrow \mathrm{Dec}(osk, C)$
    $M \equiv <i, pk, cert, s>$
    if $\mathbf{reg}[i] \neq \epsilon$ then
        $\mathbf{reg}[i] \equiv (pk_i, sig_i)$
    else
        $pk_i \leftarrow \epsilon$; $sig_i \leftarrow \epsilon$
    $\pi_2 \leftarrow P_2(1^k, (opk, C, i, pk, cert, s), (osk), R_2)$
    if $V_1(1^k, (opk, ipk, m, C), \pi_1, R_1) == 0$ then
        Posts $BC.post(<m, opk, \sigma>, osk, <0, \epsilon>)$; triggers tx: Open Request; (break)
    if $pk \neq pk_i$ then
        Posts $BC.post(<m, opk, \sigma>, osk, <0, \epsilon>)$; triggers tx: Open Request; (break)
    $\tau \leftarrow (pk_i, sig_i, i, pk, cert, s, \pi_2)$
    Posts $BC.post(<m, opk, \sigma>, osk, <i, \tau>)$; triggers tx: Open Request
Function $Judge(gpk, ipk, \mathbf{upk}[i], m, \sigma, \tau)$
begin $Judge$ by $U_i$
    $gpk \equiv (1^k, R_1, R_2)$; $ipk \equiv pk_s$; $\sigma \equiv (C, \pi_1)$
    if $(i, \tau) == (O, \epsilon)$ then
        Posts $BC.post(<gpk, ipk, \mathbf{upk}[i], m, \sigma, \tau>, 0 \equiv V_1(1^k, (opk, ipk, m, C), \pi_1, R_1))$
        triggers tx: Judge; (break)
    $\tau \equiv (\widetilde{pk}, \widetilde{sig}, i', pk, cert, s, \pi_2)$
    if $V_2(1^k, (C, i', pk, cert, s), \pi_2, R_2) == 0$ then
        Posts $BC.post(<gpk, ipk, \mathbf{upk}[i], m, \sigma, \tau>, 0)$; triggers tx: Judge; (break)
    if $(i == i')$ && $(Vf(\mathbf{upk}[i], \widetilde{pk}, \widetilde{sig}) == 1)$ && $(\widetilde{pk} == pk)$ then
        Posts $BC.post(<gpk, ipk, \mathbf{upk}[i], m, \sigma, \tau>, 1)$; triggers tx: Judge
    else
        Posts $BC.post(<gpk, ipk, \mathbf{upk}[i], m, \sigma, \tau>, 0)$; triggers tx: Judge
```

The transaction types and corresponding published data, defined in the $\mathcal{DOGS}$ protocol, are summarized in Table 4.1. The table gathers the definitions of eight transaction types for Phase 1 (e.g. the *bootstrap transaction* corresponds to the publication of bootstrapping information); two transaction types for Phase 2, which illustrate the signing action of a user and verifying process of other members of the group; and finally, four transaction types for Phase 3.

Since the information published does not hold private or sensitive information, the public bulletin board can be accessed by anyone. However, we restrict the writing permissions to authenticated users only.

Phase 1: Distributed Generation of the Opening Keys

In the following paragraphs, we will explain each step of Phase 1.

Bootstrap. The scheme starts with the bootstrapping of the system. It consists in the Issuer executing the `GKg` algorithm and results in the publication of both the group public key *gpk* and the Issuer's public key *ipk* (Bootstrap transaction). It then designates the set of sub-opener entities O_i with $1 \leqslant i \leqslant p$. This action triggers the OK Generation Request event.

Opening Keys (OK) Generation. Since the Opener is no longer a single entity, from now on, we will use the wording “Opening" to designate the functionality it was in charge of.

Each O_i executes the `sharing` function inherited from [39]. Subsequently, O_i generates and publishes the required information (commitments (C_{ik}) with $k = 0..t$, shares (s_{ij}) with $j = 1..p$ and value h^{s_i} according to [39]) for the computation of an Opening public key.

The broadcasting of this new transaction triggers an update of the public bulletin board of the users. They individually execute the `share_verification` function to check the correctness of the share that has been sent to them by O_i. Let us stress out that each share s_{ij} is encrypted with a symmetric encryption scheme of key k_{ij}:

$$k_{ij} = g^{sk_i sk_j} = pk_i^{sk_j} = pk_j^{sk_i}$$

Note: since the sub-openers are designated during the Bootstrapping by the Issuer, their public identities, the pk_i for $i = 1..p$, are broadcast within the BOOTSTRAP transaction. Therefore, for any sub-opener i with their public/private keypair pk_i/sk_i, they can compute $pk_j^{sk_i}$ for all j between 1 and p.

The execution of this function triggers a CLAIM transaction and publishes the result as $< status, data, key, proof >$. Depending on $status$, there are two different results: if $status$ contains the value "*no_claim*" or U_j does not reveal k_{ij} or its proof $\pi(k_{ij})$, then the share is accepted and U_j in turn has to broadcast its own shares (see **Case 1**). Else if, $status$ value is equal to "*claim*", then the protocol holds as others check the claim (see **Case 2**).

Case 1. Therefore, O_j in turn executes the `sharing` function, distributes the resulting shares and publishes related data. Again, `share_verification` function is used to check the validity of the shares, but this time either the peers $\{O_k\}_{k \neq j \in p}$ accepts O_j's share, and it ends there, or it rejects it.

Case 2. In this case, a share has been rejected, and a claim has been broadcast. Hence, users execute `claim_verification` function to check its validity. Doing so, they trigger a CLAIM VERIFICATION transaction, which results in either the claim being denied or accepted.

Once all the shares and claims have been verified, the Issuer can execute `key_derivation` function. It browses the public records and determines which shares are used to compute the final Opening public key. It finally publishes this key opk and $\mathcal{R}$ the list of authenticated local identities which participated in the creation of this Opening key.

Once the distributed opening authority is set up, users can join the group and benefit from the new signature scheme.

Registration. User U_i enters the RoI. First, U_i executes the `UKg` algorithm to obtain a personal public/private key pair, referring to its local identity (usk_i, upk_i). Then, U_i starts the `Join`/`Iss` interactive protocols. It executes the `Join` function and triggers the Join Request transaction. The generation and publication of related information (Tab. 4.1) is compliant with Bellare *et al.*'s $\mathcal{GS}$ `Join` algorithm, but enhanced with logging functionalities for traceability purposes. The Issuer receives encrypted data from U_i, including identifying information such as the public key pk_i used to verify U_i's signed messages. It compares it to the public record for data integrity checks; then executes the `Iss` function, triggering the publication of resulting data and the Issuing Reply transaction. The **Registration** step has two issues: the first ends up in U_i's request being rejected due to faulty data; the second grants U_i with a certificate $cert_i$ and allows it to perform subsequent actions such as signing a message or verifying a signature, but also issuing an opening request. If so, the Issuer records, in its local database, the relationship between the user's local identity given by `UKg`, and the authenticated identity (i, pk_i, sig_i) used in the RoI.

In the following sub-section, we explain how anyone in the RoI can use opk in `GSig` to encrypt its signature, protecting its anonymity in communications while still not being able to escape action traceability.

Phase 2: Inter Communications and Application-related event logging

Application. Our $\mathcal{DOGS}$ scheme has been initially thought to be applied in the context of local logging of events, especially for ITSs. Let us consider U_i has road-safety information to share with vehicles in the neighbourhood, for instance, an alert about a car accident. However, U_i does not want to reveal its identity nor its position at this particular time (to prevent location-based identity inference [176]). If U_i applies `GSig` function on the alert message, it can produce a signature that refers to the group but protects its identity. It additionally triggers the Signature transaction. The use of [34] 's `GSig` along with the DKG-computed Opening public key ensures that no single sub-opener can open this signature, hence identify U_i. However, users in the neighbourhood can still individually execute `GVf` [34] to assert the correctness of the received signature and trust U_i's alert. The result gets published along with VERIFICATION transaction.

The application of $\mathcal{DOGS}$ in the context of ITSs is fully detailed in sub-section 4.2.5.

Phase 3: Auditing and de-anonymizing

Audit. This module regroups the required functionalities implemented in $\mathcal{DOGS}$ for providing action traceability.

With `Request`, the requester U_i triggers the OPEN REQUEST transaction, hence summoning the sub-openers in the RoI to collaborate to reconstruct an Opening secret key. U_i therefore communicates the signature σ it wants to open, the message m it signs and the corresponding Opening public key opk. For traceability guarantees, this request gets published on the bulletin board.

Then, the `Collaborate` function is executed. All U_j in the set $\mathcal{R}$ related to opk will collaborate to reconstruct the Opening secret key osk (by consecutively summing their shares to previous partial results as they do not exist in any single location). The last peer to add its secret s_{t+1} also publishes the result while triggering the SECRET KEY PUBLICATION transaction. It includes the initial data $< m, opk, \sigma >$, the set $\mathcal{R}$, and the result of their work osk.

Note: For the sake of concision, we considered in DOGS that the secret key was reconstructed. However, since we are dealing with dlog-based protocols, we can also decrypt a message without reconstructing the key. It would be done as follows. All the peers U_i would publish their partial decryption of c with their secret s_i such that by combining all these partially decrypted ciphertexts, anyone can recover the plaintext (according to Appendix A.2.3). Each publication comes with the emission of a PARTIALDECRYPTION event. Once enough contributions are sent, anyone can retrieve and combine them to obtain the message. According to Appendix A.2.3, we observe that osk does not need to be reconstructed in order to access the identity of a signer.

Finally, by retrieving osk, the requester U_i can identify the origin of σ and publishes the result via the `dOpen` function (OPEN transaction). Other peers in the RoI can consequently check this identification by executing the `Judge` function (JUDGE transaction).

Let us remark that in practice, *osk* does not need to be reconstructed, as this would hinder forward secrecy. Traditionally, the encryption scheme is a hybrid construction that leverages dlog-based cryptosystems to prevent the release of the signing key while enabling the decryption of one message [177].

4.2.4 Security Analysis of $\mathcal{DOGS}$

In this section, we show that $\mathcal{DOGS}$, resulting from the combination of BSZ group signature scheme [34], GJKR-DKG protocol [39] and a public bulletin board [175], is an anonymous, traceable and non-frameable group signature scheme. More specifically, we establish that the use of DKG functionalities is compatible with the security environment related to group signature schemes as presented in [34], hence that $\mathcal{DOGS}$ presents all the security properties we aimed for. Most of the security proofs are directly inherited from the BSZ construction and the GJKR-DKG protocol. Only the anonymity property requires careful treatment. Below, we recall the main security features and arguments that constitute the proofs, and then refer the reader to BSZ [34] for a complete description of the experiments for each security feature.

Arguments of security.

Correctness. Properties i) and iii) of Definition 12 are directly satisfied using BSZ. Assuming the Opening key is correctly reconstructed, which is the case with overwhelming probability (in λ) thanks to DKG, then ii) is also satisfied. □

Anonymity. In the original experiment [34], $\mathcal{A}$ does *not* have access to the opening oracle. In $\mathcal{DOGS}$, we weaken this assumption to "$\mathcal{A}$ has access to at most t shares of the Opening secret key (including her own, if she is a registered user)". Then, *w.r.t.* Definition 2, $\mathcal{A}$ has negligible advantage in λ in recovering *osk* [39] and our anonymity feature boils down to the original one, which is fulfilled by using BSZ. □

Traceability. Similarly, in [34], $\mathcal{A}$ is granted access to the Opening secret key *osk*. Therefore, *w.r.t.* to Definition 18, $\mathcal{A}$ learns nothing more by corrupting users and $\mathcal{DOGS}$ traceability is inherited from BSZ. □

Non-frameability. Here again, since $\mathcal{A}$ has already access to osk in the original experiment, she obtains no additional advantage by exploiting the shares of the Opening secret key, and $\mathcal{DOGS}$ satisfies non-frameability as BSZ, *w.r.t.* Definition 4. □

Discussion. Most of the security features are directly inherited from the BSZ and GJKR-DKG constructions. However, the anonymity experiment as described in [34] needs to be slightly modified. Indeed, an adversary $\mathcal{A}$ against the original property could create and corrupt sufficiently many (more than t) users to obtain their respective shares of the Opening secret key osk, and hence reconstruct it. By using osk, $\mathcal{A}$ could trivially break the original anonymity property. Therefore, an upper bound on the number of corruptible users has to be integrated to compensate for the extra knowledge $\mathcal{A}$ gets by corrupting the sub-openers.

4.2.5 Description of the $\mathcal{DOGS}$-based new-DEN messaging pattern

① **Setup.** The setup of the new-DEN messaging protocol corresponds to Phase 1 of $\mathcal{DOGS}$, namely, it consists in identifying the set of sub-openers and generating, in a distributed manner, the opening public key opk. The TA bootstraps the system (**Bootstrap**). It generates a pair of keys (ipk, isk) and selects the RSUs that will act as sub-openers. This event triggers the generation of the opening key (**Opening Key Generation**). At the end of the setup, the Issuer, sub-openers and corresponding public information are known.

② **OBU Registration.** Similarly to the OBU registration in the previous section, each OBU interacts with TA to get a certified temporary public key.

③ **OBU Signing.** Once the registration phase is completed, the OBUs can use the new group signature scheme similar to what was presented in the previous section.

④ **OBU Verifying.** The new construction does not modify the `GVf` algorithm. Therefore, other OBUs can verify that the previous signature is correct by employing the `GVf` function defined in $\mathcal{DOGS}$.

⑤ **Distributed opening.** In case of dispute, an opening request is broadcast and logged inside the public bulletin board. The RSUs must collaborate if they want to perform the Opening functionality (**Audit**).

⑥ **Judging.** The result of the opening is publicly verifiable, thanks to its publication inside the bulletin board. The OBUs can check the result of the `dOpen` function with the `Judge` function.

The strategy adopted upon detection of a malicious node depends on the application. For instance, the RSU can issue a blockchain transaction to revoke the group member by publishing its identity and the proof output by the `dOpen` () algorithm.

4.2.6 Analysis of the $\mathcal{DOGS}$-based new-DEN messaging protocol

In the following paragraph, we demonstrate that replacing the initial group signature field `GSig` by a $\mathcal{DOGS}$ signature provides the same privacy and accountability features in a stronger security model where the adversary can corrupt up to $t < p/2$ RSUs. This threshold is chosen *w.r.t.* the security requirements related to the blockchain network dynamics.

Theorem 2. *If $\mathcal{DOGS}$ is anonymous, traceable and non-frameable, then the proposed new-DEN messaging pattern provides privacy-preservation and accountability* w.r.t. *the adversary model described in Sub-section 4.2.1.*

Argument of security.

The proof follows the same reasoning as Proof 4.1.4. Instead of relying of the security features of $\mathcal{GS}$, the new-DEN messaging protocol inherits from $\mathcal{DOGS}$'s proven security features (Sub-section 4.2.4).

Discussion. The distribution of the Opener over a set of RSUs handicaps the adversary. It becomes more difficult to achieve the same attacks on privacy and accountability as previously since it involves compromising more $(t-1)$ RSU nodes. Controlling one RSU no longer gives the adversary the power to arbitrarily de-anonymize the DENM traces or to escape the traceability mechanism (Open functionality). This was made possible thanks to the developed construction of a group signature scheme with a distributed opening functionality. Being the combination of a BSZ group signature scheme and GJKR-DKG protocol, we were able to construct $\mathcal{DOGS}$ and to show that, by distributing the Opener among a group of users, we preserve the traceability feature of group signatures while enhancing their anonymity feature. Indeed, our scheme is proven stronger than BSZ's in terms of anonymity due to the use of DKG. Applied in the context of ITSs, this enables the design of a new service of road hazard alerting called *new-DEN messaging protocol*. It provides the stated privacy-preservation and accountability requirements even when the system is attacked by an adversary that can corrupt at most $p/2$ RSU entities.

However, there are some limitations to the system model considered. Firstly, the use of the bulletin board is controversial. Indeed, we overlooked the publication of the tracing information and are relying on the fact that the publications are authenticated and correct. Yet, how do we ascertain the validity of the published data? And how do we prevent privacy leaks related to the publication of this information? We observe that **the blockchain technology is a better candidate** for the implementation of our traceability mechanism. Secondly, the place of the TA and its role as the Issuer is questionable. It would be interesting to complete the current scheme with the addition of a distributed Issuing functionality. These ideas will be explored in Chapter 5 where we developed a *blockchain-based threshold encryption scheme* that supports an *anonymous-yet-decryption service* named $\mathcal{TOAD}$.

5

$\mathcal{TOAD}$ for censorship-resistance in ITSs

"Censorship reflects a society's lack of confidence in itself."

— Potter Stewart

This chapter mainly focuses on setting up the infrastructure nodes of an ITS so they can provide a privacy-preserving accountable road hazard warning reporting system through the use of the developed new-DEN messaging protocol. Therefore, in this chapter, we explain why the presence of the TA in Chapter 4 is a threat to users' privacy and accountability. More specifically, we demonstrate the presence of availability risks and analyze how this impacts the aforementioned security requirements. To alleviate the power of TA, we slightly modify the ITS setting and redefine our objectives (Section 5.1.2). Then, we present a new construction of the new-DEN messaging protocol that relies on the concept of Issuer-as-a-service. The novel protocol leverages

DKG protocols and threshold encryption to distribute the role of the Issuer, similarly to what was done in $\mathcal{DOGS}$. We analyze the result *w.r.t.* the state objectives and draw its limitation. Finally, we propose a mitigation technique under the form of a new cryptographic primitive called $\mathcal{TOAD}$.

5.1 The problem of censorship

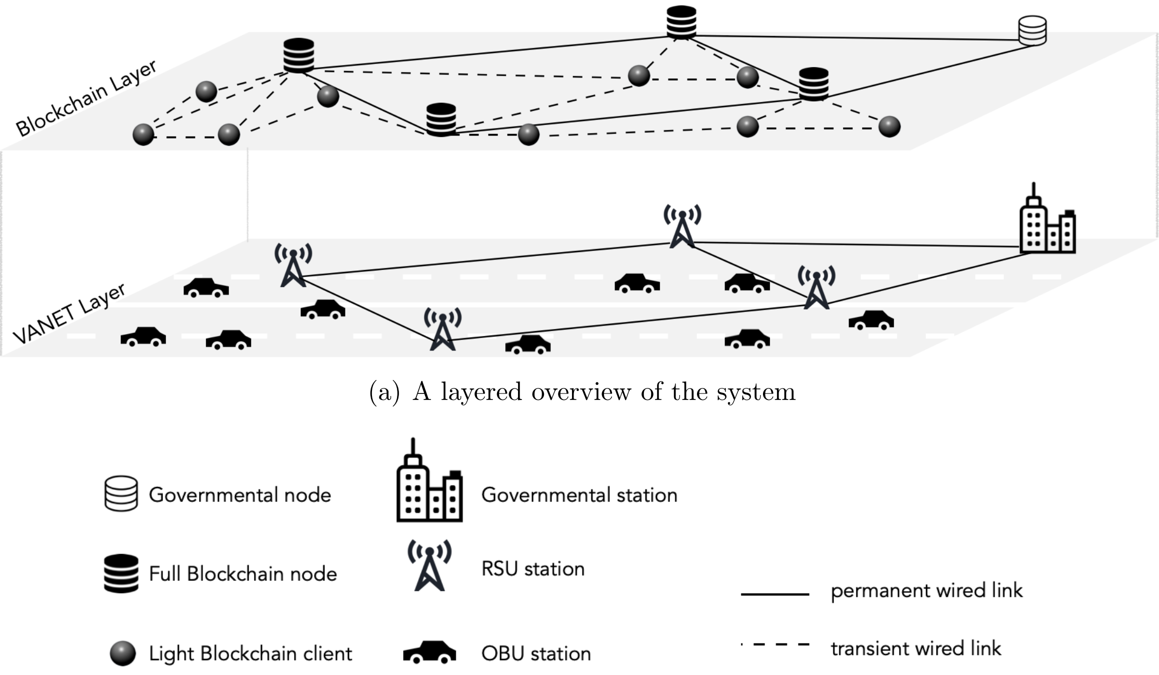
The presence of the TA in Chapter 4 is a considerable liability if we consider a more realistic environment, especially when we relax the security assumptions and consider that this third authority is no longer fully trusted. In that case, threats towards users' privacy and accountability rise due to the risks that weigh upon the communications services' availability. In the following sub-sections, we design a new system model in which the power of TA is significantly weakened. We refine our objectives and present an enhanced new-DEN messaging protocol that fits into this new setup.

5.1.1 The setup

System model. Figure 5.1 presents a refined overview of the system model. We consider the existence of two distinct networks: the ITS and the blockchain network. As explained in Chapter 1, both networks present some similarities, for instance, their distributed architecture. Moreover, as mentioned in Chapter 4, we decided to replace the append-only web bulletin board by a blockchain as the latter technology presents some specificities that makes it more suitable to our current research (notably the transparency of the logged transactions and the public auditability of its records).

The mapping between the two layers is explained as follows. We assume that RSUs can handle the role of full nodes in the Blockchain layer. Similarly, despite their limited computational and storage resources, vehicles can still embed light nodes and communicate with other light and full nodes. The TA ensures two roles in the system. Firstly, it acts as the Issuer for the vehicles. Secondly, it selects the RSUs during the creation of the Opening authority. Having such a double-hatted authority can lead to various problems of censorship. At the vehicles' level, TA can deny group

Figure 5.1: A new overview of the system model



(a) A layered overview of the system

(b) Legend

members' registration or tamper with the registering vehicles' information. At the RSUs' level, some RSUs may never be selected as potential sub-opener nodes. In this chapter, we split TA into two distinct entities: TA and an Issuing Authority. The TA part acts as a third entity which does not require to be trusted and is only used to initiate the bootstrapping. The Issuing Authority ensures the registration of the vehicles and issuance of certificates. We also consider the presence of a distributed Opening authority, set up via $\mathcal{DOGS}$, and explain how we replaced the bulletin board by a blockchain-based implementation of a recording system.

Communication model. The two-layer nature of the system implies the definition of two types of communications: the first is used between vehicles and infrastructure nodes and is already known as Vehicle-to-Anything (V2X) (Chapter 2). We define the second type as Anything-to-Blockchain (X2B). It is bidirectional and allows vehicles and RSUs to communicate with the blockchain network and retrieve information. We therefore distinguish the P2P connection through which they broadcast *messages* M_i

to addresses $addr_i$ enabled by the ITS; from the blockchain client through which they publish *transactions* t_i via their account numbers acc_i. By using a blockchain implementation, we achieve partial synchrony between the ITS nodes. Indeed, they can all monitor and broadcast messages. Thanks to the blockchains, parties all agree on a common view and order of these messages. Therefore, any message broadcast by a participant during some protocol phase is received by all other parties before the next phase starts.

Adversary model. We still consider the active, rational adversary that evolves inside the ITS. It focuses on corrupting RSUs and can compromise up to a threshold number t of RSUs, with $t < p/2$ where p is the total number of RSUs deployed. The TA and the Issuing authority are no longer trusted. The goal of the adversary is to breach vehicles' privacy or to escape traceability. In this chapter, we study the impact of censorship. We consider two cases. The first relates to TA colluding with malicious RSUs. And the second concerns the Issuing authority denying services to the users requesting group membership. We illustrate the two cases with two attacks:

> **Attack 1.** TA selects $p = 1$ RSU (instead of $p > 2$) resuming the Opening functionality to a single node as in Chapter 4 Section 4.1 and makes it easier for the adversary to corrupt the RSU.
>
> **Attack 2.** The Issuing authority receives a join request from a user and discards it. The user in question cannot access the anonymous-yet-traceable functionality of the proposed new-DEN messaging protocol.

5.1.2 Objectives

We aim at designing a new-DEN messaging protocol that is privacy-preserving, accountable and censorship-resistant. More specifically, we want to inherit from the previous construction and protect the privacy-preservation and accountability properties built up in Chapter 4. Yet, we want to augment the initial structure of the new-DEN messaging protocol with censorship-resistance *w.r.t.* the described adversary

(Sub-section 5.1.1). To this end, we concentrate on providing a non-biased mechanism for the selection of the sub-openers and in distributing the role of the Issuing authority likewise the work proposed in $\mathcal{DOGS}$.

About Censorship-resistance. As suggested by Khattak *et al.*'s systematization of knowledge [178], web content has constantly been attacked by governmental or corporate organizations which tried to censor it for political or commercial reasons. Censorship-resistance is defined as the property guaranteeing that considering an adversarial removal of an arbitrarily large and constant fraction of the nodes in the network, the remaining nodes can recover (most of) the original data [179]. The notion of censorship-resistance is traditionally applied to *data censorship.* We formalize the idea of *service censorship-resistance* as follows.

Definition 8 (Service Censorship-resistance)**.** *Let $P_1, \ldots, P_N$ be $p > 3$ nodes constituting the network that proposes a service S, and let $\mathcal{A}$ be an adversary that can compromise t nodes in the system (without loss of generality, we consider that the corrupted parties are $P_{p-t+1}, \ldots, P_p$). The* censorship-resistance property *guarantees that there exists $r > t$ such that $P_{i_1}, \ldots, P_{i_r}$ can still provide S with $i_j < p - t + 1\ \forall j = 1, ..., r$.*

When it comes to data protection, a censorship resistance system thwarts the censor's attempts to corrupt the information or its publication/access. We draw from the existing requirements [178] to state the following security properties for service censorship resistance:

1. *Unblockability or Service Availability* - even after identifying a threshold number of service providers, the adversary cannot block the service.

2. I*ntegrity or Service Robustness* - despite an adversary's intervention and its attempts to tamper with the system, the service is provided correctly.

5.1.3 Our approach

The surveyed literature establishes that one way to ensure censorship resistance is to opt for distribution [178, 180, 181]. Recently, He *et al.* also suggested the use of blockchains to provide censorship-resistance in IoT management systems [182]. In this chapter, we pursue the same idea and propose to leverage full-distribution to tackle Service Censorship against the $\mathcal{DOGS}$ service and the Distributed Issuing service. To this end, we aim to provide the following functionalities.

> **Functionality 1:** Anyone should be able to trigger the creation of the group of sub-openers from the blockchain network. This refers to the $\mathcal{DOGS}$ service.
>
> **Functionality 2:** The Issuing authority should be distributed. This consists of the Distributed Issuing service.

Functionality 1. To provide a non-biased mechanism for the selection of the sub-openers, we define a smart contract $\mathcal{C}_O$. When a node wants to create a new distributed Opening authority, it deploys the $\mathcal{DOGS}$-Registration smart contract denoted $\mathcal{C}_O$, which triggers the event 'SUB-OPENER REGISTRATION OPENED'. The RSUs can apply through their blockchain client by locking coins via the smart contract. When a defined number N of candidates have participated, the contract triggers another event 'SUB-OPENER REGISTRATION COMPLETED' which indicates to the volunteer nodes to start $\mathcal{DOGS}$. Since no intervention from the TA is required during the process, this method ensures the properties mentioned above of 2) Service Availability and 3) Service robustness for the $\mathcal{DOGS}$ service.

Functionality 2. We suggest combining blockchains and DKG protocols to implement the distribution of the issuing authority. Anyone can deploy the Sub-Issuer-Registration contract $\mathcal{C}_I$, informing the volunteer nodes that they can candidate to become a sub-Issuer. Once the group issuing keypair $ipk, (isk_1, isk_2, \ldots)$ is generated, it can be used during the `Join`/`Iss` joint protocols. A vehicle encrypts its join request with the ipk s.t. the sub-Issuers must collaborate to decrypt the request. Then, one of them is elected to issue the car a certificate. Once again, no intervention from the TA is required during the process. Moreover, the Issuing Authority is no longer entitled to a single node but ensured by a group of sub-issuers. This distribution provides the properties mentioned above of 2) Service Availability and 3) Service Robustness for the Distributed Issuing service.

5.2 Towards Issuer-as-a-service

In this section, we show how by naively combining a threshold cryptosystem and a distributed key generation protocol, we can transform the model and add more decentralization by distributing the Issuing authority.

5.2.1 Assumptions

For the construction of the new-DEN messaging protocol, we make the following assumptions.

- A blockchain, supporting smart contract executions, is bootstrapped and running. There are two contracts: $\mathcal{C}_O$ for the selection of sub-openers and $\mathcal{C}_I$ for the sub-issuers.
- Each node in the network possesses a blockchain account acc_i corresponding to a pair of public/private keys pk_i, sk_i, and an P2P address $addr_i$.
- The vehicles use a threshold encryption scheme $\mathcal{TE} = (\texttt{KeyGen}, \texttt{Encrypt}, \texttt{Partial-Decrypt}, \texttt{Combine})$ to encrypt their join request before sending it to the sub-issuers. Let us note that we will denote $\mathcal{TE}. < \texttt{function\_name} >$ the call to one of the four algorithms that define this threshold encryption (according to the description given in Chapter 3).
- $H(\cdot)$ is a one-way Collision-Resistant Hash Function (CRHF).

5.2.2 Description of the enhanced new-DEN messaging protocol

① **Setup.** The initialization of the new-DEN messaging protocol is divided into two steps: (1.1) setup of the distributed Opening authority; (1.2) setup of the distributed

Issuing authority.

(1.1) **Setup of the distributed Opening authority.** This phase starts with the deployment of the smart contract $\mathcal{C}_O$. Anyone in the system can do this. This action triggers the emission of the Sub-Opener Registration opened event that informs the RSU nodes in the system of the possibility to participate in the establishment of the distributed Opening authority. The RSUs can become candidates by locking some coins through $\mathcal{C}_O$. Upon reception of a predetermined number N of applications, the smart contract emits the Sub-Opener Registration Completed event. Upon reception of this event, the RSU candidates know that they can execute the **Opening Key Generation** phase of $\mathcal{DOGS}$. The execution ends with the publication of the group (public) opening key opk. Moreover, the sub-opener nodes are identified, and the Opening authority is set up for further use.

(1.2) **Setup of the distributed Issuing authority.** Similarly to what is described in the **Opening Keys Generation** phase in $\mathcal{DOGS}$, we use a blockchain-supported DKG protocol to generate the Issuing Keys. The phase starts with the deployment of $\mathcal{C}_I$. This action triggers the emission of the Sub-Issuer Registration opened event that informs the RSU nodes in the system of the possibility to participate in the establishment of the distributed Issuing authority. The RSUs can candidates by locking some coins through $\mathcal{C}_I$ similarly to what they did in step (1.1). Upon reception of a predetermined number N' of applications, the smart contract emits the Sub-Issuer Registration Completed event. Upon reception of this event, the RSU candidates know that they can execute a blockchain-based DKG protocol for the issuing key (by following the protocol of Schindler *et al.* [38] for instance). At the end of this execution, the sub-issuer nodes are identified, and the public issuing key is advertised for further uses.

② **OBU Registration.** In previous registration processes, the OBU V_i was interacting with a single TA to get its temporary keys certified. Now, the V_i will engage in the `Join`/`Iss` joint protocols with a group of RSUs. To this end, it will encrypt its request for registration, which contains (i, pk_i, sk_i, sig_i), with the threshold encryption scheme $\mathcal{TE}$.`Encrypt` and the public key of the distributed Issuing authority. The group of RSUs, chosen to incarnate the sub-issuers, collaborate to decrypt the registration request and further processing. They individually apply the $\mathcal{TE}$.`PartialDecrypt` function to the ciphertext and then combine the resulting partially deciphered messages with $\mathcal{TE}$.`Combine` to reconstruct the request in plaintext. *Then, one of them is randomly elected to certify V_i's temporary keys.* It then encrypts $(i, pk_i, sk_i, sig_i, cert_i)$ with $\mathcal{TE}$.`Encrypt` and ipk, and concatenates $H(cert_i)$ before adding the result to the registration table reg.

③ **OBU Signing.** Once the registration phase is completed, the registered OBUs can use the new group signature algorithm `GSig` in the same way as initially described in Chapter 4 Section 4.1.

④ **OBU Verifying.** The new construction does not modify `GVf` either. Therefore, the signatures can be verified in similar ways as before by applying the `GVf` algorithm.

⑤ **Distributed Identification.** This step is divided into two parts: the Opening of the signature and then the identification of the signer.

(5.1) **Distributed Opening.** This refers to $\mathcal{DOGS}$ processing of the signatures. In case of dispute, an opening request is broadcast and logged inside the blockchain. The sub-openers collaborate to, if they want, the Opening functionality. The result of this process reveals the certificate $cert_i$. To identify the owner of this certificate, the sub-openers forwards this value to the sub-issuers.

(5.2) **Distributed Identification.** All the sub-issuers look-up in the reg table to find the corresponding $H(cert_i)$. Then, they collaboratively decrypt the full associated entry and finally reveal the identity of the signer.

⑥ **Audit.** Of course, since the RSUs can be corrupted, they must produce proof of identification. It is the combination of the proof of Opening provided by the sub-openers and the proof of identification. This includes demonstrating the relationship between all the elements in $(i, pk_i, sk_i, sig_i, cert_i)$ (since sig_i is the signature of pk_i with usk_i, it can be verified with upk_i).

5.2.3 Analysis and limitations

As of now, we only focus on the secure bootstrapping of the infrastructure nodes and Issuing/Opening authority, such that the system can provide a privacy-preserving accountable new-DEN messaging protocol. The distribution of the Issuer and the automation of the selections of sub-openers and sub-issuers do not impact the subsequent vehicular communications directly. However, they do affect the robustness of the system against attacks. In this sub-section, we analyze the proposed new-DEN messaging protocol *w.r.t.* the properties of censorship- and coercion-resistance.

Theorem 3 (Censorship-resistance). *If the sub-opener and the sub-issuer registration mechanisms are censorship-resistance, then the proposed* new-DEN *messaging protocol is censorship-resistant.*

Argument of security.
TA was the only entity that could apply censorship either by non-randomly selecting the sub-opener nodes or by refusing to register users as group members. The two processes are now automated via the smart contracts $\mathcal{C}_O$ and $\mathcal{C}_I$. Moreover, the adversary can only corrupt up to $t < p/2$ RSUs. Therefore, it cannot prevent service provision ensured by either the sub-opener nodes or by the sub-issuers. The proposed new-DEN messaging protocol **is censorship-resistant**.

However, the system is not coercion-resistant. It is due to the presence of a single point of failure in the issuing process. Indeed, the issuing authority is given to each RSU : “one of them is randomly elected to certify V_i's temporary keys".

About Coercion-resistance. Coercion resistance is prevalent in e-voting systems. Informally, it is defined as the

> *'[Incapacity] of a voter to cooperate with a coercer to prove to him that they voted in a certain way'* [183].

In contrast, the privacy property focuses on the unlinkability of the voter and the vote cast, coercion-resistance guarantees that the coercer cannot learn how the voter voted. Later, Juels *et al.* formalize the property as a stronger form of privacy in which the voter can interact with the others [184]. Subsequent works have been proposed to define this coercion-resistance property as illustrated by Haines and Smyth's systematization of knowledge [185]. Our second functionality is very similar to e-voting systems as the sub-issuers are asked to collaborate to access the user's request for further processing. We define the notion of Service Coercion-resistance as follows.

