10-24-2019

A Privacy Framework for Decentralized Applications using Blockchains and Zero Knowledge Proofs

David Gabay
Florida International University, dgaba002@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

A PRIVACY FRAMEWORK FOR DECENTRALIZED APPLICATIONS USING BLOCKCHAINS AND ZERO KNOWLEDGE PROOFS

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

David Gabay

2019

To: Dean John L. Volakis
College of Engineering and Computing

This thesis, written by David Gabay, and entitled A Privacy Framework for Decentralized Applications using Blockchains and Zero Knowledge Proofs, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

Arif Selcuk Uluagac

Alexander Perez-Pons

Kemal Akkaya, Major Professor

Date of Defense: October 24, 2019

The thesis of David Gabay is approved.

Dean John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development and
Dean of the University Graduate School

Florida International University, 2019

DEDICATION

To my parents, thanks for supporting me through all of the years, I couldn't have done it without you guys. To my friends and family, thank you for your continuing support.

ACKNOWLEDGMENTS

Thank you Professor Dr. Kemal Akkaya for being my advisor, and mentor, without your guidance I would not have been able to succeed in my schooling. Thank you Professor Dr. Arif Selcuk Uluagac for supporting me through schooling. Thank you National Science Foundation for sponsoring me with the Scholarship For Service Cybercorps. Thank you Florida International University for providing the resources to succeed in school. Thank you for the lab mates in Cyber-Physical and Adwise Labs for continuing help throughout the projects. Thank you to my supporting friends who guided me in the right direction.

ABSTRACT OF THE THESIS

A PRIVACY FRAMEWORK FOR DECENTRALIZED APPLICATIONS USING BLOCKCHAINS AND ZERO KNOWLEDGE PROOFS

by

David Gabay

Florida International University, 2019

Miami, Florida

Professor Kemal Akkaya, Major Professor

With the increasing interest in connected vehicles along with electrification opportunities, there is an ongoing effort to automate the charging process of electric vehicles (EVs) through their capabilities to communicate with the infrastructure and each other. However, charging EVs takes time and thus in-advance scheduling is needed. As this process is done frequently due to limited mileage of EVs, it may expose the locations and charging pattern of the EV to the service providers, raising privacy concerns for their users. Nevertheless, the EV still needs to be authenticated to charging providers, which means some information will need to be provided anyway. While there have been many studies to address the problem of privacy-preserving authentication for vehicular networks, such solutions will be void if charging payments are made through traditional means. In this thesis, we tackle this problem by utilizing distributed applications enabled by Blockchain and smart contracts. We adapt zero-knowledge proofs to Blockchain for enabling privacy-preserving authentication while removing the need for a central authority. We introduce two approaches, one using a token based mechanism and another utilizing the Pederson Commitment scheme to realize anonymous authentication. We also describe a protocol for the whole process which includes scheduling and charging operations. The evaluation

of the proposed approaches indicates that the overhead of this process is affordable to enable real-time charging operations for connected EVs.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

1.1 Motivation

New generation vehicles are becoming much smarter at various levels including but not limited to infotainment, brake assistance, adaptive cruise control, parking assistance, driver fatigue detection, and more [Mil15]. This is in particular due to recent developments on Vehicle to Infrastructure communication (V2I), which enables a smart car to communicate with its environment (i.e., determine traffic details, communicate with service providers, etc.) as well as Vehicle-to-Vehicle (V2V) communication that enables safety and social applications. Ultimately, there is a lot of potential applications due to such developments in vehicular networks [RKX+18].

One subclass of smart vehicles is Electric Vehicles (EVs) which operate via electricity as opposed to gasoline vehicles. With a single charge, EVs can operate anywhere from 50-200 miles [BKA19] which means they will need frequent charging. They can rely on V2I and V2V to communicate with other EVs and EV Charging Service providers (EVSP) to schedule charging. With the advancements in autonomous vehicle technologies, they are expected to conduct charge scheduling and even charging automatically. Even charging through another EV would be possible this manner which will enable social communication among EVs [BKA19].

However, frequent charging would bring privacy issues as studied in the literature. Specifically, in the case of charge scheduling, the service providers would have access to all of the EVs information since this is similar to a client-sever model. For instance, over time the service providers can learn a lot about their clients including their personal data, location patterns, and habits. This data can be shared with other third parties such as marketers, insurance companies, etc. Also, if any pay-

ment is made for the service, that can also raise privacy concerns as the payment information will still be collected. These issues with a centralized system lead to the implementation of various privacy laws as in [IH18] but this does not address the problem as full enforcement of these policies is not easy.

An alternative to address the privacy issue that is being considered is moving to a decentralized architecture that will also prevent issues due to single point of failure. In particular, Blockchain technologies that utilize distributed ledger architecture can be a viable option since it enables a truly distributed trust [Swa15] as well as enabling payments through cryptocurrencies that will not expose any payment info to third parties.

Applications that are run on such a distributed ledger are now commonly referred to as *Decentralized Applications (DApps)*. There are several examples such as supply chain management [NMNB18], decentralized voting [DKM$^+$18], Internet of Things [Che18], social media [CR17], digital rights management (DRM) [MJGW18] which aim to mitigate the risks accompanied by centralized systems. EV Charging can also be moved to a distributed architecture by using smart contracts on a distributed ledger. In the case of EV Charging applications, the EV communicates with a smart contract, and with the EVSP before it can be allowed to charge at a charging station. Even though the distributed architecture can support these operations, developing an architecture through a smart contract will still be a challenge. This is because there should be a mechanism to allow only registered EVs to access to the service through authentication. Second, this authentication mechanism should also ensure privacy.

To address these issues, this thesis proposes an efficient and feasible framework that allows an EV user to request charging scheduling from an EVSP in an Ethereum blockchain [W$^+$14] application while ensuring the privacy of the EV user. To this

end, this work utilizes the concept of zkSNARKs [BSCTV14] which aims to provide an efficient variant of a zero-knowledge proof [GMR89]. zkSNARKs allows a person to prove to another person the correctness of a statement without revealing the contents of the proof in one message.

In the proposed framework, it uses zkSNARKs efficiently by having a single contract verify every EV using the decentralized application, as opposed to having a smart contract generated for each EV. Once verified by the smart contract without being identified, it uses an Ethereum token system to issue tokens to that EV, which will be used for scheduling and charging service requested through a pseudonym. The token approach will ensure the authentication and charging operations are kept separate and thus EVs remain anonymous to both the EVSP and the public using the blockchain.

However, since this approach still requires writing to blockchain for scheduling and charging services which brings some overhead, an improvement to the scheme to eliminate this overhead is proposed. Specifically, this new scheme entails using *Pederson Commitment* scheme [Ped91] for authentication as well as charge scheduling purposes. In the Pederson approach the commitment is both binding and hiding, meaning nobody can infer who submitted the commitment and only the committer can prove s/he submitted the commitment. This new approach still follows the same protocol but the steps for token-based operations are eliminated with the help of Pederson scheme. In this way, it is able to eliminate access to blockchain for charge scheduling and charging and hence improve efficiency and cost.

This work implements both approaches of the proposed framework and assesses their feasibility and performance. The analysis shows that our proposed model provides anonymity for the EV (user) during communication with the EVSP and charging stations. Additionally the framework's underlying protocol Zokrates [ET18] pro-

vides strong proofs that enable that only legitimate users can generate valid proofs. Finally, the results show that the cost and computation time for the framework on the Ethereum Blockchain are acceptable ranging from $0.23 to deploy a smart contract and $0.13 to verify the proof. In addition, scheduling a charging slot for the EV can be done in less than 38 secs when using the Pederson approach.

The rest of this thesis is organized as follows: a summary of the related literature next. In Section 3, any background information related to DApps, Zero-knowledge proofs, Pederson commitment scheme, and the current vehicle communication model. In Section 4, we present our token based approach using zkSNARKs and Blockchain, followed by our improved Pederson approach in Section 5. In Section 6, we discuss the security and privacy analysis of our framework. In Section 7, we evaluate our proposed framework, comparing both approaches, and conclude in Section 8.

CHAPTER 2

RELATED WORK

EV charging stations, at the time of writing, do not offer scheduling services for EVs. In this chapter, we look into the different applications of EV charging scheduling, zero knowledge proofs and Blockchain (Decentralized Applications – DApps). We also consider the use of Blind Signatures for use in EV charging scheduling which, unlike ours, is a centralized approach.

2.1 Traditional Privacy Approaches in EV Charging

There have been many works for privacy-preserving of EVs in the past [HX16]. These focused on several approaches including homomorphic encryption, blind signatures, and pseudonyms to preserve either the location or identity privacy for the EVs.

Pseudonyms are widely used for privacy purposes. Basically anonymous identities are assigned to EVs as also suggested in the IEEE 1609.2 standard [LDN14]. However, these pseudonyms are assigned to EVs by a set of trusted authorities. Also, the mechanism, by design, allows the trusted authorities to track the EVs actions if desired. The suggested pseudonym mechanism in IEEE 1609.2 only protects the privacy of EV from untrusted parties. Thus, the pseudonym mechanism does not provide the desired privacy of the EV against all parties. In our case, we rely on the pseudonyms too but we ensure the privacy even against authorities/EVSP by utilizing zero-knowledge proofs.

Homomorphic encryption is used in various contexts but mainly to enable location privacy and data aggregation, if any [YAB18]. The idea is to hide the information in the distance and other computations. In EV charging, this can apply to make schedules on the encrypted data but this would require fully homomor-

phic encryption which is very slow and limited in capability to deal with all search functions.

2.1.1 Blind Signatures

Blind signatures are another option to provide privacy in authentication. This is a cryptographic protocol [Cha83] which allows the user to send a message which has been blinded to be signed by a third party. Therefore, the content of the message is not known to third parties. The signature on the blinded message can then be unblinded by the source and be attached to its future messages.

While blind signatures were not directly used for EV charging, they have been employed for smart meter privacy protection from the utility in [CYHL12]. Basically, the idea is to do in-advance power allocation for the smart meter through the concept of credentials. These are sent to the control center which signs the requests without any knowledge of the message originator. The smart meter then uses these signatures to make actual power demand through anonymous IDs. Finally, at the end of the month during reconciliation, the smart meter identifies itself to the Power provider and sends the number of credentials used by the smart meter by subtracting the total credentials by the unused credentials and pays for the extra power usage. In this particular work, privacy will be exposed at the end of the period, when the smart meter identifies itself to the power provider. This work is successful in preserving location privacy for the specific use-case of smart meter power scheduling and assignment.

The use of blind signature cryptography could be used as an alternative approach to our proposed scheduling scheme for a centralized architecture. Take for example our EV charging model, the EVSP would be the central authority (also the signer),

and the EV would be the client looking to schedule charging times with the EVSP meanwhile preserving his location privacy and anonymity. First, both the EV and the EVSP will register each obtaining a private/ public key pair, by providing their identities. Next EV would request a number of credentials to be used for scheduling charging time at the EV charging station. Next, whenever the EV wishes to schedule a charging period he encrypts a message with a random session key, the credentials required for charging, as well as the time he wishes to use the charging station. Following this, the EV would encrypt the session key with the EVSP public key and send this message to the EVSP. Once the EVSP decrypts the session key he would, in turn, be able to decrypt the message containing an authorized anonymous EV requesting to charge. Finally, at the end of a period of time, the EV would report back to the EVSP with its used number of credentials to verify their EV charging usage. While these signatures are useful when used with anonymous IDs, the issue of payments for charging is still there which can expose privacy. Therefore, blind signatures need to be combined with a privacy-preserving payment system as well. Our proposed approach in this paper address both issues by combining the DApps and zero-knowledge proofs.

2.2 Blockchain

Other work has been done in the realm of authentication using Blockchain, but most have been geared towards the mutual authentication of IoT devices in either fog scenarios or wireless sensor networks. Blockchain is is a decentralized framework and inherently works well for decentralized applications like fog computing and Wireless sensor networks. However, in these authentication mechanisms it either focuses on mutual authentication or does not take into affect payment.

2.2.1 EV Location Privacy in DApps

Location privacy is important when considering Decentralized applications since users interact on a public ledger where everyone can see the transactions. In [ABB+19] Amiri et al. use the Blockchain and a Private Information Retrieval technique to design a smart parking system that preserves the users' privacy. In this way, a user could anonymously request a parking offer from another peer directly, and authenticate using short randomizable signatures.

In [KUE18] Knirsch et al. designs a transparent EV charging framework with consideration of smart grid supply and demand. Knirsch uses blockchain to offer transparency and dynamic pricing to EVs. The EV can choose a charging slot based on it's distance to the station and the offered price. Knirsch uses a SHA256 hash of the requested charging period to hide the details of the charging period and commit to the schedule. EVs in this approach remain anonymous, and can only be identified by some randomly generated blockchain ID. This approach bypasses EV authentication and instead considers every EV an authorized customer of the EVSP. In our approach, we implement a method of authentication for the EV to prove they are authorized customers' of the EVSP.

Similarly, in [PKS16] Pustiek et al. provides an excellent model for autonomous charging station selection using blockchain smart contracts. However, in [PKS16], the authors do not consider the privacy of the EV, but rather focus on the protocol used to optimally select a charging station for that EV.

