Blockchain-based Data Sharing of Vehicle Passports

by

Mohammed Fattah Saqib
B.Sc., American International University-Bangladesh, 2018

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Blockchain-based Data Sharing of Vehicle Passports

by

Mohammed Fattah Saqib
B.Sc., American International University-Bangladesh, 2018

Supervisory Committee

Dr. Riham AlTawy, Supervisor
(Department of Electrical and Computer Engineering)

Dr. T. Aaron Gulliver, Department Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Cryptocurrency has been one of the sectors which arguably saw the biggest technological innovation in the past decade. The technology behind this new revolution is called blockchain and it has the potential to innovate other sectors too. Its core concepts of decentralization, immutability and anonymity are traits that can revolutionize any sector if implemented properly.

In this work, we investigate how blockchain technology can be beneficial for the used vehicle market without intervention of any third party. Currently, most people cannot get basic information of the used vehicle they are buying and even when they do they cannot verify most of them. Thus various companies proposed solutions centered around vehicle passport. Staying true to its name, a vehicle passport contains all the relevant information of the vehicle which will help both the buyer and seller. The buyer will have a detailed report on any vehicle he is potentially buying and will have more chance of safeguarding himself against any kind of fraud. On the other hand, the seller will be able to sell his vehicle for the correct valuation because he has proof of the vehicle condition.

The goal of this work is to provide a better solution where blockchain-based decentralized data sharing of vehicle passports prevents any centralized authority from possessing all the power. By using smart contract of blockchain we provided *fair exchange*. This removes the necessity of having an entity overseeing the transaction because no transaction will take place unless all parties are guaranteed service/payment. One of the main requirements is proving the authenticity of the vehicle passport before the transaction takes place. Accordingly, we design a zero-knowledge proof system that verifies the consistency of encrypted data against its publicly committed value. The verification is done onchain. After all the requirements for the transaction are met, fair exchange takes place where the buyer receives the vehicle passport encrypted with their public key and service providers receive their payments.

Contents

List of Tables

List of Figures

List of Acronyms

PoW	Proof of Work
PoS	Proof of Stake
ZKSM	Zero-Knowledge Set Membership
QAP	Quadratic Arithmetic Program
zk-SNARK	Zero-Knowledge Succinct Non-Interactive Argument of Knowledge
DLP	Discrete Log Problem
CSH	Cryptographically Strong Hash
KOE	Knowledge of Exponent
PPT	Probabilistic Polynomial Time
DDH	Discrete Diffie-Hellman
HH	Homomorphic Hiding
CRS	Common Reference String

Chapter 1

Introduction and Motivation

A vehicle passport is a document which contains all the relevant information of a specific vehicle. Apart from basic details, an ideal vehicle passport should hold information like manufacturing and purchase date, mileage, fuel consumption statistics, accident history, and in depth details about vehicle servicing throughout its use.

Currently there is a lot of ongoing research on vehicle passports and most is led by vehicle manufacturers [8]. Their proposed solutions either use a standard database or a blockchain system. In both cases there is a centralized system controlling the whole system or the transaction part without full transparency. So, the trust in the system is equivalent to the trust in the central authority.

To eradicate control over the system by a centralized authority, a system has to provide authenticity of the vehicle passport by the system itself while ensuring its confidentiality. To resolve a portion of this dilemma, zero-knowledge proofs can be used. By using zero-knowledge proofs the system authenticates the vehicle passport using its encrypted value, thus not revealing any sensitive information to the blockchain nodes/observers. After verifying the report against its publicly committed value, payment is transferred to the

data collectors known as service providers. The buyer gets access to verified confidential vehicle report when the transaction occurs. In this thesis, ElGamal encryption is used for confidentiality. Lastly, a hashing algorithm is added to the system and the hash value of the report is saved in the blockchain to ensure commitment. If the hash of the unencrypted report is not the same as the hash saved in the blockchain then the transaction won't take place. Poseidon hashing algorithm is used in this thesis to ensure the commitment.