Definition 9 (Service Coercion-resistance)**.** *Let $P_1, \ldots, P_p$ be $p > 3$ nodes constituting the network that proposes a service S, and let $\mathcal{A}$ be an adversary that can compromise t nodes in the system (without loss of generality, we consider that the corrupted parties are $P_{p-t+1}, \ldots, P_p$). The* coercion-resistance property *guarantees that the probability that $\mathcal{A}$ determines if an honest P_i provided S is equal to $1/2 + negl(\lambda)$.*

Yet, why do we need coercion-resistance? The property is fundamental during the Issuing process. Indeed, consider a node that requests access to the network, *i.e.* ask to become a group member, and let assume that the adversary compromised the i^{th} sub-issuer node I_i. As such, the adversary learns: who is the user that requests to join the group, if the sub-issuer participated to the issuing process; and can also determine whether the node was issued a certificate at all. By being able to determine how a sub-issuer acts, the adversary can more easily influence its behavior (by bribing it into not issuing certificates). To prevent the adversary from learning this information and gain substancial advantage over the distributed issuing process, we suggest improving the previous construction by providing sub-issuer anonymity when they participate in the Issuing process. To this end, we develop a new cryptographic primitive, called $\mathcal{TOAD}$, that we detail in the following section.

5.3 Construction of $\mathcal{TOAD}$

In this section, we introduce our second contribution, $\mathcal{TOAD}$ which stands for "Thresh-Old Anonymous Decryption". To the best of our knowledge, this is the first *blockchain-based threshold encryption scheme* that supports an *anonymous-yet-accountable decryption service.* The blockchain has two purposes. It is used to advertise the service (*e.g.,* nodes that want to provide a decryption service declare their availability and generate proof that they have the necessary resources). And it also serves to trace the actions within the system (*e.g.,* logging each request and tracking anonymous participation for further analysis if necessary). The threshold encryption alleviates the problems related to a central decryption server: namely censorship and availability. The anonymity during decryption is provided by a combination of a group signature scheme controlled by a distributed authority external to the main application and an anonymous-yet-traceable DKG protocol.

5.3.1 Construction

This study aims to design a secure threshold encryption scheme that is entirely distributed, protects the anonymity of the decryption nodes while ensuring their accountability.

System model

The system is composed of a requester P_0 and a set of designated decryption servers $\mathcal{Q} = \{P_1, \ldots, P_n\}$, where $n \geqslant 3$ is an integer. We distinguish the P2P network through which they broadcast *messages* M_i to IP addresses $addr_i$; from the blockchain network through which they publish *transactions* t_i via their account number acc_i.

Assumptions

For the construction of our protocol, we assume that the following statements hold:

- A blockchain, supporting the use of mixers and the pegging of sidechains is bootstrapped and running. It implements bridges that support the transfer of assets from the main to the sidechains. This blockchain is called the *mainchain*.

- The secondary blockchain, which we will call *sidechain*, supports smart contract executions and is also bootstrapped and running. The use of smart contracts enable the parties to publish transactions t_i and therefore commit to the broadcast messages m_i. Moreover, it allows for disseminating events that inform the blockchain network that a new action was performed.

- Each decryption server owns a mainchain account acc_i corresponding to a pair of public/private keys pk_i, sk_i.

- A group signature scheme is already deployed (and we can consider either a centralized model such as defined in [34] or a distributed blockchain-oriented one [186, 187]). Therefore, the Opening authority is publicly identified. Moreover, since the scheme is deployed, *i.e.* the Issuer and the Opener are known, and the group is empty, new nodes that request to join the set of decryption servers engage in the `Join`/`Iss` joint protocols to integrate the group. They are registered.

- We use the ETHDKG contract to perform DKG on the blockchain as defined in [38].

- There exists a blockchain-based anonymous-yet-traceable construction of a DKG protocol that we denote `NewDKG`.

Adversarial model

We assume that the adversary $\mathcal{A}$ can corrupt up to t parties in the network $\mathcal{P}$ for any value $t < n/2$. We consider a malicious adversary that may cause corrupted parties to divert from the specified protocol in any way. We assume that a probabilistic polynomial-time Turing machine adequately models the computational power of the adversary. Our adversary is static, *i.e.*, chooses the corrupted parties at the beginning of the protocol.

Description of the protocol

The protocol is divided into six phases (① INSTANTIATE, ② ANONYMOUS-DKG, ③ ENCRYPT, ④ CONTRIBUTE, ⑤ DECRYPT) and involves the use of one smart contract $\mathcal{C}_1$ and one routine `newDKG`. The contract $\mathcal{C}_1$ contains all the functions related to the setup and usage of the threshold encryption scheme, namely (`ethdkg`, `encrypt`, `contribute`, `decrypt`). The `ethdkg` function triggers the deployment of the ETHDKG contract. Then, the other functions generate the publishing of hash values inside transactions. The routine `NewDKG` is used as an oracle that improves anonymity by delivering new accounts, to the users, that are not linked to their previous ones and performing a blockchain-based DKG. In later sections, we will detail the implementation of these components and how they are used throughout the execution of the protocol to comply with the stated goals.

① **Contact.** The protocol starts with P_0 identifying its decryption servers $\mathcal{P} = \{P_1, \ldots, P_n\}$. P_0 calls the function `ethdkg` from $\mathcal{C}_1$ on inputs the list of the P_i's accounts $\{acc_1, \ldots, acc_n\}$, and esk_0 a symmetric encryption key used to encrypt the P_i's exchanges during the establishment of a common threshold encryption keypair. This triggers the deployment of ETHDKG and forwarding of the input data.

② **Anonymous-DKG.** Before they can execute the DKG protocol, the chosen P_is must obtain new account numbers that are not linked to their public ones. To do so, they collaboratively engage in the `NewDKG` protocol. **This protocol is for now a black box** and we assume that it provides the following features:

- At the end of the execution, the contacted decryption servers have new anonymized accounts;
- They have used these new accounts to execute the DKG (by calling an instance of the ETHDKG contract [38] with their anonymized accounts).

At the end of the protocol, one elected P_i, among the parties that correctly participated, will broadcast the public key and publish the corresponding hash value to terminate the DKG-related contract and update $\mathcal{C}_1$.

③ **Encryption.** Now that the threshold cryptosystem is entirely set up, *i.e.* the encryption key is set and the decryption servers are known, P_0 can encrypt a message. To do so, it applies a traditional hybrid encryption function $\mathcal{HE}$.`Encrypt` along with the group encryption key gek on the payload m_0. Then, it broadcasts this encrypted message, and use $\mathcal{C}_1$'s encrypt function `encrypt`. The output is the publication of the hash of the transmitted payload and the distribution of an event New Encryption.

④ **Contribute.** Upon the reception of a New Encryption event and corresponding transaction t_0 and message m_0, and once the decryption servers have new accounts $acc_i^{'}$, unlinkable to their previous ones acc_i, the decryption process can occur. First, the $P_i \in \mathcal{R}$ decrypt the received data with their shares s_i of the group decryption key gdk. Then, each $P_i \in \mathcal{R}$ broadcasts the partially decrypted message denoted $m_{0,i}$, along with the temporary identifier u_i[1] used in the DKG protocol. Once partially decrypted, P_i calls the `contribute` function in $\mathcal{C}_1$ to commit to both $m_{0,i}$ and u_i, and to emit the New Partial Decryption event.

⑤ **Decrypt.** Once all $P_i \in \mathcal{R}$ participated, all the decryption servers can combine the given shares $m_{0,j}$, decipher the encrypted file and check that the decrypted message matches the published hash value $H(m_0)$. Each P_i publishes with their non-anonymized account acc_i the hash value of the decrypted message.

Figure 5.2 illustrates the exchanges and published transactions throughout the execution of the protocol. In parallel with the publications of these hash values, the following data are broadcast:

① $m_{contact} = \langle \overline{(esk_0, addr_{DKG})}^{k_{0i}}, sig_0 \rangle$ where esk_0 is the symmetric key used to protect the communications between the parties during the DKG, $addr_{DKG}$ is the address of the DKG contract, and sig_0 the signature of P_0 obtained with

[1] Traditionally, u_i is the account number $acc_i^{'}$ or the corresponding public key of the participating party. Thanks to the Mixing step, this value is indeed temporary. It is used in the Lagrange Interpolation theorem to obtain gdk.

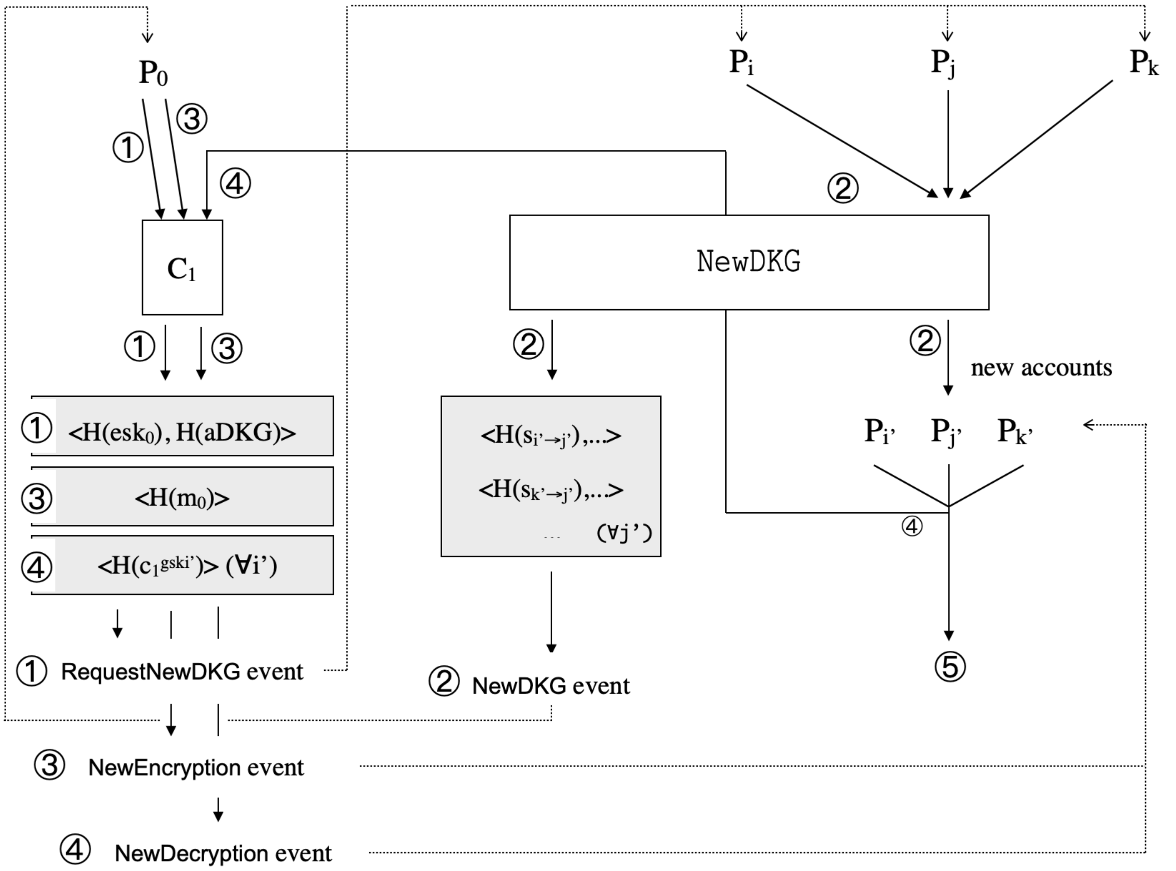


FIGURE 5.2: Illustration of the protocol execution and the generated blockchain transactions

a traditional signature scheme. Moreover, we note that this message is sent encrypted n times with k_{0i} the symmetric key resulting of the combination of P_0 and P_i keys ($k_{i0} = pk_0^{sk_i} = pk_i^{sk_0}$). However, the values are committed only once.

② m_{DKG} are DKG messages and can be equal to: $\langle \overline{s_{i'\to j'}}^{k_{ij}}, \sigma_i \rangle$ during the sharing phase; or $\langle \overline{k_{ij}, \pi_{ij}}^{esk_0}, \sigma_i \rangle$ if there is a dispute claim. σ_i is the group signature of the individual P_i with its true initial and registered account acc_i and related information pk_i, sk_i.

③ $m_{encrypt} = \langle c_0 = \overline{m}^{gek}, sig_0 \rangle$ the encrypted message.

④ $m_{contribute} = \langle \overline{c_1^{gsk_i}}^{esk_0}, \sigma_i \rangle$ where c_1 is the first component of the ElGamal encryption of m_0, $c = (c_1, c_2) = (h^r, gek^r \cdot m)$.

⑤ Decryption can be done individually.

5.3.2 Security analysis

This section recalls and formally defines the two main targeted security properties: anonymity and traceability.

Anonymity. As stated by Pfitzmann and Köhntopp [70], the anonymity property asks that an element is *not identifiable* within a set of subjects, referred to as the *anonymity group*. Here, we recall that $\mathcal{P} = \{P_1, \ldots, P_n\}$ is the set of selected decryption servers and we denote $AG(P_i)$ the anonymity group associated to the user P_i. In order to prove that the proposed protocol provides anonymity to the decryption servers when they participate to the decryption process, we have to prove that for any active adversary which corrupted at most t-out-of-n party, the probability to successfully identify the origin of a contributing transaction $t_{contribute} = \langle H(c_1^{gsk_i}) \rangle$ and message $m_{contribute} = \langle \overline{c_1^{gsk_i}}^{esk_0}, \sigma_i \rangle$ is equal to $\frac{1}{n-t+1}$. By following the framework proposed by Mauw *et al.* in [188], we formalized the notion of anonymity as follows.

Definition 10 (Anonymity)**.** *Let the system be as stated in 5.3.1, let $\mathcal{A}$ be the PPT adversary which corrupted exactly t-out-of-n parties from $\mathcal{P}$. $\mathcal{O}bs$ is a set of observable actions* i.e. *a set of transactions t and a set of messages m. For a user $P_i \in \mathcal{P}$, we define its anonymity group $AG(P_i)$ as the set of P_j in $\mathcal{P}$ that, upon the publication of a transaction t and the broadcast of the corresponding message m, has the same probability as P_i to be identified as the origin of the pair (t, m).*

Theorem 4. *For any uncorrupted party $P_i \in \mathcal{P}$, let $AG(P_i)$ be the anonymity group of P_i. We say that the proposed protocol is $(n - t)$-anonymous if and only if:*

$$AG(P_i) \subseteq \mathcal{P} \text{ and } |AG(P_i)| = n - t \tag{5.1}$$

Proof. The proof is twofold: we have to prove that this construction ensures the anonymity of the peers both at the blockchain and network layers.

1. Blockchain layer. In this paragraph, we must demonstrate that no anonymity leak occurs neither from the publication of the transaction nor from its content.

1.1 Publishing transactions. From a pure blockchain-based approach, since `NewDKG` is an anonymous DKG protocol, the probability to determine which P_i contributed to the decryption process and issued $c_1^{gsk_i}$ is $1/N$. Therefore, if we consider that the adversary corrupted the maximum parties t, we have $N = n - t$ and can conclude that our protocol provides anonymity when publishing transactions. Now, we have to verify that the content of the transactions does not release identifiable information.

1.2 Indistinguishable transactions. Now, we show that it is not possible with a high probability of distinguishing two transactions issued during the same phase. More specifically, we will show that during ③ and ⑤, the adversary is not able to link back a transaction to the P_i that issued it. Let us consider two uncorrupted participants $P_i, P_j \in \mathcal{P}$. The transactions issued by both parties contain only hash values, which are considered as random values in the Random Orable Model, and are published with anonymized accounts. Concerning part 1 of the proof, the second statement implies that the publication does not reveal the identity of the publishing node.

Now another question stays unanswered: what happens in the peer-to-peer network? Does the information broadcast hinder the privacy-preservation efforts done at the blockchain level? We will explain ways to mitigate the impacts of these exchanges on users' anonymity and show that the data itself does not reveal any additional personal information about the decryption servers.

2. Network layer. Once again, we need to address two facets. We must prove that there is no anonymity breach at the network layer, neither from the broadcasting of messages nor from their content.

2.1 Zero-knowledge data. Now, we show that it is not possible with a high probability of distinguishing two messages of the same type (meaning issued during the same phase). More specifically, we will show that during phase ③, and ⑤ of the protocol, the adversary is not able to exploit the content of a message broadcast by P_i. Let us consider two uncorrupted participants $P_i, P_j \in \mathcal{P}$.

③ During this phase, the protocol executes a blockchain-based DKG with temporary information (the polynomial, the secret value, the account number i,...). Therefore, since all these parameters are not linked to P_i, they do not leak any identifiable information.

⑤ In this phase, the pseudo-identifying information are $c_1^{gsk_i}, acc_{i'}$. Since $acc_{i'}$ is obtained after mixing, it does not reveal anything about the broadcast P_i (*w.r.t.* parts 1 and 2 of the proof). Moreover, due to the discrete logarithm assumption, there is no retrieving of the gsk_i information, nor is there telling whether the c_1 component is an actual contribution (impossible to distinguish $c_1^{gsk_i}$ from c_1^{α} with $\alpha \xleftarrow{\$} \mathbb{Z}_q$). The encryption with esk_0 does not reveal any additional information regarding what is already known.

In addition, the group signature attached to each of the P_i's transactions is generated by an anonymous-yet-traceable group signature scheme. This means that, unless the Opening authority opens the signature, it remains anonymous. Therefore, the broadcast data does not leak personal information that an adversary could exploit to infer which decryption servers P_i participate.

2.2 Anonymous network addresses. The last issue to discuss is related to the P2P network and linkability of the broadcasting identities. This can easily be discarded as there are a plethora of options to dynamically change the address/identifier of the communicating nodes at the network layer [61, 189].

Proof conclusion. In presence of an adversary $\mathcal{A}$ that corrupts exactly t-out-of-n parties from $\mathcal{P}$, and under the aforementioned assumptions, the anonymity group of an uncorrupted user P_i in $\mathcal{P}$ denoted $AG(P_i)$ is equal to AG the set of potential publishers $P_k \in \mathcal{P}$ that have not been corrupted by $\mathcal{A}$. Then, the probability that $\mathcal{A}$ successfully identifies the sender of a transaction, which is by definition $1/|AG(P_i)|$, is equal to $1/(n-t)$ verifying the Definition 10. Theorem 4 holds. □

Definition 11 (Traceability)**.** *Informally, the traceability property applies to digital signature schemes and ensures that:*

1. *an honest user that does not sign a message m should not be convincingly declared*

*as a possible signer of this message (*i.e. *no one can produce a signature on behalf of another member), it is the* non-frameability *property;*

*2. nobody should be able to produce a valid signature that cannot be linked to an identifiable user (*i.e. *any valid signature can be linked back to the identity that issued it), this is* unforgeability.

Considering Theorem 4 and the corresponding proof, we state the following theorem.

Theorem 5. *If the group signature scheme used to design our protocol is* ***non-frameable and traceable****, then our protocol is so.*

Proof. Indeed, it is quite immediate to understand, from the anonymity proof, that the protocol's non-frameability and traceability properties are guaranteed by the underlying group signature scheme used to sign the broadcast messages. From a pure blockchain point of view, the P_is are anonymous. Therefore, the only identifying information is inserted in the broadcast messages: the group signature σ_i used to authenticate a message. Since, the scheme chosen [34] is proved to be non-frameable and traceable under the Discrete-Log assumption [34], then our protocol is too, under the same conditions. Theorem 5 holds. □

Summary. This sub-section presented the first description of a blockchain-based threshold encryption protocol that supports an anonymous-yet-accountable decryption service. We extended the standard notion of threshold cryptosystems with new security properties such as the anonymous participation to the decryption process and the traceability a posteriori of the participants' actions, even in the presence of a strong adversary that can corrupt up to half of the participants. We designed two use cases that justify the need to develop such new anonymous threshold primitives and why it is essential to support them with accountability-enhanced systems such as the blockchain. We proved that our construction respects the targeted objectives of anonymity and traceability.

5.3.3 How does the new-DEN messaging protocol benefit from $\mathcal{TOAD}$?

Our goal is to set up a secure infrastructure such as to develop a privacy-preserving accountable new-DEN messaging protocol. This implies designing a bootstrapping framework that ensures censorship- and coercion-resistance. To benefit from $\mathcal{TOAD}$, we modify the step (1.2) of the protocol described in Sub-section 5.2.2.

(1.2) Setup of the distributed Issuing authority. This phase is divided into two steps: (1.2.1) The nomination of the sub-issuers; (1.2.2) The selection of the sub-issuers for the ongoing round. This notion of 'round' is similar to an election. With (1.2.1) The sub-issuers are identified in the network. However, thanks to (1.2.2) the adversary cannot know which group of issuers acts as the Issuing Authority during a specific round.

(1.2.1)Nomination of the sub-issuers. This corresponds to the previous (1.2) step. The RSUs register via the deployed $\mathcal{C}_I$ smart contract. Upon reception of a predetermined number N' of applications, the smart contract emits the Sub-Issuer Registration Completed event, and the registered RSUs can execute a blockchain-based DKG protocol for the issuing key. At the end of this execution, the sub-issuer nodes are identified and possess a common Issuing key.

(1.2.2) Election of the sub-issuers. Let ΔT be a predetermined period of time. Every ΔT, the group of acting sub-issuers changes. This change is triggered by the smart contract $\mathcal{C}_I$, which emits the Sub-Issuer election started event. Among the N' sub-issuers, some of them will therefore anonymously engage in the $\mathcal{TOAD}$ protocol to create a common issuing public key (which corresponds to the encryption key output by the protocol) in such a way that they all possess a share of the corresponding issuing secret key. As such, when a user asks for registration to the group, the sub-issues can choose whether to participate or not. Since $\mathcal{TOAD}$ is anonymous and traceable, the elected sub-issuers are accountable for their actions. Yet, they are protected against privacy and coercion related attacks.

Theorem 6. *(Coercion-resistance) If $\mathcal{TOAD}$ is anonymous and traceable, then the proposed* new-DEN *messaging protocol is* coercion-resistant w.r.t. *the specified adversary model (Sub-section 5.1.1).*

Argument of security.

Since $\mathcal{TOAD}$ is traceable, the elected sub-issuers are accountable for their actions. Yet, since the scheme is anonymous, their identity remains private. This anonymity property is what makes the new-DEN messaging protocol resistant against coercion attacks as no adversary can force sub-issuers into doing something that they don't want to do as they cannot learn which one of them have been elected for a specific round. Again, since the group of selected issuers change over time, the adversary should adapt to the new setup, which makes its task more difficult. □

Conclusion. In this chapter, we focused on the infrastructure nodes' censorship- and coercion-resistance. Indeed, both requirements are critical in the designing of a privacy-preserving accountable road hazard warning reporting system. We explained why the presence of the TA in Chapter 4 was a threat to users' privacy and accountability, as the centralization of its power induces availability risks.To alleviate the power of TA, we slightly modified the previous construction of the new-DEN messaging protocol by relying relies on the concept of Issuer-as-a-service. To ensure the privacy-preservation and accountability properties of the protocol, while providing at the same time censorship- and coercion-resistance, we have to develop a new blockchain-based cryptographic primitive: called $\mathcal{TOAD}$.

However, the argumentation above and the construction of $\mathcal{TOAD}$ leans on the existence of a primitive that has yet to be proven: NewDKG. In the following chapter, we propose a construction the said primitive, namely of a blockchain-enabled anonymous-yet-traceable DKG protocol that present the necessary security properties to guarantee $\mathcal{TOAD}$'s ones, and should complete our framework.

6

$\mathcal{BAT}-\mathit{Key}$: a Blockchain-enabled Anonymous-yet-Traceable DKG for $\mathcal{TOAD}$

"Start by doing what's necessary; then do what's possible; and suddenly you are doing the impossible."

— Francis of Assisi

In this chapter, we propose a DKG protocol that utilizes a blockchain to provide trust among the participating distrusting entities. More specifically, we design the missing component of $\mathcal{TOAD}$, denoted `NewDKG` in the previous chapter. In the following, `NewDKG` is renamed $\mathcal{BAT}-\mathit{Key}$ for "Blockchain-based Anonymous-yet-Traceable DKG protocol". The targeted properties are anonymity and traceability of the participants' contributions, and scalability of the system. The novelties of the scheme are multiple. Firstly, we improve traditional DKG implementations with anonymity, which guarantees to the peers willing to participate to our protocol, that their identities will remain private. Secondly, we ensure that peers remain accountable for their actions by ensuring traceability of their participation. Thus, the protocol guarantees that a node, that deviates from the initial protocol, can be identified and its identity exposed. Thirdly, we fork from standard blockchain applications by considering the scalability aspect as a main issue to address. That is why we develop our new technique on a sidechain and demonstrate, from a theoretical and practical point of view, how it provides the scalability needed for mass-market applications.

While $\mathcal{BAT}-Key$ can be used in various contexts (as will be shown in Chapter 8), in the remaining of this thesis, we decided to explore and define anonymous-yet-traceable DKG protocols in the context of VANETs.

6.1 Construction

This research aims to design a Blockchain-supported *Anonymous*-yet-*Traceable* secure DKG protocol. In the following subsections, we highlight the construction of our protocol and fully document each steps of the protocol w.r.t. Figure 6.1.

6.1.1 Cryptographic tools

We denote by $\mathcal{GS}$ a group signature scheme that satisfies the anonymity, traceability and non-frameability requirements [32]. We fix a symmetric $\mathcal{SE} = (\texttt{SEnc}, \texttt{SDec})$ and asymmetric $\mathcal{AE} = (\texttt{AEnc}, \texttt{ADec})$ encryption schemes, both satisfying the standard notion of indistinguishability under adaptive chosen-ciphertext attacks (IND-CCA2). We also consider (P, V) a simulation-sound NIZK proof system for NP languages as an essential underlying primitive for the construction of our protocol. We assume that (P, V) satisfies the properties of completeness, soundness, zero-knowledge and is simulation-sound [190]. Finally, we choose a blockchain environment that enables the creation of sidechains. The main blockchain is called *mainchain* and uses a scripting system for transactions (e.g. Bitcoin); and secondary chains are called sidechains and supports Turing-complete programming languages (e.g. Ethereum).

6.1.2 System and Communication models

The system is composed of a sender P_0 and a set of intended receivers $\mathcal{P} = \{P_1, \dots, P_n\}$, where $n \geqslant 3$ is an integer. We assume they can all monitor and broadcast messages on a shared public and authenticated communication channel. Furthermore, parties all agree on a common view and ordering of these messages. We assume synchrony in the sense that, any message that is broadcast by a participant during some protocol phase is received by all other parties before the next phase starts (partial synchrony model [191]). Finally, we assume that the group signature is already deployed, hence that there is a node $\mathcal{O}$ identified as the opening authority.

6.1.3 Protocol Description

The protocol is divided into eight phases and composed of one public routine KeyEstablishment, two chains, and one client application as illustrated on Figure 6.1.

The mainchain is public and acts as the Coordination Blockchain. It also handles the anonymization of the participants via a *mixing technique*. In parallel, there is one on-demand sidechain administrated by KeyEstablishment which manage the group creation and subsequent actions. We consider that each participant P_i owns one public account acc_i on the mainchain, available through a public directory.

① **Group creation.** The protocol starts with P_0 deploying the KeyEstablishment routine on the sidechain, and identifying its intended receivers $\{P_1, \dots, P_n\}$. It calls the function `groupCreation` in the KeyEstablishment routine on inputs the list of accounts associated to receivers in $\mathcal{P}$, esk_0 a symmetric encryption key used to encrypt sidechain transactions (for confidentiality purposes), and opk the Group Signature Opener's public key. The last two parameters are summarized as psp for *private sidechain parameters*. To ensure their confidentiality, the broadcast message contains n encryptions of psp with an asymmetric encryption function $\mathtt{AEnc}_{pk}(\cdot)$ for $pk = pk_1 \dots pk_n$ the public keys corresponding to the P_is'. Finally, for traceability and verification purposes, a

GROUP CREATION event is emitted and a blockchain transaction publishes the hash value of *psp*.

To participate to further steps of the protocols, P_i must anonymously: 1) commit to its signature onto the mainchain, 2) lock some mains coins onto the mainchain. These two steps are described below.

② **Mixing.** Upon reception of a GROUP CREATION event, P_i can decide whether to anonymize its future blockchain communications. We denoted $\mathcal{P}'$ the set of P_is that decide so, we have $\mathcal{P}' \subseteq \mathcal{P}$. Therefore, if $P_i \in \mathcal{P}'$, P_i participates in a mixing transaction. It engages by building a transaction that spends a standardized amount of main coins and sends them to a new address acc_i'. The core idea is that this transaction will get mixed with other similar transactions in such a way that both P_i's accounts acc_i and acc_i' are unlinkable to any external observer, nor other peers that participated to the mixing. By doing so, P_i covertly owns N main coins at the address acc_i'.

③ **Commit signature and lock main coins.** In order to ensure traceability and authentication, P_i must commit a signature onto the mainchain. P_i will use its own identity to produce an anonymous, unforgeable and traceable group signature. It then publishes it with its new account acc_i'. By doing so, it anonymously ties acc_i' to its identity P_i. Therefore, it acts as an authentication proof and can be opened on P_i's public identity by the Opener authority. Then, the remaining main coins (those that were not used for paying the transaction and fees) can be locked on the mainchain to participate to further steps of the protocol on the sidechain. Let us mention that transactions from acc_i' are accepted on the sidechain iff this commit transaction was correct. If so, P_i only uses acc_i' for the next transactions.

④ **Sharing** The sharing process is backed up by the sidechain (Figure 6.1).

(4.1) **Share initialization** At the beginning of the sharing phase, each $P_i \in \mathcal{P}'$ generates a temporary keypair $\langle tpk_i, tsk_i \rangle$, and draws at random a value u_i from $\mathbb{Z}_q$. Then, it calls the function `temporaryKeysSharing` to broadcast the tuple $\langle tpk_i, u_i \rangle$ and stored

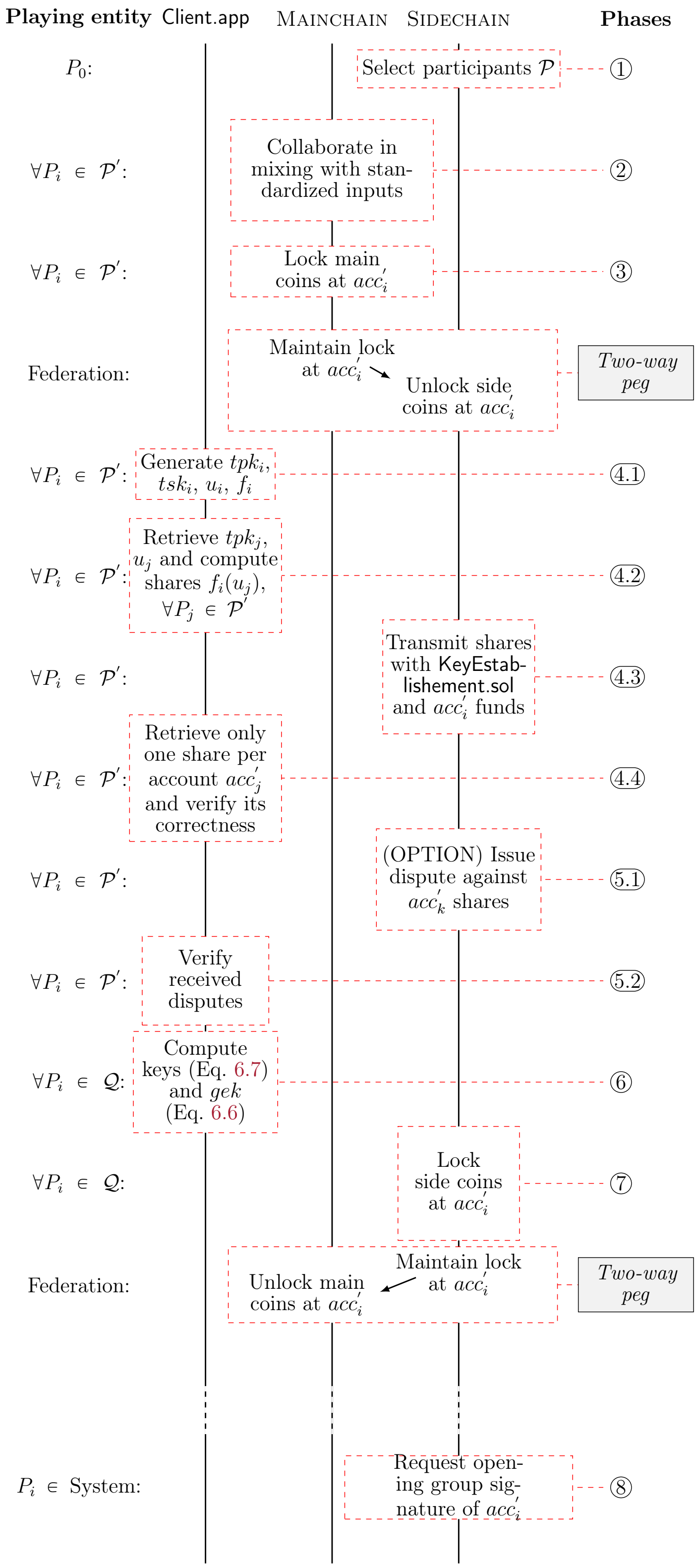

Playing entity
Client.app
Mainchain
Sidechain
Phases
P_0:
Select participants $\mathcal{P}$
①
$\forall P_i \in \mathcal{P}'$:
Collaborate in mixing with standardized inputs
②
$\forall P_i \in \mathcal{P}'$:
Lock main coins at acc_i'
③
Federation:
Maintain lock at acc_i'
Unlock side coins at acc_i'
Two-way peg
$\forall P_i \in \mathcal{P}'$:
Generate tpk_i, tsk_i, u_i, f_i
4.1
$\forall P_i \in \mathcal{P}'$:
Retrieve tpk_j, u_j and compute shares $f_i(u_j)$, $\forall P_j \in \mathcal{P}'$
4.2
$\forall P_i \in \mathcal{P}'$:
Transmit shares with KeyEstablishement.sol and acc_i' funds
4.3
$\forall P_i \in \mathcal{P}'$:
Retrieve only one share per account acc_j' and verify its correctness
4.4
$\forall P_i \in \mathcal{P}'$:
(OPTION) Issue dispute against acc_k' shares
5.1
$\forall P_i \in \mathcal{P}'$:
Verify received disputes
5.2
$\forall P_i \in \mathcal{Q}$:
Compute keys (Eq. 6.7) and gek (Eq. 6.6)
⑥
$\forall P_i \in \mathcal{Q}$:
Lock side coins at acc_i'
⑦
Federation:
Unlock main coins at acc_i'
Maintain lock at acc_i'
Two-way peg
$P_i \in$ System:
Request opening group signature of acc_i'
⑧

Figure 6.1: Overview of the execution flow of the proposed protocol

the hash values $\langle H(tpk_i||u_i)\rangle$ inside the sidechain. The function emits a Temporary Keys Sharing event which informs other selected members that new keys are available for further processing.