2.3 Zero-knowledge proofs and Blockchain

Zero-knowledge proofs have started to be used in various DApps. Some of these approaches followed zkSNARKs but to date, none of these studies targeted the EV

charging problem. For instance, Zerocash [SCG+14] is the core of the cryptocurrency Zcash which utilizes zkSNARKs to preserve the privacy of transactions that occur on the Blockchain. [NT16] used zkSNARKs to develop a novel image authentication approach used to determine if a photograph taken represents the original and has not been modified. Other studies using zkSNARKs include [WZC+18] and [ZNP15] which are geared for ensuring the integrity of machine learning approaches and data sharing privacy, respectively.

In [WZC+18], Wu developed a distributed framework for zero-knowledge proofs for the authenticity of photographs and the integrity of machine learning models. Wu from the University of California, Berkeley in DIZK: Distributed Zero-Knowledge Proof System aims to improve the time and memory overhead of the off-chain processes such as the setup phase conducted by the trusted third party and the proof generation phase performed by the prover. Wu introduces compute clusters to improve the performance and scalability of zero-knowledge proofs, which he hopes to test the feasibility of integration into the Blockchain.

In [ZNP15], the authors use the Blockchain as an access control moderator combined with an off-chain storage system to allow users to share information with others including service providers based on their privacy needs. The work features Compound Identities which involve at least two users one of which is the owner and the others are guest with limited access to the data, included are key signing pairs for the owner and guests as well as a symmetric key to encrypt/decrypt the data. The application presented in the work is a mobile phone user downloads a third party application and creates a compound Identity with the service provider and restricts access to the service provider using the ***access*** transaction on the Blockchain, and data retrieval using the ***data*** transaction.

In [BLM+19] Baza et al. use Blockchain smart contracts along with Zero Knowledge Set Membership (ZKSM) Proofs to provide a decentralized ride-sharing service called "B-Ride". B-Ride preserves the users' privacy including personal identity and preserves users' location privacy all the while using the Blockchain to provide decentralization. Since B-Ride does not rely on a third party or service provider they implement a reputation model to track drivers based on ride completion.

2.4 Pederson Commitment Scheme for Authentication

The only relevant work in the literature on Pederson is reported in [GB18]. The authors used the Pederson Commitment scheme to hide the Biometric IDentity (BID) of a user. Then, during authentication the user can use a zero knowledge proof of knowledge protocol to prove to the SP that they are the correct user without revealing the hidden BID. This allows the SP to authenticate the user without inferring who the user is. While the objective is similar, our context is very different in this paper.

CHAPTER 3

PRELIMINARIES

Before we explain the framework we provide some background on EV charging and zero-knowledge proofs.

3.1 EV Charging Procedure

Unlike traditional gas service, EV charging can take more time (from 30 minutes to 2 hours) and thus in-advance scheduling is needed. We assume that the EVs can communicate with EVSP and EV charging stations using some of the existing communication solutions such as LTE or IEEE 802.11p which can connect with Roadside Units (RSU) to access the Internet infrastructure. Through these communications, we follow the following protocol [RABK17]: the EVs send charging scheduling requests that will include the EV's location, area of interest for charging, EV's state of charge (SoC), the available time slots and other related constraints. The EVSP will first need to verify that this is a genuine request and then respond with a reserved time slot and location for the EV charging station. Once the EV arrives at the charging station, it can verify that the EV is there. Note that in the current protocols the EVSP needs to collect EV information. Also, the EV user needs to make a payment via a certain method such as a credit card.

3.2 DApps, Blockchain and Smart Contracts

Today most of our data is hosted in a centralized client-server architecture where the application user is the client and the server holds all the data. The problem with this architecture is that there exists a single point of failure for the servers. DApps aim to mitigate this risk by distributing data storage across all nodes. Blockchain

and *smart contracts* are building blocks of DApps. Blockchain is a decentralized ledger where each user, or node, has a ledger or list of all past blocks, where each block contains transaction(s), a timestamp, and a hash of the previous block on the chain of blocks. Since each block contains a hash of the previous block, changing the contents of one block would require changing the hash of each of the preceding blocks. Inherently blocks on the Blockchains are immutable, cannot be changed, deleted, or added.

Ethereum is a Blockchain platform that uses *smart contracts* to dictate the correctness of transactions. A smart contract is a block of code that allows a transaction to occur if and only if the parameters are correct. Using smart contracts on the Ethereum blockchain allows us to make DApps, where the code is executed correctly in a decentralized manner.

Here are a few applications of Decentralized Applications (DApps):

- *Decentralized Voting*: In [DKM+18] the authors' design a DApp framework used for decentralized voting, based on a Proof of Authority (PoA) algorithm, where voters prove to an authority that they submitted their ballot to a Blockchain without revealing the contents of the ballot. In this case, the authority is selected based on his vested interest in the network, where an authority with a lot of funds in that network is less likely to approve false transactions.
- *Supply Chain*: Current supply chain management systems are centralized, hosted on the company's server, this makes access to data available in one location, leaving it vulnerable as it has a Single Point Of Failure (SPOF). In [NMNB18] they provide a framework for developing a Decentralized supply chain using Ethereum smart contracts. In their work, they use IoT devices to minimize the amount of human interaction. They use a tree-like structure

where each node on the tree is an entity of the supply chain and thus they can update the supply chain and nodes can view the update only if they have authority, representing a parent node of that entity.

- *Digital Rights Management*: Digital Rights Management (DRM) is the process of protecting digital content from being illegally used or accessed thus protecting the content owners interest and rights. [MJGW18] create DRM-Chain a Blockchain empowered method of preserving lawful use of content and tracking illegal use or content violations.

3.3 Zero-knowledge Proofs

Zero-knowledge proofs [GMR89] allow a person to verify the correctness of a statement from another person without revealing the data used in the statement to himself or anybody else. In this way, the privacy of the person submitting the information is protected. zkSNARKs [BSCTV14] is an efficient variant of zero-knowledge proofs that allows the prover to prove his knowledge in a single message. In this subsection, we provide an overview zero-knowledge proof scheme, then we describe zkSNARKs and SE-SNARKs[GM17] that we used in our work. First, we describe the different roles involved in a zero-knowledge proof:

- ***Trusted Third Party (TTP)***: The TTP is in charge of registration of the user or *Prover*. The TTP creates a secret function which the Prover will solve to authenticate himself to the TTP. The TTP will also generate a contract with a verification key embedded in it. The contract is used by the *Verifier* to verify the generated proofs.
- ***Prover***: The Prover is the client registered with the TTP. The Prover will be in charge of solving the secret function to create a witness. Using the witness,

it will generate a zero-knowledge proof attesting to its knowledge of the secret function, thus, authenticating itself.

- ***Verifier***: The verifier is the party that could verify the Prover's proof without learning the details of the proof or the Prover. The Verifier uses the contract from the TTP to check the correctness of a proof.

3.3.1 Zero-knowledge proof scheme

In the zero-knowledge proof scheme, there exists a setup and verification phase. Let U denote the user and Prover, t a trusted third party, V the verifier, and π an honestly generated proof. In order for a zero-knowledge proof to exist it must contain three properties [May16]: 1) *Completeness:* V always accepts an honestly generated proof π from U; 2) *Knowledge:* A Prover U who does not know a secret will unlikely be able to generate a valid π therefore unable to convince V; and 3) *Zero Knowledge:* A valid π does not reveal any information about the secret. Protocol 1 provides the details for a *zero-knowledge proof of knowledge.*

Protocol 1 Zero-knowledge Commitment Scheme

---

1. **Setup**
   (a) t generates some secret function using some knowledge U possess
   (b) U inputs values satisfying the condition of the function, returning the witness to U
   (c) U uses the witness to generate π_U
2. **Verification**
   (a) U sends π_U to V, who verifies the validity of π_U

(b) V grants or denies access to U, without learning the contents of π_U

One weakness in the above scheme is the setup phase, because if the details of the secret function are known to anybody but authorized users, then somebody can perform one of two attacks: 1) An adversary can impersonate a prover by generating a fake verifying function; 2) Anyone can generate valid proofs if they know the secret function. Existing zero knowledge proof protocols mitigate this vulnerability by using a trusted setup phase as in Zcash in which they use an extensive Multiparty Computation (MPC)[BSCG$^+$15] mechanism to generate the setup parameters. This trusted setup is secure in that unless all the parties are compromised the setup will hold. Other work removes the trusted setup phase as in [WTS$^+$18]. They show that their protocol is comparably efficient to existing zero knowledge proof systems. Additionally, you can avoid a trusted setup phase by using a zero-knowledge proof scheme which is interactive, as opposed to non-interactive.

Zero-knowledge proofs can be broadly categorized into two types: 1) Interactive; 2) non-interactive. In an interactive scheme, lets assume user U, wishes to prove some knowledge to verifier V. In this case, V can generate some challenge and send it to U for an n number of rounds. As n increases the probability of U guessing what the secret is decreases exponentially. So, n is chosen by V based on his desired level of confidence that U knows the secret. On the other hand, in an non-interactive scheme, U can prove to V their knowledge of the secret with one message, this is done with the help of the trusted setup phase mentioned above. Non-interactive schemes are more efficient but come at a security vulnerability, the need for a trusted setup. Nonetheless, it would be very costly to have a interactive verifying scheme on a distributed ledger, as their would need to be several transactions for a single verification. Thus, we use a non-interactive and succinct zero-knowledge proof scheme in our work, namely zkSNARKs, which we discuss in detail, in the following section.

3.3.2 zkSNARKs

Zero-Knowledge Succinct Non-interactive ARguments of Knowledge (zkSNARKs) [BSCTV14] are a form of zero-knowledge proofs which are distinguished by the following properties:

1. *Non-interactive:* a prover can prove awareness of a piece of knowledge to a verifier with a single message.
2. *Succinct*: verification time of the proofs is small
3. *Constant size*: proofs generated by zkSNARKs are the same size regardless of the complexity of the program being proven.
4. *Private fields/inputs*: ability to prove using private inputs that are not revealed to the verifier.

The properties mentioned above make using zkSNARKs a perfect candidate for a verification scheme using Blockchain. That is, since they are non-interactive then the prover needs to perform only one blockchain transaction to be verified. Verification time is both short and constant, regardless of the complexity of the underlying program. Additionally, since the verifier cannot infer any details from a proof, this guarantees privacy, even in the public blockchain ledger. Finally, since the proof size is small (around 288 bytes in the case of [Gro16]) it is feasible for proofs to be sent over most networks, in our case it is an EV transmitting the proof over a VANET.

3.3.3 SE-SNARKs

For our use case it is important to have a proving scheme which is non-malleable. A malleable proving scheme means that using an existing proof, somebody can gen-

erate another valid proof for the same program. For our use case which is a one verifying function for all, one must not be able to derive a valid proof from an existing proof. Simulation-Extractable Succinct Non-interactive ARgument of Knowledge (SE-SNARKs)[GM17] come with the same security benefits as zkSNARKs. Additionally, SE-SNARKs provide two key enhancements:

1. *Smaller proof sizes*: around 127 bytes, compared to [Gro16] which have proofs that are about 288 bytes.
2. *Non-Malleable*: a new proof cannot be generated using a previously submitted proof.

SE-SNARKs, [GM17] utilize Signatures of Knowledge (SoK) [CS97, CL06] to make their proving scheme simulation-extractable. Basically, this works by allowing the prover to create a signature based on the witness, where the signature doesn't reveal details of the witness, and somebody cannot look at previous signatures to make a new signature. In short, this means a prover must know a valid witness to a program in order to sign it, thus this proving scheme provides non-malleability.

3.4 Pederson Commitment

Pederson commitment [Ped91] is a non-interactive verifiable secret sharing scheme. The Pederson commitment scheme allows the committer to commit to a message and send it to some verifier without disclosing any of the commitment details to the verifier or any middleman. The committer can at a later time reveal the secret to the verifier, and the verifier can reconstruct a new commitment using the secret. If the reconstructed commitment equals the initial commitment, then the verifier is sure of the committed value. Pederson commitment is based on difficulty of solving discrete logarithms and can be shown using the following three steps:

Setup: Let p and q denote large primes such that q divides p - 1, G_q is the unique subgroup of Z_p of order q, and g is a generator of G_q. Let $h \in G_q$ be chosen such that nobody knows $log_g h$. All p, q, g, h are chosen by a verifier or trusted third party and made public.

Commit: Let m denote a secret message, or commitment, and $r \in Z_q$ be a random value, and compute:

$$c_1 = C(m, r) = g^m h^r \bmod q \tag{3.1}$$

We will denote the result of this computation as commitment, c_1. The committer sends c_1 to the verifier at time τ_1.

Open: To verify the commitment the verifier receives m and r from the committer, at some time τ_2, where $\tau_2 > \tau_1$ and computes a second commitment c_2 using the same method:

$$c_2 = C(m, r) = g^m h^r \bmod q \tag{3.2}$$

It then proceeds to verify the authenticity of the commitment:

$$c_1 == c_2 \tag{3.3}$$

The work in [Ped91] shows that the commitment c_1 reveals no information about the secret message m and nobody can open a commitment made with m using $m' \neq m$ unless they know $log_g h$. This is called the hiding and binding property respectively.

3.5 Threat/Network Model

In this section we describe our Network and Threat models considered.