1.1 Problem Statement and Motivation

Any system controlled by a centralized authority is as trustworthy as the authority itself. Under the control of a corrupt corporation, a vehicle passport system leads to their own vehicles having enhanced performance and underwhelming performance report of vehicles of their competitors. Moreover, the authority can hide specific sections altogether to their benefit which provides a false perception of the vehicle. These potential alterations diminish the advantages of vehicle passports. Thus, an alternative solution where the system is decentralized will allow users to gain the full benefit of vehicle passports and with the help of blockchain and zero-knowledge proof it is well within the reach. Both centralized and decentralized system also relies on the service providers to be honest. Unfortunately there is no way of verifying that they are acting honestly in both systems but as they are compensated financially in the blockchain-based system they have less incentive to do so.

1.2 Research Objectives

The research goal is to provide a blockchain based decentralized option for vehicle passport which with the help of zero-knowledge proofs provides fair exchange. The objectives of the research are as follows.

- Analyze relevant work on vehicle passports and highlight their vulnerabilities. Investigate how the ones that are using blockchain offer decentralization and fair exchange.

- Propose a zero-knowledge based solution where potential buyers are guaranteed fair exchange.

- Utilize the autonomous execution of blockchain-based smart contracts to carry the zero-knowledge verification and fee payment in a fair way.

- Use of the Poseidon hashing function and ElGamal encryption to implement commitments and enforce confidentiality, respectively.

1.3 Related Work

In this section we summarize the existing proposals/systems for vehicle passports.

1.3.1 VINChain

VINChain [1] is a hybrid blockchain network which is built on the Graphene platform to store relevant vehicle information. Hybrid blockchain allows VINChain to hold all the access and selectively allow permission to access portion of the data to collaborators. Each vehicle has a unique identification number which is referred as the VIN number. Write access to the blockchain is provided to other service/data providers by making them delegates in the consensus algorithm Delegated Proof of Stake (DPoS) who are then in charge of the proof of stake process. As for the car owners, they will not have any write access to it but they will be able to control which data will be shared and how frequent the data will be collected. As depicted in Figure 1.1, the data providing entities will add hash of their records to the blockchain and the original data will be stored in the service provider's database. When a buyer pays for the paid version of the car passport via the smart contract, the full report will

be generated by accessing data from all the service provider's databases. The reliability will be ensured by the hashes of the data stored in the blockchain as everyone has read access to it. However, VINChain does not provide fair exchange where both parties receive their desired assets simultaneously. In VINChain, a potential buyer always pays the fees first. Afterward they receive the passport and can check the validity against its committed hash. Moreover, the blockchain utilized is permissioned which leads to a system controlled only by the blockchain owners.

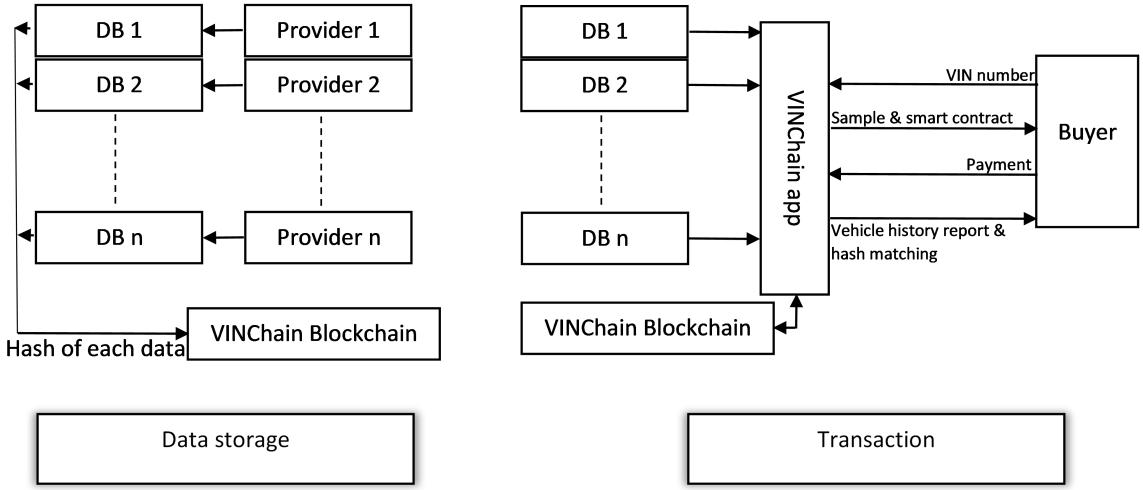

Figure 1.1: VINChain data storage and transaction scheme [1].