(4.2) **Share generation** Then, each $P_i \in \mathcal{P}'$ executes the sharing of their key by executing the first step of Schindler *et al.* DKG protocol [38], which is a relaxed version of Gennaro *et al.* DKG construction. It obtains a secret value s_i, a private polynomial f_i s.t. $f_i(0) = s_i$, and publishes the commitments $C_{i0} = g^{s_i}, \ldots, C_{it} = g^{c_{it}}$ to the coefficients $c_{i0}, c_{i1}, \ldots, c_{it}$ of f_i.

$$f_i(x) = \sum_{k=0}^{t} c_{ik}x^k$$

These committed values will be used later in the verification process of the shares (along with P_i's public polynomial $F_i : \mathbb{Z}_q \to \mathbb{G}_q$ in Equation 6.1).

(4.3) **Share transmission.** Next, each P_i has to securely send its shares $s_{i\to j}$ to all concerned parties $P_j \in \mathcal{P}'$. Since there is no private communication channel between parties, P_i uses a symmetric key encryption algorithm $\texttt{SEnc}_{k_{ij}}(\cdot)$ to ensure the secrecy of the share sent to P_j. The corresponding encryption key k_{ij} is derived from both parties' temporary public keys (Equation 6.2). Finally, P_i broadcasts the encrypted shares $\overline{s_{i\to j}} = \texttt{SEnc}_{k_{ij}}(s_{i\to j})$ for all $P_j \in \mathcal{P}'$ as well as the commitments. For traceability purpose but complying with scalability, in parallel of sending data to other P_j's, P_i proposes a transaction that contains a hash of the concatenation of C_{ik}, for $k = 1..t$, the commitments (Equation 6.3). Each P_j monitors both the broadcast channel and blockchain network for messages and event notifications issued by other parties. Upon receiving encrypted shares and commitments from P_i, P_j decrypts its intended share to obtain $s'_{i\to j} = \texttt{SDec}_{k_{ij}}(\overline{s_{i\to j}})$.

(4.4) **Share verification** P_j employs the verification procedure described in [39] to check the validity of the shares $s'_{k\to j}$ received from other P_k's. Under these conditions, a share is valid if and only if 1) both the published hash of commitments and the one reconstructed from the received data are equal, and 2) the share verification equation holds (Equation 6.4). In case either condition is found invalid, P_j can perform further actions during the Dispute Phase. Let us note that since $P_j \in \mathcal{P}'$ expects to receive only one message per party, it will process the first message received from each sidechain account.

$$F_i(x) = C_{i0} \cdot C_{i1}^{x} \cdot \ldots \cdot C_{it}^{x^t} \tag{6.1}$$

$$k_{ij} = tpk_j^{tsk_i} = tpk_i^{tsk_j} = g^{tsk_i \cdot tsk_j} \tag{6.2}$$

$$H_{commitments} = H(C_{i0}||C_{i1}|| \ldots ||C_{it}) \tag{6.3}$$

$$g^{s'_{k\to j}} = F_k(s'_{k\to j}) \tag{6.4}$$

At each stage of the *Sharing* phase ④, P_i transacts with its new account acc_i'. Since acc_i' is anonymized, other P_j's cannot tell which identity is hidden behind this account.

⑤ **Dispute phase.** Suppose a party P_j receives invalid commitments or an invalid share from acc_i' during the previous phase. P_j must broadcast a dispute claim to ensure that P_i is excluded from further steps of the protocol execution. To prevent false allegations from an adversarial P_j, we adopt the same non-interactive proof technique as [38]: P_j must present elements proving that P_i indeed violated the protocol.

5.1 **Issuing a dispute claim** To do so, P_j will publish the symmetric encryption key k_{ij}. By using this key, other parties may decrypt the corresponding broadcast share and verify P_j's claim. Since k_{ij} is temporary to the execution, revealing it does no impact former, nor later messages than the one being audited. In addition to the publication of k_{ij}, and in order to prevent an adversarial P_j from publishing a random key instead of k_{ij}, P_j must join a NIZK proof $\pi(k_{ij})$. The proof technique shows the equality of two discrete logarithms. The instantiation of the proof $\pi(k_{ij})$ is $P(g, tpk_j, tpk_i, k_{ij}, tsk_j)$.

⑤.2 Verifying a dispute claim. The dispute claim is $\langle k_{ij}, \pi(k_{ij})\rangle$ the symmetric key and the proof of correctness related to that key. P_j broadcasts the claim and commits the hash value of the key and the proof onto the blockchain. This transaction emits a Dispute Claim event. Upon reception of the event notification and corresponding data, P_k uses $V(g, tpk_j, tpk_i, k_{ij}, \pi(k_{ij}))$ to check the claim. If the proof is incorrect or the hash value of the received key/proof does not match the one transmitted through the sidechain network, P_k rejects P_j's claim. Otherwise, P_k decrypts the conflicting share and check its correctness (w.r.t. Equation 6.4). The dispute is valid if and only if 1) hash values of the received key and proof match hash values stored on the sidechain, and 2) the verification condition of the share does not hold.

⑥ Key derivation. The key derivation phase is divided into two steps. First, each participant must determine who are the qualified nodes. Then, they must derive the keys *w.r.t.* the relevant information they saved from the previous step.

Deriving the set of qualified nodes. The first step in the key derivation phase is determining which of the initially selected participants are qualified to participate in the elaboration of the joint group encryption keypair $\langle gek, gdk\rangle$.

At the end of the previous phase, any party $P_i \in \mathcal{P}'$ belongs to either category:

1. parties that correctly shared their secret s_i with all other parties denoted $\mathcal{Q}$;
2. parties that incorrectly shared their secret s_i, meaning that there exists at least one valid dispute claim against the party;
3. parties that did not participate, meaning that they did not share their secret.

The set $\mathcal{Q}$ is the set of qualified nodes. These parties are characterized by coherent pairs of broadcast/transaction (e.g., the commitments received match the published hash value of their concatenation), and no other node has filed a valid dispute against P_i so far.

Deriving the keys. Let h denotes a generator of $\mathbb{G}_q$ such that $dlog_g(h)$ is not known (and difficult to compute in $\mathbb{G}_q$). To prevent biases in the generated keys, each $P_i \in \mathcal{Q}$ will broadcast $\langle h^{s_i}, \pi(h^{s_i})\rangle$ a share of the final group encryption key and the proof which shows the correspondence between h^{s_i} and the first commit $C_{i0} = g^{s_i}$. Notice that in case any (adversarial or faulty) party $P_k \in \mathcal{Q}$ does not reveal the value h^{s_i} or a valid proof $\pi(h^{s_i})$ by the end of this phase, a set of $t+1$ correct parties can recover s_i with a Lagrange Interpolation-based recovery procedure.

We let gek be the common group encryption key. Each $P_j \in \mathcal{Q}$ obtains a group keypair $\langle gsk_j, gpk_j\rangle$. Therefore, we can compute the (distributed) common group decryption key gdk. We can check that $gek = h^{gdk}$ (Equation 6.6).

⑦ **Unlocking.** At the end of the derivation phase, P_0 can tell the protocol successfully terminated and can determine which shares are valid. It can therefore derive the value of gek. Also, it knows which $acc_i^{'}$ participated, and can uses these new accounts to contact the P_i's for further processing. The participating P_i's can also claim back their main coins. To do so, they lock the remaining side coins on their accounts $acc_i^{'}$. Through a consensus, the corresponding main coins are unlocked on the same accounts.

$$s_i = \sum_{P_j \in \mathcal{R}} s_{i\to j} \prod_{\substack{P_k \in \mathcal{R} \\ k \neq j}} \frac{u_k}{u_k - u_j} \tag{6.5}$$

$$gek = \prod_{P_i \in \mathcal{Q}} h^{s_i} \;\; ; \;\; gdk = \sum_{P_i \in \mathcal{Q}} s_i \tag{6.6}$$

$$gsk_i = \sum_{P_j \in \mathcal{Q}} s_{j\to i} \;\; ; \;\; gpk_i = h^{gsk_i} \tag{6.7}$$

$$gdk = \sum_{P_j \in \mathcal{R}} gsk_j \prod_{\substack{P_k \in \mathcal{R} \\ k \neq j}} \frac{u_k}{u_k - u_j} \tag{6.8}$$

6.2 Security analysis

In this subsection, we briefly sketch the formal security proofs w.r.t. the targeted functionalities (Subsection 6.2.1) and the adversary considered (Subsection 6.2.2).

6.2.1 Targeted functionalities

Our protocol should present the following features:

- The distributed generation of the public/private keypair is *correct, robust and entirely distributed* to the set of receivers $\mathcal{P}$ selected by P_0.
- The participation to the distributed generation of the public/private keypair is *anonymous* w.r.t. the adversary;
- The participation to the distributed generation of the public/private keypair is *traceable* w.r.t. the adversary.

6.2.2 Model of adversary

We consider an adversary targeting both properties: anonymity and traceability. The adversary is either external to the system or internal, passive or active. It may want to: de-anonymize users from the collected broadcast messages, or from the collected published transactions; impersonate a user; covertly participate to the protocol. As such, we will show that our protocol is secured against a malicious P_0 as P_0 is the node that has the most knowledge about the system (it selected the P_is) and the most power (as it can participate to the protocol too).

6.2.3 Correctness, Robustness and Distribution

Now, we formally define these three requirements and show that our protocol complies with them.

Correctness. The designed scheme should be correct w.r.t. secure DKG protocols. We reuse the definitions provided by Gennaro *et al.* in [39] (Definition 12).

Definition 12 (Correctness)**.**
(R1) All subsets of $t+1$ shares provided by honest parties define the same unique secret value x.
(R2) All honest parties have the same value of public key $y = g^x \mod p$, where x is the unique secret guaranteed by (R1).

Robustness. This property implies that the reconstruction of x should be possible also in the presence of malicious parties that try to undermine the computation; [39] formalizes this notion as follows.

Definition 13 (Robustness)**.**
(R3) There is an efficient procedure that on input the n shares submitted by the parties and the public information produced by the DKG protocol, outputs the unique value x, event if up to t shares are submitted by faulty parties.

Uniform distribution. Former DKG protocols were proven against an adversary that was able to corrupt less that $t+1$ parties and could still reconstruct the secret value x. These attacks are the result of non-uniform distribution of the shares, meaning that two distinct shares of x may contribute differently to its reconstruction. In our case, we must ensure that Definition 14 is respected.

Definition 14 (Uniformity)**.**
(R4) s is uniformly distributed in $\mathbb{Z}_q$ (i.e. $gek = g^s$ is uniformly distributed in the subgroup generated by g).

Secrecy. Finally, we require that our protocol complies with the secrecy requirement as defined by [39] and recalled in Definition 15.

Definition 15 (Secrecy)**.** *(R5) No information on s can be learned by the adversary except for what is implied by the value $gek = g^s \mod p$.*

Theorem 7 (Secure DKG)**.** *Under the Discrete-Log Assumption, the presented protocol, as defined in sub-section 6.1.3, is secure for DKG in dlog-based cryptosystems; namely, it satisfies the correctness, robustness, uniform distribution and secrecy requirements of sub-section 6.2.3 with threshold* t*, for any* $t < n/2$*.*

Proof. The proof directly flows from Gennaro *et al.*'s demonstration and Schindler *et al.*'s analysis [38, 39].

6.2.4 Anonymity and Traceability

Below, we define the anonymity and traceability properties, and we show that our protocol is compliant with both requirements.

Anonymity This property aims to ensure that users' identity will stay private during the protocol execution. In our case, this entails two features: *unlinkability of blockchain accounts* (Definition 16), and *indistinguishability of sidechain transactions* (Definition 17).

Definition 16 (Unlinkability of Blockchain/Sidechain accounts)**.**
(R6) Given a mixing transaction characterized by a set of inputs $\mathcal{I}$ *and a set of outputs* $\mathcal{O}$ *with* $|\mathcal{I}| = |\mathcal{O}|$*, considering a honest user* P_i *for who* $\exists in_i \in \mathcal{I}$ *and* $out_i \in \mathcal{O}$ *s.t.* in_i *maps to* out_i*, we say that the mainchain/sidechain accounts are unlinkable iff the probability that* $\mathcal{A}$ *successfully maps (denoted by the function* **map***)* P_i*'s input to the corresponding output is equal to* $1/m + \textsf{negl}(k)$*.*

Definition 17 (Indistinguishability of sidechain transactions)**.**
(R7) Given a set of sidechain transactions $\mathcal{S}_t = \{stx_1, \ldots, stx_m\}$ *with* $m \geqslant n$*, and given a user* P_i *that sends transaction* $stx_i \in \mathcal{S}_t$*, we say that the sidechain transactions are indistinguishable if and only if the probability that* $\mathcal{A}$ *successfully maps* stx_i *to* P_i *is equal to* $1/m + \textsf{negl}(k)$*.*

Traceability The notion of traceability usually applies to digital signature schemes or any other authentication protocol. In our case, we need to ensure that the protocol is compliant with Definition 18.

Definition 18 (Traceability).
(R8) The traceability property guarantees that 1) an honest user that did not send a transaction tx should not be convincingly declared as a possible sender of this tx; 2) nobody should be able to produce a transaction tx that cannot be linked to an identifiable user.

Theorem 2 (Anonymous and Traceable DKG).

Theorem 8. *Under the Discrete-Log assumption and the existence of a CRHF, the presented protocol, as defined in sub-section 6.1.3, is an anonymous-yet-traceable protocol for DKG in dlog-based cryptosystems; namely, it satisfies the indistinguishability (of transactions), unlinkability (of Blockchain accounts) and traceability requirements of sub-section 6.2.3 with threshold t, for any $t < n/2$.*

Proof. **(R6)** P_0 knows who are the selected $P_1, \ldots, P_n$. Therefore, it can identify within the set of inputs which belong to the P_i's. But then, the mixing transaction provides relationship anonymity. Therefore, the probability to successfully select the corresponding n outputs out of m is one out of $\binom{m}{n}$ possibilities. Therefore, the presence of the mixing step makes sure that the strongest adversary P_0 cannot link the original accounts to the P_i's to the one they get after mixing (the participants do not learn anything either).

(R7) P_0 knows that $P_i \in \mathcal{P}$. However, since the blockchain accounts are unlinkable (requirement R6), P_0 cannot tell from mainchain transactions whether P_i still participates. Moreover, since the protocol respects the *secrecy requirement* (property R5), P_0 does not learn any additional information about the participating P_j's from the broadcast data nor the published transactions. Hence, the probability that it guesses which transaction from $\mathcal{S}_t$ was published by P_i is equivalent to the one of a random guess, thus $1/|\mathcal{S}_t| = 1/m$.

(R8) This traceability property asks that the adversarial P_0 be unable to produce a transaction s.t. the honest opener is unable to identify the origin of the transaction. This property is twofold. First, we remark that the blockchain is immutable and the consensus power cannot be centralized, no record can be changed: hash values cannot be tampered. Let P_0 own the anonymized account acc_0. It is able to produce a transaction, accepted by honest users, that cannot be linked back to its identity iff: Case 1) P_0 participates in a mixing transaction and does not commit its signature with its new anonymized account. If that is the case, P_0 obtains $acc_0^{'}$ which is not linkable to acc_0. However, since P_0 did not execute the commit signature phase (phase 3), the honest users will only ignore sidechain transactions from $acc_0^{'}$. Therefore, P_0 won't be able to participate to the DKG protocol and there is no notion of traceability left. Case 2), P_0 participates in the mixing and commit to a group signature with $acc_0^{'}$. Since the group signature scheme is correct, anonymous, traceable, and non-frameable, this group signature is either correct, hence traceable; or incorrect, in which case honest users will proceed as before and ignore sidechain transactions $acc_0^{'}$.

Moreover, the commit signature phase does not leak any information about the signing entity which completes both the demonstration for secrecy (R5) and anonymity (R7). □

6.3 Implementation and Evaluation

In this section, we discuss technical implementation choices for our protocol, and present our experiment that illustrates the benefits of sidechains.

6.3.1 Choice of the cryptographic tools

For the implementation, we selected the following instances of the cryptographic tools described in sub-section 6.1.1. The client application Client.app is in Python, HTML and JavaScript. The group $\mathbb{G}_q$ of order q is the optimized Barreto-Naehrig curve over a 128 bit prime field. We selected the optimized_bn128 function from the py_ecc library developed by Ethereum's community. The collision-resistant hash function $H(\cdot)$ is Keccak-256. The group signature scheme $\mathcal{GS}$ is Bellare, Shi and Zhang (BSZ) scheme proven to be anonymous, traceable and non-frameable (under the existence of a trapdoor permutation) [34]. AES is used for the symmetric encryption $\mathcal{SE}$ =

$(\texttt{SEnc}, \texttt{SDec})$.The Python implementation of the NIST Curve P-256 which provides 128-bits of security (library PyCryptodome) stands for the asymmetric $\mathcal{AE} = (\texttt{AEnc}, \texttt{ADec})$ ECC-based Encryption scheme. Finally, for our pair (P, V) of simulation-sound NIZK proof system we re-use the proving and verifying functions described in [38] respectively as $\texttt{DLEQ}(\cdot, \alpha)$ and $\texttt{DLEQ-verify}(\cdot, \pi)$ where α is the secret to prove and π the proof to verify. The instantiation of the proving function is DLEQ and the proof $\pi(k_{ij})$ is $\texttt{DLEQ}(g, tpk_j, tpk_i, k_{ij}, tsk_j)$; the verifying function is denoted DLEQ-verify and the verification consists of $\texttt{DLEQ-verify}(g, tpk_j, tpk_i, k_{ij}, k_{ij})$. Finally, we decided to use Bitcoin and RSK for the sidechain architecture. The mixing technique follows a CoinJoin protocol ??; and the KeyEstablishment routine is a smart contract developed in Solidity.

6.3.2 Why did we choose Bitcoin and RootStock?

In 2009, Satoshi Nakamoto introduces Bitcoin [13] to the world. While revolutionising payments online, Bitcoin remains limited due to the lack of integration of smart contracts. Since then, several cryptocurrencies have been launched with stateful virtual machines capable of executing Turing complete programming languages, hence smart contracts. RSK is presented as the safest smart contract blockchain as it is secured by the Bitcoin network. RSK is a platform that enhance Bitcoin by enabling smart contract development. It also inherits some other Ethereum characteristics: account format, virtual machine and web interface. RSK provides a way to create Bitcoin sidechains. RSK is based on proof-of-work and, as of mid-2021, RSK has 46.6% of Bitcoin's hash rate, meaning that almost half of the mainchain mining power also mines the sidechain (RSK). To transfer Bitcoins to and from the RSK sidechain, the technology implements a two-pay peg. When Bitcoins get transferred into the RSK blockchain, they become “Smart Bitcoins" (RBTC). Smart bitcoins are equivalent 1:1 to bitcoins but live in the RSK network. They can be transferred back into Bitcoins at any time.

We chose RSK because it re-uses most of the infrastructure software developed for Ethereum. Therefore, it facilitates the migration of Ethereum applications to RSK. In addition, RSK increases the transaction throughput of the Bitcoin blockchain from 7 transactions per second (tps) to about 300 tps. This higher number of transactions per second helps reduce the transaction cost. As a conclusion, we selected RSK sidechain environment for the development and implementation of our protocol for its promises of scalability both in terms of transaction throughput and execution cost.

However, since Bitcoin and RSK are two public chains, we still have our pending issue of anonymity.

6.3.3 CoinJoin and CoinShuffle

Bitcoin is not anonymous [192], it can be de-anonymized via transaction graph analysis. To prevent this unwanted tracing, mixing techniques were developed. The underlying idea for mixing is to obfuscate the relationship between inputs and outputs, thereby preserving the relationship anonymity. This concept was introduced as the CoinJoin technique [193]. It is based on the fact that Bitcoin is not an account-based blockchain (unlike Ethereum). Instead, Bitcoin leverages Unspent Transaction Outputs (UTXOs). Each UTXO can be seen as a coin, and holds a certain amount of value in its respective currency. Each UTXO represents a chain of ownership. Moreover, Bitcoin transactions can have multiple inputs and outputs. The CoinJoin technique leverages both properties by combining multiple UTXOs from different users into a single transaction. That is, each participant chooses one UTXO to spend (input) and an output address to which send the coins (output). Therefore, the final transaction has a number of pairs of input/output UTXOs. From an external observer, it is difficult to determine which input corresponds to which output. CoinShuffle [194] is an example of generic decentralized mixing protocols. In this section, we propose to reuse this protocol for mixing participants' coins. CoinShuffle is a decentralized protocol that allows users to mix their coins with those of other interested users. The CoinShuffle protocol is proven to provide unlinkability, verifiability, robustness, and double-spending resistance.

6.3.4 Interfacing Bitcoin, RSK and CoinJoin

Firstly, P_0 initiates the protocol by selecting the participants $\mathcal{P} = \{P_1, \ldots, P_n\}$ ①. It deploys the KeyEstablishment smart contract on the sidechain and uses the `groupCreation` function to transmit the necessary information to the P_i. Upon receiving the Group Creation event and corresponding data, the P_i's who wish to participate in the DKG execute the CoinShuffle protocol to mixing their coins ②. They obtain new anonymized addresses $acc_i^{'}$ that contain some bitcoins. To authenticate these new addresses, they individually publish with $acc_i^{'}$ a group signature generated with their original identity ③. Then, they locks these tokens on the mainchain. The Federation, a set of entities in charge of the locking and unlocking of coins, ensures that the bitcoins stay locked, and unlocks the corresponding amount of smart bitcoins (RBTC) on the same accounts. Let us note that the members of the RSK are publicly known. At the end of this first phase, the contacted P_i's have all the information to continue the execution of our protocol, they are completely anonymous from the system point of view, yet they are traceable thank to the committed group signatures.

Then, they engage in the DKG protocol and follows the steps described in Section 5.3.1. Calls to the routine are only replaced by calls to the KeyEstablishment contract ④ ⑤.

Once the DKG is finished, every selected participant, including P_0 can derive the common secret key gek ⑥, and terminate this instance of the KeyEstablishment smart contract ⑦. Eventually, they can beforehand issue an opening request ⑧. If so, the Opening authority will take the group signature associated to the address against which the request has been issued and open it. Thus, this will de-anonymize the misbehaving or faulty participant.

6.3.5 Evaluation in terms of gas

Our blockchain-based DKG protocol provides two additional features compared to existing literature, namely anonymity of the participants and traceability of their actions (Chapter 3 and Section 6.2). In this subsection, we show that the protocol is also *relatively cheap* (gas cost).

Table 6.1 illustrates the savings that can be made by migrating the DKG contract (denoted KeyEstablishement.sol) from Ethereum to RSK. For deployment only, we observe a saving of almost 91 USD. This is due to the difference in the gas price. AS of September 2021, 1 gas costs approximately 20 gwei on the Ethereum network[1]. Instead, on RSK, it costs 0.072 gwei[2]. We extrapolate this cost difference in Tables 6.1 and 6.2. In the tables, the ETH gas price is equal to 20 gwei; and the RSK gas price is of 0.072 gwei.

Table 6.1: Cost of deploying KeyEstablishment.sol contract on Ethereum and RootStock

Operation	Ethereum testnet	RSK testnet
Deployment - Gas used	1,713,516	2,101,072
Gas Price (gwei)	20	0.072
Deployment - Final cost (ETH)	0.038	0.0017
Deployment - Final cost (USD)	**91.0488**	**0.3978**

Let us note that it is not clear why we observe a difference in the gas consumption when deploying the smart contract on RSK. One possible guess is that RSK must handle the sidechains in addition to the deployment.

Knowing the gas consumed by a function in our Ethereum-based implementation, we can estimate its execution cost in USD w.r.t. the gas price (in gwei) in Ethereum and RSK, and Ethereum's current price. We observe that the expected cost of execution on Ethereum is multiplied by a factor 273 compared to what a user could pay by using RSK. This colossal difference transforms a scheme somehow usable in specific settings where anonymity and traceability are two critical features worth the price; to a scheme that can be used in a daily basis.

Table 6.2: Cost of executing our protocol on the Ethereum testnet vs. the expected cost on RSK testnet (in USD)

Operation	Gas Used	Cost ETH testnet	Expected Cost RSK testnet
Group Creation	1,437,090	68.4	**0.25**
Share Transmission	66,729	27	**0.0114**
Tpk Publication	27,646	1.3	**0.0047**
Group Key Publication	53,488	2.5	**0.0092**

[1] https://etherscan.io/gastracker

[2] https://rskgasstation.info/

Conclusion In this section, we propose a Blockchain-supported Anonymous-yet-Traceable DKG protocol. Our protocol can be used to securely and distributively generate a common dlog-based encryption/decryption keypair (cryptosystem); or a group signing/verifying keypair (threshold signature). This is particularly useful in anonymous trading or in coercion-resistant e-voting systems. In such applications, participants are untrustworthy and have conflicting interests. They want to be anonymous, and keep their contribution secret. Yet, the system needs to be auditable hence, traceable and transparent. With the introduced protocol, we extend existing blockchain-based implementations of DKG protocols with the anonymity and traceability properties required in such applications. We prove that our new scheme is correct, robust and may achieve uniform distribution under the Discrete Logarithm assumption and the existence of a Collision-Resistant Hash Function. Moreover, we showed that existing implementations can greatly gain from using sidechains. Hence, we were able to obtain a reduced execution cost of our protocol, on RSK, by 99% compared to a pure Ethereum-based implementation.

Part III

Application to ITSs, conclusion and perspectives

7

Towards a complete Blockchain-enabled Privacy-enhanced Traffic Reporting system for ITSs

“The devil is in the details."

— Gustave Flaubert

This chapter completes our analysis of the challenge of providing auditability while preserving users’ privacy in a vehicular network environment. More specifically, we discuss the design of a Blockchain-based Privacy-enhanced Traffic Reporting system that offers Public Auditability in the context of ITS. Road safety is the thriving goal when designing ITS. It is currently enforced by sharing critical information related to the positioning of vehicles (*e.g.,* the CAMs) and hazard warnings (*e.g.,* the DENMs). This chapter focuses on the sending of DENMs and their impact on vehicular users' privacy. We investigate ways to keep users' identities and location private while providing a strong auditing system for further investigation a posteriori by a distributed public authority. The proposed system leverages the key features of proposed blockchain-based cryptographic primitives presented in Chapters 4, 5 and 6. In this chapter, we present our final Blockchain-enabled Privacy-enhanced Traffic Reporting system for ITSs.

7.1 Summary of the thesis context

In this first section, we recall the context and summarize the challenges related to securing the broadcast of DENMs. We list the security requirements an authentication scheme in ITSs must comply with. We will use this authentication scheme afterwards (Section 7.2) for the construction of the Traffic Reporting protocol.

7.1.1 About the DENMs

DENMs are the messages that enable traffic hazard warnings. Unlike CAMs, used for the vehicle to vehicle positioning, DENMs are sent to alert the network about specific hazardous events, *e.g.,* a car accident. There are defined as

> “facilities layer message[s] that [are] mainly used by ITS applications to alert road users of a detected event using ITS communication technologies"
> [6].

The construction of the DENM is triggered by an ITS-S application upon detection of a road hazard or abnormal traffic conditions. The DENM is transferred to other relevant (*e.g.,* in the specified geographical area) ITS-Ss through the DEN basic service. This transmission is performed via V2V or V2I communications. Upon reception of a DENM, the service processes the message and forwards its content to the ITS-S application. The application, in turn, displays it to the user through the HMI modules.

DENM generation. The generation and propagation of DENMs are described in the ETSI standard specifications [6]. When it comes to communications, ETSI standards identify vehicles and RSUs as ITS-S, which embed several ITS Applications as specified in [9]. The exchange of these messages is operated through the DENM protocol, which consists of the following steps:

- Upon event detection, the station broadcasts a DENM to disseminate the information about the hazard to other stations in a specific area. This first ITS-S is called the originating ITS-S.
- Then, the message is relayed by forwarding ITS-Ss.
- The transmission of a DENM may be repeated and persist as long as the event is present.
- The termination of a DENM can occur automatically(*e.g.,* if an expiry time has been specified) or upon request (*e.g.,* the event has terminated).
- The ITS-S receiving a DENM processes the information and may decide to discard the warning or inform the user.

Privacy risks. By design, and to prevent the propagation of fake information, the DENMs (of which the structure is shown in Figure 2.6) contain information about:

- the identity of the alerting/source node that first detected the event, in `StationID`;
- the location of the alerting/source node in the ⟨event position⟩ field of the DENM.

Therefore, the broadcast of DENMs can breach users' identity and their location privacy either directly (*e.g.,* via the `StationID` field) or indirectly (*e.g.,* by analysis of the location traces). Growing efforts towards improving the safety of vehicular networks are focusing on securing the wireless communications and data transferred between vehicle and infrastructure nodes to communicate safety-critical information.

Motivation. The core *motivation* of our research, in blockchain-based anonymous-yet-traceable distributed cryptographic primitives, is the proposition of a reporting protocol that guarantees:

1. quick user response to safety events (whether they are true or not);
2. secure and privacy-preserving sending of the warning messages;
3. accurate and effective auditing method a posteriori.

7.1.2 DENMs security according to ETSI

Several approaches attempt to provide privacy in ITSs [195]. The easiest strategy consists in implementing a *fixed pseudonym change parameter*. The parameter can be time (*e.g.,* change pseudonym every 5 minutes), the number of signed messages (*e.g.,* change pseudonym every 100 messages), or even distance (*e.g.,* change pseudonym every 500 meters travelled) [196]. However, its simplicity makes the method vulnerable to an eavesdropping attacker who can quickly infer the fixed parameter.

In order to cope with the monotony of parameter changes, *randomness* can be inserted. The pseudonym is changed *w.r.t.* the same parameter after adding a random value. The addition of a random factor contributes to preventing the attacker from inferring the pseudonym change periodicity. However, the attacker can combine the observations of these changes with vehicular mobility patterns and still detect which pseudonym changed into which other one.

To tackle the later issue, vehicles can implement *silent periods* [197]. These are periods of time, after a pseudonym change, during which they do not communicate. The advantage lies in that tracking becomes more difficult if the pseudonym changes occur over a large group of vehicles in a location where trajectory predictability is reduced (*e.g.,* cross-roads). However, it comes with the drawback that vehicles concerned by the pseudonym change cannot send safety messages during the silent period.

The *vehicle-centric strategy* regroups all three previous methods at the vehicle level. It supposes that the vehicle changes pseudonym at random periodicity and applies silent period *w.r.t.* their mobility pattern. This strategy addresses some of the issues above. However, it may reduce the actualization rate of the pseudonyms and hinder the efforts towards privacy preservation.

A mitigation measure will trigger pseudonym change if the environment is dense enough, *i.e.* in the presence of a sufficiently large number of surrounding vehicles. This idea was elaborated in the *mix-zones-based strategy* [72]. A mix-zone is a delimited geographical area where no location-aware applications are running, *i.e.* no CAM nor DENM broadcast. This creates an area in space and time during which the pseudonyms are mixed s.t. it becomes difficult for the eavesdropping attacker to determine the new mapping. There are three main ways to propose a mix-zone. The first method is to involve an RSU. However, it necessitates trusting the RSU, which learns the new mapping.

Another method is to rely on *collaborative change*. It relies on broadcasting messages to surrounding vehicles advertising their readiness to change pseudonyms. Once they all agree, no vehicle sends location change messages until all have a new pseudonym. This method is highly robust against an eavesdropping attacker as it provides k-anonymity (the new pseudonym belongs to one out of the k nodes that participated in the pseudonym change). However, it requires synchrony and is not efficient in low-density networks. The previous strategy can be combined with pseudonym swaps and thus, tackles both issues. Two vehicles sharing the same mobility pattern can swap their pseudonyms.

Combining collaborative mix-zones with random pseudonym swaps may significantly increase privacy. However, it poses the question of traceability. Indeed, as it becomes complicated to reveal the link between pseudonyms and real identities, how does the system ensure the accountability of the road users and enforce traffic laws? Moreover, it brings the risks of a Sybil attack, in which a malicious entity tries to use multiple pseudonym identities to simulate various (fake) ITS stations. By doing so, the adversary can conduct other types of attacks, including DoS attacks or modification attacks.

7.1.3 Other cryptographic approaches to secure vehicular communications

Pseudonym change is the method privileged by ETSI. However, there are other strategies to propose privacy-preserving authentication as suggested by Manvi *et al.*'s survey [198].

For instance, *asymmetric cryptography*, also called Public Key Cryptography, it is a trendy method to provide data security (via encryption) in most communication networks. It leverages a pair of keys to enable a receiver to check the authenticity of a message with the public key without knowing the private key; it refers to the digital signatures. In *PKI-based authentication* schemes, users run the asymmetric cryptography algorithm and rely on a TTP (the Certificate Authority (CA)) for the management of certificates and the traceability feature. The CA is often different from the RSUs, which are rarely entirely trusted. The idea is that before broadcasting the safety message, the user will sign it with its private key and append its certificate (additional information are also present, such as a timestamp). On the receiving side, the node “only" has to check the CRL, maintained by the CA, to accept or reject the incoming message. These schemes are widely adopted in ITSs, but they still have some limitations according to [198]: subject to flooding and location tracking attacks and introduction of high computation and communication overheads. In [199], authors propose one attempt to improve the latter drawback. They replaced the CRL checking process with **keyed HMACs** which is said to reduce the message loss and the computational and memory footprint (CRLs can be considerable). Other methods favor ECC-based asymmetric schemes. The advantage of ECC over traditional algorithms is that for a smaller key size (160 bits against 1024), it achieves the same or better communication security. Therefore, ECC-based authentication algorithms are lighter, faster and require less memory space; consequently, they also introduce less computation overhead. However, compared to Public Key Infrastructure (PKI)-based authentication algorithms, they are more complex systems and introduce higher communication overheads. An other alternative is known as ID-based cryptography. This method consists in deriving the public keys of a user from their identity information, such as email address. It does not rely on certificates for message authentication, but pseudo-identities hence reducing communication overhead and does no longer require the maintenance of a CRL, thus improving the computation overhead. **Single-User Signature** is the initial proposition that considers ID-based cryptography for authentication scheme. However, existing schemes fail to provide privacy preservation, efficient key management and usually introduce high computation and communication overheads.

Symmetric cryptography is also an option as it is faster and simpler than asymmetric cryptography. The elementary primitive of symmetric cryptography is certainly the hash functions. They are essentially used to check message integrity, mainly one-way hash functions, as they provide unique output values. They are used in various declination as building blocks for more complex and complete systems. For instance, they are prevalent in Message Authentication Code (MAC) algorithms. These algorithms are famous because they require less computation overhead compared to PKI-based and Elliptic Curve Digital Signature Algorithm (ECDSA)-based authentication schemes. They are known as keyed hash functions. They rely on a shared secret key and a hashing function that generates a message authentication code (the tag) from the key and the message. The receiver can check the tag as it knows the secret key. MAC ensures both message integrity and authenticity but, alone, it does not provide non-repudiation as the sender and receiver use the same private key. Yet, combined with a digital signature, the method can provide non-repudiation. While being faster and simpler, symmetric primitives require the sender and receiver to share a common secret knowledge (a key). Moreover, there are other severe drawbacks including: the absence of the non-repudiation feature by design, weak security, *i.e.* “easy" loss of confidentiality in case of key corruption.