3.5.1 Network Model

Our network model is depicted in Figure 3.1, and each step is described below:

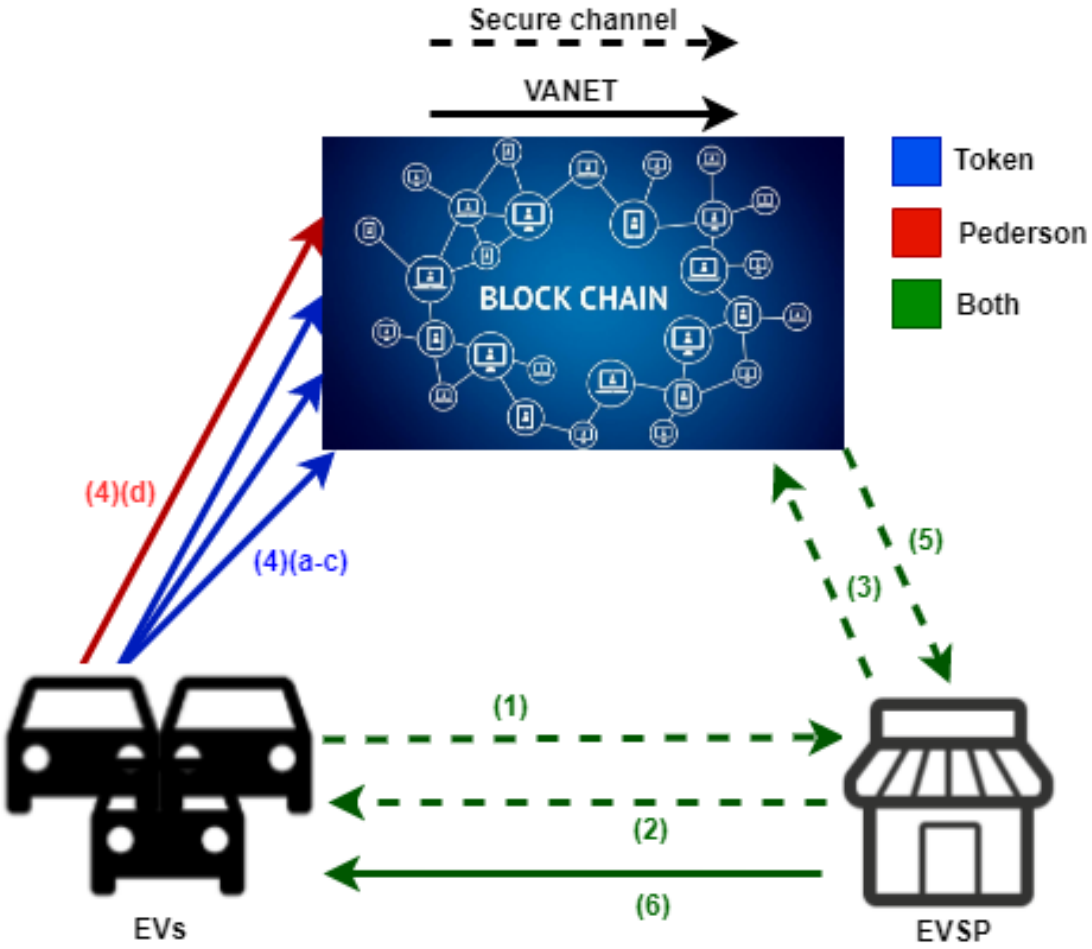

Figure 3.1: Network Model: the steps indicated represent the communication between Electric Vehicles (EVs), Electric Vehicle Service Provider (EVSP), and Blockchain When scheduling charge slots.

1. The first step is where EVs register with the EVSP. To do so, EVSP first authenticates the EV and prepares required parameters for zero-knowledge commitment.
2. EVSP returns common proof generation parameters to authenticated EVs.
3. EVSP deploys the verification smart contract to the blockchain.
4. In this step phases a-c represent the token approach meanwhile phase d represents the Pederson approach.

(a) EV submits the zero knowledge proof for authentication to the Blockchain verification smart contract. If verified the contract issues a *scheduling-token* to the EVs pseudorandom address.

(b) EV submits his/her charging scheduling request to the verification smart contract passing it's *scheduling-token* in the process. In return the contract returns a *charge-token* to be used at the charging station.

(c) EV arrives to the charging station and authenticates itself with a *charge-token*.

(d) EV submits a zero knowledge proof, encrypted schedule, and Pederson Commitment to the verification smart contract. If verified successfully the EVSP will store the commitment for charging station verification.

5. EVSP retrieves scheduled EV from Blockchain emitted event and reserves the charging station for that period.

6. EVSP approves charging for EV. For Pederson approach this is done by verifying the Pederson Commitment. For Token approach the EV passes the *charge-token* and is verified.

3.5.2 Threat Model

The adversary in our model can see all blockchain transactions, and can also be a nefarious EVSP. Threats pose risks to the EV that include but are not limited to location privacy leakage, physical security, loss of PII, service availability, etc. We consider the following threats in the decentralized setup as shown in Figure. 3.2.

- *EVSP Attack:* We assume the EVSP is not trustful and curious about the EV's information when EVs would like to request a scheduling service. A

nefarious EVSP can use the Ev's personal information or charging habits for unauthorized purposes, or distribute the data either unwillingly or with intent.

- *Public Ledger Attack:* Since the ledger is public, anyone can have access to it and analyze the transaction details of an EV with EVSP or charging stations. Thus, if an EV authenticates on the public ledger his/her credentials will be exposed.
- *Man in the Middle Attack:* A malicious adversary can capture EV's scheduling requests and analyze the packets to obtain private information.
- *Replay Attack:* A malicious adversary can use the submitted proofs to Blockchain and re-submit them to authenticate itself.
- *Charging Station Attack:* The charging station can analyze EV's request and proofs to find out private information.
- *Denial of Service Attack:* A EV can produce excessive amounts of charging periods, causing a shortage of available charging periods for other EVs.

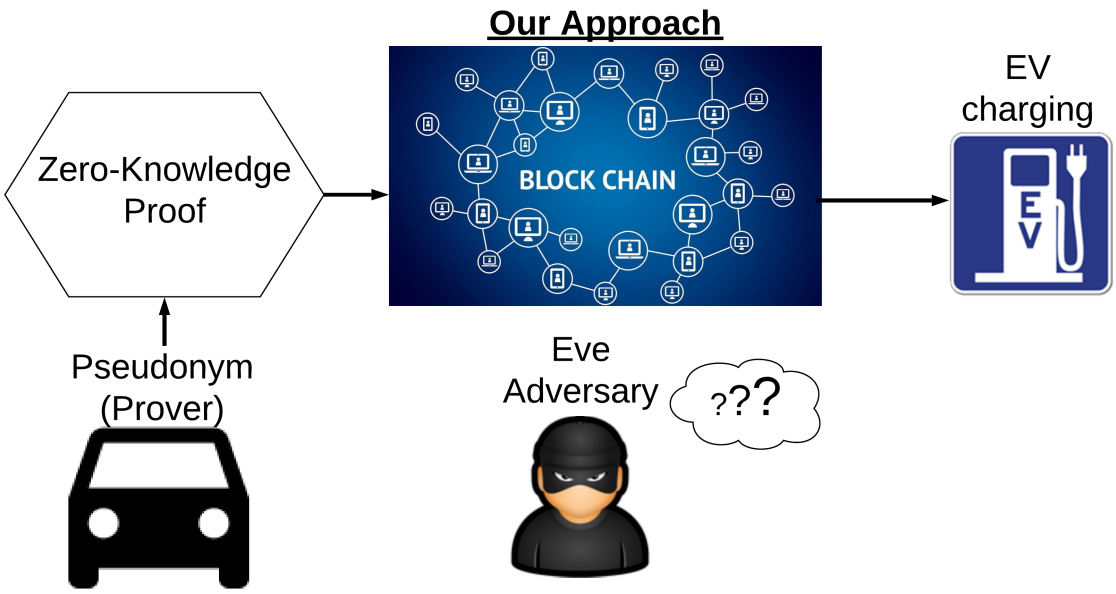

Figure 3.2: Attack Model for our Application: Eve the adversary can see all blockchain transactions, she can also be part of the EVSPs organization.

CHAPTER 4

PROPOSED TOKEN-BASED APPROACH

In this chapter, we describe our proposed token-based framework in detail. This approach is one of two approaches we deploy and we first start this chapter with an overview and then move onto its components.

4.1 Overview

To protect the EVs' privacy, we propose a novel framework that combines the zk-SNARKs and the Ethereum distributed ledger. The idea is to integrate the zero-knowledge proof process in Protocol 1 to the blockchain environment. Specifically, the elements of Protocol 1 are mapped to EV charging setting as follows: The *Prover* is the *EV*, the *TTP* is the *EVSP*, and the *Verifier* is the *Blockchain* which holds smart contracts. In our case, we assume Ethereum-based smart contracts for our approach.

Briefly, the process works as follows: The EVSP creates a secret function and passes it along with a proving key to the EVs. An EV solves the secret function to create a witness, then uses the proving key and witness to generate a proof to show that it knows the secret function EVSP created. The EV then contacts a Blockchain authentication contract and presents the proof. Once the contract verifies the proof, it generates a Service-token for the EV to be used for scheduling charging. This service-token is spent in scheduling and the EV uses a returned charging token at any charging station. In this way, the EV does not need to authenticate itself to the EVSP.

4.2 Registration of EVs

We assume that the EVs will register with the EVSP in advance. Basically, the EV identifies itself to the EVSP where it is looking to use the services of the EVSP. This registration is much like the registration of any web services where the EV would provide information such as name, address, EV model, etc. Registration information is not used during scheduling or charging as the EVs will remain anonymous to the EVSP. Basically, it assures the user is a legitimate EV, and once registered the EV will obtain the necessary information for charge scheduling, outlined in the next section.

4.3 Setup Phase

Once the EVs are registered, the rest of the process starts with the **Setup phase** where the EVSP generates two elements to be given to the registered EV users and one element for writing to the blockchain as summarized in Figure. 4.1.

First, the EVSP generates a *secret function*, $f(x)$, to be shared with the registered users so that they can generate a *witness* and a *proof* to authenticate themselves and schedule EV charging later. $f(x)$ might employ a series of conditions for its input variables including arithmetic operators and flow control statements. Algorithm 1 provides an example of such a secret function where X is a list of 3 inputs. In this function, the 3 inputs are accumulated into a variable *sum* which is eventually compared to another variable *tot*. If *sum* is greater or equal than *tot*, the function returns *sum*, else it terminates. Note that finding the variables that satisfy the conditions of the secret function is called *witness* which happens in the **Attestation generation phase** as will be detailed next.

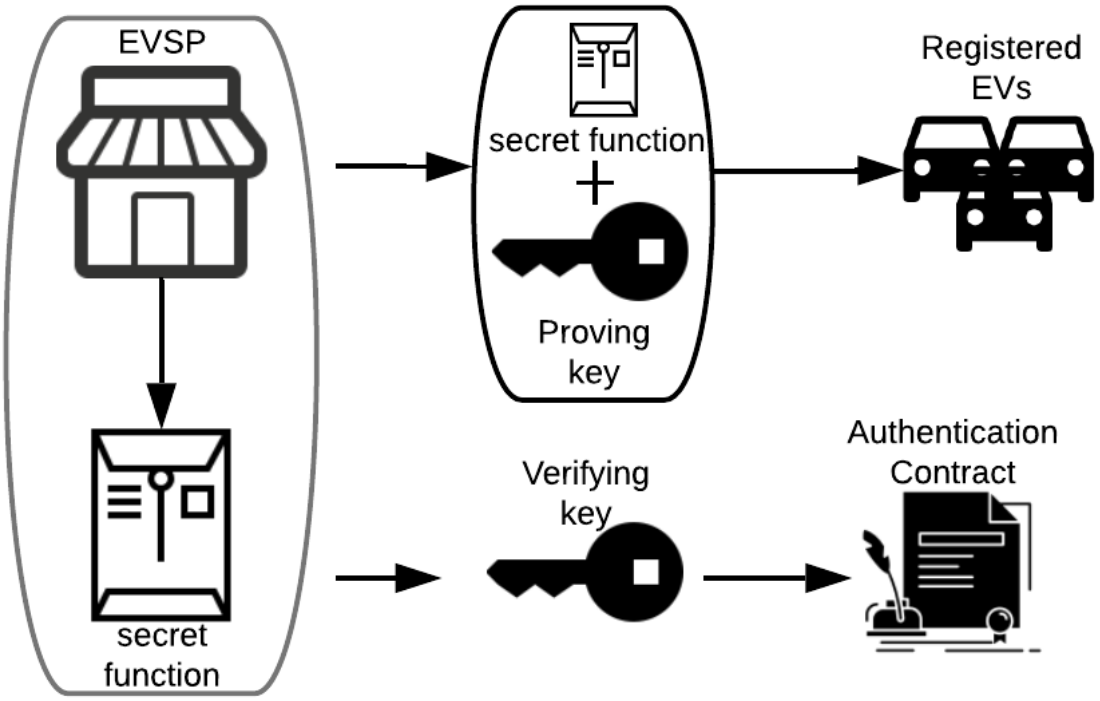

Figure 4.1: Setup Phase: EVSP distributes proving keys, and the secret function to authorized users. EVSP also creates an Authentication smart contract with the verifying key embedded.

In the **Setup phase**, the EVSP also creates *Proving* and *Verifying* keys from a Common Reference String (CRS) which will be used to create and verify proofs. While the *Proving* key is sent to EV users, the *Verifying* key is sent to a Blockchain smart contract which is created for EV Authentication. This smart contract, which is shown as *Authentication Contract* in Figure. 4.1 is public on the Ethereum network and is used for authenticating the EV.

During the Setup phase, the EVSP deploys two tokens, *service-token* and *charge-token* from a pool of tokens maintained by it. These tokens will be used in the next phases.