1.3.2 Blockchain Xdev

All service providers and stakeholders are considered as reliable sources in Blockchain Xdev [2]. Together the service providers and stakeholders log all relavant information within a consortium Blockchain. The consortium blockchain also allows the stakeholders to collaborate with different sets of data to allow them to build new services on top of it such as insurance according to car use or predicting maintenance schedules according to car needs. The collaboration is incentivised by the trading of stable coins between the consortium members. The ledger is maintained with manual data input from various consortium

members. To ensure the reliability of the source, Blockchain Xdev attaches a quality index with the consortium member. Sensitive data are kept offchain for confidentiality. Blockchain Xdev uses a protocol which matches the onchain data to its offchain counterpart before providing a report. Similar to VINChain, Xdev does not provide fair exchange and the trust of the system mainly relies on the stakeholders.

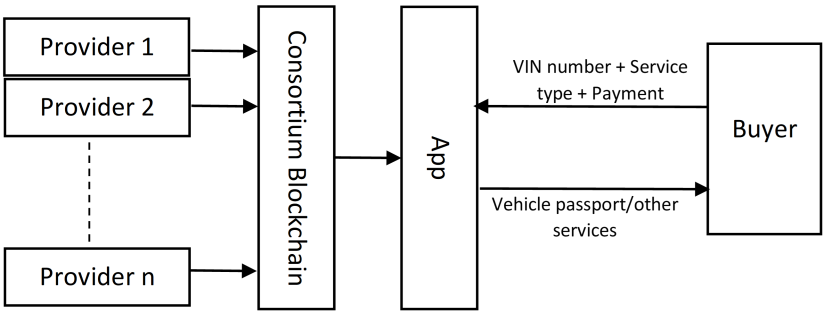

Figure 1.2: Blockchain Xdev data storage and transaction scheme [2].

1.3.3 Carfax

Carfax [9] has been widely used since late 1980s for buying and selling cars and can be considered a leader in the vehicle passport sector judging solely by longevity. Unlike the two prior systems, Carfax does not use blockchain technology. It collects vehicle data from insurance companies, auto repair facilities, rental companies, fire departments, law enforcement agencies and other sources. Carfax stores all the data in their personal database and use it to generate report. There is no incentive for the data providers to share the data, thus a lot of information is not included in Carfax's database. Also, the report generation and storage is done solely by Carfax. Hence, trusting that Carfax does not manipulate the data and is capable of preventing any malicious attacks on its database is mandatory.

1.3.4 AutoCheck

AutoCheck [10] is the closest competitor of Carfax. Similar to Carfax, AutoCheck does not utilize blockchain technology and thus it has the same challenges as Carfax. They collect data from similar sources. So the reports are similar but variations can occur due to different sources. Plus, the sources don't have an incentive to provide up to date data so the databases vary. Autocheck's database is also centralized like other big corporations and the buyer has to assume that the data haven't been manipulated by a malicious party. There is also no way of verifying the report, so the buyer has to trust that Autocheck has generated an honest report according to its database.

1.4 Organization of the Thesis

The remainder of this thesis is organized in the following manner. Chapter 2 provides the literature background on blockchain technology and breakdowns all the technologies that it uses. Chapter 3 shows how Zero-knowledge proofs can be utilised to verify sensitive data without the need of sharing them beforehand. Chapter 4 starts with the preliminaries and background of the prototype. Afterwards it goes into more detail about the proposed prototype. Section 4.8 explains how the system is going to be secure. Chapter 5 evaluates the performance of the proposed system. The last chapter, Chapter 6 has the final discussion regarding our work.

Chapter 2

Blockchain Background

In this chapter we provide a summary of the basic concepts of blockchain technology. We first give a general overview and then introduce the cryptographic primitives that are employed to construct it. Finally, we briefly discuss consensus protocols in Peer to Peer (P2P) networks.

Blockchain is mainly known for being the record-keeping technology behind the cryptocurrency networks. Decentralization is the core concept of blockchain and one of the main traits which make it unique. In a traditional database a centralized authority has all the access but here nodes communicate and collaborate with each other to create a consensus between them to manage and validate the data. This network of nodes is called a P2P network. Time, transaction amount and addresses of both the sender and the receiver is broadcasted whenever a transaction is made. This broadcasted information also allows the blockchain to learn the balances of addresses which reduces additional computation for balance query. After the transaction is verified all the data and hashes are saved in the blockchain, a copy of it is stored in every full node across the network. Every copy is updated whenever an exchange is made.