Group Signatures have been developed to address the aforementioned drawbacks. Their anonymity property enables the design of anonymous authentication schemes. Other important properties include the prevention of ID disclosure, vehicle non-traceability, unlinkability and unforgeability. However, it introduces high computation overhead during the verification of signatures and increases the message loss ratio. Later works attempt to shorten group signatures [116]. Cryptographic tools are mainly used to provide data security (*e.g.,* through encryption). Instead, signature schemes are implemented to ensure source authentication. They ensure node authentication, message integrity and non-repudiation of safety messages. However, traditional signature schemes are not compliant with recent efforts toward privacy preservation. As part of these efforts, the design of **anonymous authentication schemes** has been recently prevalent in ITSs.

Anonymous authentication schemes for VANETs. In lights of the literature, it appears that a practical anonymous authentication scheme for ITSs should comply with the following properties:

1. **source authentication**, algorithm run by each receiver;
2. **data integrity**, the checks are performed by each receiver;
3. **time synchronization**, for information freshness evaluation;
4. **communication overhead**, which should be as low as possible in a safety-critical environment such as ITSs;
5. **computation overhead**, both at sending and receiving sides, and especially for the verification of the DENMs validity;
6. **privacy-preservation**, to protect the anonymity of the users;
7. **accountability** mechanism to ensure they can be held accountable for their actions.

7.1.4 Group-signature based authentication schemes

In the family of anonymous authentication schemes, **group-signature-based authentication schemes** have been increasingly popular in ITSs (Chapter 2). In Table 7.1, we list the most important contributions in the field. These primitives are particularly interesting because they provide users' anonymity, yet, they also implement a traceability mechanism for accountability.

In [200], Zhang *et al.* propose a decentralized group-authentication protocol based on group signatures and signcryption (the later concept is a method that provides the properties of both digital signatures and encryption schemes in a way that is more efficient than signing and encrypting separately [201]). The underlying idea is to cope with certificate distribution and revocation challenges, avoidance of computation and communication bottlenecks, and reduction of reliance on tamper-proof devices. The decentralization comes from considering that each RSU maintains and manages an on-the-fly group of vehicles evolving in its range. The Opener authority is played by an external TTP and is called whenever a message seems fraudulent. The scheme is designed to apply the signature to an ITS message. It defines the safety message as a concatenation of a group ID, the payload (*i.e.* the actual ITS message as, for instance, a DENM), a timestamp and the signature. The total length of the new safety message structure is 474 bytes (including a payload of 100 bytes and a signature of 368 bytes). The signature scheme employed is defined in [202]. Among other parameters, the simulation presented shows that the scheme achieves a density-constant message delay that grows linearly with the time to batch verify the signatures.

In [203], Zhu *et al.* follow the same idea. They leverage the advantages of a group signature scheme and consider the RSUs as group managers. It facilitates the distribution of the secret keys and reduces the communication overhead. In addition, they use Hash-based Message Authentication Code (HMAC) to avoid the time-consuming step of verifying the revoked certificates and propose a cooperative message authentication mechanism to reduce the number of messages a vehicle must verify. Once again, the signature scheme is applied to the safety payload of CAM only. The structure of the messages broadcast by the vehicles for vehicle-to-vehicle positioning consists of a group ID, message ID, timestamp, location, signature and HMAC. The signature is 56-byte-long, and the size of the final message is 72 bytes. These beacons are broadcast every 300ms. The simulation shows that their average message loss ratio is null no matter the number of OBUs in the communication range. Similarly, the message verification

Ref	Year	Title	Authors
[200]	2009	A scalable robust authentication protocol for secure vehicular communications	Zhang *et al.*
[203]	2013	Privacy-preserving authentication based on group signature for VANETs	Zhu *et al.*
[204]	2015	A threshold anonymous authentication protocol for VANETs	Shao *et al.*

Table 7.1: References in Group-Signature-based Authentication Schemes for ITSs.

time is constant independently of the number of entities in the CRL and better than existing schemes when it contains more than six identities.

In [204], Shao *et al.* also work in the decentralized group model that considers an ITS as separate groups, each one controlled by an RSU. In this configuration, they too develop a group signature-based authentication scheme for ITSs. However, the novelty lies in the threshold authentication of messages. Instead of verifying all the messages, the vehicles may equally accept messages that have been checked by a *threshold* number of peers. They adopt the same technique as the two previous works and apply the signature scheme to the payload. Therefore, the broadcast message consists of a message identifier (shorten as ID), a payload (100 bytes), a timestamp, a Time To Live (TTL), a group ID, and a signature of 826 bytes. The size of the messages is 935 bytes. The group signature scheme leverages bilinear pairings, which also allow for batch verification. Their simulation shows that the verification of the messages costs approximately 1.2 seconds. Additionally, this time grows with the number of signatures to verify (with or without batching).

Our contribution. Similarly to the listed references, we focus on designing a group signature-based anonymous-yet-traceable authentication scheme to build a new Traffic Reporting protocol that is privacy-preserving, accountable, censorship- and coercion-resistant. However, instead of augmenting an ITS message (*e.g.,* CAM or DENM) with a group signature for authentication purposes, we propose to redefine the DENM structure to embed privacy security by design. Indeed, in the presented propositions, the ITS message is always considered as the payload of a new authenticated message. However, we will explain, in the following section, that this way of implementing anonymity brings redundant information and does not fulfill its goal of privacy-preservation.

7.2 Terminology and System Overview

In this thesis, we are particularly interested in how vehicular communications, and especially the new-DEN messaging pattern, provide safety to road users while impacting their privacy. We work on designing a new-DEN messaging protocol that respects the aforementioned security requirements. In the following section, we recall the definition of our system model, the nodes and networks involved in the propagation of the DENMs. We present the modified version of the DENM structure and all related P2P exchanges that support the designing privacy-preserving yet accountable Traffic Reporting system.

7.2.1 Nodes

The vehicular network architecture presented in Fig. 7.5 is made of three main components defined as follows:

- **The Vehicle nodes (VNs)** simplified vehicles. The vehicles are the users of the network. They witness road events and produce hazard **warning messages**, under the form of DENMs, to be broadcast to their peers for safety reasons. The vehicles are mobile. As such, we may assume that they have limited communication, computational and storage resources.

- **The Infrastructure nodes (INs)** also called RSUs. The RSUs are similar to vehicles in that they participate in hazarding warnings, and the difference is that they are static nodes. Therefore, we can assume that their resources are more important than the vehicles' and that additional backup mechanisms are implemented to increase their availability and make their logs resilient to data loss and component failures.

- **The Administration nodes (ANs).** Lastly, the Administration nodes perform the deployment of the different blockchains, later referred to as *sidechains*, before runtime and ensure the auditing process. It is the authority entitled to query the network regarding one specific event and request that sensitive data be released for further investigations.

7.2.2 Networks

As illustrated in Figure 5.2(a) (Chapter 2), we define two distinct networks: the vehicular network and the blockchain network as follows:

- the **vehicular network** is defined by the combination of vehicles and RSUs (Fig. 7.1). In the vehicular network, ITS nodes essentially broadcast data either in plaintext (such as the DENMs) or encrypted (*e.g.,* the join requests).

- the **blockchain network** is layered on top of the vehicular network. It is used to archive digital events in a privacy-preserving way. The blockchain network has the same size as the vehicular network: each vehicular node is associated with a node in the blockchain environment. The RSUs will play full nodes, and the vehicles will embed light nodes. The ledger that records the warning message hash values is the same for all the nodes. Everyone has access to the data stored inside.

7.2.3 Context

The context is presented on Figure 7.1 and goes as follows:

1. The vehicle A witnesses a hazardous event e_0 on the road (materialized by the “caution" panel) at location L_0 and time T_0.

2. A processes this information and the environmental data captured by its sensors and produces the **warning message** *denm*. In parallel, it computes the corresponding **transaction** $tx = \langle \text{seq no.}, denm\_ch, \sigma \rangle$, where seq no. is the sequence number associated to the *denm*, *denm_ch* is an integrity check of the warning message and σ authentication metadata corresponding to vehicle A.

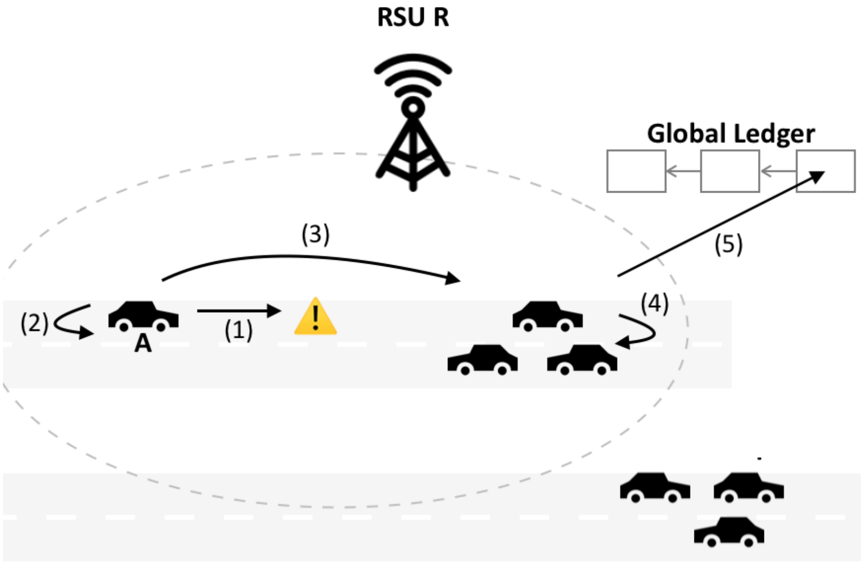
3. Then, vehicle A broadcasts $\langle denm, tx \rangle$ to its surrounding peers.

4. Each user processes the **warning message** and the **transaction**, taking actions in the real world if needed (*e.g.,* avoid the location where the hazard has been declared).

5. Finally, the *denm* is saved inside a distributed database (*e.g.,* InterPlanetary File System (IPFS), Cloud storage), and the transaction is logged inside a *tamper-proof immutable ledger*.

For the following discussion, we fix $H(\cdot)$ a CRHF and assume that a blockchain is already deployed. Therefore, the value $denm\_ch = H(denm)$ is unique (probabilistically), and σ is the signature computed with the blockchain algorithm and secret data from vehicle A.

FIGURE 7.1: Illustration of the context of the study describing one node reporting a hazard to its peers and resulting in the archiving of the event.



7.2.4 Structure of the DEN messages

Figure 7.2 illustrates the frame of a DENM as standardized by ETSI in [2]. It is divided into 16 blocks of different sizes to result in a at least 40-byte-long message.

The *header* container is 8-byte-long and contains the protocol version, *i.e.* the current version of the protocol used by the *Decentralized Situation Management* container. It also contains a message ID associated to each DENM, and a generation time, namely a timestamp. The management container is 14-byte-long. It specifies the identifier of the ITS station that broadcast the DENM (in 'origin ID'), and a sequence number, denoted 'seq no.', unique to the event being reported. The field 'data version' indicates an update of the situation (*e.g.,* 255 is for cancellation of the event). The 'expiry time' sets a timestamp after which the event is obsolete. The 'frequency' value defines the transmission frequency of the DENM, and the 'reliability' represents the probability for the event information to be true. Lastly, the boolean 'isNegation' confirms or not the existence of the event. In the *Decentralized Situation* container, we find three bytes related to the situation itself. More specifically, the 'CauseCode' identifies the event direct cause according to a predetermined table of referenced values; the 'SubCauseCode' provides additional information; and the 'severity' value evaluates the seriousness of the event. Finally, the *Decentralized Situation Location* container provides geographical information about the event, including its latitude with 'RefPosition_SituationLat', its longitude through 'RefPosition_SituationLon' and altitude via 'RefPosition_SituationAlt'. It also precises the accuracy of the position in the field 'accuracy' and can provide more details in the 'Other DEs and DFs' field of variable size (DE: Data Element, DF: Data Frame).

As mentioned in the previous section, the ITS messages already contain the necessary information of location, timestamp, and message and event unique identifiers. In addition, the payload contains identifying information in the `StationID` field. We argue that applying an anonymous authentication scheme on top the current structure of DENMs will not have the desired effect to protect the anonymity of the users. Instead, we suggest to modify the structure of the DENM itself by replacing the `StationID` field by a group signature. A modified version of the DENM structure is therefore proposed in Figure 7.3.

In order to support this slight modification, we developed a whole blockchain-based framework. The idea is to provide a way to securely bootstrap the group signature

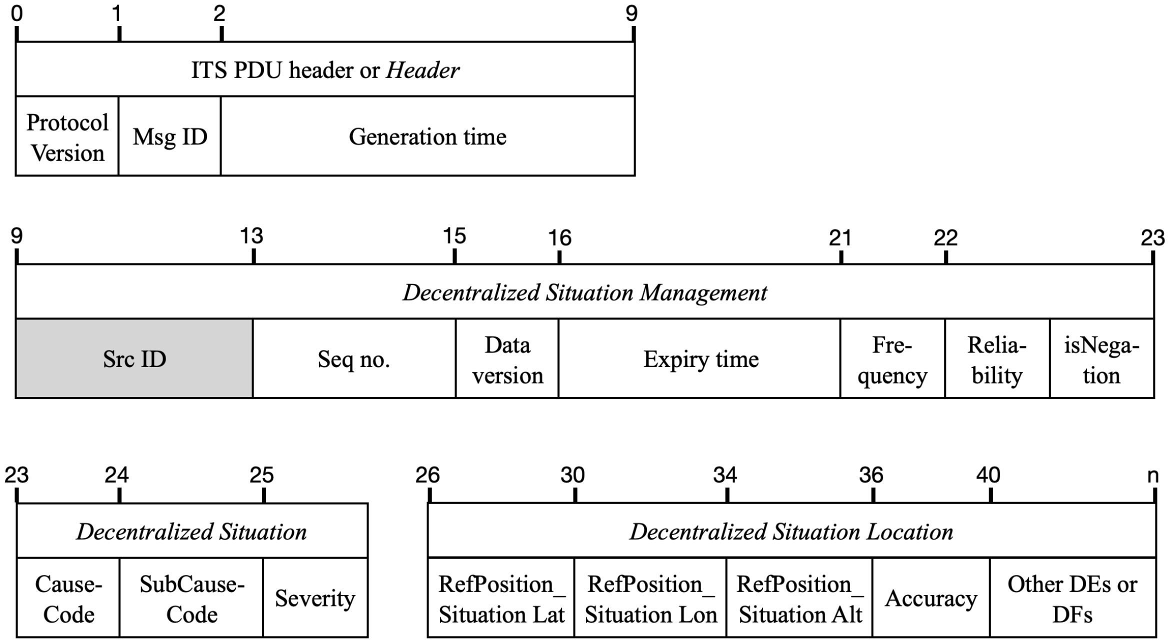

0
1
2
9
ITS PDU header or *Header*
Protocol Version
Msg ID
Generation time
9
13
15
16
21
22
23
Decentralized Situation Management
Src ID
Seq no.
Data version
Expiry time
Fre-quency
Relia-bility
isNega-tion
23
24
25
Decentralized Situation
Cause-Code
SubCause-Code
Severity
26
30
34
36
40
n
Decentralized Situation Location
RefPosition_ Situation Lat
RefPosition_ Situation Lon
RefPosition_ Situation Alt
Accuracy
Other DEs or DFs

Figure 7.2: Content and format of a DENM as specified by ETSI in [2]

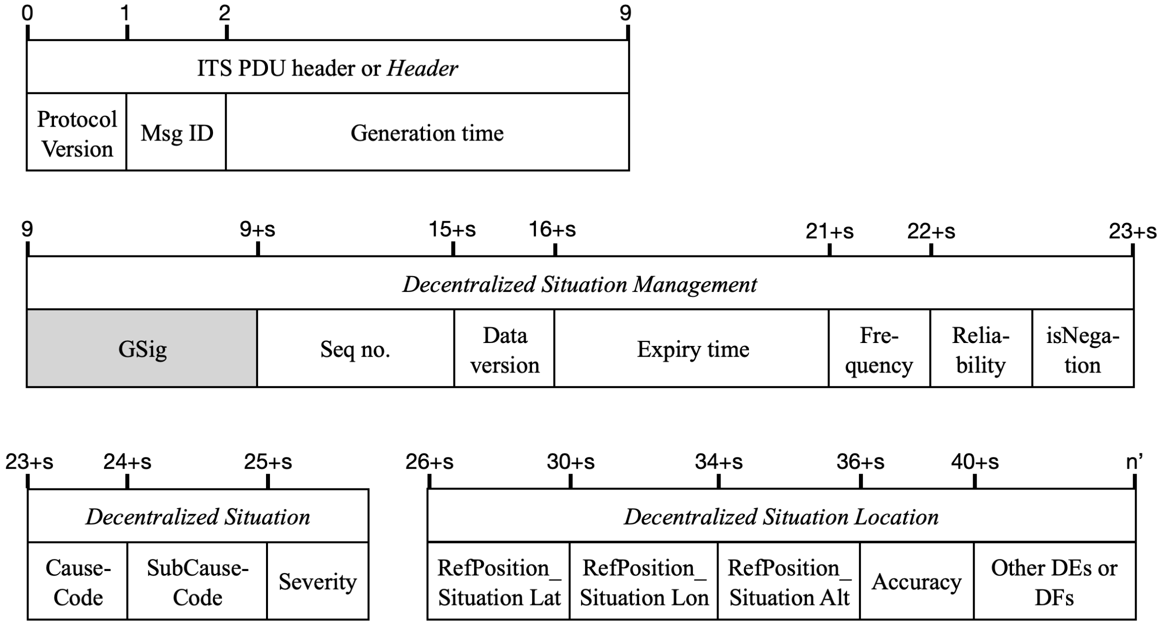


Figure 7.3: Content and format of the proposed new DENM

infrastructure (Opening and Issuing authorities) such as to define an anonymous-yet-traceable group signature-based authentication scheme. The new primitive, which relies on blockchains and distributed cryptography, enables the design of the new-DEN messaging protocol and the definition of the privacy-preserving, accountable, censorship- and coercion-resistant Traffic Reporting system.

7.2.5 The designed sidechain-based logging functionality

The sidechain and contracts. In our system, the RSUs and the vehicles all maintain a shared and synchronized database that is distributed across the vehicular network, namely a blockchain. More specifically, we define *one mainchain and one sidechain* (Figure 7.4). The mainchain is called the Coordination chain. It supports UTXO scripting and mixing services to anonymize the blockchain accounts, and enables the execution of the $\mathcal{BAT}-Key$ protocol. The sidechain is called the VANET chain. It supports smart contract programming and can support the deployment of five smart contracts:

- the Register contract: any RSU or vehicle that want to become a blockchain node and subsequently be called to play a sub-Opener node or a sub-Issuer node, can register via this smart contract.
- the DOGS contract: it focuses on the generation of the distributed Opening authority and logs all the events emitted during $\mathcal{DOGS}$ protocol.
- the TOAD contract: which enables the tracing of all events related to the setup of the Issuing Authority or the issuance of group members' certificates.
- the Reporting contract: it is the contract by which participants can issue and log the transactions associated with the broadcast of DENMs.
- Finally, the Auditing contract: which tracks all audit requests and subsequently releases the requested information.

We will show in the following section how the broadcast information is securely and accurately stored inside the corresponding ledger using cryptographic primitives such as hash functions (*e.g.,* for data integrity check, misbehaviour prevention) and digital signatures (*e.g.,* for access control and authentication).

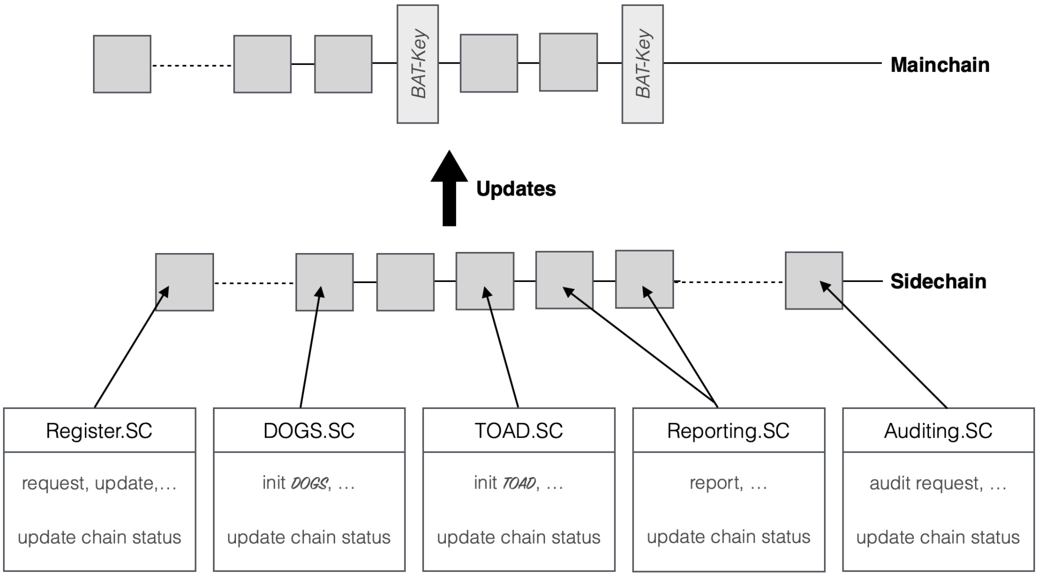


Figure 7.4: Detailed overview of sidechain-based blockchain architecture

Interacting with the chains. When it comes to the blockchain layer, we consider that each ITS node has two local stacks at its disposal for managing and prioritizing incoming transactions: the **block** and the **pending queue**. The block is defined by its size s_{block} and consists of a list of warning messages $list_{block} = \{denm_i^{block}\}_i$ of sizes $\{s_i^{block}\}$. Let's note $denm_{new}$ the last incoming warning message.

- As long as $s_{block}(= \sum_{i=0}^{n-1} s_i^{block}) + s_{denm_{new}} \leqslant s_{block}^{max}$, $denm_{new}$ is appended to $list_{block}$.
- If $s_{block} + s_{denm_{new}} \geqslant s_{block}^{max}$, then $denm_{new}$ is added to the pending queue and the miner starts the mining of the block. The messages in the pending queue are not discarded, they will be treated in the next block.

7.3 Description of the framework

In this section, we propose a description of the final blockchain-based cryptographic framework that will enable the new-DEN messaging protocol while enforcing the targeted security requirements.

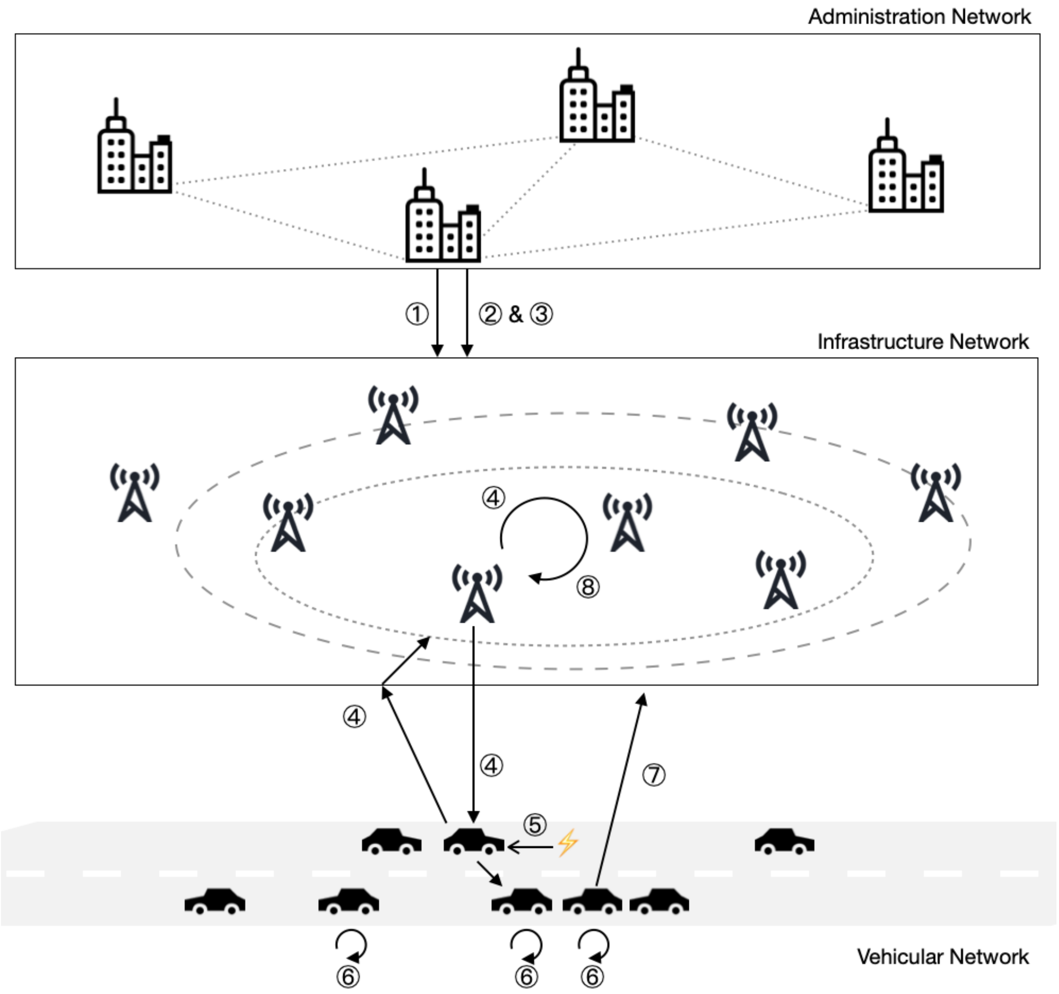


Figure 7.5: A top-down description of the proposed framework

7.3.1 Primitives

The framework is organized around the presented contributions. Namely, it leverages the $\mathcal{DOGS}$, $\mathcal{TOAD}$ and $\mathcal{BAT}-Key$ protocols as presented respectively in Chapters 4, 5 and 6.

$\mathcal{DOGS}$

A group signature scheme usually relies on the presence of a group manager [32]. It can also be implemented with a duo of authorities, the Issuer and the Opener [34]. These three roles are not compliant with the spirit of Blockchain and decentralization. In Chapter 4, we presented a more decentralized group signature scheme, named

$\mathcal{DOGS}$ [186]. The chapter introduces a group signature scheme with a distributed opening functionality. Unlike previous works that implement a single Opener authority, $\mathcal{DOGS}$ leverages an Ethereum-based blockchain to support the execution of a (traceable) DKG protocol, and therefore, the setup of a distributed opening authority. The full description of the $\mathcal{DOGS}$ protocol is detailed in previous chapters.

$\mathcal{TOAD}$

Threshold cryptosystems are particularly useful in settings where the centralization of the power to decrypt is a concern. In such cryptosystems, there is a single public encryption key; however, the corresponding private key is shared among a set of users, also called the decryption servers, in such a way that the collaboration of a certain number of them is required to decrypt a message. Threshold encryption schemes often rely on a TTP for the sharing of the decryption key shares. Aligning on previous comments regarding decentralization, we described a more decentralized threshold encryption scheme named $\mathcal{TOAD}$ in Chapter 5. $\mathcal{TOAD}$ implements a mechanism for a sender to encrypt a file for a set of receivers that can be recovered only if at least t receivers participate in the decryption process. The full description of the $\mathcal{TOAD}$ protocol is detailed in previous chapters.

$\mathcal{BAT} - Key$

Many threshold systems require some kind of secret sharing during their setup phase. Often, threshold cryptography papers do not develop the details of how such secret sharing is performed. There have been several recent works in doing secret sharing in a distributed manner through Blockchain so that one does not have to rely on trusted third parties or strong assumptions about the communication medium. In Chapter 6, we re-use the protocol developed in [205] that continues this line of work and presents a new protocol for DKG. Instead of using the mainchain of a popular blockchain platform like Bitcoin, we opted for a sidechain called RootStock, which is pegged to the Bitcoin mainnet. The full description of the $\mathcal{BAT} - Key$ protocol is detailed in previous chapters.

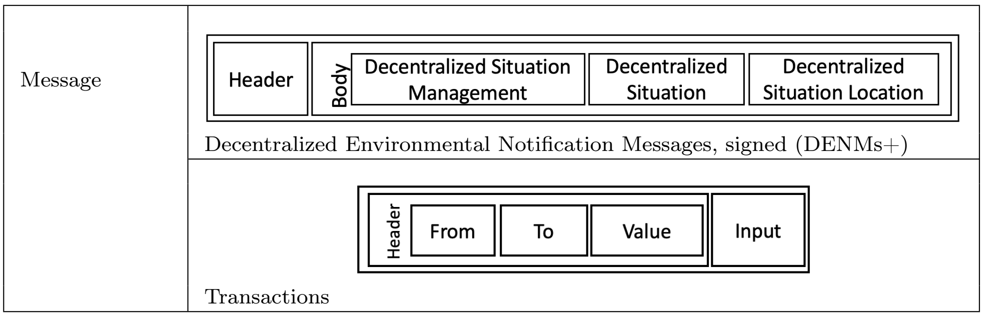

Message
Header
Body
Decentralized Situation Management
Decentralized Situation
Decentralized Situation Location
Decentralized Environmental Notification Messages, signed (DENMs+)
Header
From
To
Value
Input
Transactions

Table 7.2: The two types of messages exchanged in the current blockchain-based VANET system

7.3.2 Description of the framework

There are two phases in the use of the framework. The first phase relates to the secure bootstrapping of the distributed authorities. It uses the blockchain-based cryptographic primitives developed in this thesis to establish the distributed Opening and Issuing authorities of a modified group signature scheme. The second phase presents the use of the infrastructure with the new-DEN messaging protocol and the construction of the blockchain-based Traffic Reporting system.

Setting up the Infrastructure

Figure 7.5 illustrates a top-down description of the proposed framework. Each step, from ① through ⑧, is detailed in the following paragraphs. The description discusses the secure distributed bootstrapping of the Infrastructure and the life of the network as a Traffic Reporting system. The setup of the infrastructure consists in the following steps: the registration of the RSUs in the blockchain network ①, the establishment of the Opening authority ② and the Issuing authority ③, the registration of the vehicles as group members ④ so that they can protect their anonymity in further communications. After this bootstrapping, the vehicles can use the system to anonymously report road hazardous events. This reporting protocol consists in: the generation, signing and broadcast of the new-DEN messages ⑤; the reception the DENMs and verification of their validity ⑥. The usage of the Traffic Reporting system also defines the reporting of malicious events ⑧ and the distributed opening of the associated signatures ⑦.

① The Registry contract. When the system is up and running, there is a contract, called the Registry contract, that tracks full nodes that are joining and leaving the network. The Registry contract enables the registration of the capacitated and willing ITS nodes to play a role in the Blockchain network and in the blockchain-based cryptographic primitives developed. It gives an overview of the nodes available at a given time t to setup the ITS infrastructure and plays the role of sub-Opener or sub-Issuer. The Registry contract is deployed on the sidechain. To setup the whole ITS/Blockchain joint infrastructure, a governmental node starts by deploying an instance of the Register smart contract. This action triggers the deployment of the DOGS and TOAD smart contracts too. This results in parallel emissions of the Sub-Opener Registration opened and the Sub-Issuer Registration opened events informing the set of RSUs that they can join the blockchain network and apply to become one and/or the other. In that case, they will play the roles of sub-openers in the execution of the $\mathcal{DOGS}$ scheme and sub-issuers in the execution of the $\mathcal{TOAD}$ protocol.

② Setup of the Opening authority. We adapt our blockchain-based group signature scheme with a distributed opening to the case where the blockchain and its nodes are already setup and publicly known. This means that the Bootstrap and Registration phases from the initial protocol are no longer necessary. It goes as follows.

(2.1) Listing sub-openers. A sub-opener is a selected node that actively participates in the computation of a distributed key for the opening functionality. The Registry smart contract triggers the deployment of an instance of the DOGS contract. By accessing the Registry contract's public information, the DOGS smart contract can inform the RSUs that the sub-opener selection has started (emission of the Sub-Opener Registration opened).

(2.2) Setup of the distributed Opening authority. Upon reception of this event, the RSUs apply by locking a predetermined amount of coins through the DOGS smart contract. Once a predetermined number N of applications is collected, the contract emits the Sub-Opener Registration Completed event. Upon reception of this event, the RSU candidates know that they can execute the **Opening Key Generation** phase of $\mathcal{DOGS}$. The registered sub-Openers nodes are denoted O_i. Thus, the group $O_1, \ldots, O_N$ engages in the Blockchain-supported DKG protocol. The protocol's output is a dlog-based public/private keypair where the public part is the opening public key denoted opk, and the secret key osk is never reconstructed. Instead, the protocol outputs a set of (private) opening secret keys $osk_1, \ldots, osk_N$ s.t. $osk = \sum_{i=1}^{N} osk_i$, and $opk = g^{osk}$. The execution of our $\mathcal{DOGS}$ protocol has been fully detailed in Chapter 4. The execution ends with the publication of the group (public) opening key opk.

⇒ At the end of this phase, the system has a distributed opening authority for its group signature scheme.

⇒ For the U_i to be able to use $\mathcal{DOGS}$ scheme, it needs to register towards the sub-openers and thus, belong to the group. However, this registration process would occur between one vehicle and one RSU. This is an issue. Indeed, the RSU could apply censorship and deny the service to the car. No one would know it except for both of them. We suggest enhancing $\mathcal{DOGS}$ with a privacy-preserving registration process (addressing the aforementioned flaws in the `Join`/`Iss` joint protocol). The idea is to protect the communication between the vehicle and the group of RSUs by encrypting it with a threshold encryption scheme. Hence, all the RSUs would receive a request, but they would have to collaborate to decrypt it and provide the requested service. In the next paragraph, we describe how $\mathcal{BAT} - Key$ offers such functionality and apply it in the context of $\mathcal{TOAD}$ scheme to improve the registration process of $\mathcal{DOGS}$.

③**Setup of the distributed Issuing authority.** This phase is divided into two steps. The first step consists of $\boxed{1.2.1}$ the nomination of the sub-issuers. The second step is $\boxed{1.2.2}$ the selection of the sub-issuers for the ongoing round. This notion of 'round' is similar to an election. With $\boxed{1.2.1}$ The sub-issuers are identified in the network. However, thanks to $\boxed{1.2.2}$ the adversary cannot know which group of issuers acts as the Issuing Authority during a specific round. A sub-issuer is a selected node that actively participates in computing a distributed key for the issuing functionality. In parallel to the deployment of $\mathcal{DOGS}$, the Registry smart contract deploys an instance of the TOAD contract. By accessing the Registry contract's public information, the TOAD smart contract can perform two main actions. 3.1 First, it triggers the emission of the Sub-Issuer Registration opened event that informs the RSU nodes in the system of the possibility to apply for becoming a sub-issuer node. Then, 3.2 it periodically demands to the registered sub-issuers to generate a new group and corresponding keypair for the acting Issuing authority.