Algorithm 1 Secret Function 1

Require: $X.length() = 3$

```
1: procedure MAIN(X)
2:     sum = 0
3:     tot = 10
4:     for each integer i in X do
5:         sum = sum + i
6:     if sum ≥ tot then return sum
7:     else
8:         terminate
```

4.4 Attestation Generation

Once an EV user is registered and receives the authentication elements, it starts the **Attestation generation phase** where the EV generates a proof which attests its knowledge of the secret function, $f(x)$.

The EV begins this process by assigning a set of variables that will satisfy the parameters of the $f(x)$. It is assumed that the EV knows $f(x)$ and thus can provide satisfying values. This assignment of variables is called generating a *witness*. If we consider the $f(x)$ portrayed in Algorithm 1, a witness for this $f(x)$ will be any 3 variables whose *sum* is greater than or equal to 10 (e.g., 2, 3, 5). The witness also includes the return value (i.e., the output) based on the selected parameters, in this case, 10. Although Algorithm 1 is a simple function, we provide it as an example for making it easier for the reader to understand the concept.

Next, using the witness along with the proving key, the EV generates a *proof*, namely π, attesting to its knowledge of the $f(x)$. We also append a timestamp as public input to the proof to prevent potential replay attacks. The reason for making the timestamp a public input is to allow it to act as a plaintext input to the smart contract.

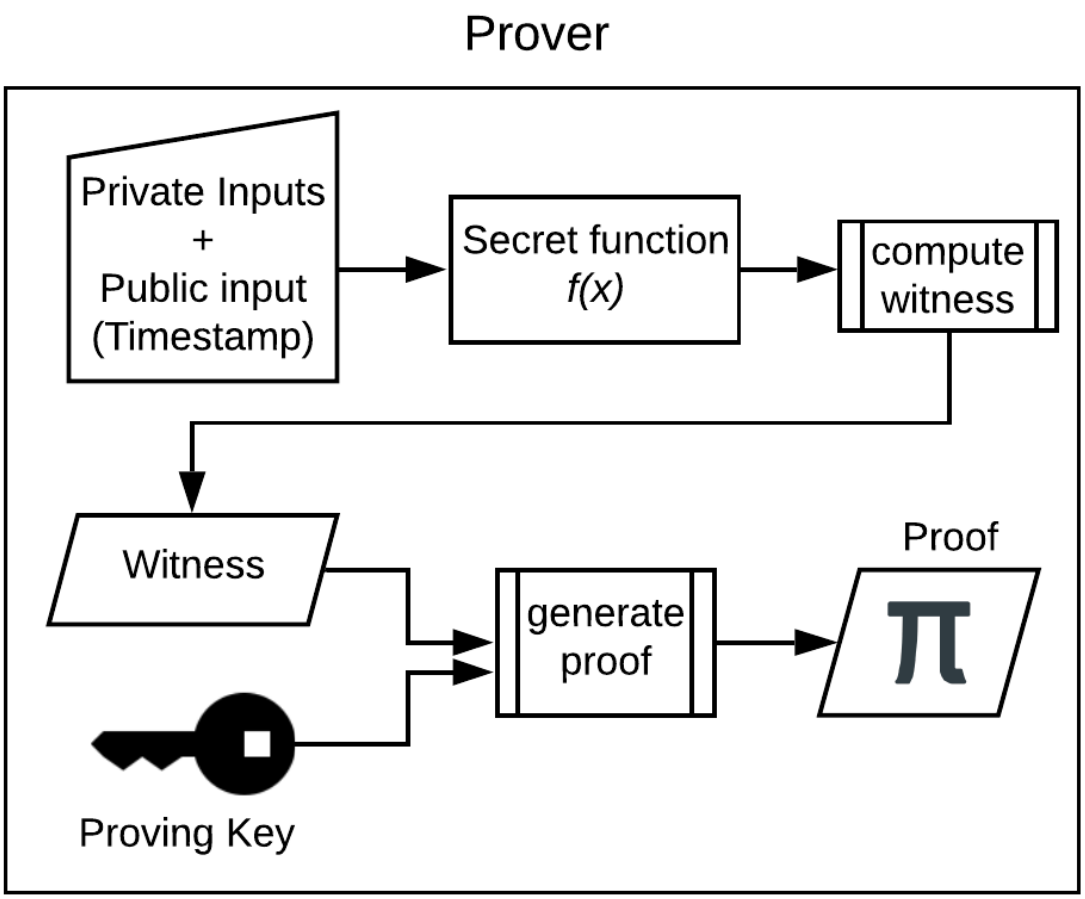

Figure 4.2: Proof Generation: EV computes a witness using values that satisfy the secret function and a public inputs. With the witness and proving key, the EV generates a proof.

This process is shown in Figure. 4.2. Note that all of the steps except the deployment of the *Authentication contract* up to now are performed off-chain, meaning they are not written to the blockchain. The *Authentication smart contract* is created on the blockchain for achieving our goal of distributed authentication.

4.5 Verification for Authentication

Upon possessing the valid proof, π, for $f(x)$, the EV submits his π to the *Authentication smart contract* on Ethereum for authenticating itself. The EV uses his pseudonym address, to interact with the contract. Importantly, the EV will generate a new pseudonym address each time it wishes to schedule a charging service with the EVSP. Generating new pseudonym addresses is easy (performed almost instantly), free of cost, and done off-chain. Along with π, an array of public inputs including the output of $f(x)$ (given the witness values), and optional public parameters to $f(x)$, are also passed to the Blockchain. Once the authentication is successful, the smart contract issues a *service-token* that belongs to the EV. Note that this token is assigned to the same pseudonymous address that was used to interact with the smart contract by the EV. The authentication process is shown in Figure. 4.3.

4.6 Charging Scheduling

The *service-token* received after authentication is later used by the EV for scheduling the charging on another smart contract which is referred to as *scheduling smart contract*. Basically, the EV submits the *service-token* as well as its desired available time slots to the scheduling smart contract on Ethereum. The address used to submit this *service-token* is by default the address which received it from the authentication smart contract.

The scheduling information is sent encrypted with the *public key* of the EVSP to hide the charging details from the public which is depicted as λ in Figure. 4.4. EVSP also interacts with the scheduling smart contract and hence can receive the encrypted scheduling information to properly schedule a time slot for the spender of that service-token. The scheduling smart contract returns a new *charging-token* to

the EV, which can be spent during its scheduled charging slot for charging. During this phase, the EV also sends a deposit fee in the form of Ethereum which serves the purpose of a security deposit for charging. The deposit fee is the same for everyone, this way there can be no pattern tracing to identify a specific EV. Without this fee, scheduling would be almost free and this could be abused by malicious EVs to schedule a large number of fake charging appointments. This process is shown in Figure. 4.4.

4.7 EV Charging Station Verification

When the EV arrives at the charging station at it's allocated charging time slot, the charging station will receive the charging-token at its own address. Then the EV charging station verifies with the EVSP that the EV which submitted the charging-token is the currently scheduled EV. Basically, it contacts the smart contract with the token and receives the information about the ID and time (encrypted) to make a comparison.

Note that with the charging-token, the EV also makes a payment (in addition to deposit fee) in the form of Ether cryptocurrency to the charging station. Using the proposed blockchain framework allows the EV to pay anonymously in this step which is important for privacy. Following the verification of the charging-token and sufficiency of funds, the EV will proceed to charge and send a message to EV. We assume there will be a communication channel (i.e., cable or wireless) that will run the protocol between the EV and the charging station to send this approval message. The charging fee paid by the EV in this phase represents payment for the amount of charging it needs. Figure. 4.5 demonstrates this charging verification process.

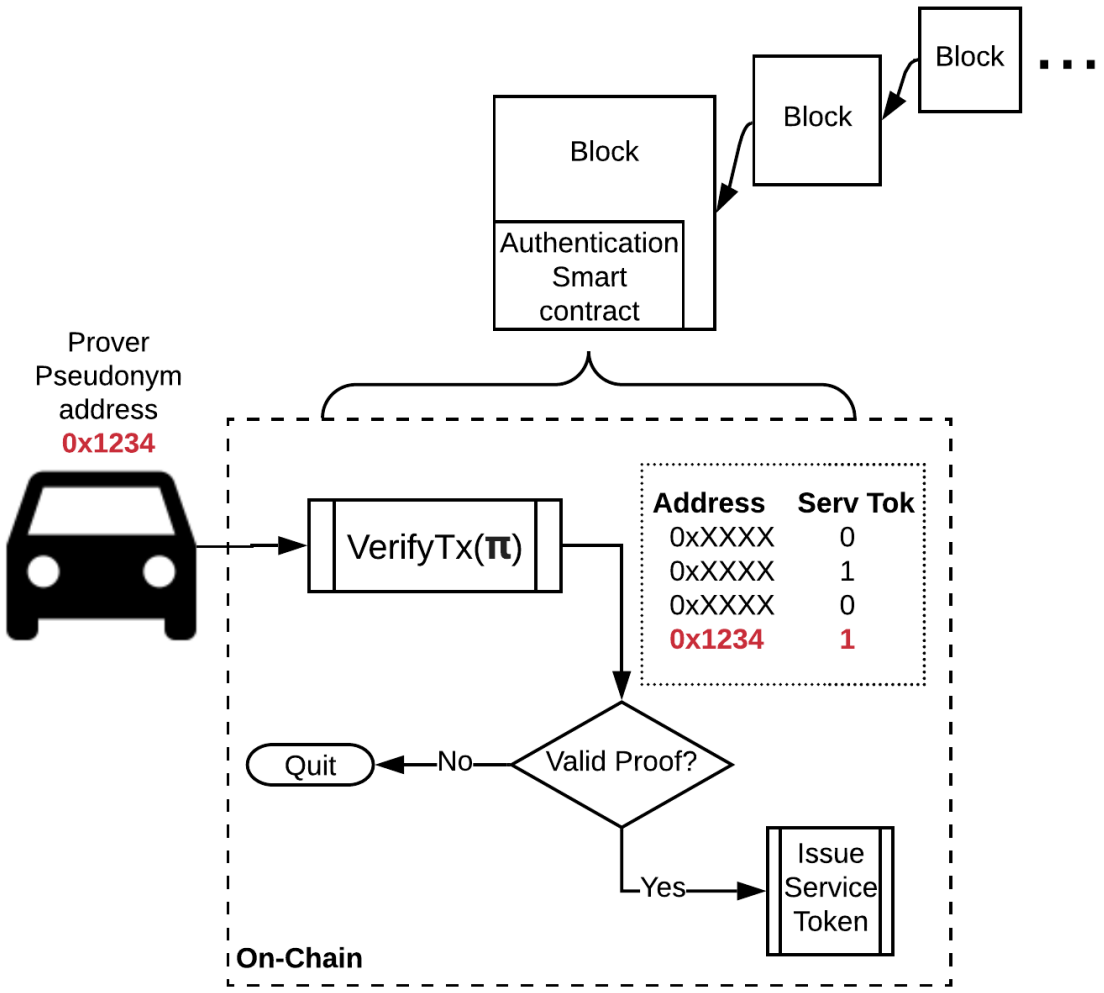

Figure 4.3: Authentication: EV submits a proof to the Authentication smart contract. If the input is valid, then the smart contract returns a service-token to the EV's pseudonym address.

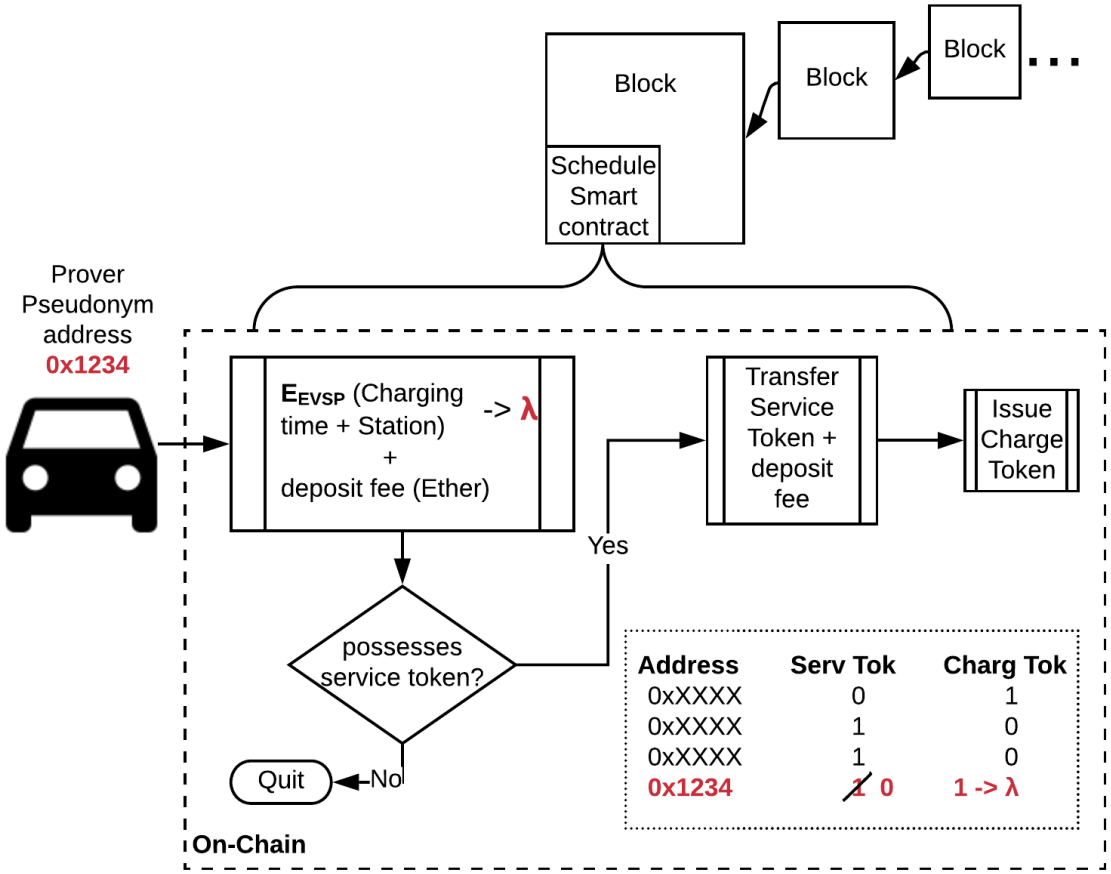

Figure 4.4: Scheduling: EV spends its service-token on a scheduling contract to receive a charging-token for EV charging.