The origin of Blockchain technology is from a 1991 paper [11] where the earliest version of the technology was used for time-stamping digital documents. They introduced the idea of using chain of hashes as commitment for growing sets of documents. Here, a Time-Stamp Service creates a Merkle tree from all the certification requests and sends the requesters the Merkle Proof of their documents being committed to it. Merkle tree is a hash tree where every leaf is a hash of a data block. Merkle proof authenticates any transaction represented by a leaf within the Merkle tree. Users can validate each document's time stamp by using Merkle Proof on relevant interval hash. This made the digital documents tamper resistant which is still one of the biggest traits of blockchain. This technique was later adopted by Satoshi Nakamoto in 2009 [12] to create the very first digital cryptocurrency, bitcoin.

The immutability of the blockchain network can be understood by going through the content of each block as depicted in Figure 2.1. Each block contains some data, a unique hash of the block and the hash of the previous block. Even a slight change in the data will result in a drastic change in the hash. Thus, it is very easy to detect any change in the blocks just by looking at the hash. As each block stores both its own hash and the one from the previous block, tampering of any block will also make all the blocks after it invalid. But additional security is still needed because modern computers are fast in computing hashes. So they can easily calculate new hashes for all the invalid blocks which will make any tampering undetectable very quickly. To mitigate this issue, blockchains have various consensus algorithms. One of the most widely used consensus algorithms is Proof of Work (PoW). In PoW, miners compete against each other to solve complex computational puzzles which use a nonce and are time consuming and difficult to solve but very easy to verify. After a miner finds a solution to the puzzle, he broadcasts the block to the whole network and every other miner will verify it. After verification, the new block is added to the blockchain

and the miner who solved the puzzle will get the block reward.

Due to being such a secure system, blockchain is being integrated into various other areas to provide different benefits. The introduction of smart contracts also made integration into other areas very easy. Smart contracts are programs on a blockchain that run when their predetermined conditions are met. Because it allowed programmers to write additional programs on top of blockchain which makes the technology extremely flexible and creates unlimited possibilities.

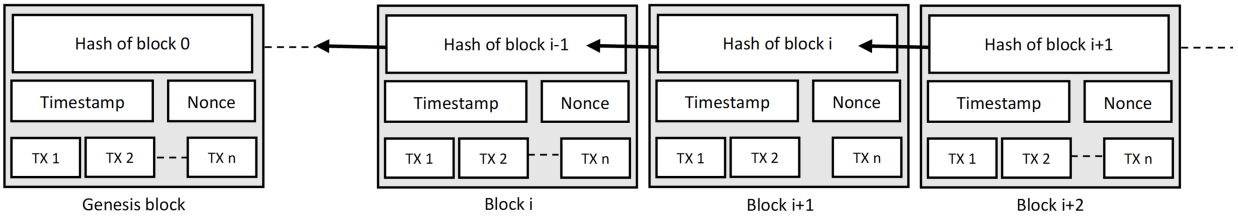

Figure 2.1: An example of blockchain which consists of a continuous sequence of blocks [3].

2.1 Types of Blockchain Networks

Depending on the type of data and the purpose of the network, blockchain networks fall under four types.

2.1.1 Public Blockchain

These kind of blockchain networks were the first to be introduced to the world as the technological backbone for cryptocurrency. As its name suggest this blockchain network is open for all and allows anyone with an internet connection to participate in the consensus process which determines which blocks get added to the blockchain. Being a decentralized network, everyone will be able to see all the transactions in the network which creates transparency and trust but this also makes it unsuitable for sharing sensitive data because everyone has access to it.

2.1.2 Private Blockchain

In this network, the write permission is always kept to the organization and read permissions are given according to one's level of access of the data. By being a centralized network it does not have the same trustless environment like decentralized public blockchains. This type of blockchains is only applicable to a closed network and usually adopted by corporations. It's a good alternative to distributed storage as it insures nobody can modify any previous data.