3.1 Nomination of the sub-Issuers. Similarly to what is described in the **Opening Keys Generation** phase in $\mathcal{DOGS}$, the TOAD implements a function to perform a blockchain-supported DKG protocol and generate the first Issuing keypair. The deployment of the contract triggers the emission of the Sub-Issuer Registration opened event. It informs the RSU nodes in the system of the possibility to apply as sub-issuers. (Later on, we describe how registered sub-issuers can be selected in the acting Issuing authority.) The RSUs start by locking some coins through TOAD contract similar to what they did in step 2.1. Once the contract collects a predetermined number N' of applications, it emits the Sub-Issuer Registration Completed event. Upon reception of this event, the RSU candidates know that they can execute a blockchain-based DKG protocol for the issuing key (the function calls an instance of Schindler *et al.* protocol [38]). At the end of this execution, the sub-issuer nodes are identified, and the public issuing key is advertised for further uses.

3.2 Election of the sub-issuers. Let ΔT be a predetermined period of time. Every ΔT, the group of acting sub-issuers changes. This reset is triggered by the smart contract TOAD, which emits the Sub-Issuer election started event. Some sub-issuers, among the N' available, anonymously engage in the $\mathcal{TOAD}$ protocol to create a common issuing public key (which corresponds to the encryption key output by the protocol) in such a way that they all possess a share of the corresponding issuing secret key. As such, upon a registration request from the users, the sub-issuers can choose whether to participate or not. Since $\mathcal{TOAD}$ is anonymous and traceable, the elected sub-issuers are accountable for their actions. Yet, they are protected against privacy and coercion related attacks.

All the exchanges performed in $\mathcal{TOAD}$ and corresponding transactions (as detailed in Chapter 5) are logged inside the sidechain. As such, the latest Issuing public key ipk published on the chain corresponds to the current acting Issuing authority.

7.3.3 Using the Infrastructure

Registering the vehicles

④ OBU Registration. In previous registration processes, the OBU V_i was interacting with a single TA to get its temporary keys certified. Now, the V_i will engage in the `Join`/`Iss` joint protocols with a group of anonymous RSUs. To this end, it will encrypt its request for registration, which contains (i, pk_i, sk_i) and (pk_i, sig_i), with the threshold encryption scheme used in $\mathcal{TOAD}$ and the public key of the distributed Issuing authority (conformally with $\mathcal{TOAD}$'s description in Chapter 5). The group of acting sub-issuers collaborate to decrypt the registration request and further processing. Then, one of them is randomly selected to certify V_i's temporary credentials. It then encrypts $(i, pk_i, sk_i, sig_i, cert_i)$ with $\mathcal{TE}$ and ipk, and concatenates $H(cert_i)$ before adding the result to the registration table reg. All the transactions corresponding to the messages broadcast during the execution of $\mathcal{TOAD}$ (as described in Chapters 5 and 6) are logged in the sidechain.

Reporting road hazard events

⑤ OBU Signing and ⑥ Verifying. Once the registration phase is completed, the registered OBUs can use the new group signature scheme in the same way as initially described in Section 4.1. The new construction does not modify `GVf` either. Therefore, the signatures can be verified in similar ways as before by applying the `GVf` algorithm. The transactions generated from the DENMs (Table 7.2) are logged in the sidechain via the Reporting smart contract.

Exposing fake news and malicious nodes

When a fake message is suspected, the node can issue an audit request via the Auditing contract. It sends the message, its identity and locks a predetermined amount of coin to the Auditing smart contract. An New Audit Request event is emitted for further processing. If the request is accepted (*i.e.* validated by more than half of the nodes), the Opening and the Issuing authorities can jointly de-anonymize the signer, and the requester retrieves its coins. If the request is rejected, the requester is considered malicious: the coins remain locked.

⑦ **Distributed Identification.** This step is divided into two parts: 7.1 the Opening of the signature and then 7.2 the identification of the signer.

7.1 Distributed Opening This refers to $\mathcal{DOGS}$ processing of the signatures. In case of dispute, an opening request is broadcast and logged inside the blockchain. The sub-openers collaborate to, if they want, the Opening functionality. The result of this process reveals the certificate $cert_i$. In order to identify the owner of this certificate, the sub-openers forwards this value to the sub-issuers.

7.2 Distributed Identification All the sub-issuers look-up in the reg table to find the corresponding $H(cert_i)$. Then, they collaboratively decrypt the full associated entry and finally reveal the identity of the signer.

The role of the Auditing smart contract

⑧ **Audit** Of course, since the RSUs can be corrupted, they must produce proof of identification. It is the combination of the proof of Opening provided by the sub-openers and the proof of identification. This includes demonstrating the relationship between all the elements in $(i, pk_i, sk_i, sig_i, cert_i)$ (since sig_i is the signature of pk_i with usk_i, it can be verified with upk_i).

All the transactions associated with the broadcast messages are logged in the sidechain through calls to the Auditing contract.

7.3.4 Security review

From the aforementioned security requirements applicable to anonymous authentication schemes in ITSs, we have drawn the following security definitions for the proposed new-DEN messaging protocol.

Privacy-preservation. Let $\mathcal{M} = (m_1, \ldots, m_k)$ be a list of k broadcast new-DEN messages such that exactly 2 of them have been sent by the same node. The privacy-preservation property requires that no PPT adversary $\mathcal{A}$, with the knowledge that the same user broadcast exactly two messages, can guess which messages with non-negligible (in λ) advantage.

Valid new-DEN message. A new-DEN message is said to be valid if and only if all the fields inherited from the initial new-DENM satisfy the original requirements AND the GSig value is correct.

Accountability. Let m_0 be a broadcast new-DEN message. The accountability property requires that no PPT adversary $\mathcal{A}$ can generate a valid m_0 which does not link to its identity with non negligible (in λ) advantage.

Service Censorship-resistance. Let $P_1, \ldots, P_N$ be $N > 3$ nodes constituting the network that proposes a service S, and let $\mathcal{A}$ be an adversary that can compromise t nodes in the system (without loss of generality, we consider that the corrupted parties are $P_{N-t}, \ldots, P_N$). The Censorship-resistance property guarantees that there exists $r < N - t$ such that $P_1, \ldots, P_r$ can still provide S.

Service Coercion-resistance. Let $P_1, \ldots, P_N$ be $N > 3$ nodes constituting the network that proposes a service S, and let $\mathcal{A}$ be an adversary that can compromise t nodes in the system (without loss of generality, we consider that the corrupted parties are $P_1, \ldots, P_t$). The Coercion-resistance property guarantees that the probability that $\mathcal{A}$ determines if an honest P_i provided S is equal to $1/2 + negl(\lambda)$.

Being the combination of $\mathcal{DOGS}$, $\mathcal{TOAD}$ and $\mathcal{BAT} - Key$, we have provided, throughout the chapters of this thesis, evidences that the resulting new-DEN messaging protocol, and consequently the resulting blockchain-based Traffic Reporting system, is correct, privacy-preserving, accountable, censorship- and coercion-resistant.

7.3.5 Comparing with existing authentication schemes for ITSs

Through theoretical analysis, we obtain a comparison Table 7.3 between the three schemes presented in Table 7.1 and our scheme.

In theory. From a theoretical point of view, our scheme provides the same security services as those of the best popular references but at the expense of lower trust assumptions. Indeed, we do not consider a trusted infrastructure nor do we rely on a trusted tracing authority. Instead, we acknowledged that RSUs can be corrupted and assumed that at least a threshold number of them $t < n/2$, where n is the total number of RSUs in the system, is dishonest. In addition, we opted for total distribution of the powerful authorities namely the tracing authority, which refers to the Opener in our construction, and the Issuing authority as well.

While references [200] and [203] focus on the CAMs, we target the DENMs which are bigger in size and contain the PII of their origins. Future work should include an implementation of the scheme, based on those of $\mathcal{DOGS}$ and $\mathcal{TOAD}$ described in Appendix A.

Table 7.3: Security services and trust assumptions - a comparative table with the referenced schemes

Ref	Message authentication	Conditional anonymity	Anonymity revocability	Tracing authority	Trusted infra. (TA, RSU)	Type of message	Safety message size (bytes)	Signature size (bytes)
[200]	●	●	●	Centralized	Yes	CAM	474	368
[203]	●	●	●	Centralized	Yes	CAM	72	56
[206]	●	●	●	Centralized	Yes	unknown	935	826
Our	●	●	●	Decentralized	Threshold t	DENM	200	160 [KLAP20]

In practice. Indeed, we demonstrated throughout the chapter that our proposition theoretically improve on the referenced schemes. Yet, we need to evaluate the impact of changing the structure of the DENMs and determine which group signature fits best in our model such as to identify where the proposition ranges between 72 bytes, the lowest safety message size provided by [203], and 935 bytes, the largest one obtained in [206]. Based on the work of Kim *et al.* in [207], we were able to foreseen a size amplitude. The benchmarked group signatures are ranging from 128 bytes with the PS16 scheme [208], to 672 bytes with the BBS04$^+$ scheme [202]. Among them, three are defined in the same BSZ model as considered in this thesis (Table 7.4). While DP06 and DS18 provide a stronger anonymity property compared to KLAP20, they generate longer signatures, respectively 288 and 448 bytes against 160. The CCA anonymity is known as *selfless anonymity* [209]. It is a relaxation of the CCA-full anonymity defined by Bellare *et al.* in [112]. While the definition of selfless anonymity is weaker, it is often sufficient for realistic applications [207]. Considering the performance aspects of group signature schemes, KLAP20 is quite interesting: smallest signature size, fastest signing, verifying and batch verifying processes. In addition, it provides the opening soundness by which the Open algorithm can always find the user as the owner of a valid signature. Yet, additional work on the implementation and evaluation of the proposition must be done to validate this hypothesis about the viability of a KLAP20-based Traffic Reporting system.

Table 7.4: Comparing existing group signature schemes that may fit our system's design

Scheme	Anonymity	Opening	Signature	Sign (ms)	Verify (ms)	Batch Verify (ms)
[Ref]			size (bytes)			for n=100
DP06 [161]	CCA2	Sound	288	6.5	10.8	1080.0
DS18 [210]	CCA2	Weak	448	4.8	14.4	489.6
KLAP20 [207]	**CCA-**	**Sound**	**160**	**1.2**	**7.8**	**68.1**

In addition, [200] and [203] both analyze their protocol *w.r.t.* the *message loss ratio* and the *time to verify signatures*. It would be interesting to complement the above analysis with an evaluation of the performance of the implementation *w.r.t.* these two metrics.

7.3.6 Conclusion

This chapter concludes our current construction of a blockchain-based Traffic Reporting system. We presented how the designed distributed cryptographic primitives $\mathcal{DOGS}$, $\mathcal{TOAD}$ and $\mathcal{BAT}-Key$ can be used in the context of ITS to provide privacy and accountability in vehicular communications. In our last chapter, we will elaborate on the contributions of this thesis, its identified limitations and open up on perspectives for future work.

8

Conclusion and perspectives on future research

"The journey, Not the destination matters."

— Thomas Stearns Eliot

This chapter will conclude the present thesis. In Section 8.1, we draw the final conclusions and summarize our contributions presented throughout the chapters. Then, in Section 8.2, we establish the limitations of the current research. Finally, in Section 8.5, we share some thoughts on perspectives for future work and improvements.

8.1 Conclusion

In this thesis, we were mainly interested in studying the trade-off between privacy and accountability in distributed systems. We entrenched our analysis in the context of ITSs. In a nutshell, we presented an experimental setup on the use of blockchains in distributed cryptography to secure the bootstrap of a Road-Traffic Reporting system. In Chapter 1, we set the stage for our story to take place, namely we introduced ITS, vehicular communications and related challenges. This vehicular environment was further explored in Chapter 2 along with the Blockchain technology to stress out their similarities. In Chapter 3, we discussed the challenges and core motivations for our research. We also reviewed the existing literature that tries to address them and detail their limitations. More specifically we focused on three cryptographic primitives: group signatures, distributed key generation protocols (DKG) and threshold encryption schemes. We established that existing protocols could benefit from the use of blockchains to reinforce traceability and accountability in their execution, but also to provide anonymity to the involved parties. In Chapter 4, we described a group-signature-based authentication protocol for traffic reporting in ITSs and highlighted its limitations. We then proposed our construction of a blockchain-based group signature scheme with distributed opening, called $\mathcal{DOGS}$. The resulting authentication protocol was refined in Chapters 5 and 6 with the addition of both: a blockchain-enabled threshold encryption that supports an anonymous decryption service, protocol called $\mathcal{TOAD}$; and a blockchain-based anonymous-yet-traceable DKG protocol. These contributions added on the distribution of the central authorities traditionally present in cryptographic constructions applied to ITSs. Chapter 7 concluded the framework and wrapped up the thesis by combining the presented blockchain-based cryptographic contributions $\mathcal{DOGS}$, $\mathcal{TOAD}$ and $\mathcal{BAT}-Key$. The resulting blockchain-enabled Traffic Reporting system is theoretically analyzed in terms of security properties. Yet, as we will see in Section 8.5, while both $\mathcal{DOGS}$ and $\mathcal{TOAD}$ were developed and briefly analyzed in Appendix A *w.r.t.* the metrics of *gas consumption* and *time of execution*, the implementation and performance evaluation of the Traffic Reporting system itself is still incomplete.

8.2 Limitations

The blockchain-based cryptographic contributions described in Chapters 4, 5 and 6 were analyzed in a static setting where the groups of sub-openers and sub-issuers do not change over time. More specifically, there is no addition while it is theoretically supported by the underlying schemes. Thus, our constructions are meant to investigate how authoritative parties, such as international agencies, can enhance the current specifications with distributed cryptography and blockchains. The end goal being to find a practical framework that protects vehicular actors' privacy while enforcing accountability mechanisms for safety purposes.

The BSZ security model chosen to build up our group signature scheme with distributed opening is a strong security model often impractical in realistic applications. While the resulting cryptographic primitives are more secure than others, they may lack practicality. Consequently, as our future work suggests, more work can be done in weakening the initial security assumptions and investigating the impact of such relaxation on the performance of the resulting framework.

Our blockchain-based Traffic Reporting system is extensively detailed through the thorough description of its constituting components, namely $\mathcal{DOGS}$, $\mathcal{TOAD}$ and $\mathcal{BAT} - Key$, and their right combining. Yet, a full implementation is missing. Some work has been done in evaluating the underlying primitives, $\mathcal{DOGS}$ and $\mathcal{TOAD}$ in the Appendix A. The metrics of gas consumption and time of execution are both extremely important: the first establishes the practicality of the scheme on the blockchain chosen; the second determines the feasibility of the cryptographic operations in a CPS (ITS) environment. However, some other metrics may be interesting to evaluate, notably: the *message loss ratio*, which establishes the rate at which messages are discarded instead of being received; and the *time to verify signatures* which may confirm the feasibility of the `ActionID` replacement *w.r.t.* the real-time aspect of the ITS application.

8.3 Openings

The developed cryptographic primitives, $\mathcal{DOGS}$, $\mathcal{TOAD}$ and $\mathcal{BAT} - Key$, are of utmost importance, even outside the field of ITSs.

8.3.1 Using $\mathcal{TOAD}$ for the decryption of data under shared governance

In this sub-section, we outline two use cases for a threshold encryption service that justify the necessity of guaranteeing the anonymity and accountability of the decryption servers. It is closely related to the trade-off between research interests and the world-wide political balance. Several research topics necessitate the study of confidential data to make technological and sociological advances. The following paragraph details them: the first concerns the release of confidential data after they fall into the public domain; the second explores the sharing of classified pictures captured by a satellite orbiting around the Earth.

Rwanda, François Graner's fight to opening the archives. In 2020, after five years of fighting, the French Council of State has authorized a researcher, François Graner, to access the documents tabled by late president François Mitterrand over to the National Archives [211]. This triggered a huge outrage in France, where the government was accused to hide the participation of former french governments in the slaughter of thousands of Tutsi back in 1972. The main issue here is the censorship operated on data. They should have been released if the process had been automated. Indeed, the period during which the data were supposed to stay confidential elapsed.

Resolution. We propose to exploit the digital technology in the following way to prevent this from happening again. The digitization of the archives and their encryption guarantee the confidentiality of the data. The distribution of the decryption authority ensures the availability of the service and reduce the risk of censorship. Finally, the logging of these actions onto a publicly verifiable ledger ensures the accountability of the decrypting authority. Using the proposed protocol (described in Section 5.3.1), we alleviate any trusted party's censorship and improve the data availability by distributing the role of the data centre and decryption server to a set of entities. In addition, we ensure the anonymity of these entities to prevent coercion and corruption in the decryption of sensitive data. Yet, we provide their accountability via the implemented traceability mechanism.

Sharing satellites to fight global warming. Let us consider a satellite orbiting around the Earth and capturing pictures of the Earth topology. The satellite is equipped with a camera, a unique hardware key, an artificial intelligence program, a trusted environment and communication modules. The satellite works as follows. ① The camera takes a picture of the earth surface. ② The trusted environment retrieves the picture and runs the IA, which outputs a flag that characterizes the image (*e.g.,* “danger"). Then, it encrypts the image with the hardware key, attaches the flag and sends it to the communication modules. ③ The communication modules encode the encrypted data and the flag and forwards the packet to the Earth station. ④ On the Earth side, the communication modules of the station decode the packet and get the flag “danger".

Resolution. Currently, there is traditionally only one country that possesses the corresponding decryption key of the one embedded in the satellite. Therefore, they may choose to share or hide the potentially important information captured by the satellite. For instance, this could be catastrophic for the research community and hinder technological advances. However, access to a picture of a foreign country is prohibited by international laws. We suggest sharing the access to the satellite and the data it captures by using a threshold encryption scheme instead. The decryption key corresponding to the hardware key embedded in the satellite could be split among actors of different nationalities. In that case, the access to flagged information is subject to a consensus. Preserving the anonymity of the peers, whether they participate in the decryption of the picture, is of utmost importance as it protects free will and prevents coercion within actors with diverse and sometimes conflicting interests.

8.4 $\mathcal{BAT}-Key$ for fairness and anonymity

In this sub-section, we outline several application domains that outline the importance of anonymous-yet-traceable DKG protocols, and could benefit from the use of $\mathcal{TOAD}$.

Anonymous trading and fairness. Anonymous trading occurs when high profile investors execute trades that are publicly visible while keeping their identity private. Usually traders trade non-anonymously, however, there are several reasons for which some prefer to keep their participation in a market a secret. For instance, anonymous trading is primarily used to avoid informing the market of a pending action, as this could lead to unwanted behaviors. Let us take the following example: a large institutional buyer is interested in acquiring millions of shares and may not want to make their intentions known before they can complete the purchase. Making the transaction non-anonymously may lead to smaller investors bidding up the price and hoping to sell it to the institutional buyer for a quick arbitrage profit.

Currently there are three different ways to do anonymous trading: though 1) Anonymous exchanges, in which case the stock exchange provides the service itself; 2) Dark pools, which are private asset exchanges designed to provide the additional anonymity in trading; 3) Inter-dealer brokers, which are intermediary and perform trades on behalf of their clients.

In all three methods share a common flaw: the centralized model chosen to provide anonymous trading, as the anonymity of a trader lies in the hand of a single entity.

Example. Let us consider a set of identified traders. At one point, they may want to execute a trade anonymously without revealing this intent to other competitors. The advantages of using our Blockchain-based anonymous-yet-traceable DKG protocol are twofold: firstly, it ensures that traders can covertly create a group behind which they can hide to execute a trade anonymously (by using the DKG protocol to instantiate a group signature scheme with distributed opening). Secondly, the designed protocol ensures traceability of their trades, meaning two things: they cannot repudiate having executed a trade; the retribution of that trade cannot be claimed by anyone else.

Coercion-resistant e-voting systems. Electronic voting sees a lot of adoption recently, in Nigeria [212], in Ghana [213], in Indonesia [214]... To one day surpass traditional paper ballots, e-voting systems must guarantee, to the voters, the same or even better privileges *w.r.t.* to their anonymity and integrity of their votes. One important condition is coercion-freeness [184], which is defined as the inability for an attacker, also called coercer, to force a voter to behave in a certain way during the election. As a consequence, in a coercion-free system the coercer must be unable to force the voter to disclose their voting credentials, to vote for a certain party or to prevent them from voting at all.

The threat of coercion depends on multiple parameters related to the nature of the election and the properties of the e-voting systems. Therefore, there exists several ways to fight it ranging from simple re-vote-based protocol combined with physical countermeasures, or usage of smart cards and trusted devices, to cryptography-based approaches with public key infrastructure, ring signatures, and randomisable encryption schemes [215]. More recently, coercion-free blockchain-based evoting systems started to emerge [216]. For the design of our protocol, we adopt a similar strategy: we provide anonymous communications to prevent an attacker from identifying a voter, and use the blockchain as an immutable record to ensure accountability.

Example. Let us consider a CEO wants to present a motion to the employees of their company and ask them to vote in favor or against this motion. In this case, our scheme presents the following advantages. Firstly, the CEO can pre-select the voters as the employees. Then, the employees can decide for themselves whether they want to vote. If so, they can anonymously engage in our anonymous-yet-traceable DKG protocol to create a group of anonymous voters behind which they can hide their identity (from the others employees, but also from the CEO) to cast their vote. Secondly, and similarly to the previous use case, the designed scheme ensures that: the voter cannot repudiate having sent the vote; and no one can vote on behalf of someone else.

8.5 Future work

Throughout the chapters we gave some insights on what could be improved in the current construction of our blockchain-based cryptographic primitives and the resulting Traffic Reporting system. In this final section, we reflect on what could be further developed in future works.

8.5.1 In-depth analysis of the proposed implementations

We present in the Appendix A a preliminary evaluation study of both $\mathcal{DOGS}$'s and $\mathcal{TOAD}$'s implementations. Our goal is to give a tangible overview of how they can be implemented and what a naive programming can already perform.

We outline in the Appendix A that the choice of the NIZK proofs system is critical. Existing implementations may induce too much computation and communication overhead. Benchmarking the zero knowledge proofs in the present context can be a trail to follow. Another interesting axis of research would relate to the generation of the secret shares. We present in the Appendix three methods that bring their advantages and drawbacks. Some work can be done to further examine these possibilities and propose better ones. Moreover, we could have talked about the on-chain and off-chain communications. More specifically, we delegated two things:

- the anonymizing of the P2P communications to the network layer specialists.
- the mixing of accounts by leveraging an already famous mixing technique.

8.5.2 State of knowledge on mixing techniques

The mixing of accounts in blockchain-based systems is not trivial and still quite new. It may be interesting in a future work to further analyze the current state-of-the-art via a state of knowledge and propose a new mixing technique. An idea is that existing solutions, which leverage a trusted third party, may benefit from the presented blockchain-based distributed cryptographic primitives to offer a distributed, privacy-preserving and accountable mixing service.

8.5.3 Multiple-sidechain-based infrastructure

The propose Traffic Reporting system is described by considering one mainchain and one sidechain. Therefore, depending on the rate of exchanges in the blockchain network, one might prefer to have multiple sidechains, each maintaining the states of particular data (as shown on Figure 8.5.3). Yet, sidechains can read and interpret data from the mainchain. As such, each sidechain smart contract can retrieve information from the mainchain. Unlike RSK, the Horizen[1] network provides a new type of data container called the Cross-chain Certificates. It is a transfer mechanism, initiated on the sidechain, that informs the mainchain of incoming backward transactions. RSK does not provide a way to publish data from the sidechain to the mainchain by design. Yet, an alternative process would be to submit the transaction/payload to the Federation nodes that belongs to both the mainchain and the sidechain networks. As such, since the majority of these nodes is assumed to be honest, this guarantees the truthful publication of the requested information. It would be interesting to figure out whether such a mechanism can be implemented in the RSK framework. And if not, it would be valuable to determine if the current sidechain-based framework is implementable on the Horizen platform instead.

[1] https://academy.horizen.io/horizen/expert/sidechains/

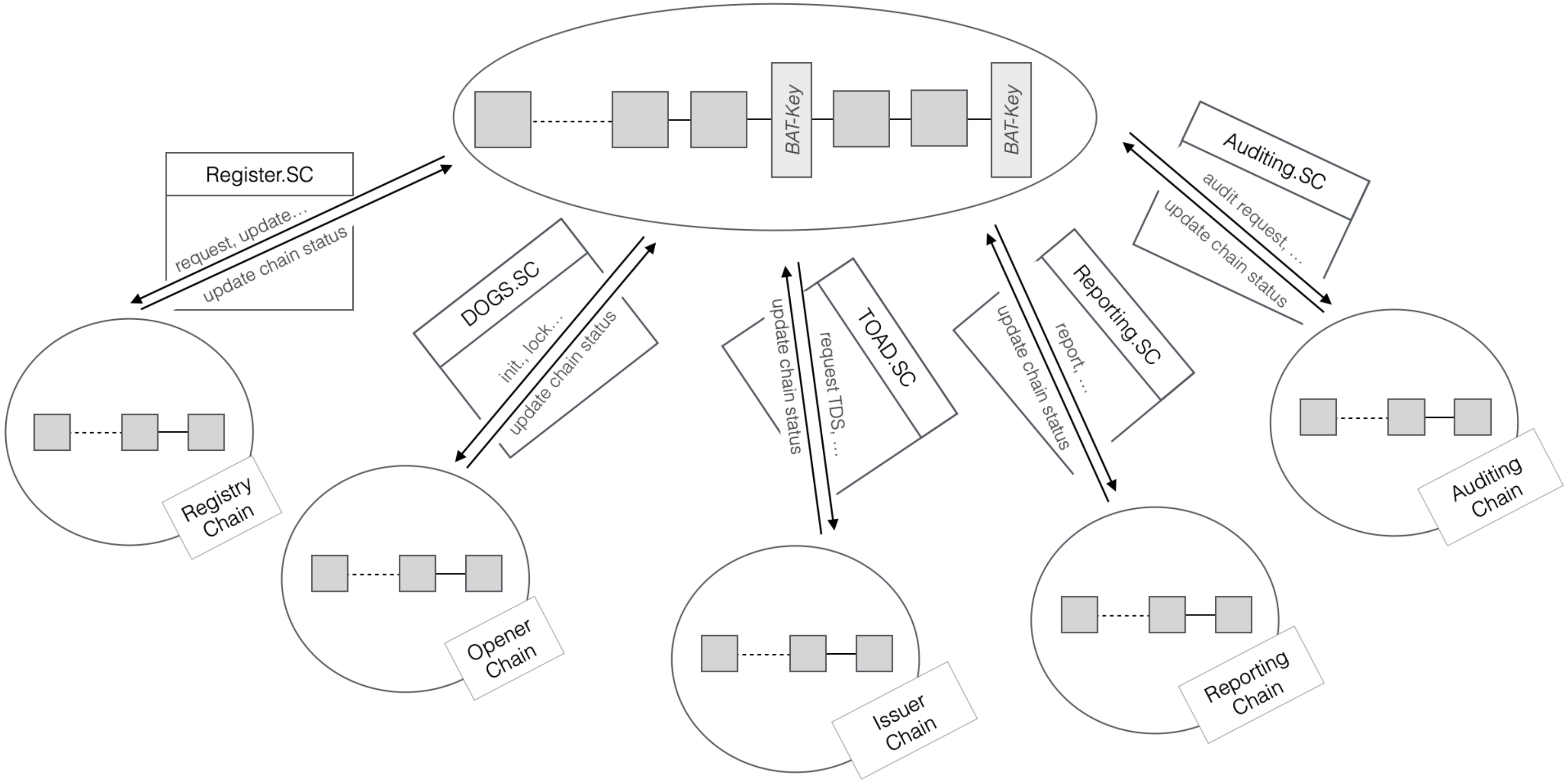


Figure 8.1: Detailed overview of multiple-sidechain-based blockchain architecture

A

Implementations of $\mathcal{DOGS}$ and $\mathcal{TOAD}$

"No amount of experimentation can ever prove me right; a single experiment can prove me wrong."

— Albert Einstein

In this appendix, we present the implementations of $\mathcal{DOGS}$ and $\mathcal{TOAD}$ protocols. Our goal is to determine the practicality and feasibility of the proposed cryptographic primitives. We only restricted ourselves to the programming of the group signature scheme and the encryption cryptosystem which propose both more functionalities than $\mathcal{BAT}-Key$. Hence, demonstrating their practicality induces $\mathcal{BAT}-Key$'s one. In the following sections, we detail each implemenation and analyze the main results in terms of gas consumption and time of execution.

A.1 Implementation of $\mathcal{DOGS}$

As explained in Chapter 4, the protocol proposes a Group Signature scheme with a Distributed Opening functionality. The goal of the protocol is to enable users evolving in the same Region of Interest, denoted RoI (which might be virtual or geographical) to communicate anonymously. Each RoI is controlled by an entity called the Issuer. At the same time, the protocol implements by design a traceability mechanism that ensures the accountability of these users. When one node sends a message, it group-signs it with the $\mathcal{DOGS}$ protocol. Therefore, any receiver can determine if the signature is valid. However, it cannot tell to which user it belongs. Yet, the identity of the signer can be revealed (we also say that its signature can be opened) by the Opener. The novelty brought by $\mathcal{DOGS}$ is the distribution of this Opening authority. Compared to what was described in Chapter 7, we make some simplifications, notably on the centralization of the Issuer.

A.1.1 Scenario

In this subsection, we recall the execution workflow of a classic group signature scheme and highlight the novelties implemented in $\mathcal{DOGS}$'s implementation.

Registration. A user enters the RoI and contacts the Issuer by sending its identity (see `Join`/`Iss` joint protocols in group signature schemes, Chapter 3). The communication channel is secured by the use of a symmetric key encryption scheme. The Issuer advertises its public key ipk. The user i owns a pair of public/private keys too as upk_i, usk_i. It can therefore compute a symmetric key as a combination of ipk and upk_i. Considering a dlog-based encryption scheme, let g be a generator of a group $\mathbb{G}$ (associated with the chosen scheme). Let assume that:

$$ipk = g^{isk} \quad \text{, and:} \quad upk_i = g^{usk_i}$$

Consequently, the user i and the Issuer can both compute the same symmetric key as:

$$ipk^{usk_i} = upk_i^{isk} = g^{usk_i \cdot isk}$$

By repeating the above process, the Issuer registers the new users i as group members.

Opening Keys generation. Once the Registration phase has terminated, the next one can start: it is the generation of the Opening keys. Instead of generating a simple public/private keypair as opk, osk (for the Opener public key and the Opener secret key), the users will collaborate to generate a common public key still denoted opk but for '*Opening* public key'. The corresponding private key is not constructed. Instead, each user owns a share of it, denoted osk_i. If we still consider a dlog-based cryptosystem, we can let:

$$osk = \sum_i osk_i \quad \text{, and thus:} \quad opk = g^{osk}$$

To this end, the users engage in a DKG protocol as described in Chapter 4. These nodes are called *sub-openers* thereafter.

Signing and verifying. The registered users can use the advertised Opening public key opk to protect their identity when communicating with surrounding nodes. Therefore, they sign their messages with the $\mathcal{DOGS}$ protocol (Chapter 4). The resulting signature σ is added to the message to authenticate its source without revealing its identity.

Distributed Opening. One user can sign as many messages as they want and cannot be identified as the source of a message, despite the numbers of signatures they issued. Yet, if a message is suspicious, t among n honest sub-openers can open the signature and reveal the identity of the signer.

About the threshold t. $\mathcal{DOGS}$ is developed over a blockchain similar to Ethereum (i.e. that enables the creation and execution of smart contracts). The current implementation considers that all the users of the scheme participate in the setup of the Opening authority. Moreover, it assumes that these users also contribute to the validation of the transaction and the blocks (i.e. they act as miners). Thus, the blockchain network too consists of these nodes. Consequently, we fix the threshold to:

$$t \geqslant \frac{n}{2}$$

And, we add that the adversary can corrupt up to t nodes in the system. This value ensures, therefore, that the majority of the nodes maintaining the blockchain network is honest, thus guaranteeing the scheme's security and preventing the attack of the 51%.

Discussion on the `dOpen` algorithm. Let us denote $\mathcal{Q}$ the set of sub-openers. Inspired by ETHDKG [38], the DKG protocol (during the Opening keys generation phase) starts with each user i in $\mathcal{Q}$ choosing s_i a secret value such that:

$$osk = \sum_i s_i$$

and a polynomial function f_i of degree t such that:

$$f_i(0) = s_i$$

Then, each user i generates $(n-1)$ shares of s_i, denoted $s_{i \to j}$, from to f_i and the identities j of the other participants s.t. :

$$s_{i \to j} = f_i(j) \text{ , with: } j \in \mathcal{Q}$$

Therefore, each user i sends $(n-1)$ shares. Thus, there are $(n-1) * n$ shares sent in total. At the end of the sharing phase, each user i owns n values of n distinct polynomial functions (one value per polynomial including its own). Consequently, when $t+1$ distinct users collaborate and put their shares related to the same polynomial in common, they can reconstruct that polynomial, and the corresponding secret s_j, via Lagrange interpolation. Let $\mathcal{R}$ denote this subset of $t+1$ users.

$$\forall j \in \mathcal{Q},\ s_j = f_j(0) = \sum_{i \in \mathcal{R}} s_{j \to i} \cdot \prod_{k \in \mathcal{R}, k \neq i} \frac{k}{k-i}$$

Or also:

$$\forall j \in \mathcal{Q},\ s_j = \sum_{i \in \mathcal{R}} f_j(i) \cdot \prod_{k \in \mathcal{R}, k \neq i} \frac{k}{k-i}$$

A.1.2 Implementation choices

The implementation of $\mathcal{DOGS}$ necessitates to distinguish the programming of the client application and the development of the smart contracts.

The client application

The client application is implemented in **Python version 3.7**. It handles the cryptographic operations and the calls to the smart contracts. The primitives used in $\mathcal{DOGS}$'s implementation are **AES-256-GCM** used with nonces, **ECDSA** signatures and **Keccak-256** for hashes. They are used interchangeably on two elliptic curves: **secp256k1** and **alt_bn128**. The first is the curve selected by Ethereum and represents the Ethereum keypairs. The second is a pairing-friendly curve used by ETHDKG. This is where the opening public keys live. Arithmetic operations are inherited from `ethdkg.ethdkg.crypto` (a library developed in [38]). The client also manages the off-chain communications via a TCP module. In addition, it manages the calls to the DOGS smart contract through the `web3` library. Finally, it defines a script for performance evaluation in terms of the *time of execution* and *gas consumption*.

Smart contracts

The smart contracts are written in Solidity and deployed on a local testnet. Ganache is used to simulate this testnet and facilitate the testing of the protocol. It automatically generates accounts with Ethereum funds (called ether) and automatically handles the consensus on transactions and blocks.