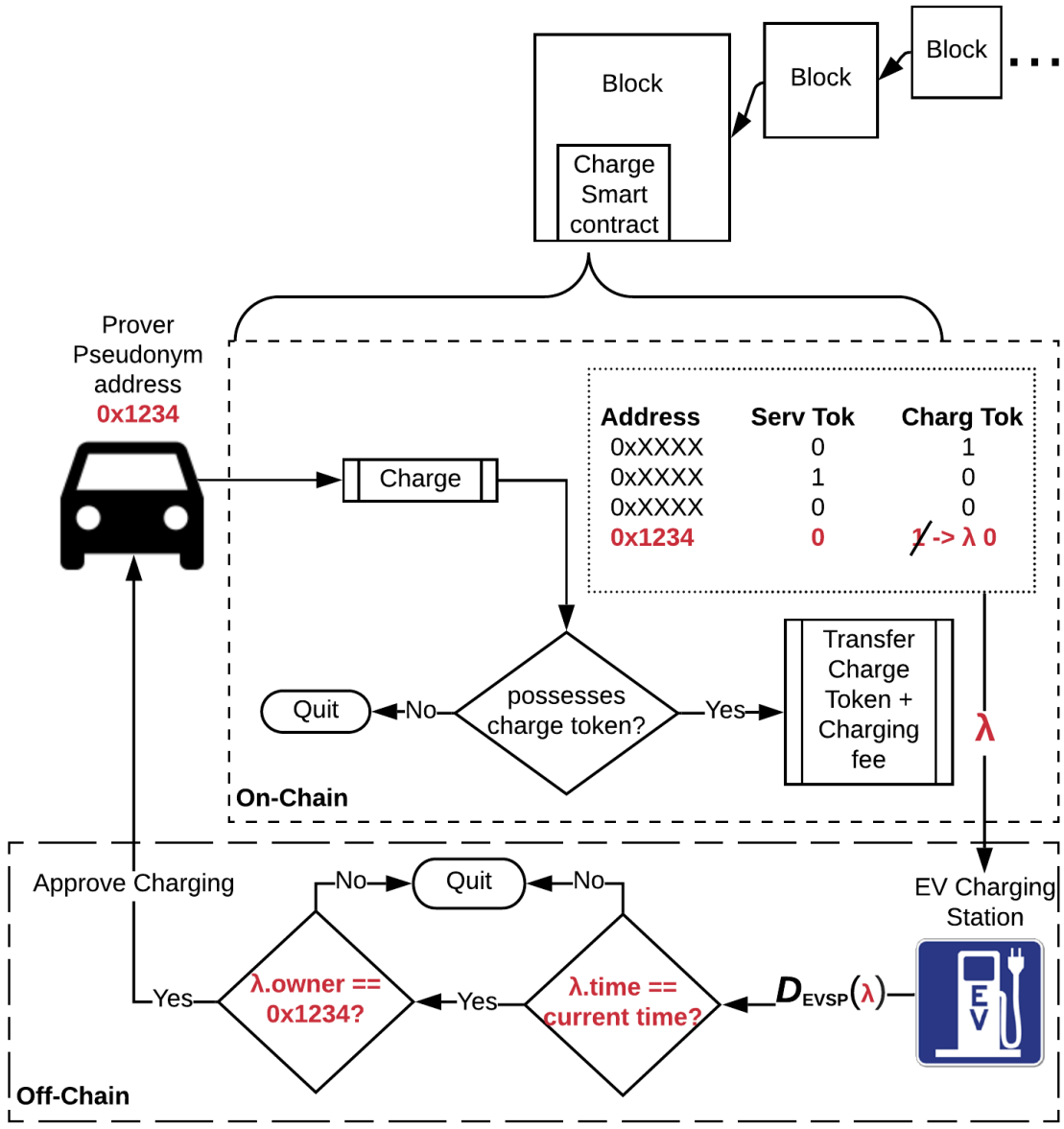

Figure 4.5: Charging: EV uses his charging-token at a charging station during its allotted period.

CHAPTER 5

PEDERSON COMMITMENT APPROACH

The aforementioned approach provides zero-knowledge and anonymity when scheduling charging for EVs. However, there is still some overhead associated with the use of tokens in performing authentication, scheduling and charging on Ethereum in terms of efficiency and transactions costs.

Therefore, as an alternative to tokens, we propose adapting Pederson commitment scheme which is a method of sharing a secret that is both binding and hiding. In our privacy framework, we propose using this commitment scheme to allow an EV to commit to a scheduled charging slot. This improves the efficiency of the framework and reduces the cost by reducing the transactions performed on-chain, including the removal of the scheduling, and charging contract completely. In the following sections, we describe the changes needed at each step in the proposed framework.

5.1 Setup Phase

The Pederson Commitment parameters along with $f(x)$, and proving key are given to registered EVs, as shown in Figure. 5.1. For the this approach, it is necessary for the EVSP to generate the required parameters for a Pederson Commitment scheme. That is, the EVSP will perform the setup phase of the Pederson Commitment scheme which is described in section 3.4. The outcome of this setup step will be the public Pederson Commitment parameters, namely p, q, g, h. These parameters are sent to registered EVs to be used in generating a commitment that contains the requested schedule for charging.

In this approach, we implement a different $f(x)$ which can be proven to be difficult enough that it is cryptographically secure for our use case. In this case, we

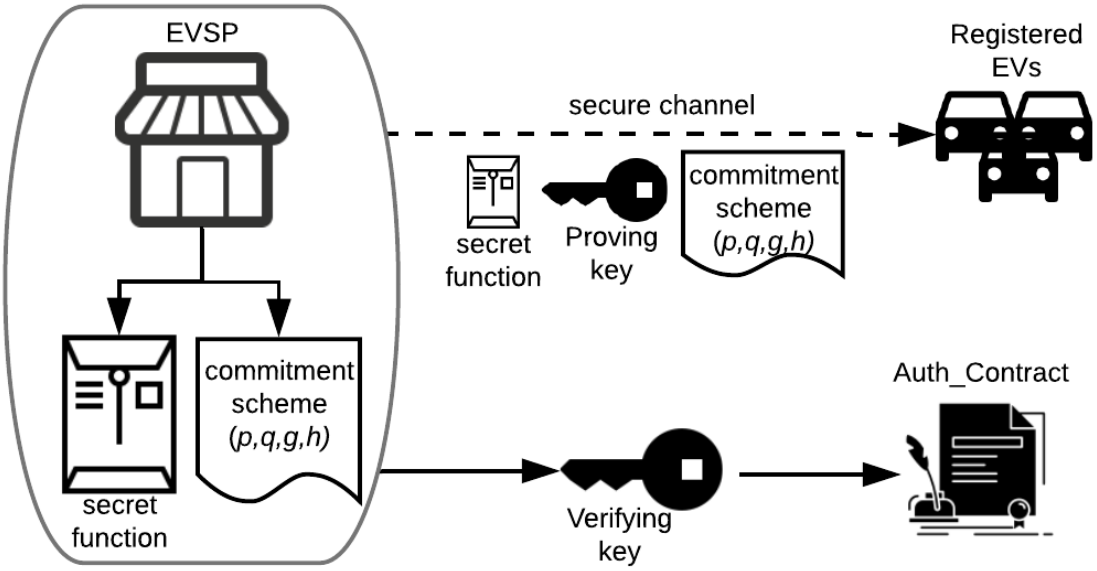

Figure 5.1: Improved Setup: EVSP distributes proving keys, the secret function and the Pederson Commitment parameters p, q, g, and h to authorized users. EVSP also creates an Auth_Contract with the verifying key embedded.

use an $f(x)$ that requires the prover to prove his/her knowledge of the preimage of a hash function. Thus, the difficulty of providing an invalid input to solve the secret function lies in the difficulty of generating an X' such that:

$$X' \neq X$$

$$\&$$

$$H(X') == H(X)$$

.

The new $f(x)$ is shown in Algorithm 2. It accepts a private input X, and returns a boolean value, true or false. It is true if the hash of a user's input, $hash\_image_1$, is equal to the hash of a secret value, $hash\_image_2$, known only to the EVSP and authorized EVs.

Algorithm 2 Secret Function 2

```
1: procedure MAIN(private X) returns (bool result))
2:     hash_image_1 = H(X)
3:     hash_image_2 = b94d27b9934...
4:     if hash_image_1 == hash_image_2 then return 1
5:     else return 0
6:         terminate
```

During the setup phase, the EVSP also deploys a single smart contract, called *Auth_contract* for brevity. The *Auth_contract* performs the following functions in order:

1. Performs authentication of zkSNARK π granting access to authorized EVs.
2. Accepts a Pederson Commitment containing a scheduled charging slot chosen by the EV.
3. Accepts a message encrypted with the public key of the EVSP that states the desired charging slot.
4. Accepts payment in the form of Ethereum for the associated charging appointment.

The *Auth_contract* posts an event whenever an EV authenticates and schedules charging, from which the EVSP uses the private key to decrypt the message and book the charging slot for the EV who can provide the secrets to open the Pederson Commitment. The setup phase ends when the EVSP sends the Pederson Commitment parameters along with the $f(x)$, and proving key to registered EVs, as shown in Figure 5.1.

5.2 Attestation Generation

Next, the EV must attest his knowledge to $f(x)$ and the parameters of the Pederson Commitment. In this case attesting knowledge to $f(x)$ means he/she knows the preimage of hash image using $H(.)$. The EV inputs the preimage to $f(x)$ and if the

$$H(input) == hash\_image$$

then the EV will receive a witness. It is important to note that the input is private and not made public during the authentication *on-chain*. In this approach, we again utilize the notion of a public input that is a timestamp to avoid the reuse of the same proof. Combining the generated witness and the proving key received from the EVSP during setup, the EV will generate a proof, π. The proof generation process is shown in Figure 4.2.

Additionally, the EV must attest his knowledge of the Pederson Commitment parameters and thus be able to use these parameters for creating his own commitment. A security benefit of the Pederson Commitment is that even if the parameters, namely p, q, g, h, were to be released to the public or published on a public site, an adversary cannot use this information for an attack on the commitment. For our sake, the EVSP sends the Pederson Commitment parameters along with the proving key and $f(x)$ as the combined size of the parameters is only 188 bytes and thus does not introduce heavy overhead. The EV uses equation 3.1 to create a commitment. In this case m, the secret, is a *unsigned integer* representing the charging slot chosen. Additional digits are appended to represent the charging station where the EV wishes to schedule the charging. The other input to a commitment is a random value, r, which is randomly generated by the EV and stored for opening the commitment in a later period. Using m, r, p, q, g, and h the EV calculates a commitment, c_1, of order q using Equation 3.1.

5.3 Authentication and Commitment

At this point, the EV has a π, and a commitment, c_1. Additionally, the EV knows what charging slot it would like to schedule, as this is contained within c_1. Knowing the charging slot, the EV writes a message, λ, and encrypts it using the public key of the EVSP. This message is designed to allow the EVSP to book the charging station for that charging period. Now possessing these three parameters, the EV uses a pseudonym address to send them to the *auth_contract*. The EV also includes any payment in the form of Ethereum that is required for the charging, as well as public inputs and the output to $f(x)$.

Assuming the EV has provided a valid π, he/she would have committed to that charging slot and no longer needs to interact with the blockchain. The process of authenticating and committing to a EV charging schedule is shown in Figure 5.2.

5.4 Scheduling

The *auth_contract* verifies the π triggering an event, passing λ and c_1 to the EVSP. The EVSP decrypts λ to book the charging station for that period and associates it with the c_1 and the payment. For maintaining the commitments and associated charging details, we store them *off-chain* in a database called *EVSP commitment store*. The charging station will retrieve the commitments from this storage for use in authenticating the EV for charging.

5.5 Charging Station Verification

The final step to this approach is verifying the EV when it arrives for charging during its assigned period. When the EV arrives to the charging station, it uses a

secure channel to transmit the secrets to open the *Pederson Commitment*, that is m and r which will open c_1. Using m and r, the EVSP creates a new commitment c_2 using Equation 3.2 and compares it to c_1 that is associated with that charging time using Equation 3.3. If the commitments are equal then the EVSP is sure the EV that has arrived is the same one that committed to that charging slot when it authenticated using the *auth_contract*. During this step the EVSP also verifies that the current c_1 represents the scheduled EV. This process is shown in Figure 5.3.