About the implementation

The resulting project is available on Github<link>. It started from a fork of ETHDKG. The idea was to propose a modular implementation that could facilitate and support subsequent updates on the original work of Schindler *et al.*. We added about 2000 lines of code to modify ETHDKG into $\mathcal{DOGS}$, namely to transform a blockchain-based DKG protocol into a Group Signature scheme with Distributed Opening. In addition to the definition of the Issuer and the user roles, three types of adversaries were developed, including:

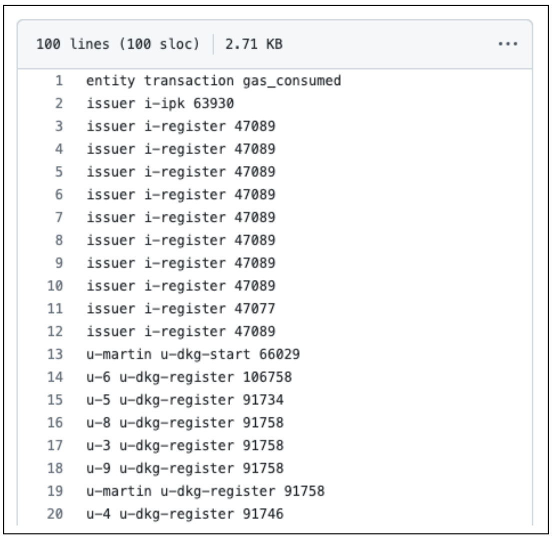
1. an adversary that sends invalid shares to other users during the distribution phase of ETHDKG.
2. an adversary that does not send their key share when reconstituting the opening public key during ETHDKG.
3. an adversary that does not send their secret share when opening a group signature.

The implementation has been fully documented, by using the tool Sphinx, to open-source it and facilitate its use and further improvements.

A.1.3 Evaluation and Discussion

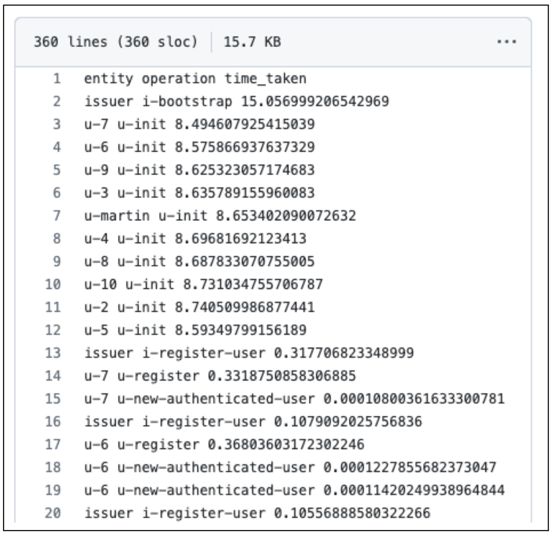
The following is a discussion of the implementation. It shows some results and details the process by which we obtained them. We also discuss ways to improve the proposed implementation.

The metrics. As aforementioned, *time of execution* and *gas consumption* were the two metrics chosen to evaluate the performance of the implementation. The tests have been done using the `measurer` file in the client implementation. The raw data (Figures A.1 and A.2) are then post-treated with a Matlab script.



```
entity transaction gas_consumed
issuer i-ipk 63930
issuer i-register 47089
issuer i-register 47089
issuer i-register 47089
issuer i-register 47089
issuer i-register 47089
issuer i-register 47089
issuer i-register 47089
issuer i-register 47089
issuer i-register 47077
issuer i-register 47089
u-martin u-dkg-start 66029
u-6 u-dkg-register 106758
u-5 u-dkg-register 91734
u-8 u-dkg-register 91758
u-3 u-dkg-register 91758
u-9 u-dkg-register 91758
u-martin u-dkg-register 91758
u-4 u-dkg-register 91746
```

Figure A.1: Excerpt of the raw data file for the gas consumption



```
entity operation time_taken
issuer i-bootstrap 15.056999206542969
u-7 u-init 8.494607925415039
u-6 u-init 8.575866937637329
u-9 u-init 8.625323057174683
u-3 u-init 8.635789155960083
u-martin u-init 8.653402090072632
u-4 u-init 8.69681692123413
u-8 u-init 8.687833070755005
u-10 u-init 8.731034755706787
u-2 u-init 8.740509986877441
u-5 u-init 8.59349799156189
issuer i-register-user 0.317706823348999
u-7 u-register 0.3318750858306885
u-7 u-new-authenticated-user 0.00010800361633300781
issuer i-register-user 0.1079092025756836
u-6 u-register 0.36803603172302246
u-6 u-new-authenticated-user 0.0001227855682373047
u-6 u-new-authenticated-user 0.00011420249938964844
issuer i-register-user 0.10556888580322266
```

Figure A.2: Excerpt of the raw data file for the time consumption

Experiment. In the following, we show the result of the `performance.sh` script. The experiment considers 10 users, including 3 adversaries (one node per type) and a threshold value t equal to 4. Once all users have been authenticated, the main user (chosen randomly) starts ETHDKG and follows the steps described in Schindler's implementation. Then, users exchange messages for a while. After that, the main user requests to open the last group signature it received. The honest users agree. The test ends after the group signature is opened and the signer is identified.

The user names are `martin` for the main user and `2` through `10` for the others. The adversaries are nodes `8`, `9` and `10` respectively of type 1, 2 and 3.

Analysis of the gas consumption

Total gas consumed per entity. Figure A.3 shows the total amount of gas spent by each entity. We can observe that malicious parties consumed less than honest users, and we could have predicted it. Indeed, node 8 sends invalid shares to other users during the distribution phase of ETHDKG. Therefore, it is quickly detected and ignored for the rest of the protocol. Since it cannot spend gas after being removed from the group, its consumption is reduced. Similarly, nodes 9 and 10 do not participate at one point in time, and the extra consumption, corresponding to the data they did not share, is not added to their current consumption. The user `martin` consumed more than the other because it is the main user for this round of execution. As such, it performed more transactions and was also responsible for the publication of *opk*.

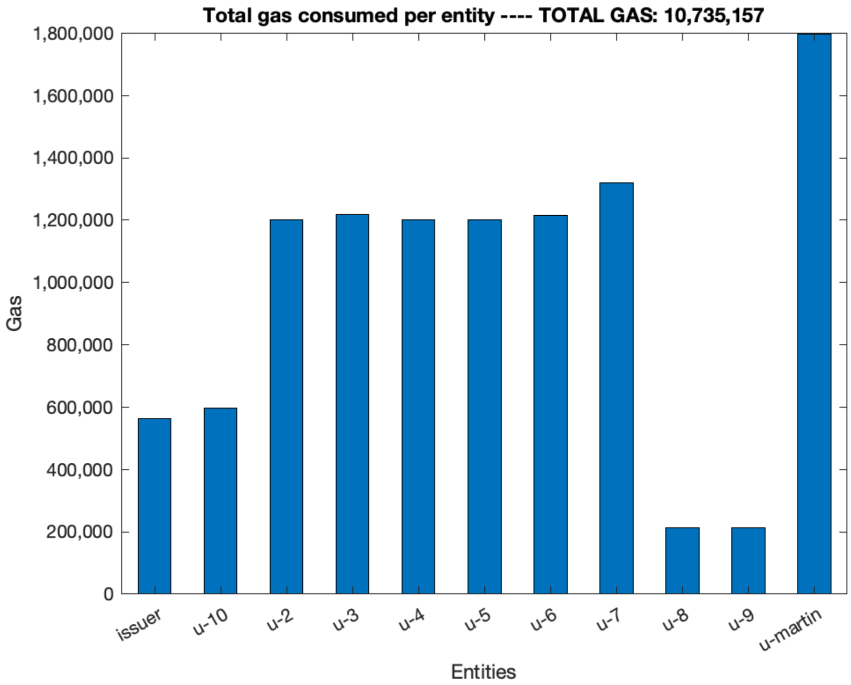


FIGURE A.3: Total gas consumed per entity

Gas consumed per transaction. Figure A.4 shows the gas spent per transaction time. It reveals that the publication of *opk* is quite expensive — the difference between the max amount of gas spent in blue and the median value in orange — which explains why `martin` spent more as than others. Overall, the transactions cost less than 250,000 gas, with a mean of around 50,000, except for the opening functionality. Indeed the `u-open-secret-share` transaction costs about 600,000 gas. This is due to the fact that both functions publish data inside the contract.

Grouping gas consumption per phase. There are three main phases in the *DOGS* protocol: the issuing phase played at the Issuer, the running of the DKG protocol via

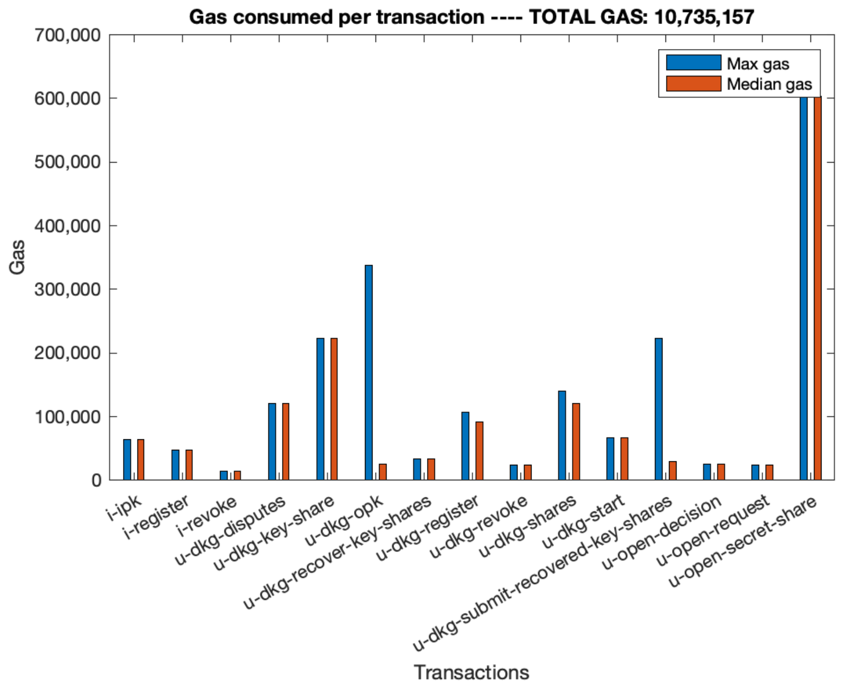


Figure A.4: Total gas consumed per transaction

ETHDKG, and the opening phase. Figure A.5 reveals that running the blockchain-based DKG protocol is what costs the most (between approximately 4,700,000 when there is no adversary and 5,800,000 gas). We can also observe that the ETHDKG and the opening process are quite comparable.

Analysis of the execution time

Figure A.7 shows the time taken by each operation; and Figure A.6 illustrates the execution time of the protocol for each entity. The *DOGS* protocol is executed in approximately 500 seconds, i.e. 8 minutes and 20 seconds, in the presence of three adversaries. Thanks to the detailed graph, we can determine that the most time-consuming operation remains the blockchain-based DKG protocol (approx. 6 minutes and 40 seconds). Apart from the DKG, the other operations are almost instantaneous. Since the setup of the Opening authority belongs to the setup phase of any system, this time consumption is not a big issue. Indeed, its impact is smoothen over time and protocol's usage.

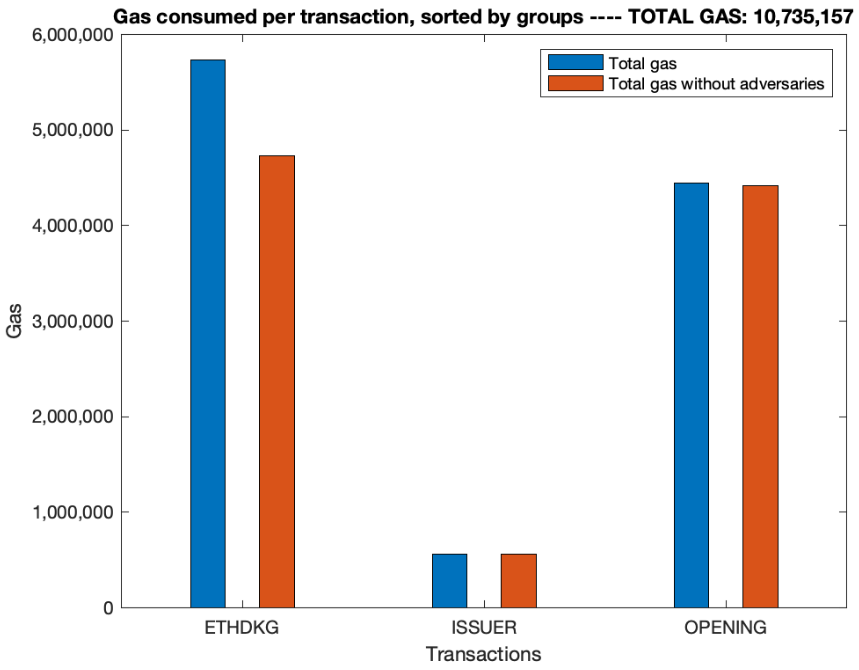


FIGURE A.5: Total gas consumed per transaction, sorted by phases

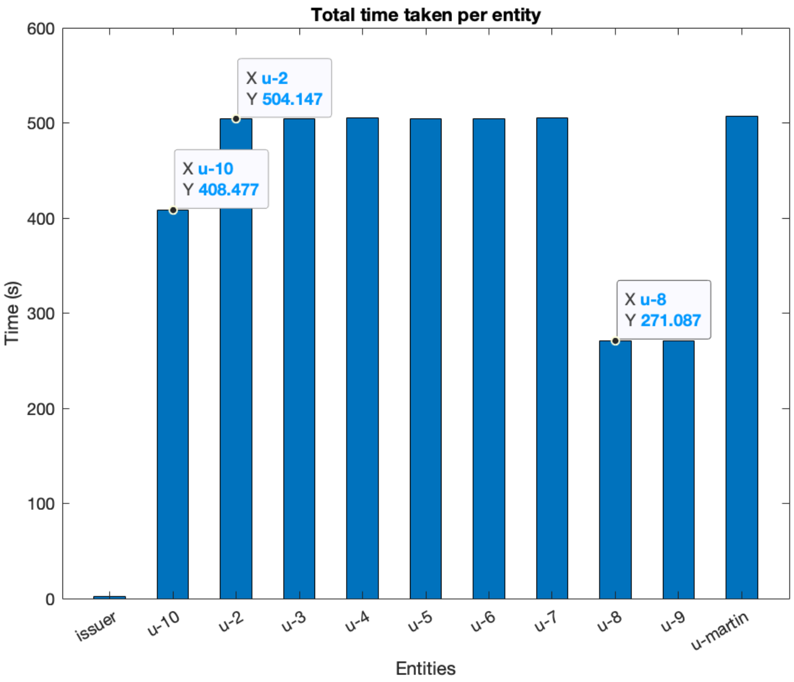


FIGURE A.6: Total time taken by the protocol at each entity

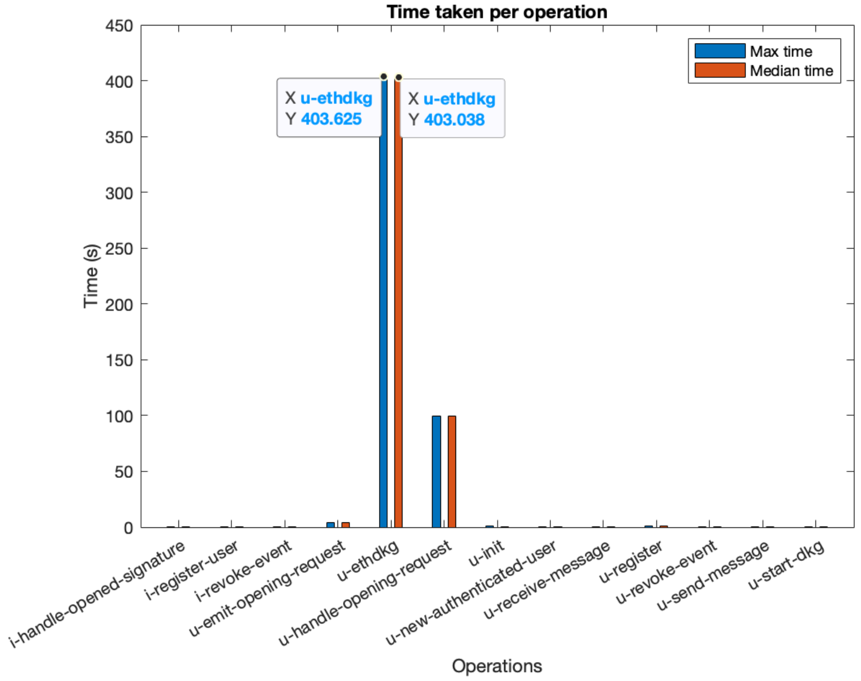


Figure A.7: Total time spent per operation

A.1.4 Future work

About the GVf algorithm

In the current implementation, the signature verification algorithm is still under exploration. The algorithm must provide a way to differentiate a signature generated by a group member (registered user) from a random string. There are several ways to implement such a verification protocol.

Zero Knowledge proofs. The theoretical protocol leverages a Non-Interactive Zero-Knowledge proofs system. Examples of such systems include: SNARKs, SNORKs, STARK, PBC-based NIZK... and other zero-knowledge proof system of which a list is given in [217].However, these methods may be too heavy to be implemented in a practical setting.

Using confidential knowledge. The opening public key *opk* is available to anyone after the execution of the ETHDKG, even to a non-authenticated user, because it is published onto the blockchain. We could restrict its access to only authenticated users. Yet, if only one adversary is present in the system, they could simply leak the key to the outside, nullifying our efforts to conceal it.

In this implementation, `GVf` is represented by the function `doggy_crypto.g_verify()`. We implemented two ways to verify a group signature, based on whether the user is qualified or not.

A user is labelled 'non-qualified' if it did not participate in the DKG protocol that set up the Opening authority. Therefore, it must contact the Issuer and ask them to deliver a certificate to still use the scheme. The certificate is a ECDSA signature of the first coordinate of the public secret joined with the group signature, signed with the Issuer secret key:

$$cert_i = \texttt{Sign}(\text{public\_secret\_x}, isk)$$

Verifying a certificate comes down to verifying this signature. It is possible because Issuer public key ipk is known by all. This verification is already implemented in `doggy_crypto.g_verify()`. As such, non-qualified users can actually both send and verify messages.

Qualified users could adopt the same process. However, this would give twice as much power to the Issuer, which is not ideal. Qualified users have participated in ETHDKG. They played a part in generating opk: they each have a secret s_i and a group key pair gsk_i, gpk_i. An idea could be to leverage the knowledge of this latter information to write the `GVf` algorithm as it is independent from the Issuer. This method is not implemented currently. Since ETHDKG comes with a ZK-proof called `DLEQ`, the method could reuse that function to prove the signer has one of the secrets generated during ETHDKG. A qualified user would therefore join with the group signature a proof π:

$$\pi = \texttt{DLEQ}(h, h^{s_i}, b, b^{s_i}, s_i)$$

where s_i is the user i secret of which each other user j knows a share $f_i(j)$ and b a public value associated to s_i (e.g. b^{gsk_i}?).

Secret shares

The secret shares refer to the `alt_bn128` points post by the users on the blockchain when they open a group signature. These shares are collected to reconstruct the key k_0 used to encrypt the signatures. We have:

$$k_0 = opk^a = b^{osk}$$

where (a, b) is an `alt_bn128` keypair generated during the signature. The secret shares are used to recover k_0 without revealing osk. There are two methods to do that.

Method ①: b^{s_i}. In that case, the secret shares are equal to:

$$b^{s_i}, i \in \mathcal{Q}$$

with s_i the secret generated by the user i during ETHDKG.

In that case, the reconstruction method consists in summing the secret keys:

$$\prod_{i \in \mathcal{Q}} b^{s_i} = b^{\sum_{i \in \mathcal{Q}} s_i} = b^{osk} = k_0$$

Method ②: b^{gsk_i}. Here, the secret shares are equal to:

$$b^{gsk_i}, i \in \mathcal{Q}$$

with gsk_i the group secret key of user i, also generated during the execution of ETHDKG.

Here, the reconstruction leverages the Lagrange interpolation theorem:

$$\prod_{i \in \mathcal{R}} (b^{gsk_i})^{L_i} = \prod_{i \in \mathcal{R}} b^{gsk_i \times L_i} = b^{\sum_{i \in \mathcal{R}} gsk_i \times L_i} = b^{osk} = k_0$$

with:

$$L_i = \prod_{k \in \mathcal{R} k \neq i} \frac{k}{k - i}, i \in \mathcal{R}, \text{ the Lagrange coefficients.}$$

Both methods come with their advantages and their drawbacks. In the following paragraphs, we explore them and draw their limitations.

A word on commitments. Commitments are, as their name suggests, values users must commit to throughout the protocol. In ETHDKG, they are sent along with the encrypted shares during the distribution phase of the protocol. They are shared through events. Initially, the only data stored in the smart contract are the first commitments of every user i:

$$C_{i,0} = g_1^{s_i},\ i \in \mathcal{Q}$$

along with a hash value of the form:

$$\mathit{commitments\_hash} = \texttt{keccak256}(\texttt{abi.encodePacket}(\mathit{encrypted\_shares}, \mathit{commitments}))$$

This hash value is computed from all the commitments of a certain user i and used, later on, to verify that users send the correct data throughout the protocol.

The other commitments are denoted:

$$C_{i,k},\ 1 \leqslant k \leqslant t$$

For a given user $i \in \mathcal{Q}$, the *encrypted_shares* variable is a list of shares encrypted with a symmetric key encryption function as:

$$\overline{s_{i\to j}}^{k_{ij}},\ j \in \mathcal{Q},\ j \neq i$$

where $s_{i\to j} = f_i(j)$, k_{ij} is a symmetric key obtained from j's public information and i's secret key. (The overline line stresses out the encryption of the data under.)

The commitments are used to verify various proofs, as explained in the following paragraph.

Proof of correct secret share. In addition to the sharing of the secret share (either as b^{s_i} or b^{gsk_i}), we need to enable the users to prove the correctness of the share they hold. To this end, we reuse the ZK proofs system proposed in ETHDKG. The nature of the proof depends on the method chosen.

In method ①, the proof π is:

$$\pi = \mathtt{DLEQ}(g_1, g_1^{s_i}, b, b^{s_i}, s_i)$$

which proves that the secret is s_i without leaking its value. The requirement for the proof is to assume that the commitment to s_i is correct. Yet, the first commitment sent during ETHDKG is:

$$C_{i,0} = g_1^{s_i}, i \in \mathcal{Q}$$

The proof π can be therefore crosschecked against this first commitment. Since all the first commitments are stored in the ETHDKG smart contract, verifying this proof does not cost any additional storage space.

In method ②, the proof π becomes:

$$\pi = \mathtt{DLEQ}(g_1, g_1^{gsk_i}, b, b^{gsk_i}, gsk_i)$$

Yet, to access $g_1^{gsk_i}$, we need to store all the commitments for every user as:

$$g_1^{gsk_i} = g_1^{\sum_{j \in \mathcal{Q}} s_{j \to i}} = \prod_{j \in \mathcal{Q}} \prod_{0 \leqslant k \leqslant t} C_{j,k}^{i^k}$$

In the first case, we need all the secret shares to reconstruct the key. Therefore, all the sub-openers must participate to open a signature. Consequently, if a user disconnects from the protocol, we have to recover their secret s_i. This recovery process drastically increases the communication cost of the protocol as more exchanges are required to recover the missing s_j. In the second case, we only need (at least) $t + 1$ secret shares to reconstruct the key and open the signature, via Lagrange interpolation. At first glance, the second choice is less troublesome and more elegant. Yet, to verify the validity of the secret share given during the opening, the second method requires a lot more storage space, and thus gas, than the first one.

Testing alternative methods for the computation of the shares

The Github project presents four branches for the testing of four methods to compute the secret shares (summarized in Table A.1). The differences between these methods lie in how the share is computed and which commitments are stored in the contract.

Methods ① and ② are described above. The method ① (branch `si`) consists in computing the shares as b^{s_i} and storing only the first commitments $C_{i,0}$ in the smart contract. The method ② (branch `gski_old`) consists in setting the shares equal to b^{gsk_i} storing all the commitments $C_{i,k}, i \in \mathcal{Q}, 0 \leqslant k \leqslant t$ in the smart contract. In both cases, the proving function only has to retrieve the necessary information inside the smart contract to compute the proof π.

Method ③. This method (branch `gski`) consists in computing the shares as b^{gsk_i}. Still, instead of storing the commitments inside the contract, they are sent as parameters of the `DLEQ` function along with the encrypted shares. Yet, thanks to the *commitments_hash* variable computed above, the function can check that the given commitments were used to compute this value in previous processes.

Method ④. This method (branch `gski_alt`) is similar to method ③. However, instead of joining the encrypted shares, it stores the *shares_hash* value along with the *commitments_hash* as two distinct values:

$$\text{commitments\_hash} = \texttt{keccak256}(\texttt{abi.encodePacket}(\text{commitments}))$$

$$\text{shares\_hash} = \texttt{keccak256}(\texttt{abi.encodePacket}(\text{encrypted\_shares}))$$

Comparison In Table A.1, we compare the four methods w.r.t. the storage and the bandwidth occupation parameters. The storage occupation evaluates the space taken inside the smart contract to store the required data. The bandwidth occupation considers the amount of data given in the parameter of the functions. A commitment is represented by a list of two 256-bit-long integers, and therefore *weigh* two times more

Branches	Storage occupation	Bandwidth occupation
① `si`	$n \times 2$ (commitments) $+$ $n\times$ (hashes) $= 3 \times n$	0
② `gski_old`	$n \times (t+1) \times 2$ (commitments) $+$ n (hashes) $\approx n^2 + 3 \times n$	0
③ `gski`	$n \times 2$ (commitments) $+ n$ (hashes) $= 3 \times n$	$n \times (t+1) \times 2$ (commitments) $+$ $n \times (n-1)$ (hashes) $\approx 2 \times n^2 + n$
④ `gski_alt`	$n \times 2$ (commitments) $+$ $n \times 2$ (hashes) $= 4 \times n$	$n \times (t+1) \times 2$ (commitments) $\approx n^2 + 2 \times n$

Table A.1: Comparison between the four methods to compute the secret shares

Branches	Gas consumed
① `si`	3,121,197
② `gski`	4,122,572
③ `gski_old`	4,129,735
④ `gski_alt`	4,687,552

Table A.2: Gas consumption of the execution of the protocol for the four methods

that hashes, which are too represented as 32 bytes. Encrypted shares are represented by one 256-bits integer. Therefore, the comparison is made based on the amount of 256-bit-long integers stored.

In the best case, method ① is the more efficient implementation. However, it still requires all the secret shares, which generates a costly recovery process for each secret that the shares are missing. In Table A.2, we give the results of a test with five honest users, values that confirms that in the best-case scenario, the execution of the protocol is less costly with method ①. We do not observe a notable difference between methods ②, ③ and ④, which is certainly due to the low number of users.

Future testing To further compare these four methods, here are some hypotheses to test.

(H1) *The method ① decreases in efficiency as the number of adversaries increases.*

In order to reveal that, we can perform a test with an important number of users (e.g., 50) and t adversaries.

(H2) *The method ④ becomes better than method 3 as the number of users increases.* Figure A.8 shows the amount of storage space occupied and size of parameters for methods ③ and ④ according to the theoretical analysis. We expect that method ④ performs better than method ③. Performing tests with a significant number of users are necessary.

(H3) *Storing the secrets s_i after their recovery, or the points gsk_i after their computations, should reduce the gas consumed per opening in the long run.* This could be revealed with the evaluation of two subsequent opening of group signatures.

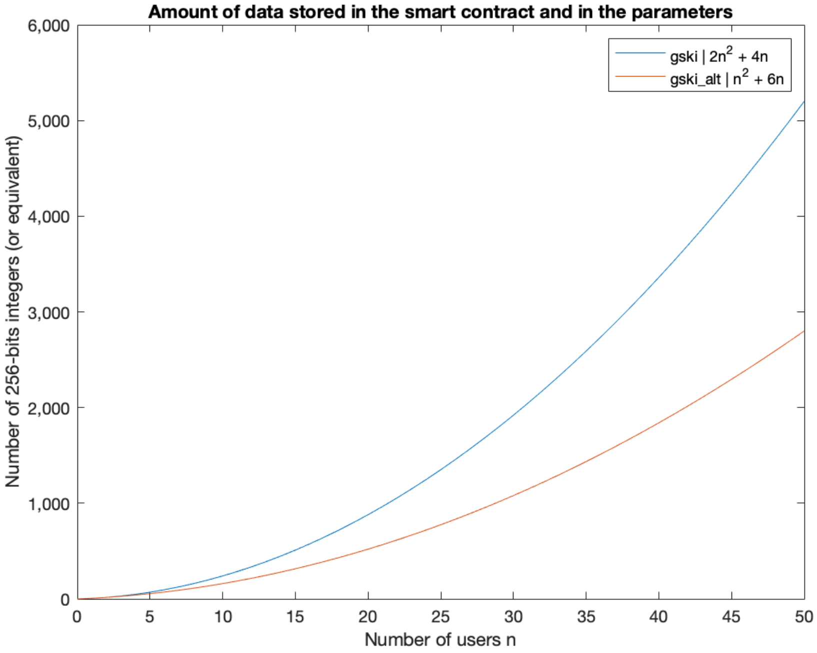


FIGURE A.8: Comparison between Methods 3 and 4 on the expected evolution in the amount of data stored in the smart contract and parameters

Off-chain communications

On-chain or off-chain? During certain parts of the protocol, users can communicate directly with each other and with the Issuer using the methods in `doggy_tcp`.

Communications can also happen on the blockchain, notably via the emission of events. While being in theory possible, it would increase significantly the quantity of gas consumed, making the solution impractical. The choice of making an interaction off-chain or on-chain is generally a compromise between how much gas we are willing to consume and whether or not the data needs to take advantage of the blockchain's immutability.

We chose to send messages off-chain because sending long and many messages will result in heavy gas consumption and longer execution times (due to the delay of validation of blocks). However, we decided to distribute the encrypted shares and the commitments on-chain. We leverage its immutability property and the presence of a consensus to guarantee their integrity. Indeed, when users retrieve the corresponding event, they can be sure they get the correct data. If this distribution was performed off-chain, an adversary could send the wrong data to a particular user (e.g. by performing a man-in-the-middle attack).

Anonymity issues. While being more practical, off-chain communications bring additional problems of anonymity.

Indeed, our initial goal is to enable the users to send messages anonymously by using a group signature scheme to sign their messages. This provides source authenticity without revealing the identity of the users. The $\mathcal{DOGS}$ was designed to this end with the particularity that the Opener should be distributed.

Therefore, to protect the anonymity of the users in off-chain communications, users should be receiving messages from anonymous addresses and only determine whether an authenticated user sent it with a correct group signature. Furthermore, they should not distinguish between two messages written by two different authenticated users, i.e. they cannot identify the source of a message.

Unfortunately, neither feature is respected. Indeed, we use Python sockets to connect users so they can send their messages. Consequently, each user is associated with an IP address. A user can differentiate the origins of two messages. One way to solve that issue is to use a broadcast channel instead. However, users leverage a symmetric encryption scheme with a symmetric key computed from their keys and the receiver's keys to protect the communications. As such, encrypted communications can be uniquely associated with a user, even though the data was broadcast.

This problem of anonymity is further explored in $\mathcal{BAT}-Key$'s design and $\mathcal{TOAD}$'s implementation.

Synchrony issues

While the blockchain provides partial synchrony, it comes with inherent limitations.

Asynchronous transactions. When sending a transaction, especially subsequently to a call to a smart contract function, this transaction must first be included in a block before being processed. Indeed, it is only after the transaction has been included that the data in the smart contract can be updated. However, the time needed for that inclusion depends on various factors related to the nature of the underlying P2P network, the time before consensus, the verification time of the transactions. When using a private blockchain, some of these factors can be fastened, including the time before consensus. If we use a public, Proof-of-Work-based blockchain like the Ethereum mainnet, we need to wait for the miners to include our transaction in a block. Miners will handle transactions faster if the fees associated are higher. However, this suggests that a faster $\mathcal{DOGS}$ protocol is also more expensive.

Forking issues. This issue happens when different blocks are added to the blockchain at the same time. This creates multiple valid states of the blockchain, and transactions can be located in whichever one. It is called a fork. A fork happens naturally and usually do not last very long. At one point, one chain of blocks will grow faster than the others, and this longest chain will be considered as the reference for the valid state of the blockchain. The wrong chains of blocks and transactions they contain are discarded.

This is a big problem in our case. Indeed, if some transactions made during the $\mathcal{DOGS}$ protocol are inserted in the wrong chain of blocks, then users can act as if these transactions are valid, only to find them reverted once the correct chain has been chosen.

Fortunately, to deal with this issue, the ETHDKG protocol introduces some rules to recover some synchronicity. ETHDKG is broken down into phases that are of limited duration (in block number). Users know in advance the duration of each phase and act accordingly. As such, when users interact with the smart contract, they make the corresponding transactions then wait for the end of the current phase. They process the results of the transactions only after waiting. This is called *consensus stabilization* and is done to prevent the issue of forking.

While $\mathcal{DOGS}$ inherits from ETHDKG, some parts of the implementation are not currently resilient against forking. For example, there is no 'phase' during the opening process. It could be implemented but would require more coordination between the users.

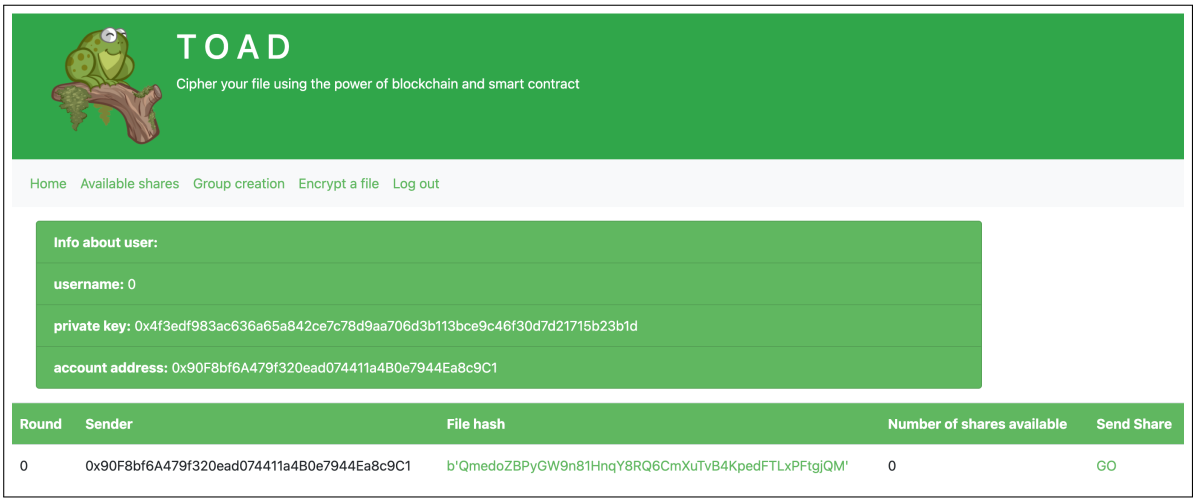
A.2 Implementation of $\mathcal{TOAD}$

We worked on the implementation of $\mathcal{TOAD}$ from November 2020 to March 2021. As explained in Chapter 5, the protocol proposes a ThreshOld encryption scheme that supports a distributed Anonymous-yet-traceable Decryption service. The goal of the protocol is to spread the decryption functionality among a set of decryption servers. Moreover, it aims at protecting the anonymity of the decryption servers to prevent coercion and limit targeted attacks. In addition, the protocol ensures that these servers remain accountable for their action by implementing a traceability mechanism based on a group signature scheme. Similarly to the previous section, we simplified the system model, compared to what was presented in Chapter 7, so as to determine the practicality of this building block.

A.2.1 Scenario

The application allows a group of users to generate a private and public keypair. These keys are distributed over the group members and can be retrieved with the collaboration of at least $t + 1$ of them. The value t is a fixed threshold between 1 and the number of members in the group.

Thereafter, any user can encrypt a file using the public key of the group. Then, for decryption to happen, the group members must put their shares of the corresponding secret key in common. At least $t+1$ members are required to decrypt the file. Once this number is reached, any user can access the plaintext of the encrypted file. These functionalities are provided through a contract deployed on the Ethereum testnest, called TOAD and detailed below. Finally, the client web application (shown in Figure A.9) enables the users to obliviously manage the interactions with these contracts.


TOAD
Cipher your file using the power of blockchain and smart contract
Home
Available shares
Group creation
Encrypt a file
Log out

Info about user:

username: 0

private key: 0x4f3edf983ac636a65a842ce7c78d9aa706d3b113bce9c46f30d7d21715b23b1d

account address: 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1

Round	Sender	File hash	Number of shares available	Send Share
0	0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1	b'QmedoZBPyGW9n81HnqY8RQ6CmXuTvB4KpedFTLxPFtgjQM'	0	GO

Figure A.9: Screenshot of $\mathcal{TOAD}$'s client web application

Deployment. We assume that a blockchain is already up and deployed. This is done in practice via the `launch.sh` script. The script performs the following actions:

- activate a Python virtual environment;
- run a ganache-cli with deterministic keys and IDs for 20 accounts;
- compile and deploy the TOAD contract;
- launche the script `event_retrieve.py` to capture the blockchain events;
- run the web application that enables users to encrypt/decrypt files.

Registration and subsequent actions. The users register via a web application by giving their username and public/private blockchain keypair. Once registered, they can perform the following actions:

- create a group: this action generates the creation of the group of N decryption servers, enumerated with $0 \leqslant i \leqslant N$, and their corresponding new encryption/decryption keypair $gek/\{gsk_i\}_i$ via a blockchain-based DKG protocol.
- encrypt a file: a user can upload a file via the web portal and encrypt it with the newly generated encryption key gek (public output of the DKG). Once encrypted, the ciphertext is sent onto the IPFS for storage.
- send a share: when a decryption server wants to participate in the decryption, it can send its share of the encryption key (the one used to encrypt the file) by clicking the corresponding button.
- decrypt a file: once enough key shares have been sent, the pool of shares is combined to obtain the decryption key. Then, the users can individually download and decipher the ciphertext.

A.2.2 Implementation choices

Similarly to $\mathcal{DOGS}$, the implementation of $\mathcal{TOAD}$ requires to distinguish between the client program and the web interface regrouped as a client application and the smart contract that supports the connection to the local blockchain.

The client application

The client application is divided into two parts: the client program that ensures the cryptographic operations, manages interactions with other peers and the IPFS server, triggers the calls to the smart contract; and the web interface that enables oblivious access to the client program's functionalities.

The client program is implemented in **Python version 3.7**. $\mathcal{TOAD}$'s implementation leverages **AES-128** with **CCM** mode (Counter with Cipher block chaining

Message authentication code), the optimized **BN128** elliptic curve, the **SHA256** hash function (Python libraries: `Crypto` and `py_ecc`).

The web application is supported by **Flask**, a web development open-source framework written in Python, and leverages the **`werkzeug`** web application library. This library manages the communications between a web server and the web apps and creates page **`html`** templates via the Web Server Gateway Interface (WSGI). In addition, the web app initializes a sqlite3 database that enables the users' registration. Users that are not registered cannot participate in the $\mathcal{TOAD}$'s protocol.

The smart contract

There is only one smart contract called
textsfTOAD. It contains eight functions:

- `group_creation`: this function retrieves the data sent by the client program and pushes them onto the blockchain (as a storage variable stored in the contract and persistent over time). In addition, it emits the `Group Creation` event.

- `publish_tpk`: This function forwards the temporary public keys given by the client program to other users via the emission of the `Public Key` event. No data are published; they are broadcast off-chain along with the event.

- `encrypt_shares`: This function ensures the fowarding of the encrypted shares of user i's secret s_i via the sending of $f_i(tpk_j)$ where f_i is i's secret polynomial and tpk_j j's temporary public key (received previously). The data is also broadcast under the form of an event.

- `publish_group_key`: This function concludes the initialization of the group by publishing onto the blockchain the resulting group public key gpk. The data is pushed onto the ledger (also as a storage variable).

- `send_file`: This function informs; via the emission of an event, other users that an encrypted file has been shared and accessible via the IPFS.

- `send_share`: This function triggers the broadcast of partial decryption data related to the previously mentioned encrypted file via the emission of an event.

The full project is available on Github <link>. It also starts from a fork of ETHDKG and improves the initial construction of a blockchain-based DKG protocol with new cryptographic functionalities and security properties with small gas cost and complexity overheads (see Sub-section A.2.4).

A.2.3 Application to the generation of the group's keypair

The generation of the private and public keypair for the selected group is done using the ETHDKG protocol. The protocol goes as presented in the previous section. Once the execution is finished, every group member has a part, also called share, of the keypair, denoted (gpk_i, gsk_i). The resulting group keys denoted gek for 'group encryption key', and gdk for 'group decryption key' are computed as:

$$gdk = \sum_{P_i \in \mathcal{R}} gsk_i \cdot \prod_{k \neq i} \frac{k}{k-i}$$

$$gek = \prod_{P_i \in \mathcal{R}} gpk_i^{\sum_{k \neq i} \frac{k}{k-i}}$$

with $\mathcal{R}$ still a subset of $\mathcal{Q}$ with $t+1$ honest group members. In practice, gdk is never reconstructed.

In the following, we describe the implementation of the threshold cryptosystem. The cryptographic operations are handled by the client application and can be found in the `crypto_utils.py` Python program.

Let m be the file to encrypt, $\mathbb{G}$ be the same group as in ETHDKG (from which the keypairs are extracted) and h a generator of this group.

Encryption. The protocol for the encryption of m works as follows:

1. The client chooses a random integer r_0

2. It computes $k_{point} = h^{r_0}$

3. The symmetric key is derived as:

   $$k_0 = \texttt{HKDF}(k_{point},\ \text{key length} = 32\ \text{bytes})$$

   The function HKDF is a key derivation function based on Hash-based Message Authentication Codes (HMAC) implemented in Python. It extracts a pseudo-random key using a HMAC hash function on the salt k_{point} and the weak input key (the empty string `''` converted in bytes).

4. Once the key is generated, the client encrypts its message m with AES [218] as:

   $$(c, n) = \texttt{AES}(k_0, m)$$

   where c is the ciphertext and n a nonce. The ciphertext is sent to the IPFS for public storage.

5. Then, the client selects another random integer r_1 and computes:

   $$c_1 = h^{r_1}$$

   and

   $$c_2 = k_{point} \times gpk^{r_1}$$

   The values c_1 and c_2 are known as the two outputs of an ElGamal cryptosystem [154]. They are finally stored in the blockchain by calling the `send_msg` function of the TOAD smart contract.

Distributed decryption. The decryption of the ciphertext will occur if and only if $t + 1$ decryption servers collaborate.

1. We consider that $t+1$ decryption servers send their $c_1^{gsk_i}$ share along with a proof of validity by calling the `send_share` function in
   textsfTOAD contract.

2. Upon reception of these shares, any user can retrieve k_{point} by computing:

$$\frac{c_2}{\prod_{P_i \in \mathcal{R}} (c_1^{gsk_i})^{\prod_{k \neq i} \frac{k}{k-i}}}$$

3. Then, to actually decrypt the ciphertext, the client computes:

$$\texttt{HKDF}(k_{point},\ \text{key length} = 32\ \text{bytes})$$

and obtains k_0. Finally, it downloads the ciphertext from the IPFS and applies `AES` with k_0 to obtain the original plaintext.

A.2.4 Evaluation and Discussion

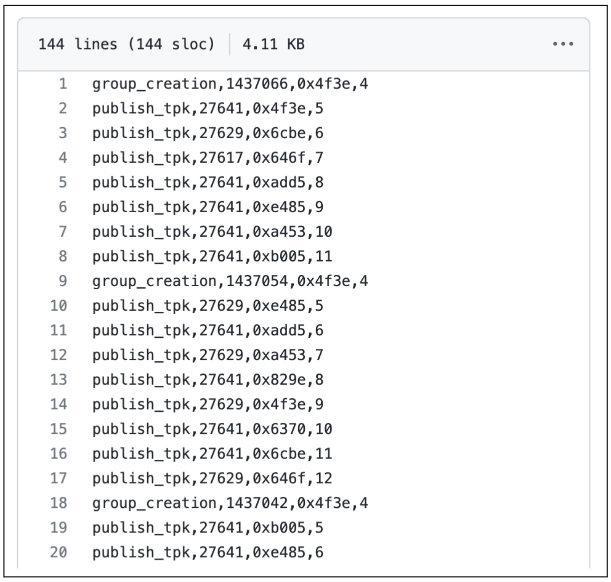
The following subsection is a discussion of $\mathcal{TOAD}$'s implementation performances. It illustrates the small gas cost and complexity overheads generated by the addition of the blockchain-based anonymous threshold decryption service compared to the initial ETHDKG protocol.

The metrics. Similarly to $\mathcal{DOGS}$, $\mathcal{TOAD}$'s implementation is evaluated over two metrics: the *time of execution*, and the *gas consumption.*To evaluate these metrics, we use wrappers and decorators to fill a `csv` file. Then we post-process the results via a python script and the `matplotlib` library to generate the graphs. Raw data are shown in Figure A.10.

Experiments. We show the results of the execution of $\mathcal{TOAD}$'s implementation considering $n = 2$, 4, 6, 8 and 10 group members. The threshold is always set to $t = n/2$. We do not consider the network aspect of P2P communications.

Analysis of the gas consumption

Gas consumed per protocol phase. Figure A.12 illustrates the evolution of the gas consumption per smart contract's function w.r.t. the number of users. There are



```
group_creation,1437066,0x4f3e,4
publish_tpk,27641,0x4f3e,5
publish_tpk,27629,0x6cbe,6
publish_tpk,27617,0x646f,7
publish_tpk,27641,0xadd5,8
publish_tpk,27641,0xe485,9
publish_tpk,27641,0xa453,10
publish_tpk,27641,0xb005,11
group_creation,1437054,0x4f3e,4
publish_tpk,27629,0xe485,5
publish_tpk,27641,0xadd5,6
publish_tpk,27629,0xa453,7
publish_tpk,27641,0x829e,8
publish_tpk,27629,0x4f3e,9
publish_tpk,27641,0x6370,10
publish_tpk,27641,0x6cbe,11
publish_tpk,27629,0x646f,12
group_creation,1437042,0x4f3e,4
publish_tpk,27641,0xb005,5
publish_tpk,27641,0xe485,6
```

FIGURE A.10: Excerpt of the raw data file for the gas consumption of *TOAD*'s implementation with 10 group members

six identifiable functions which are listed below along with their mapping to the smart contract's functions described above:

- *group creation* — `group_creation`
- *publish tpk* — `publish_tpk`
- *encrypt shares* — `encrypt_shares`
- *publish group key* — `publish_group_key`
- *file encryption* — `send_file`
- *file decryption* — `send_share`

We observe two things. Firstly, the cost of the group creation function prevails over the others. This phenomenon is due to the publication of data onto the blockchain. Indeed, the method `push` called inside the `group_creation` function stores the registration data directly inside the contract. More specifically, the data pushed are the `encrypted_accounts` of the selected group members. Secondly, we can also assert that the gas consumption related to group creation evolves linearly in the number of group members. This statement is corroborated by Figure A.13 and corresponding analysis.

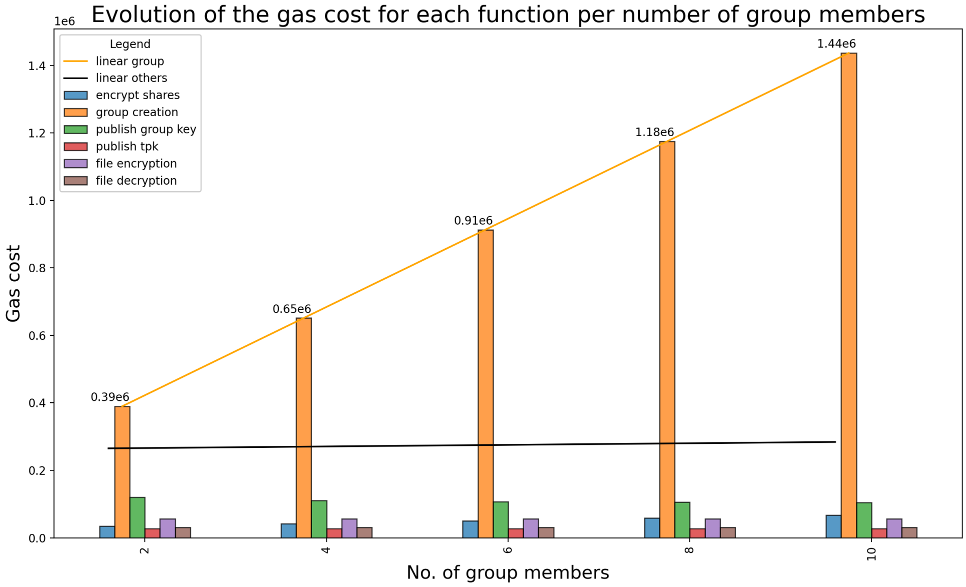


Figure A.11: Evolution of the gas consumption per smart contract's function w.r.t. the number of group members

Cumulative Gas Consumption. On Figure A.12, we illustrate the cumulative gas consumption over one execution of the $\mathcal{TOAD}$ protocol. The background illustrates the several phases associated with each smart contract's function, and the black line represents the evolution of the gas consumption w.r.t. the block number. The graph reveals three things. Firstly, we note that three phases are repeated. They correspond to the `publish_tpk`, `encrypt_shares`, and `publish_gpk` functions. This is explained by the fact that after each encryption, the shares of the same group public key are regenerated to prevent data collection and lead to the reconstruction of one group's secret polynomial (via Lagrange Interpolation). Secondly, we observe that the biggest variation occurs during the group creation phase and that the second biggest deviation happens during the publish group key phase, which both confirm the previous observations. Fortunately, the group creation phase is performed only once as a setup phase. Finally, the gas consumption is likely to grow indefinitely since the `publish_group_key` function is called periodically after each encryption and requires the publication of the new *gpk* key.

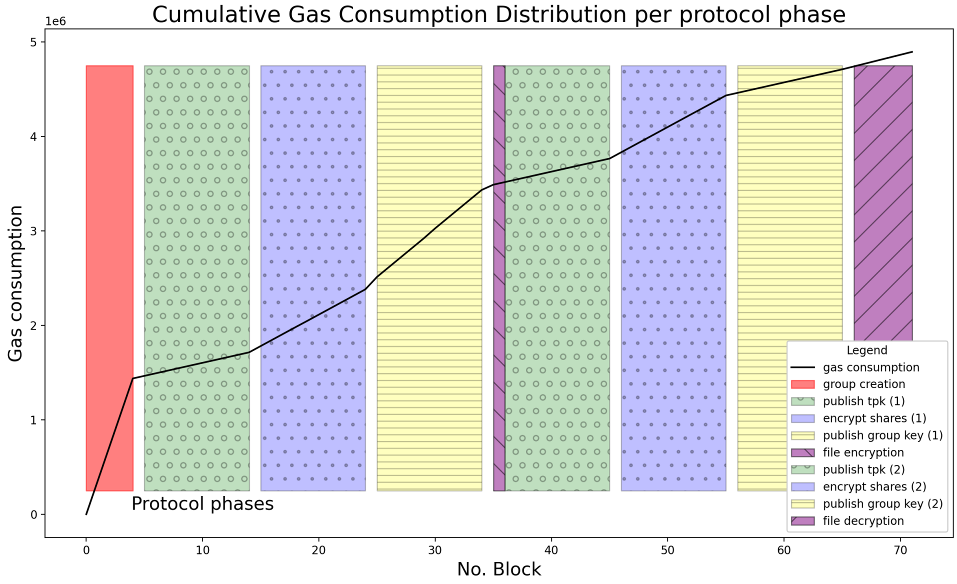


Figure A.12: Evolution of the gas consumption for $N = 10$ group members

Correlation between gas consumption and size of transiting data. Figure A.13 illustrates the correlation between the gas it costs to execute a smart contract function and the amount of data that function handles. We can observe that the group creation function inherited from ETHDKG is the most expensive function. Comparatively, all new functions that were added to build $\mathcal{TOAD}$, i.e. from the function that ensures the publication of the temporary public keys for *user anonymity* (`publish_tpk`) to the encryption and decryption functions, have little impact on the gas consumption or the amount of data generated. We make an exception for the *encrypt shares* function. We recall that it is used by user i to distribute the shares $s_{i \to j} = f_i(tpk_j)$ of its secret s_i in a *confidential* way. As such, the function handles a lot of data, but none is written inside the contract. This subtlety explains why, unlike the group creation one, the encrypt shares function does not cost a lot of gas.

Analysis of the execution time

Block production rate over time. In Ganache, the transactions are mined instantly and each transaction triggers the creation of a new block. In Figure A.14, we

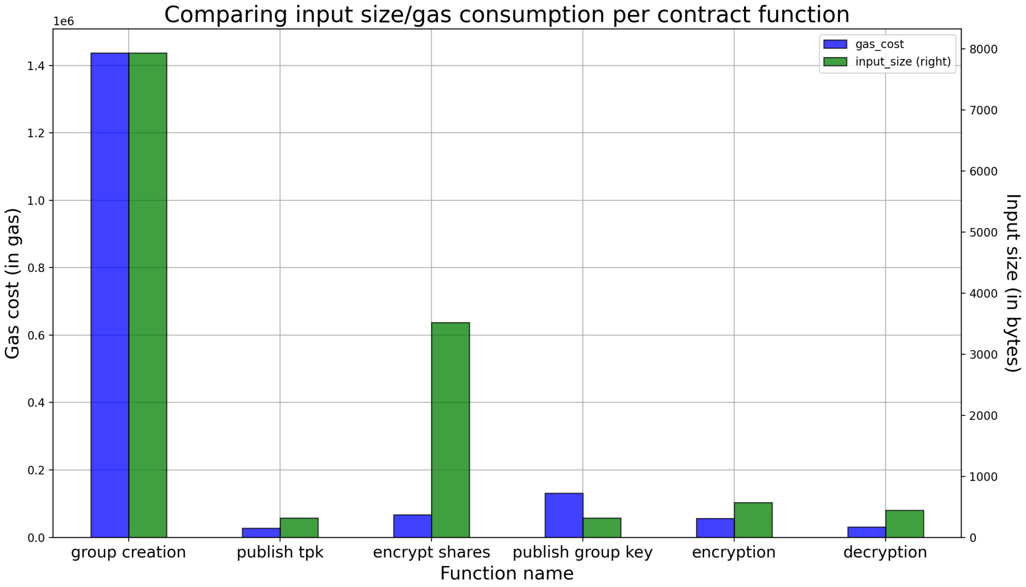


Figure A.13: Illustrating the correlation between the gas consumption of a smart contract function and the amount of data it handles.

show the block production rate over time and correlate it to the protocol phases and smart contract's functions. We observe three bursts in the production rate. The first one occurs only once at the beginning: it corresponds to the deployment of the TOAD smart contract and related migrations. Then, there are two similar outbreaks: it coincides with the execution of the three functions `publish_tpk`, `encrypt_shares` and `publish_group_key`. This is explained as follows. Upon execution of the group creation function (red dot numbered 4), the protocol automatically triggers the sharing of the temporary identifiers tpk_i, then of the encrypted shares $s_{i \to j} = f_i(tpk_j)$, and finally the publication of the resulting group public key gpk. When this group key gpk is used to encrypt a file, the protocol automatically starts generating a new group public key with the same group members. The temporary public keys tpk_j and related shares are used only once to preserve the anonymity of the group members participating in two subsequent decryption processes.

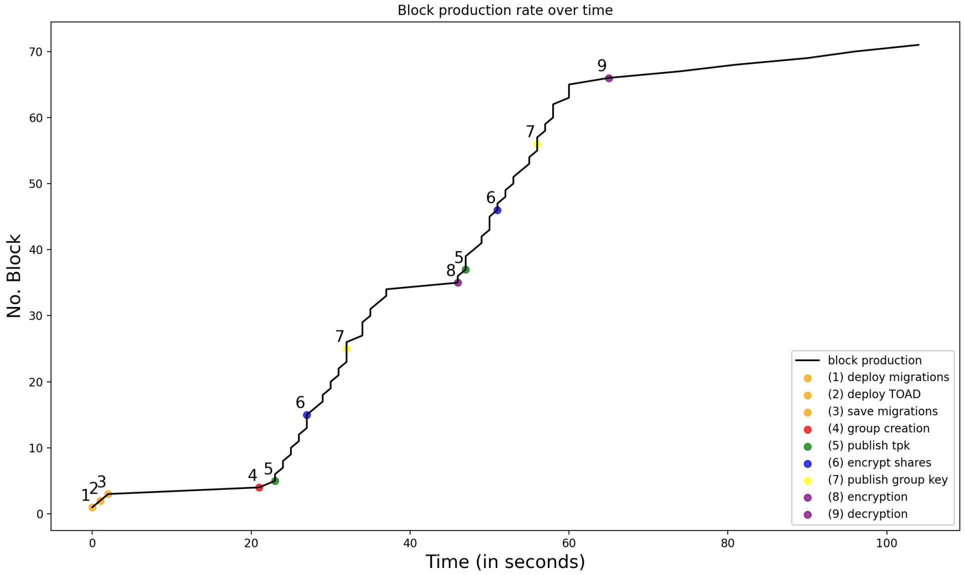


FIGURE A.14: Evolution of the production rate over time for $N = 10$ group members

Amount of data transiting in the contract over time and bandwidth occupation. We illustrate once again the correlation between the block production rate and the amount of data transiting in the smart contract with Figure A.15. The figure shows the cumulative amount of data shared through the contract for the $N = 10$ users (black line) and the corresponding bandwidth occupation over time (blue area). It reveals that the most costly operation in terms of communications comes from the encrypt shares function. Indeed, the function enables the sending of

$$t \times N = (N/2 + 1) \times N = 60 \text{ for } N = 10$$

shares encrypted with AES. The cross-comparison between Figures A.13 and A.15 confirms that each user sending 3,500 bytes of data during the `encrypt_shares` function amounts to approximately 35 kilobytes in total for the same phase.

From both Figures A.14 and A.15, we see that:

- The group public key is available in about 10 seconds (group creation at 21 and publication of the first *gpk* at 32 seconds for the first, and respectively 46 and 56

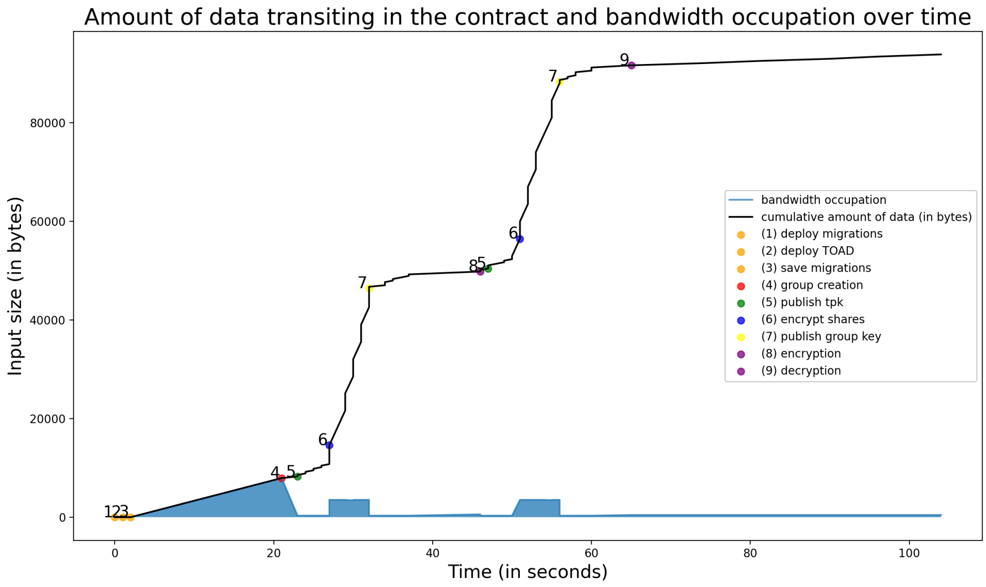


Figure A.15: Evolution of the amount of data transiting through the contract over time for $N = 10$ group members

for the second generation).

- Decryption occurs in 57-58 seconds and generates 15 Kb of data.
- Most of the data (around 92%) is shared in 9 seconds (from 6 to 7) and is about 31 kilobytes (approximately the size of a small text file).
- The execution of the protocol, from the group creation to the decryption of one file, takes about 100 seconds (1 min and 40 seconds) and 93 kilobytes.

A.2.5 Future Work

The current implementation of $\mathcal{TOAD}$ could be improved in the two following ways.

Adversaries

As explained above, the users are supposed to be honest. The implementation preserves the anonymity of these honest users w.r.t. an external eavesdropper that may want to infer the identity of the decryption servers. The implementation of $\mathcal{TOAD}$ would benefit from the development of stronger adversaries. Firstly, it could be interesting to analyze the same metrics of gas consumption and time of execution when considering the same adversaries as the ones programmed in $\mathcal{DOGS}$'s implementation. Then, it could be interesting to design more specific adversary profiles that would target the encryption/decryption functionalities of the implementation to reveal information about the decryption servers.

Optimization

We envision two main axes for optimizing the implementation: ① speeding up the decryption process; and ② reducing the communication cost of the `encrypt_shares` function.

① Speed of decryption. An easy modification to speed up the decryption of a file would be to swap the (9) decryption function with the (5-7) functions. However, this would bring the problem that the same decryption servers cannot be solicited twice in a row as the switch would incur additional delays before publication a new *gpk*.

② Reduction of communication costs The most expensive phase is the sharing of user i's shares of its secret s_i. We could explore data aggregation techniques [219] to compress the broadcast messages (the $f_i(tpk_j)$) without loss of information.

“I was taught that the way of progress was neither swift nor easy.”

— Marie Curie

References

[1] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K.-K. R. Choo. *Sidechain technologies in blockchain networks: An examination and state-of-the-art review.* Journal of Network and Computer Applications **149**, 102471 (2020). xix, 41

[2] ETSI. *102 637-3, intelligent transport systems (its); vehicular communications; basic set of applications; part 3: Specification of decentralized environmental notification basic service* (2010). xx, 155, 157

[3] World Health Organization. *Global status report on road safety 2018* (2018). Last accessed 15 September 2021, URL `http://www.who.int/violence_injury_prevention/road_safety_status/2015/en`. 2

[4] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan. *A comprehensive survey on vehicular ad hoc network.* Journal of network and computer applications **37**, 380 (2014). 2, 29

[5] ETSI. *Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service* (2011). 2, 23

[6] ETSI. *302 637-3 v1. 2.2 (2014-11), intelligent transport systems (its); vehicular communications; basic set of applications; part 3: Specifications of decentralized environmental notification basic service* (2014). 2, 23, 144, 145

[7] H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti. *Vanet security challenges and solutions: A survey.* Vehicular Communications **7**, 7 (2017). 2, 29

[8] Z. El-Rewini, K. Sadatsharan, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan. *Cybersecurity challenges in vehicular communications.* Vehicular Communications **23**, 100214 (2020). 2

[9] ETSI. *101 539-1 v1. 1.1: Intelligent transport systems (its)* (2013). 3, 145

[10] ETSI. *102 941 v1. 1.1—intelligent transport systems (its); security; trust and privacy management* (2019). 3

[11] C.-Y. Chow and M. F. Mokbel. *Trajectory privacy in location-based services and data publication.* ACM Sigkdd Explorations Newsletter **13**(1), 19 (2011). 3

[12] L. Lamport, R. Shostak, and M. Pease. *The byzantine generals problem.* In *Concurrency: the Works of Leslie Lamport*, pp. 203–226 (2019). 4

[13] S. Nakamoto *et al.* *Bitcoin: A peer-to-peer electronic cash system* (2008). 4, 32, 40, 135

[14] M. Fleder, M. S. Kester, and S. Pillai. *Bitcoin transaction graph analysis.* arXiv preprint arXiv:1502.01657 (2015). 4

[15] N. Saberhagen. *Monero: Crypto v2. 0.* Whitepaper Database, Oct (2013). 4

[16] S. Noether, A. Mackenzie, *et al.* *Ring confidential transactions.* Ledger **1**, 1 (2016). 4

[17] N. T. Courtois and R. Mercer. *Stealth address and key management techniques in blockchain systems.* ICISSP **2017**, 559 (2017). 4

[18] A. Ali, M. Khan, M. Saddique, U. Pirzada, M. Zohaib, I. Ahmad, and N. Debnath. *Tor vs i2p: A comparative study.* In *2016 IEEE International Conference on Industrial Technology (ICIT)*, pp. 1748–1751 (IEEE, 2016). 4

[19] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox. *Zcash protocol specification.* GitHub: San Francisco, CA, USA (2016). 4

[20] I. Miers, C. Garman, M. Green, and A. D. Rubin. *Zerocoin: Anonymous distributed e-cash from bitcoin.* In *2013 IEEE Symposium on Security and Privacy*, pp. 397–411 (IEEE, 2013). 4

[21] J. Camenisch and A. Lysyanskaya. *Dynamic accumulators and application to efficient revocation of anonymous credentials.* In *Annual international cryptology conference*, pp. 61–76 (Springer, 2002). 4

[22] E. Duffield and D. Diaz. *Dash: A privacy centric cryptocurrency* (2015). 4

[23] E. Duffield and K. Hagan. *Darkcoin: Peertopeer cryptocurrency with anonymous blockchain transactions and an improved proofofwork system.* bitpaper.info (2014). 4

[24] R. Viglione and R. Versluis. *Horizen: unbounded by design* (2017). Last accessed in November 2021, URL `https://www.horizen.io/fr/`. 4

[25] A. Zaidelson. *Beam scalable confidential cryptocurrency* (2018). Last accessed in November 2021, URL `https://beam.mw/`. 4

[26] Q. Zhou, H. Huang, Z. Zheng, and J. Bian. *Solutions to scalability of blockchain: A survey.* IEEE Access **8**, 16440 (2020). 5, 40

[27] C. Fan, S. Ghaemi, H. Khazaei, and P. Musilek. *Performance evaluation of blockchain systems: A systematic survey.* IEEE Access **8**, 126927 (2020). 5, 40

[28] I.-C. Lin and T.-C. Liao. *A survey of blockchain security issues and challenges.* IJ Network Security **19**(5), 653 (2017). 5, 40

[29] ETSI. *302 637-2 v1. 3.1-intelligent transport systems (its)* (2014). 6

[30] G. Yang, D. S. Wong, X. Deng, and H. Wang. *Anonymous signature schemes.* In *International Workshop on Public Key Cryptography*, pp. 347–363 (Springer, 2006). 7

[31] R. L. Rivest, A. Shamir, and Y. Tauman. *How to share a secret.* In *Communications of the ACM* (Citeseer, 1979). 7, 31

[32] D. Chaum and E. Van Heyst. *Group signatures.* In *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 257–265 (Springer, 1991). 7, 8, 50, 122, 160

[33] C. Lin, K. Liu, B. Xu, J. Deng, C. W. Yu, and G. Wu. *Vclt: An accurate trajectory tracking attack based on crowdsourcing in vanets.* In *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 297–310 (Springer, 2015). 9

[34] M. Bellare, H. Shi, and C. Zhang. *Foundations of group signatures: The case of dynamic groups.* In *Cryptographers' Track at the RSA Conference*, pp. 136–153 (Springer, 2005). 9, 50, 52, 77, 80, 85, 92, 94, 95, 111, 118, 134, 160

[35] Y. Sakai, J. C. Schuldt, K. Emura, G. Hanaoka, and K. Ohta. *On the security of dynamic group signatures: Preventing signature hijacking.* In *International Workshop on Public Key Cryptography*, pp. 715–732 (Springer, 2012). 9

[36] J. Blömer, J. Juhnke, and N. Löken. *Short group signatures with distributed traceability.* In *International Conference on Mathematical Aspects of Computer and Information Sciences*, pp. 166–180 (Springer, 2015). 9, 55

[37] E. Ghadafi. *Efficient distributed tag-based encryption and its application to group signatures with efficient distributed traceability.* In *International Conference on Cryptology and Information Security in Latin America*, pp. 327–347 (Springer, 2014). 9, 55

[38] P. Schindler, A. Judmayer, N. Stifter, and E. R. Weippl. *Ethdkg: Distributed key generation with ethereum smart contracts.* IACR Cryptol. ePrint Arch. **2019**, 985 (2019). 9, 63, 106, 111, 112, 126, 127, 132, 135, 165, 184, 185

[39] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. *Secure distributed key generation for discrete-log based cryptosystems.* In *International Conference on*

the Theory and Applications of Cryptographic Techniques, pp. 295–310 (Springer, 1999). 10, 60, 85, 90, 94, 126, 131, 132

[40] T. P. Pedersen. *A threshold cryptosystem without a trusted party.* In *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 522–526 (Springer, 1991). 10, 12

[41] A. Kate, Y. Huang, and I. Goldberg. *Distributed key generation in the wild.* IACR Cryptol. ePrint Arch. **2012**, 377 (2012). 10, 62

[42] P. Voigt and A. Von dem Bussche. *The eu general data protection regulation (gdpr).* A Practical Guide, 1st Ed., Cham: Springer International Publishing **10**, 3152676 (2017). 10

[43] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille. *Enabling blockchain innovations with pegged sidechains.* URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains **72** (2014). 11, 40

[44] Y. G. Desmedt. *Threshold cryptography.* European Transactions on Telecommunications **5**(4), 449 (1994). 12

[45] A. Shamir. *How to share a secret.* Communications of the ACM **22**(11), 612 (1979). 12, 66

[46] *Car 2 car – communication consortium: Manifesto.* Last accessed in November 2021, URL `https://www.car-2-car.org/index.php?id=31`. 21

[47] S. Zeadally, R. Hunt, Y.-S. Chen, A. Irwin, and A. Hassan. *Vehicular ad hoc networks (vanets): status, results, and challenges.* Telecommunication Systems **50**(4), 217 (2012). 21

[48] W. J. Mitchell, C. E. Borroni-Bird, and L. D. Burns. *Reinventing the automobile: Personal urban mobility for the 21st century* (MIT press, 2010). 21

[49] ETSI. *202 663-v1. 1.0-intelligent transport systems (its); european profile standard for the physical and medium access control layer of intelligent transport systems operating in the 5 ghz frequency band* (2009). 23

[50] ETSI. *302 636-5-1 v1. 2.1. intelligent transport systems (its); vehicular communications; geonetworking* (2014). 23

[51] ETSI. *102 894-1 intelligent transport systems (its)* (2013). 23

[52] ETSI. *102 638 intelligent transport systems (its); vehicular communications; basic set of applications; definitions* (2009). 24

[53] ETSI. *102 940: Intelligent transport systems (its); security; its communications security architecture and security management* (2012). 24

[54] M. Raya and J.-P. Hubaux. *Securing vehicular ad hoc networks.* Journal of computer security **15**(1), 39 (2007). 24, 26

[55] M. N. Mejri, J. Ben-Othman, and M. Hamdi. *Survey on vanet security challenges and possible cryptographic solutions.* Vehicular Communications **1**(2), 53 (2014). 27, 29

[56] F. Cunha, L. Villas, A. Boukerche, G. Maia, A. Viana, R. A. Mini, and A. A. Loureiro. *Data communication in vanets: Protocols, applications and challenges.* Ad Hoc Networks **44**, 90 (2016).