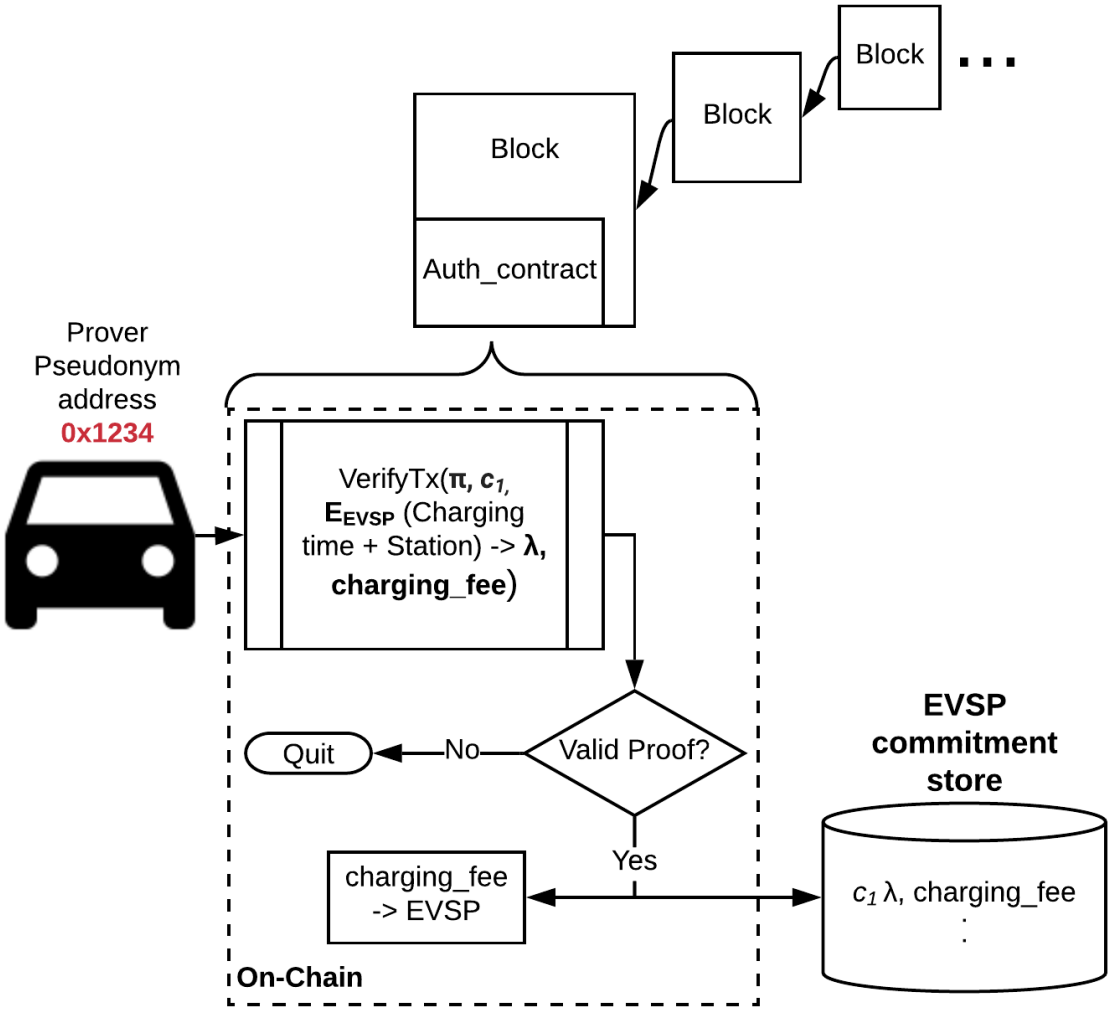

Figure 5.2: Improved Authentication and Commitment: EV submits a proof π, a commitment c_1, a message λ containing charging schedule encrypted using the public key of the EVSP, and any charging fees in the form of Ethereum.

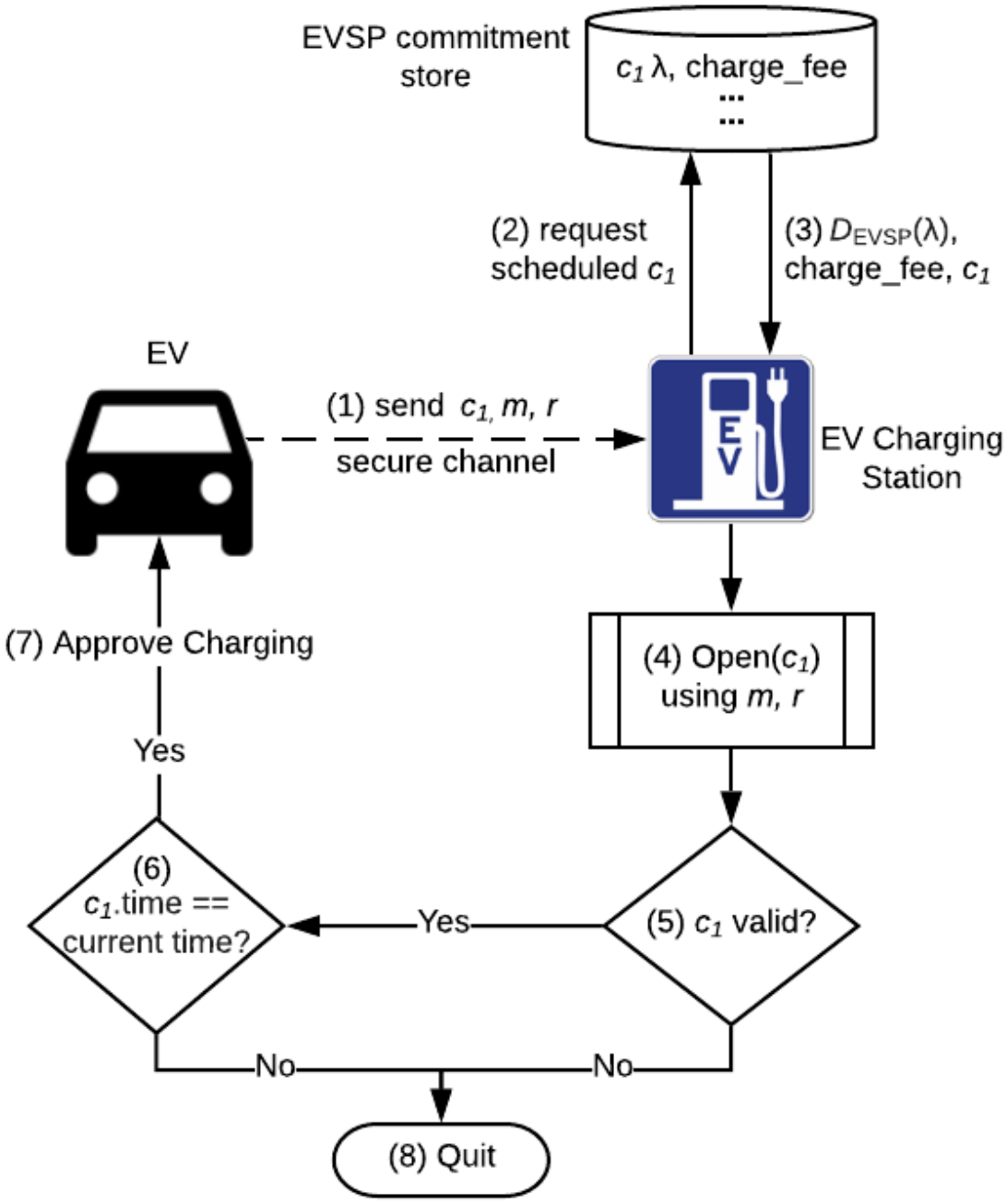

Figure 5.3: Improved Charging: EV sends its original commitment c_1, and the secrets to open it, m and r using a secure channel. Then, the charging station retrieves the scheduled commitment and compares it to the commitment from the EV. If the commitments are equal then the EV that has arrived is the scheduled EV. Next, the EVSP opens c_1 using the secrets provided by the EV. If the c_1 is valid and contains the correct time then charging is approved.

CHAPTER 6

SECURITY AND PRIVACY ANALYSIS

In this chapter, we first provide a security analysis of how our approach addresses the threats mentioned in our Threat model. Within each analysis, we mention how it relates to token-based approach or Pederson commitment if needed. Then, we provide a privacy analysis which demonstrates how the privacy of the EV is preserved using our framework.

6.1 Security Analysis

EVSP Attack: Our framework mitigates EVSP attack risk by authenticating EVs to the EVSP through merely a zero-knowledge proof, thereby, hiding the identity of the EV. In our framework, the EVSP is only aware that an authorized EV is requesting to schedule a charging slot. EVSP cannot gain any personal information from the EVs authentication process including charging habits, location, and personal information.

Public Ledger Attack: Even if different pseudonyms are used for each transaction, tracking the flow of funds may expose some privacy. A fund flow contains a variety of information such as public addresses and transferred amounts (e.g., transferred amount pattern). Then, a flow-based analysis enables eavesdroppers to identify and monitor interacting parties. Using a similar analysis, an eavesdropper, for our token-based approach approach, can try to match different token transfers to a single EV identity even though the EV uses different pseudonyms. We claim that this is not possible in our approach, because the token transfer to various pseudonym addresses is essentially the same (including the deposit fee) for all cases since the amount for the transaction will be the same (i.e., we use the same token). Thus,

an adversary cannot differentiate among these and link certain pseudonyms. Here, another possible passive attack by the adversary could be doing timing analysis on the public ledger to try to match different pseudonyms to an EV. This is also not possible due to two reasons: 1) the token transfers are very in-frequent (i.e., in order of days); and 2) since EV switches to a new pseudonym for every new charging service regularly, a consistent pattern is not possible to identify. We also note that a passive adversary can not obtain details of scheduling request either since the request is encrypted with the public key of the EVSP and therefore malicious actors can not obtain details of the scheduling event. In case of Pederson, opening and/or finding a c_1 which is equal to the commitment submitted is sufficiently difficult that an adversary cannot find the secrets to recreate the commitment.

Man in the Middle Attack: Since each EV sends the scheduling information encrypted with EVSP's public key to the blockchain, even if an adversary could capture this packet, s/he will not be able to access its contents. Also, the ID of the EV is anonymous and changing regularly.

Replay Attack: We prevent proofs from being replayed by adding a timestamp as input during the attestation generation phase. The timestamp is a public input and therefore it is submitted to the smart contract when verifying the proof. The authentication contract then verifies the current time. If the current time is within the proof validity period, then the proof is valid. Otherwise, it is denied.

Charging Station Attack: For the token-based approach, when the EV arrives for charging, it submits only a charging-token with a pseudorandom Ethereum address to get charged. During this transaction, no details that can lead to the ID of the EV are revealed. In case of Pederson, when the EV arrives, it submits the c_1, m, and r. The EV does not identify itself to the EVSP since it is sufficient that it knows the scheduled c_1 and the secrets to open the c_1.

Denial of Service Attack: In token-based approach, EVs can schedule as many charging as they want. However, this requires a deposit fee payment in advance, which will be written on the charging token and thus an adversary will have to limit its scheduling requests. Therefore, DoS attempts to make bogus scheduling will not be possible. For Pederson, payment for scheduling charging is sent upfront during authentication and commitment. Thus, excess scheduling would be a costly attack which will not be attractive.

Secret Function Attack: If an unauthorized EV gets a hold of the secret function details, then they can generate proofs and use the EVSP charging service. However, the secret function sent to authorized EVs doesn't reveal the details as it is not human readable and debugging the program will be computationally difficult, even if an adversary gets a hold of both the secret function and proving keys. Thus, it is extremely difficult for an adversary to gain knowledge from the secret function unless they know the input that satisfies the secret function. On this note, we also use a pay-per-use service as opposed to a subscription based service. That means in the unlikely event that an adversary does generate false proofs, in order to schedule and charge his EV, he will need to pay for service when submitting his proof.

6.2 Privacy Analysis

The information regarding an EV's ID, choices, location, etc. was the main concern for exposure. Through our security analysis, we have shown that this information will not be exposed to parties that are not trusted. Briefly: 1) EVSP will not have access to any of this information while it can still authenticate EVs via zkSNARKs; 2) Charging station will only know the ID but this is anonymous so no other information will be revealed. for the Pederson approach the charging station only

knows the commitment secrets, these are changed for every scheduling; and 3) Any adversary capturing traffic or accessing the public ledger will not be able to obtain any scheduling or location information and do any matching of IDs. Additionally, since the EV can provide payment to the EVSP with the same anonymous ID in a decentralized method, using blockchain, the EVs Privacy holds. Therefore, the proposed approaches enables the privacy of the EVs in a decentralized manner.

To represent the probability of an adversary predicting the EV when scheduling charge, we calculate the probability of the following attempts:

1. A **third-party** adversary predicting the EV by monitoring smart contract transactions. In this case, adversary's prediction is lower-bounded with the number of EVs that interacts with the smart contract. Let's assume that we have 10000 different EVs, and since the EV uses a different address in each charge scheduling, then the prediction of an adversary will be $p1 = 1/10000$. A prediction plot of an adversary for consecutive correctness would be drawn as:

$$p1^n \quad or \quad \frac{1}{10000^n}$$

for n consecutive predictions of the EV when scheduling charge. This means the adversary would have a 0.01% chance of predicting the EV on it's first charging schedule, and 0.000001% of predicting the same EV on it's next charging schedule.