[57] P. K. Singh, S. K. Nandi, and S. Nandi. *A tutorial survey on vehicular communication state of the art, and future research directions.* Vehicular Communications **18**, 100164 (2019). 24, 29

[58] J. Blum and A. Eskandarian. *The threat of intelligent collisions.* IT professional **6**(1), 24 (2004). 25

[59] A. Aijaz, B. Bochow, F. Dötzer, A. Festag, M. Gerlach, R. Kroh, and T. Leinmüller. *Attacks on inter vehicle communication systems-an analysis.* Proc. WIT **20**, 189 (2006). 26

[60] M. Houmer and M. L. Hasnaoui. *A risk and security assessment of vanet availability using attack tree concept.* International Journal of Electrical & Computer Engineering (2088-8708) **10**(6) (2020). 26

[61] T. Lu, Z. Du, and Z. J. Wang. *A survey on measuring anonymity in anonymous communication systems.* IEEE Access **7**, 70584 (2019). 29, 117

[62] M. A. Al-Shareeda, M. Anbar, I. H. Hasbullah, and S. Manickam. *Survey of authentication and privacy schemes in vehicular ad hoc networks.* IEEE Sensors Journal **21**(2), 2422 (2020). 29, 30

[63] P. Papadimitratos, V. Gligor, and J.-P. Hubaux. *Securing vehicular communications-assumptions, requirements, and principles.* In *Proc. ESCAR'06* (2006). 29

[64] J. Petit, F. Schaub, M. Feiri, and F. Kargl. *Pseudonym schemes in vehicular networks: A survey.* IEEE communications surveys & tutorials **17**(1), 228 (2014). 29

[65] I. Ali, A. Hassan, and F. Li. *Authentication and privacy schemes for vehicular ad hoc networks (vanets): A survey.* Vehicular Communications **16**, 45 (2019). 29

[66] A. Boualouache, S.-M. Senouci, and S. Moussaoui. *Privanet: An efficient pseudonym changing and management framework for vehicular ad-hoc networks.* IEEE Transactions on Intelligent Transportation Systems **21**(8), 3209 (2019). 30

[67] D. Chaum. *Untraceable electronic mail, return addresses, and digital pseudonyms.* Communications of the ACM **24**(2), 84 (1981). 30

[68] D. Chaum. *The dining cryptographers problem: Unconditional sender and recipient untraceability.* Journal of cryptology **1**(1), 65 (1988). 30

[69] A. R. Beresford and F. Stajano. *Location privacy in pervasive computing.* IEEE Pervasive computing **2**(1), 46 (2003). 30

[70] A. Pfitzmann and M. Hansen. *Anonymity, unlinkability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology* (2005). 30, 115

[71] K. Sampigethaya, L. Huang, M. Li, R. Poovendran, K. Matsuura, and K. Sezaki. *Caravan: Providing location privacy for vanet*. Tech. rep., Washington Univ Seattle Dept of Electrical Engineering (2005). 30

[72] C. Kalaiarasy, N. Sreenath, and A. Amuthan. *Location privacy preservation in vanet using mix zones–a survey*. In *2019 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–5 (IEEE, 2019). 30, 147

[73] R. Yamazaki, M. Yoshida, and H. Shigeno. *A dynamic mix-zone scheme considering communication delay for location privacy in vehicular networks*. In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pp. 245–250 (IEEE, 2021). 30

[74] H. Okada, S. Yamasaki, and V. Bracamonte. *Proposed classification of blockchains based on authority and incentive dimensions*. In *2017 19th international conference on advanced communication technology (icact)*, pp. 593–597 (IEEE, 2017). 35

[75] M. Walport. *Distributed ledger technology: beyond block chain (a report by the uk government chief scientific adviser)*. UK Government (2016). 35

[76] M. C. Ballandies, M. M. Dapp, and E. Pournaras. *Decrypting distributed ledger design-taxonomy, classification and blockchain community evaluation*. arXiv preprint arXiv:1811.03419 (2018). 35

[77] M. Garriga, M. Arias, and A. D. Renzis. *Blockchain and cryptocurrency: A comparative framework of the main architectural drivers* (2018). 1812.08806. 37

[78] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou. *A survey of distributed consensus protocols for blockchain networks*. arXiv preprint arXiv:1904.04098 (2019). 37

[79] E. Bellini, Y. Iraqi, and E. Damiani. *Blockchain-based distributed trust and reputation management systems: A survey.* IEEE Access **8**, 21127 (2020). 37

[80] R. Wille. *Restructuring lattice theory: an approach based on hierarchies of concepts.* In *International conference on formal concept analysis*, pp. 314–339 (Springer, 2009). 37

[81] J. Dai and M. A. Vasarhelyi. *Toward blockchain-based accounting and assurance.* Journal of Information Systems **31**(3), 5 (2017). 37

[82] S. Kozlowski. *An audit ecosystem to support blockchain-based accounting and assurance.* In *Continuous Auditing* (Emerald Publishing Limited, 2018). 37

[83] B. S. Tan and K. Y. Low. *Blockchain as the database engine in the accounting system.* Australian Accounting Review **29**(2), 312 (2019).

[84] S. B. Kahyaoğlu and T. Aksoy. *Survey on blockchain based accounting and finance algorithms using bibliometric approach.* 21st Century Approaches to Management and Accounting Research (2021). 37

[85] J. Zhang, S. Zhong, T. Wang, H.-C. Chao, and J. Wang. *Blockchain-based systems and applications: a survey.* Journal of Internet Technology **21**(1), 1 (2020). 38

[86] Y. Abuidris, R. Kumar, and W. Wenyong. *A survey of blockchain based on e-voting systems.* In *Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications*, pp. 99–104 (2019). 38

[87] R. Taş and Ö. Ö. Tanrıöver. *A systematic review of challenges and opportunities of blockchain for e-voting.* Symmetry **12**(8), 1328 (2020). 38

[88] R. Hanifatunnisa and B. Rahardjo. *Blockchain based e-voting recording system design.* In *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, pp. 1–6 (IEEE, 2017). 38

[89] F. P. Hjálmarsson, G. K. Hreioarsson, M. Hamdaqa, and G. Hjálmtỳsson. *Blockchain-based e-voting system.* In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 983–986 (IEEE, 2018). 38

[90] S. Goundar, R. Khan, K. Chand, E. Reddy, and S. Raja. *Blockchain-based e-voting application.* In *Blockchain Technologies, Applications and Cryptocurrencies: Current Practice and Future Trends*, pp. 169–188 (World Scientific, 2021). 39

[91] T. Hardin and D. Kotz. *Blockchain in health data systems: A survey.* In *2019 sixth international conference on internet of things: Systems, management and security (IOTSMS)*, pp. 490–497 (IEEE, 2019). 39

[92] E. J. De Aguiar, B. S. Faiçal, B. Krishnamachari, and J. Ueyama. *A survey of blockchain-based strategies for healthcare.* ACM Computing Surveys (CSUR) **53**(2), 1 (2020).

[93] B. Houtan, A. S. Hafid, and D. Makrakis. *A survey on blockchain-based self-sovereign patient identity in healthcare.* IEEE Access **8**, 90478 (2020). 39

[94] J. Vora, A. Nayyar, S. Tanwar, S. Tyagi, N. Kumar, M. S. Obaidat, and J. J. Rodrigues. *Bheem: A blockchain-based framework for securing electronic health records.* In *2018 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6 (IEEE, 2018). 39

[95] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar. *A survey on privacy protection in blockchain system.* Journal of Network and Computer Applications **126**, 45 (2019). 40

[96] A. Urquhart. *The inefficiency of bitcoin.* Economics Letters **148**, 80 (2016). 40

[97] S. Nadarajah and J. Chu. *On the inefficiency of bitcoin.* Economics Letters **150**, 6 (2017). 40

[98] R. L. Rivest, A. Shamir, and Y. Tauman. *How to leak a secret*. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 552–565 (Springer, 2001). 40

[99] N. Van Saberhagen. *Cryptonote v 2.0* (2013). 40

[100] I. Eyal and E. G. Sirer. *Majority is not enough: Bitcoin mining is vulnerable*. In *International conference on financial cryptography and data security*, pp. 436–454 (Springer, 2014). 42

[101] N. Szabo. *Formalizing and securing relationships on public networks*. First monday (1997). 42

[102] H. Wang, H. Qin, M. Zhao, X. Wei, H. Shen, and W. Susilo. *Blockchain-based fair payment smart contract for public cloud storage auditing*. Information Sciences **519**, 348 (2020). 42

[103] *Poa network whitepaper* (2019). URL https://github.com/poanetwork/wiki/wiki/POA-Network-Whitepaper. 42

[104] S. D. Lerner. *Rootstock rsk white paper overview* (2015). URL https://www.rsk.co/Whitepapers/RSK_White_Paper-ORIGINAL.pdf. 42

[105] O. Goldreich and Y. Oren. *Definitions and properties of zero-knowledge proof systems*. Journal of Cryptology **7**(1), 1 (1994). 50

[106] L. Chen and T. P. Pedersen. *New group signature schemes*. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 171–181 (Springer, 1994). 50

[107] J. Camenisch and M. Stadler. *Efficient group signature schemes for large groups*. In *Annual International Cryptology Conference*, pp. 410–424 (Springer, 1997). 50, 51

[108] H. Petersen. *How to convert any digital signature scheme into a group signature scheme*. In *International Workshop on Security Protocols*, pp. 177–190 (Springer, 1997). 50

[109] E. Bresson and J. Stern. *Efficient revocation in group signatures*. In *International Workshop on Public Key Cryptography*, pp. 190–206 (Springer, 2001). 50, 51, 54

[110] G. Ateniese, D. Song, and G. Tsudik. *Quasi-efficient revocation of group signatures*. In *International conference on financial cryptography*, pp. 183–197 (Springer, 2002). 50

[111] N. Attrapadung, K. Emura, G. Hanaoka, and Y. Sakai. *A revocable group signature scheme from identity-based revocation techniques: Achieving constant-size revocation list*. In *International Conference on Applied Cryptography and Network Security*, pp. 419–437 (Springer, 2014). 51

[112] M. Bellare, D. Micciancio, and B. Warinschi. *Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions*. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 614–629 (Springer, 2003). 51, 170

[113] M. Bellare and S. Micali. *How to sign given any trapdoor permutation*. Journal of the ACM (JACM) **39**(1), 214 (1992). 52

[114] D. Dolev, C. Dwork, and M. Naor. *Nonmalleable cryptography*. SIAM review **45**(4), 727 (2003). 52

[115] A. Sahai. *Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security*. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pp. 543–553 (IEEE, 1999). 52

[116] D. Boneh, X. Boyen, and H. Shacham. *Short group signatures*. In *Annual International Cryptology Conference*, pp. 41–55 (Springer, 2004). 53, 57, 150

[117] S. D. Gordon, J. Katz, and V. Vaikuntanathan. *A group signature scheme from lattice assumptions*. In *International conference on the theory and application of cryptology and information security*, pp. 395–412 (Springer, 2010). 54

[118] X. Wen, Y. Tian, L. Ji, and X. Niu. *A group signature scheme based on quantum teleportation*. Physica Scripta **81**(5), 055001 (2010). 54

[119] A. Kiayias, Y. Tsiounis, and M. Yung. *Traceable signatures*. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 571–589 (Springer, 2004). 55

[120] D. Boneh and H. Shacham. *Group signatures with verifier-local revocation*. In *Proceedings of the 11th ACM conference on Computer and communications security*, pp. 168–177 (2004). 55

[121] T. Nakanishi, T. Fujiwara, and H. Watanabe. *A linkable group signature and its application to secret voting*. Trans. of Information Processing Society of Japan **40**(7), 3085 (1999). 56

[122] H. Alshammari, K. Elleithy, K. Almgren, and S. Albelwi. *Group signature entanglement in e-voting system*. In *IEEE Long Island Systems, Applications and Technology (LISAT) Conference 2014*, pp. 1–4 (IEEE, 2014). 56

[123] J. Bringer, H. Chabanne, D. Pointcheval, and S. Zimmer. *An application of the boneh and shacham group signature scheme to biometric authentication*. In *International Workshop on Security*, pp. 219–230 (Springer, 2008). 56

[124] N. Kaur and S. Kad. *A cluster based group signature mechanism for secure vanet communication*. IJSTR (2016). 56

[125] M. S. I. Mamun and A. Miyaji. *Secure vanet applications with a refined group signature*. In *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, pp. 199–206 (IEEE, 2014). 56

[126] K. Lim, K. M. Tuladhar, X. Wang, and W. Liu. *A scalable and secure key distribution scheme for group signature based authentication in vanet.* In *2017 IEEE 8th annual ubiquitous computing, electronics and mobile communication conference (UEMCON)*, pp. 478–483 (IEEE, 2017). 56

[127] Y. Sun, Z. Feng, Q. Hu, and J. Su. *An efficient distributed key management scheme for group-signature based anonymous authentication in vanet.* Security and Communication Networks **5**(1), 79 (2012). 56

[128] M. Alimohammadi and A. A. Pouyan. *Sybil attack detection using a low cost short group signature in vanet.* In *2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, pp. 23–28 (IEEE, 2015). 56

[129] S. Zhang and J.-H. Lee. *A group signature and authentication scheme for blockchain-based mobile-edge computing.* IEEE Internet of Things Journal **7**(5), 4557 (2019). 56

[130] B. Gong, C. Cui, M. Hu, C. Guo, X. Li, and Y. Ren. *Anonymous traceability protocol based on group signature for blockchain.* Future Generation Computer Systems (2021). 57

[131] Z. Wang. *Blockchain-based edge computing data storage protocol under simplified group signature.* IEEE Transactions on Emerging Topics in Computing (2021). 57

[132] R. Clarisse and O. Sanders. *Short group signature in the standard model.* IACR Cryptol. ePrint Arch. **2018**, 1115 (2018). 57

[133] Y. Cao, Y. Li, Y. Sun, and S. Wang. *Decentralized group signature scheme based on blockchain.* In *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pp. 566–569 (IEEE, 2019). 58

[134] R. L. Rivest, A. Shamir, and L. Adleman. *A method for obtaining digital signatures and public-key cryptosystems.* Communications of the ACM **21**(2), 120 (1978). 58, 64

[135] L. Zhou, M. A. Marsh, F. B. Schneider, and A. Redz. *Distributed blinding for distributed elgamal re-encryption.* In *25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pp. 824–824 (IEEE, 2005). 58

[136] G. A. F. Rebello, G. F. Camilo, L. Guimaraes, L. A. C. de Souza, and O. Duarte. *Security and performance analysis of quorum-based blockchain consensus protocols.* Electrical Engineering Program, COPPE/UFRJ, Tech. Rep (2020). 58

[137] S. Devidas, S. R. YV, and N. R. Rekha. *A decentralized group signature scheme for privacy protection in a blockchain.* International Journal of Applied Mathematics and Computer Science **31**(2), 353 (2021). 58

[138] T. P. Pedersen. *Non-interactive and information-theoretic secure verifiable secret sharing.* In *Annual international cryptology conference*, pp. 129–140 (Springer, 1991). 60

[139] J. Canny and S. Sorkin. *Practical large-scale distributed key generation.* In *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 138–152 (Springer, 2004). 62

[140] E. Kokoris Kogias, D. Malkhi, and A. Spiegelman. *Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures.* In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1751–1767 (2020). 62

[141] I. Abraham, P. Jovanovic, M. Maller, S. Meiklejohn, G. Stern, and A. Tomescu. *Reaching consensus for asynchronous distributed key generation.* arXiv preprint arXiv:2102.09041 (2021). 62

[142] K. Gurkan, P. Jovanovic, M. Maller, S. Meiklejohn, G. Stern, and A. Tomescu. *Aggregatable distributed key generation.* In *Annual International Conference on*

the Theory and Applications of Cryptographic Techniques, pp. 147–176 (Springer, 2021). 62

[143] Y. Duan. *Distributed key generation for encrypted deduplication: Achieving the strongest privacy.* In *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security*, pp. 57–68 (2014). 62

[144] G. Kilinc, I. N. Fovino, C. Ferigato, and A. Koltuksuz. *A model of distributed key generation for industrial control systems.* IFAC Proceedings Volumes **45**(29), 356 (2012). 63

[145] W. Beullens, L. Disson, R. Pedersen, and F. Vercauteren. *Csi-rashi: Distributed key generation for csidh.* IACR Cryptol. ePrint Arch. **2020**, 1323 (2020). 63

[146] W. Neji, K. Blibech, and N. Ben Rajeb. *Distributed key generation protocol with a new complaint management strategy.* Security and communication networks **9**(17), 4585 (2016). 63

[147] L. Lei, P. Ma, C. Lan, and L. Lin. *Continuous distributed key generation on blockchain based on bft consensus.* In *2020 3rd International Conference on Hot Information-Centric Networking (HotICN)*, pp. 8–17 (IEEE, 2020). 63

[148] N. Farley, R. Fitzpatrick, and D. Jones. *Badger-blockchain auditable distributed (rsa) key generation.* IACR Cryptol. ePrint Arch. **2019**, 104 (2019). 64

[149] Y. Frankel, P. D. MacKenzie, and M. Yung. *Robust efficient distributed rsa-key generation.* In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 663–672 (1998). 64

[150] I. Damgård and M. Koprowski. *Practical threshold rsa signatures without a trusted dealer.* In *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 152–165 (Springer, 2001). 64

[151] Y. Desmedt and Y. Frankel. *Threshold cryptosystems.* In *Conference on the Theory and Application of Cryptology*, pp. 307–315 (Springer, 1989). 65

[152] C. Boyd. *Digital multisignatures.* Cryptography and coding pp. 241–246 (1986). 65

[153] R. Croft and S. Harris. *Public-key cryptography and re-usable shared secrets.* Cryptography and coding pp. 189–201 (1989). 65

[154] T. ElGamal. *A public key cryptosystem and a signature scheme based on discrete logarithms.* IEEE transactions on information theory **31**(4), 469 (1985). 65, 206

[155] V. Shoup and R. Gennaro. *Securing threshold cryptosystems against chosen ciphertext attack.* In *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 1–16 (Springer, 1998). 65

[156] V. Daza, J. Herranz, P. Morillo, and C. Rafols. *Cca2-secure threshold broadcast encryption with shorter ciphertexts.* In *International Conference on Provable Security*, pp. 35–50 (Springer, 2007). 66

[157] J. Herranz, F. Laguillaumie, and C. Ràfols. *Constant size ciphertexts in threshold attribute-based encryption.* In *International Workshop on Public Key Cryptography*, pp. 19–34 (Springer, 2010). 66

[158] D. Boneh, X. Boyen, and S. Halevi. *Chosen ciphertext secure public key threshold encryption without random oracles.* In *Cryptographers' Track at the RSA Conference*, pp. 226–243 (Springer, 2006). 66

[159] Y. Dodis and J. Katz. *Chosen-ciphertext security of multiple encryption.* In *Theory of Cryptography Conference*, pp. 188–209 (Springer, 2005). 66

[160] P. Yang, Z. Cao, and X. Dong. *Chosen ciphertext secure certificateless threshold encryption in the standard model.* In *International Conference on Information Security and Cryptology*, pp. 201–216 (Springer, 2008). 66

[161] C. Delerablée and D. Pointcheval. *Dynamic threshold public-key encryption.* In *Annual International Cryptology Conference*, pp. 317–334 (Springer, 2008). 66, 170

[162] V. Shoup. *A proposal for an iso standard for public key encryption (version 2.1).* IACR e-Print Archive **112** (2001). 66

[163] S. Myers, M. Sergi, *et al.* *Threshold fully homomorphic encryption and secure computation.* Cryptology ePrint Archive (2011). 66

[164] X. Zhang, C. Xu, C. Jin, R. Xie, and J. Zhao. *Efficient fully homomorphic encryption from rlwe with an extension to a threshold encryption scheme.* Future Generation Computer Systems **36**, 180 (2014).

[165] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. Rasmussen, and A. Sahai. *Threshold cryptosystems from threshold fully homomorphic encryption.* In *Annual International Cryptology Conference*, pp. 565–596 (Springer, 2018). 66

[166] D. Boneh, R. Gennaro, S. Goldfeder, and S. Kim. *A lattice-based universal thresholdizer for cryptographic systems.* IACR Cryptol. ePrint Arch. **2017**, 251 (2017). 66

[167] A. Baraani-Dastjerdi, J. Pieprzyk, and R. Safani-Naini. *A practical electronic voting protocol using threshold schemes* (Citeseer, 1994). 66

[168] C. Yao, L. Xu, and X. Huang. *A secure cloud storage system from threshold encryption.* In *2013 5th International Conference on Intelligent Networking and Collaborative Systems*, pp. 541–545 (IEEE, 2013). 67

[169] Y. Chen, H. Liu, B. Wang, B. Sonompil, Y. Ping, and Z. Zhang. *A threshold hybrid encryption method for integrity audit without trusted center.* Journal of Cloud Computing **10**(1), 1 (2021). 67

[170] K. Khairunas, M. Zarlis, and S. Sawaluddin. *Data security analysis against chosen ciphertext secure public key attack using threshold encryption scheme.* Randwick International of Social Science Journal **2**(3), 326 (2021). 67

[171] Q. Kong, F. Yin, Y. Xiao, B. Li, X. Yang, and S. Cui. *Achieving blockchain-based privacy-preserving location proofs under federated learning.* In *ICC 2021-IEEE International Conference on Communications*, pp. 1–6 (IEEE, 2021). 67

[172] D. Yakira, A. Asayag, G. Cohen, I. Grayevsky, M. Leshkowitz, O. Rottenstreich, and R. Tamari. *Helix: A fair blockchain consensus protocol resistant to ordering manipulation.* IEEE Transactions on Network and Service Management **18**(2), 1584 (2021). 68

[173] R. Turn and W. H. Ware. *Privacy and security issues in information systems.* IEEE Transactions on Computers **25**(12), 1353 (1976). 74

[174] M. K. Reiter and A. D. Rubin. *Crowds: Anonymity for web transactions.* ACM transactions on information and system security (TISSEC) **1**(1), 66 (1998). 75

[175] J. Heather and D. Lundin. *The append-only web bulletin board.* In *International Workshop on Formal Aspects in Security and Trust*, pp. 242–256 (Springer, 2008). 83, 94

[176] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. *Preventing location-based identity inference in anonymous spatial queries.* IEEE transactions on knowledge and data engineering **19**(12), 1719 (2007). 92

[177] R. Cramer and V. Shoup. *Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack.* SIAM Journal on Computing **33**(1), 167 (2003). 94

[178] S. Khattak, T. Elahi, L. Simon, C. M. Swanson, S. J. Murdoch, and I. Goldberg. *Sok: Making sense of censorship resistance systems.* Proceedings on Privacy Enhancing Technologies **2016**(4), 37 (2016). 103

[179] A. Fiat and J. Saia. *Censorship resistant peer-to-peer content addressable networks.* In *SODA*, vol. 2, pp. 94–103 (Citeseer, 2002). 103

[180] R. Subramanian. *The growth of global internet censorship and circumvention: A survey.* Communications of the International Information Management Association (CIIMA) **11**(2) (2011). 103

[181] M. E. Roberts. *Resilience to online censorship.* Annual Review of Political Science **23**, 401 (2020). 103

[182] S. He, Q. Tang, C. Q. Wu, and X. Shen. *Decentralizing iot management systems using blockchain for censorship resistance.* IEEE Transactions on Industrial Informatics **16**(1), 715 (2019). 104

[183] S. Delaune, S. Kremer, and M. Ryan. *Coercion-resistance and receipt-freeness in electronic voting.* In *19th IEEE Computer Security Foundations Workshop (CSFW'06)*, pp. 12–pp (IEEE, 2006). 109

[184] A. Juels, D. Catalano, and M. Jakobsson. *Coercion-resistant electronic elections.* In *Towards Trustworthy Elections*, pp. 37–63 (Springer, 2010). 109, 177

[185] T. Haines and B. Smyth. *Surveying definitions of coercion resistance.* Cryptology ePrint Archive (2019). 109

[186] M. Dehez-Clementi, J.-C. Deneuville, J. Lacan, H. Asghar, and M. A. Kaafar. *Who let the dogs out: Anonymous but auditable communications using group signature schemes with distributed opening.* In *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2020 International Workshops, DPM 2020 and CBT 2020, Guildford, UK, September 17–18, 2020, Revised Selected Papers*, p. 437 (Springer, 2020). 111, 161

[187] A. Boldyreva. *Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme.* In *International Workshop on Public Key Cryptography*, pp. 31–46 (Springer, 2003). 111

[188] S. Mauw, J. H. Verschuren, and E. P. de Vink. *A formalization of anonymity and onion routing.* In *European Symposium on Research in Computer Security*, pp. 109–124 (Springer, 2004). 115

[189] A. I. Kouachi, A. Bachir, and N. Lasla. *Anonymizing communication flow identifiers in the internet of things.* Computers & Electrical Engineering **91**, 107063 (2021). 117

[190] C. Rackoff and D. R. Simon. *Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack.* In *Annual International Cryptology Conference*, pp. 433–444 (Springer, 1991). 122

[191] C. Dwork, N. Lynch, and L. Stockmeyer. *Consensus in the presence of partial synchrony.* Journal of the ACM (JACM) **35**(2), 288 (1988). 123

[192] D. Ron and A. Shamir. *Quantitative analysis of the full bitcoin transaction graph.* In *International Conference on Financial Cryptography and Data Security*, pp. 6–24 (Springer, 2013). 136

[193] G. Maxwell. *Coinjoin: Bitcoin privacy for the real world.* In *Post on Bitcoin forum* (2013). 136

[194] T. Ruffing, P. Moreno-Sanchez, and A. Kate. *Coinshuffle: Practical decentralized coin mixing for bitcoin.* In *European Symposium on Research in Computer Security*, pp. 345–364 (Springer, 2014). 136

[195] ETSI. *103 415 v1. 1.1: Intelligent transport systems (its);security; pre-standardization study on pseudonym change management* (2018). 146

[196] M. Babaghayou, N. Labraoui, A. A. A. Ari, N. Lagraa, and M. A. Ferrag. *Pseudonym change-based privacy-preserving schemes in vehicular ad-hoc networks: A survey.* Journal of Information Security and Applications **55**, 102618 (2020). 146

[197] N. Kerkacha, N. Hadj-Said, N. Chaib, A. Adnane, and A. Ali-Pacha. *Impact of silent periods on pseudonym schemes.* In *2021 18th International Multi-Conference on Systems, Signals & Devices (SSD)*, pp. 79–85 (IEEE, 2021). 146

[198] S. S. Manvi and S. Tangade. *A survey on authentication schemes in vanets for secured communication.* Vehicular Communications **9**, 19 (2017). 148

[199] A. Wasef and X. Shen. *Emap: Expedite message authentication protocol for vehicular ad hoc networks.* IEEE transactions on Mobile Computing **12**(1), 78 (2011). 148

[200] L. Zhang, Q. Wu, A. Solanas, and J. Domingo-Ferrer. *A scalable robust authentication protocol for secure vehicular communications.* IEEE Transactions on vehicular Technology **59**(4), 1606 (2009). 151, 152, 169, 170

[201] Y. Zheng. *Digital signcryption or how to achieve cost (signature & encryption)≪ cost (signature)+ cost (encryption).* In *Annual international cryptology conference*, pp. 165–179 (Springer, 1997). 151

[202] A. L. Ferrara, M. Green, S. Hohenberger, and M. Ø. Pedersen. *Practical short signature batch verification.* In *Cryptographers' Track at the RSA Conference*, pp. 309–324 (Springer, 2009). 151, 169

[203] X. Zhu, S. Jiang, L. Wang, H. Li, W. Zhang, and Z. Li. *Privacy-preserving authentication based on group signature for vanets.* In *2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 4609–4614 (IEEE, 2013). 151, 152, 169, 170

[204] J. Shao, X. Lin, R. Lu, and C. Zuo. *A threshold anonymous authentication protocol for vanets.* IEEE Transactions on vehicular technology **65**(3), 1711 (2015). 152

[205] M. Dehez-Clementi, J. Lacan, J.-C. Deneuville, H. Asghar, and D. Kaafar. *A blockchain-enabled anonymous-yet-traceable distributed key generation.* In *2021 IEEE International Conference on Blockchain (Blockchain)*, pp. 257–265 (IEEE, 2021). 161

[206] L. Zhang, Q. Wu, B. Qin, J. Domingo-Ferrer, and B. Liu. *Practical secure and privacy-preserving scheme for value-added applications in vanets.* Computer Communications **71**, 50 (2015). 169

[207] H. Kim, Y. Lee, M. Abdalla, and J. H. Park. *Practical dynamic group signature with efficient concurrent joins and batch verifications.* Journal of Information Security and Applications **63**, 103003 (2021). 169, 170

[208] D. Pointcheval and O. Sanders. *Short randomizable signatures*. In *Cryptographers' Track at the RSA Conference*, pp. 111–126 (Springer, 2016). 169

[209] J. Camenisch and J. Groth. *Group signatures: Better efficiency and new theoretical aspects*. In *International Conference on Security in Communication Networks*, pp. 120–133 (Springer, 2004). 170

[210] D. Derler and D. Slamanig. *Highly-efficient fully-anonymous dynamic group signatures*. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 551–565 (2018). 170

[211] P. Smolar and P. Lepidi. *La lutte pour l'ouverture des archives sur le rwanda entre dans une phase décisive* (2021). Last accessed 3-oct-2021, URL `https://bit.ly/3iwf9rZ`. 174

[212] F. O. Omotayo and O. A. Adekunle. *Adoption and use of electronic voting system as an option towards credible elections in nigeria*. International Journal of Development Issues (2021). 177

[213] S. Agbesi. *Political parties and internet voting system adoption in ghana*. In *International Conference on Electronic Government and the Information Systems Perspective*, pp. 174–186 (Springer, 2020). 177

[214] D. I. Sensuse, P. B. Pratama, *et al.* *Conceptual model of e-voting in indonesia*. In *2020 International Conference on Information Management and Technology (ICIMTech)*, pp. 387–392 (IEEE, 2020). 177

[215] K. Krips and J. Willemson. *On practical aspects of coercion-resistant remote voting systems*. In *International Joint Conference on Electronic Voting*, pp. 216–232 (Springer, 2019). 177

[216] T. Dimitriou. *Efficient, coercion-free and universally verifiable blockchain-based voting*. Computer Networks **174**, 107234 (2020). 177

[217] Matter-Labs. *Awesome zero knowledge proofs (zkp)* (2018). Last accessed 3 November 2021, URL `https://github.com/matter-labs/awesome-zero-knowledge-proofs`. 191

[218] J. Daemen and V. Rijmen. *Aes proposal: Rijndael* (1999). 206

[219] R. Rajagopalan and P. K. Varshney. *Data aggregation techniques in sensor networks: A survey* (2006). 214

Acronyms

AES Advanced Encryption Standard

ASICS Application Specific Integrated Circuits

BFT Byzantine Fault-Tolerant

BLS Boneh-Lynn-Shacham

CA Certificate Authority

CAM Cooperative Awareness Message

CCU Communications Control Unit

CPS Cyber Physical System

CRHF Collision-Resistant Hash Function

CRL Certificate Revocation List

DCC Decentralized Congestion Control

DDL Double Discrete Logarithms

DDLP Double Discrete Logarithms Problem

DeFi Decentralized Finance

DEM Data Encryption Mechanism

DENM Decentralized Environmental Notification Message

DGSS Decentralized Group Signature Scheme

DKG Distributed Key Generation

dlog discrete logarithm

DLP Discrete Logarithm problem

DLT Distributed Ledger Technologies

DoS Denial of Service

DSRC Dedicated Short Range Communications

ECC Elliptic Curve Cryptography

ECDSA Elliptic Curve Digital Signature Algorithm

EHR Electronic Health Record

ETSI European Telecommunication Standards Institute

EUF-CMA Existentially UnForgeable under Chosen Message Attacks

FPGA Field Programmable Gate Arrays

GM Group Manager

GPS Global Positioning System

HMAC Hash-based Message Authentication Code

HMI Human Machine Interface

IND-CCA2 INDistinguishable under Adaptative Chosen-Ciphetext Attacks

IoT Internet of Things

IPFS InterPlanetary File System

ITS Intelligent Transportation System

ITS-S Intelligent Transportation System Station

KEM Key Encapsulation Mechanism

MAC Message Authentication Code

MANET Mobile Ad-hoc NETwork

NIZK Non-Interactive Zero Knowledge

NP Nondeterministic Polynomial time

OBU On-Board Unit

OSI Open Systems Interconnection

P2P peer-to-peer

PET Privacy Enhancing Technogologies

PII Personally identifiable information

PKI Public Key Infrastructure

PPT Probabilistic Polynomial Time

RHW Road Hazard Warning

RoI Region of Interest

RSA Rivest–Shamir–Adleman

RSK Rootstock

RSU RoadSide Unit

SGS Simplified Group signature

SPV Simplified Payment Verification

SSS Shamir Secret Sharing

TA Trusted Authority

TTL Time To Live

TTP Trusted Third Party

UTXO Unspent Transaction Output

V2I Vehicle-to-Infrastructure

V2V Vehicle-to-Vehicle

V2X Vehicle-to-Anything

VANET Vehicular Ad-hoc NETwork

VRL Verifier-local revocation

VSS Verifiable Secret Sharing

WAVE Wireless Access in Vehicular Environments

WHO World Health Organization

X2B Anything-to-Blockchain