2. A malicious **EVSP** has more information then a third-party since it is aware of the requested charge scheduling (i.e. time and location). Thus, the EVSP has a higher ability of predicting the EV that is scheduling charge. Since the EVSP knows the schedule location it can predicts the EVs identity from the registered user for that area. Let's assume that registered number of user in

a specific area is 100. Then, the prediction of EVSP will be $p2 = 1/100$. Consecutive correct prediction will be similar to the previous case and can be drawn as:

$$p2^n \quad or \quad \frac{1}{100^n}$$

In this case the EVSP has about a 1% chance of predicting the EV on its first charge scheduling, and 0.01% of predicting the same EV during it's second charge scheduling.

3. The third case is related to matching EV and related payment transaction. While considering the payment and scheduling process is decoupled (the used pseudonym IDs will be different), EVSP must guess the matching payment and EV ID. In this case, EVSP will know the time frame of the transaction due to the fact that the EV makes a payment when it shows up. So if we assume that there are 1000 transaction in that time frame, the 1 out of 1000 transaction will belong to the EV itself. Let's make this probability p3=1/1000, then the EVSP can correctly match in p2 * p3 probability. The probability of this will be drawn as:

$$p2^n * p3 \quad or \quad \frac{1}{(100^n * 1000)}$$

For an EVSP to match the EV and corresponding payment transaction on the first charge scheduling the probability is about 0.001% and predicting the same payment transaction for a second charging schedule is about 0.00001%.

CHAPTER 7

EVALUATION

In this chapter, we describe and evaluate our implementation of the framework. First, we describe the tools used and consider the overhead on the EV when creating the proof and Pederson commitment. Next, we evaluate the performance of *on-chain* transactions, including contract deployment and for the token approach the 3 transactions including authentication, scheduling, and charging. For the Pederson approach we evaluate the one and only *on-chain* transaction used for both authentication and scheduling. The results are shown and the two approaches are compared in the final section of this chapter.

7.1 Experimental Setup

Our framework for evaluation is built primarily over Zokrates [ET18] which is an all-in-one tool used to implement zkSNARKs into a distributed ledger. Zokrates was used for generating the witness and proofs, using two commands, compute-witness and generate-proof respectively. We evaluate the performance of these two phases on an EV by using a Raspberry PI 3 model B V1.2 (Quad Cortex A53 @ 1.2GHz, 1GB SDRAM, 4000Hz Videocore IV). We evaluate the performance of the protocol based on time and the size of the proof.

Additionally, we used Zokrates for implementation onto the Ropsten Test network which was used for on-chain transactions through the smart contracts. Ropsten is a Ethereum network simulator and is meant to mirror the Ethereum main network. It uses a Proof of Work (PoW) consensus algorithm, like Ethereum main net, which means miners solve some complex puzzle to verify a transaction and add a block to the blockchain. To create a smart contract on Ethereum, we used the export-verifier

command of Zokrates which generates our smart contract written in Solidity which is the native language for writing smart contracts on the Ethereum. The verifying key is embedded into the Ethereum smart contract used for authenticating the EVs. To construct our tokens, we used the guidelines in [Net13] to construct an ERC20 standard token. For the Pederson commitment approach we use a similar smart contract that verifies the π, stores c_1, accepts payment, and triggers an alert to the EVSP. It does not rely on ERC20 token contracts.

For the Pederson commitment, we reduced the computation time by using the multiplicative property of the modulo function. Basically, Equation 3.1 and 3.2 are expanded to the following equation:

$$c = C(m, r) = (g^m \bmod q)(h^r \bmod q) \bmod q \tag{7.1}$$

Using Equation 7.1, we were able to compute a commitment that is 256 bits in under 10ms and 1048 bits in less than 45ms. Since verifying the commitment is done so quickly if a charging station already has the scheduled commitment, we omitted it from the total overhead of the EV in the experiments.

7.2 Performance Metrics and Bechmarking

The following metrics are considered when evaluating the overhead of our approach:

- **Gas**: Ethereum runs on the notion of *gas*, which is the fee paid by the user when interacting with a smart contract. The amount of gas required by a smart contract transaction is determined by the complexity of the transaction. However, there is a gas limit per transaction block on the Ethereum. This gas limit determines how many transactions will fit on each block.

- **Cost**: This is the cost of gas in terms of Ethereum money which is *ether*. The user has the option to choose his/her transaction verification speed by specifying how much ether s/he will pay per each gas unit to the verifier. We used three verification speeds: *slow:* 0.000000001 Ether/gas or 1 Gwei; *medium:* 0.000000003 Ether/gas or 3 Gwei; *fast:* 0.00000002 Ether/gas or 20 Gwei. We also converted this ether cost to dollars using its approximate exchange rate at the time of writing ($144/Ethereum). This exchange rate can vary depending on the value of Ethereum. On the Ethereum main net, there is a recommended minimum gas price for each transaction, which is around 10 Gwei and 13 Gwei for a slow and fast confirmation time, respectively. It is also shown that about 50% of the transaction confirmed used between 4 and 20 Gwei. We use 1, 3, and 20 Gwei on the Ropsten test network to provide the extreme low verification times and the extreme fast.
- **Time**: This is the time required to verify a transaction on the Blockchain. While it may vary, we show average time over 30 trials for different verification speeds which can be chosen by the blockchain user (i.e., slow, medium, fastest).

For the token based approach implementing our framework on Ethereum has four primary transactions: the contract deployment, and the verification of proofs, authentication, scheduling, and charging. We benchmarked these costs in terms of gas, time and money.

To implement the Pederson commitment approach we only need to deploy the *auth_contract* and measure its proof verification costs. The cost of deployment and verification are also benchmarked in terms of gas, time and money.

7.3 Evaluation Results

In this section, we first discuss the computational overhead inhibited by the EV when generating a proof and creating the Pederson Commitment. Then, we evaluate the *on-chain* verification times and cost for the smart contract deployment, this is done only once for each of the approaches. Next, we present the evaluation of *on-chain* transactions including proof verification, scheduling, and charging station verification. Finally, we summarize the evaluations to reveal the total overhead and cost of the framework.

7.3.1 Computation Overhead on EVs

We first evaluated the feasibility of generating a witness and proof on an EV using $f(x)$ with different numbers of inputs. We used SHA3 hash function for Pederson commitment. Our experiments for the token-based approach were done using a $f(x)$ with 16 private inputs and another $f(x)$ with 2 private inputs. Each of these $f(x)$ also have 1 public input, the timestamp. For the Pederson Commitment approach, we used a $f(x)$ with 4 private inputs which represent the preimage of a sha26 hash, and 1 public input, the timestamp. In the experiments, we repeated each of the steps 30 times and at random times of the day.

The results of the experiments are shown in Table 7.1. It is shown that generating a proof for the SHA256 $f(x)$ can take up to 15 seconds. Since the EV can compute and generate a proof at anytime, it is still feasible for an EV to generate a proof(s) for a future charge scheduling. Additionally, the size of the proof is sufficiently small (i.e., 1019 bits) that sending the proof over a V2I communication will not cause any delay for the EV.

Table 7.1: Witness and proof generation time in seconds with functions that have 4, 8 and 16 inputs, number of constraints, and the size of proof in bits.

# of Inputs	# of Constraints	Compute-witness (seconds)	Proof generation (seconds)	Proof size (bits)
4	4	0.007	0.069	1019
8	4	0.008	0.071	1019
16	4	0.015	0.082	1019
16	16	0.010	0.079	1019
16 - sudoku	695	0.081	0.140	1019
4 - sha256	56,985	0.606	14.38	1019

7.3.2 Overhead of Contract Deployment

Next, we assessed the overhead of contract deployment on blockchain. Table 7.2 and 7.3 show the cost of the token based approach contract deployment for 2 and 16 inputs $f(x)$ respectively. Table 7.4 shows the cost of deploying the Pederson based approach *auth_contract*. The results indicate that if slow and medium options are chosen, the costs are almost negligible. Given that this deployment is done only once by the EVSP, time is not an issue here and thus slow mode would be a viable option. Note that both approaches do not require a contract for each EV and thus saves a lot of time and cost.

Table 7.2: Smart Contract Deployment Overhead - 2 inputs - Token Approach

	Time (s)	Gas Used	Cost (Ether)	Cost ($)
slow	15.79	5,479,043	0.00548	0.79
medium	11.95	5,479,043	0.0164	2.36
fast	11.54	5,479,043	0.110	15.78

Table 7.3: Smart Contract Deployment Overhead - 16 inputs - Token Approach

	Time (s)	Gas Used	Cost (Ether)	Cost ($)
slow	22.86	5,479,183	0.00548	0.79
medium	15.64	5,479,183	0.0164	2.36
fast	5.29	5,479,183	0.110	15.78

Table 7.4: Smart Contract Deployment - Pederson Approach

	Time (s)	Gas Used	Cost (Ether)	Cost ($)
slow	41.41	1,616,521	0.00162	0.23
medium	24.32	1,616,521	0.00485	0.70
fast	14.12	1,616,521	0.0323	4.65

7.3.3 Authentication, Scheduling and Charging Overhead

Next, we assessed the overhead for each EV through different contracts using token-based approach. First, we looked at the authentication overhead to see if a different number of inputs would make a difference. Table 7.5 and 7.6 shows the cost of verification of a $f(x)$ with 2 and 16 private inputs respectively, using the authentication smart contract. The results indicate that the gas cost associated to do the authentication transaction is the same for both cases and thus the number of inputs does not have an impact on the overhead. This mainly due to the nature zkSNARKs which does not differentiate between varying computations. The verification times, however, differ due to two reasons: 1) when there is more gas used for a transaction, the miners' process it more quickly since the fee they get will be higher; 2) there is volatility in Ethereum network transactions depending on the specific time they performed. Therefore, we see that with 16 inputs, the verification times are slightly reduced. However, overall we can see that on average the authentication time does not take more than 24 secs in any case.

Table 7.5: Average authentication overhead- 2 inputs

	Time (s)	Gas Used	Cost (Ether)	Cost ($)
slow	24.28	1,693,250	0.00169	0.24
medium	24.08	1,693,250	0.00508	0.73
fast	19.11	1,693,250	0.0339	4.88

Table 7.6: Average authentication overhead- 16 inputs

	Time (s)	Gas Used	Cost (Ether)	Cost ($)
slow	15.97	1,693,058	0.00169	0.24
medium	15.72	1,693,058	0.00508	0.73
fast	14.45	1,693,058	0.0339	4.88

After authentication, we looked at the overhead for scheduling and charging using the token-based approach. However, since these processes do not depend on any varying input, we conducted a single experiment. Tables 7.7 and 7.8 show the scheduling and charging smart contract overheads per EV. We can see from these tables that the costs are very small and almost negligible. The transaction times also do not change much compared to authentication.

Table 7.7: Average scheduling overhead (Token Approach only)

	Time (s)	Gas Used	Cost (Ether)	Cost ($)
slow	18.03	59,061	0.0000591	0.015
medium	11.52	59,061	0.000354	0.0510
fast	7.24	52,439	0.00105	0.151

Table 7.8: Average charge verification overhead (Token Approach only)

	Time (s)	Gas Used	Cost (Ether)	Cost ($)
slow	19.77	19,374	0.0000194	0.0049
medium	10.95	19,374	0.000116	0.0167
fast	10.51	34,374	0.000687	0.0990

To sum up the results for the Token approach, Table 7.9 shows the average of the total cost and time for an EV to authenticate, schedule and charge using the slowest option. Among the operations, authentication can be performed at any time but scheduling could be urgent depending on the remaining charge. We see that scheduling would only take 18secs which is almost real-time. Charging time, on

the other hand, is experienced when the EV is at the EV charging station. This time is 19.77 secs which means before starting to charge an EV driver will need to wait only 19.77 secs which is very reasonable. Looking at cost, we see that total cost is $0.44 which is again reasonable given the services the EV gets in terms of privacy-preservation, a scheduled time for charging, etc.

Table 7.9: Total scheduling overhead - Token Approach

Contract	**Time (s)**	**Gas Used**	**Cost (Ether)**	**Cost ($)**
Authentication	23.23	1,659,308	0.00166	0.42
Scheduling	18.03	59,061	0.0000591	0.015
Charging	19.77	19,374	0.0000194	0.0049
Total	**61.03**	**1,737,743**	**0.0017385**	**0.44**

For our Pederson Commitment approach, the EV only needs to authenticate and commit his scheduled charging time to the *auth_contract*. Table 7.10 shows the cost of authenticating and committing to a charge schedule. Besides proof generation and schedule commitment this is the only overhead for the EV when scheduling the charging using the Pederson approach. We can see that the total time for authentication and scheduling commitment is about 37secs and the total cost is only $0.13 which is very reasonable.

Table 7.10: Total scheduling overhead - Pederson Approach

	Time (s)	**Gas Used**	**Cost (Ether)**	**Cost ($)**
slow	37.60	875,287	0.000875	0.13
medium	35.83	875,287	0.00263	0.38
fast	20.26	875,287	0.0175	2.52

7.3.4 Putting it Altogether

To put it into a better perspective, in Figures 7.1 and 7.2 we compare both of our approaches in terms of all our benchmarks.

These results indicate that overall Pederson approach performs nearly twice as good as the token-based approach when considering the cheapest option. The total time overhead is 37.60sec while it is 61.03 for the token-based approach. Also, the total monetary cost is $0.13 in Pederson approach compared to $0.44 in token-based approach. Additionally, since the EV only needs to submit one *on-chain* transaction this reduces overall interaction between EV and the Blockchain.

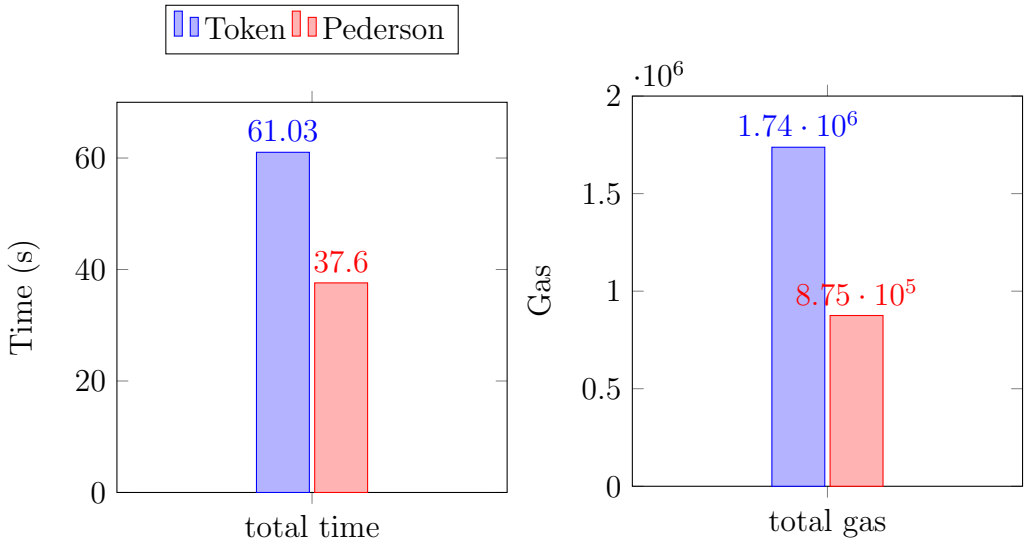

Figure 7.1: Total Time and Gas overhead on EV using Token and Pederson Approaches

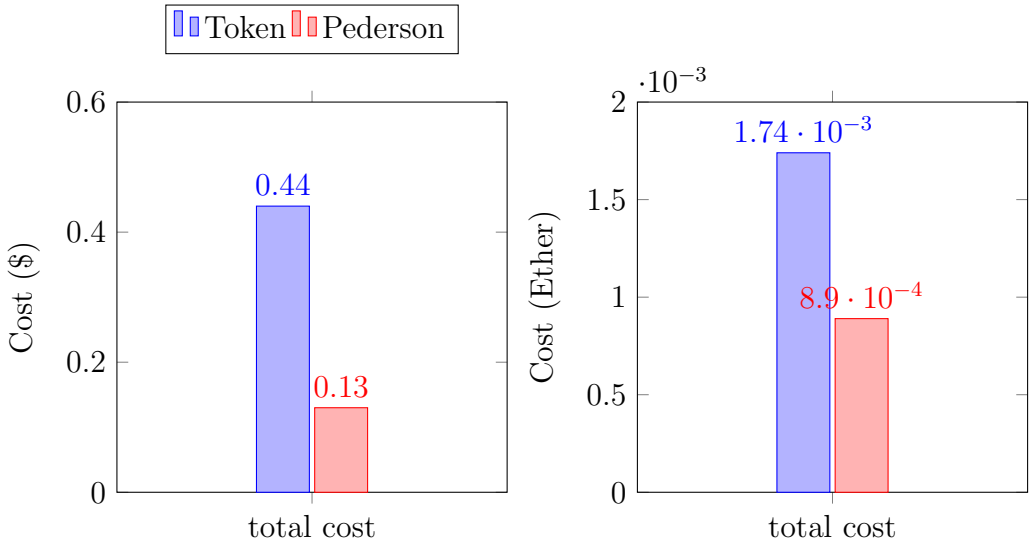

Figure 7.2: Total Cost in Ether and Dollar using Token and Pederson Approaches. (144\$/*Ether*)

CHAPTER 8

CONCLUSION

In this thesis, we introduced a framework to preserve the privacy of EVs during their charging process by combining the concept of zero-knowledge proofs and Blockchain smart contracts. This framework allows an EV to *authenticate*, *schedule*, *charge* and provide *payment* for services while preserving it's privacy. Other frameworks using a centralized model can provide anonymous authentication, scheduling, charging but privacy is broken when payment is required.

We first introduced a token-based method where we used different smart contracts and tokens to conduct authentication, scheduling, and charging without relying on any other third parties. This approach consisted of 3 separate *on-chain*transactions, including authentication, scheduling, and charging verification. The overall performance of this approach was reasonable with proof sizes at about 127 bytes and total time being about 61secs. Additionally the cost of using this approach was about $0.44 for all 3 *on-chain* transactions.

Then, we presented an improved version of our approach through the use of a Pederson Commitment rather than the token mechanism. In this approach we reduced the number of *on-chain* transactions from 3 to 1, by combining authentication and scheduling in one transaction. Since, Pederson commitments can be public and still not reveal details of the commitment, we could submit the commitment *on-chain*. From here the commitment was sent to an EVSP commitment storage for use in charging station verification. An advantage to this reasoning was that even if a charging station was only partially connected, it could still verify the EV offline. The overall performance of the Pederson approach was nearly twice as good as the token approach. The total time required was about 37secs and costs about $0.13 for authentication and scheduling, nearly 1/3 the cost of the token-based approach.

For efficiency, in both cases, we were able to use a single smart contract that will serve all the EVs. This means that contract deployment is only performed once and the EVSP needn't worry about deploying contracts for each EV, as this would be a costly overhead on the EVSP. This reasoning is twofold in that it provides efficiency and it is required for maintaining EV privacy. If a contract was deployed for every EV then the EV would be identified and linked with the corresponding scheduling.

Through security analysis, we showed that our approach will not expose any information to the involved parties such as EVSP, public and EV Charging station. We also implemented this framework using Zokrates. The results indicated that the overhead of the overall process in terms of time and Ethereum cost is affordable to be used in real-life applications. Nevertheless, Pederson approach outperforms token-based approach both in terms of time and cost overhead.

BIBLIOGRAPHY

[ABB+19] Wesam Al Amiri, Mohamed Baza, Karim Banawan, Mohamed Mahmoud, Waleed Alasmary, and Kemal Akkaya. Privacy-preserving smart parking system using blockchain and private information retrieval, 2019.

[BKA19] E. Bulut, M. Kisacikoglu, and K. Akkaya. Spatio-temporal non-intrusive direct v2v charge sharing coordination. *IEEE Transactions on Vehicular Technology*, pages 1–1, 2019.

[BLM+19] Mohamed Baza, Noureddine Lasla, Mohamed Mahmoud, Gautam Srivastava, and Mohamed Abdallah. B-ride: Ride sharing with privacy-preservation, trust and fair payment atop public blockchain, 2019.

[BSCG+15] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE, 2015.

[BSCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 781–796, 2014.

[Cha83] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[Che18] Jollen Chen. Devify: Decentralized internet of things software framework for a peer-to-peer and interoperable iot device. *SIGBED Rev.*, 15(2):31–36, June 2018.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *Annual International Cryptology Conference*, pages 78–96. Springer, 2006.

[CR17] Antorweep Chakravorty and Chunming Rong. Ushare: User controlled social media based on blockchain. In *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, IMCOM '17, pages 99:1–99:6, New York, NY, USA, 2017. ACM.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In *Annual International Cryptology Conference*, pages 410–424. Springer, 1997.

[CYHL12] Tat Wing Chim, Siu-Ming Yiu, Lucas CK Hui, and Victor OK Li. Privacy-preserving advance power reservation. *IEEE Communications Magazine*, 2012.

[DKM+18] Lalitha Devi, Devashish Kedar, Saurabh Kumar Malik, Kunal Dubey, and SRMIST Ramapuram Campus. Decentralized voting application. *International Journal of Engineering Science*, 19117, 2018.

[ET18] Jacob Eberhardt and Stefan Tai. Zokrates-scalable privacy-preserving off-chain computations. In *IEEE International Conference on Blockchain. IEEE*, 2018.

[GB18] H. Gunasinghe and E. Bertino. Privbiomtauth: Privacy preserving biometrics-based and user centric protocol for user authentication from mobile phones. *IEEE Transactions on Information Forensics and Security*, 13(4):1042–1057, April 2018.

[GM17] Jens Groth and Mary Maller. Snarky signatures: minimal signatures of knowledge from simulation-extractable snarks. Cryptology ePrint Archive, Report 2017/540, 2017. https://eprint.iacr.org/2017/540.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. Cryptology ePrint Archive, Report 2016/260, 2016. https://eprint.iacr.org/2016/260.

[HX16] Wenlin Han and Yang Xiao. Privacy preservation for v2g networks in smart grid: A survey. *Computer Communications*, 91:17–28, 2016.

[IH18] J. Isaak and M. J. Hanna. User data privacy: Facebook, cambridge analytica, and privacy protection. *Computer*, 51(8):56–59, August 2018.

[KUE18] Fabian Knirsch, Andreas Unterweger, and Dominik Engel. Privacy-preserving blockchain-based electric vehicle charging with dynamic tar-

iff decisions. *Computer Science - Research and Development*, 33(1):71–79, Feb 2018.

[LDN14] H. Li, G. Dan, and K. Nahrstedt. Portunes: Privacy-preserving fast authentication for dynamic electric vehicle charging. In *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 920–925, Nov 2014.

[May16] Hartwig Mayer. zk-snark explained: Basic principles. 2016.

[Mil15] Michael Miller. *The internet of things: How smart TVs, smart cars, smart homes, and smart cities are changing the world*. Pearson Education, 2015.

[MJGW18] Zhaofeng Ma, Ming Jiang, Hongmin Gao, and Zhen Wang. Blockchain for digital rights management. *Future Generation Computer Systems*, 89:746 – 764, 2018.

[Net13] Moritz Neto. Issue-your-own-erc20-token. https://github.com/bitfwdcommunity/Issue-your-own-ERC20-token, 2013.

[NMNB18] V. Naidu, K. Mudliar, A. Naik, and P. P. Bhavathankar. A fully observable supply chain management system using block chain and iot. In *2018 3rd International Conference for Convergence in Technology (I2CT)*, pages 1–4, April 2018.

[NT16] Assa Naveh and Eran Tromer. Photoproof: Cryptographic image authentication for any set of permissible transformations. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 255–271. IEEE, 2016.

[Ped91] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual International Cryptology Conference*, pages 129–140. Springer, 1991.

[PKS16] M. Pustiek, A. Kos, and U. Sedlar. Blockchain based autonomous selection of electric vehicle charging station. In *2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI)*, pages 217–222, Oct 2016.

[RABK17] Braden Roberts, Kemal Akkaya, Eyuphan Bulut, and Mithat Kisacikoglu. An authentication framework for electric vehicle-to-electric

vehicle charging applications. In *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 565–569. IEEE, 2017.

[RKX+18] Azizur Rahim, Xiangjie Kong, Feng Xia, Zhaolong Ning, Noor Ullah, Jinzhong Wang, and Sajal K Das. Vehicular social networks: A survey. *Pervasive and Mobile Computing*, 43:96–113, 2018.

[SCG+14] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy (SP)*, pages 459–474. IEEE, 2014.

[Swa15] Melanie Swan. *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.", 2015.

[W+14] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[WTS+18] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 926–943, May 2018.

[WZC+18] Howard Wu, Wenting Zheng, Alessandro Chiesa, Raluca Ada Popa, and Ion Stoica. Dizk: A distributed zero-knowledge proof system. 2018.

[YAB18] Fatih Yucel, Kemal Akkaya, and Eyuphan Bulut. Efficient and privacy preserving supplier matching for electric vehicle charging. *Ad Hoc Networks*, 2018.

[ZNP15] G. Zyskind, O. Nathan, and A. '. Pentland. Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE Security and Privacy Workshops*, pages 180–184, May 2015.

VITA

DAVID GABAY

November 11, 1993	Born, Plantation, Florida
2015	A.A., Computer Engineering Broward College Davie, Florida
2017	B.S., Computer Engineering Florida Atlantic University Boca Raton, Florida
2018	Cyber Systems Engineer Graduate Intern The Aerospace Corporation (FFRDC) Los Angeles, California
2019	U.S. Department of Homeland Security (DHS) Secretary's Honors Program (SHP) Cyber Student Intern Program (CSIP) The Department of Homeland Security Washington, D.C.

PUBLICATIONS AND PRESENTATIONS

Gabay, D., Akkaya, K., & Cebe, M.(Under Review, Second Quarter 2020). *Privacy-preserving Charging for Connected Electric Vehicles Using Blockchain and Zero Knowledge Proofs.* In 2020 IEEE Transactions on Vehicular Technology Journal, Special Issue on Blockchains in Emerging Vehicular Social Networks

Gabay, D., Akkaya, K., & Cebe, M.(2019, October). *A Privacy Framework for Charging Connected Electric Vehicles Using Blockchain and Zero Knowledge Proofs.* In 2019 IEEE 44th Conference on Local Computer Networks, IEEE LCN 2019.

Gabay, D., Cebe, M., & Akkaya, K. (2019, May). *On the overhead of using zero-knowledge proofs for electric vehicle authentication: poster.* In Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks (pp. 347-348). ACM.

Akkiraju, A., Gabay, D., Yesilyurt, H. B., Aksu, H., & Uluagac, S. (2017, October). *Cybergrenade: Automated exploitation of local network machines via single board computers*. In 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS) (pp. 580-584). IEEE.