2.1.3 Consortium Blockchain

Consortium blockchains also known as Federated blockchains can be considered as a good balance between public and private blockchains. It is similar to private blockchain as a portion of the data is kept private and the public does not have write access to the blockchain. But the general public has more trust in the system because it is managed by more than one company which diminishes centralization to an extent. Plus, in most cases a portion of the data will be shared as public data.

2.1.4 Hybrid Blockchain

Hybrid blockchains also known as semi-private blockchains are very similar to the consortium blockchains as it's also a mixture of public and private blockchains. The core difference between them is unlike consortium blockchains, it is not maintained by multiple companies with the same level of authority. Instead all the read and write access to the blockchain is maintained by a single corporation and they can give any access to a portion of the data or blockchain network to anyone.

2.2 Cryptographic Hash Functions

Hash functions are the fundamental cryptographic elements not only in blockchain but also in cryptography in general. Hashing is a process where the input of arbitrary length can be converted to a string of text of a fixed size using a hash function. Depending on the algorithm that is used, the size of the hash value will vary. The output with the string of text is called hash value and a slight change in the input will generate a completely different hash value. Thus, it is widely used to authenticate any input no matter the size of the file or text because any modification to the input will result in a huge alteration to the hash value. As depicted in Figure 2.2, hash function mapping is a one-way function as the original input was compressed to a fixed length. In Bitcoin, SHA2 [13] hashing algorithm is used and in Ethereum it is SHA3 [14].

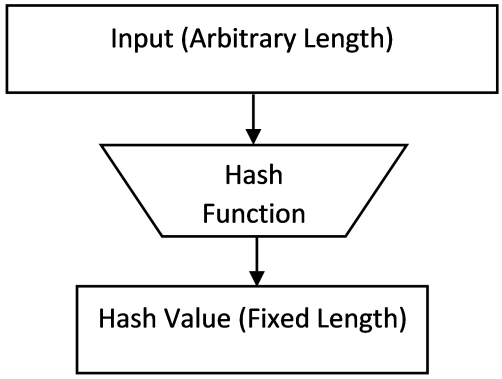

Figure 2.2: Hash generation [4]

A proper hash function $H(.)$ that outputs n-bits needs to provide the following three security properties.

Preimage Resistance. Commonly known as one wayness and it implies that it is computationally infeasible when given y, to find an x, such that $y = H(x)$. The generic attack to break one-wayness of an n-bit hash function has a success probability of 2^{-n} [15].

Second Preimage Resistance. Also known as weak collision resistance and it implies that if given x_1, it is computationally infeasible to find a second input x_2 such that $H(x_1) = H(x_2)$. The generic attack to break second pre-image resistance of an n-bit hash function has a success probability of 2^{-n} [15].

Collision Resistance. This property implies that it is computationally infeasible to find any two inputs x_1 and x_2, such that $H(x_1) = H(x_2)$. The generic attack to find collisions of an n-bit hash function has a success probability of $2^{-\frac{n}{2}}$ [15].

2.3 Digital Signature

Authentication of the nodes is one of the main hurdles any blockchain must overcome. Without authentication the receiver of any message cannot be sure if the message came from the correct node or not. A digital signature algorithm usually relies on hard problems in number theory [16,17]. It is used in blockchains to authenticate the creator of transactions. This cryptosystem consists of three procedures KeyGen, Sign, and Verify. Elliptic Curve Digital Signature Algorithm (ECDSA) is the digital signature algorithm that is used in Bitcoin and Ethereum, and it relies on the hardness of finding discrete logarithms over elliptic curves [17].

KeyGen. The key generation algorithm creates a pair of keys; secret key (SK) and public key (PK) for each node, formally, $(SK, PK) \leftarrow keygen(1^n)$ where n is the security parameter. It should be computationally infeasible to find SK given PK.

Sign. This procedure is executed by the signer (transaction sender). The sign algorithm takes a message to be signed m, and SK and some randomness r and generates a signature $\sigma = sign(m, SK, r)$.

Verify. This procedure is run by a public verifier (transaction verifier). The algorithm takes

a message m, public key PK and a signature σ. It returns true if the signature σ is verified on m using PK. Otherwise, it returns false, $\{T/F\} = verify(m, PK, \sigma)$

2.4 Merkle Trees

Blockchain networks must provide commitment for their nodes so that any decision they make becomes irreversible or binding. At the simplest level this can be done by the hashing of the message m. For a network we need to do such commitment for every message (transaction) in an efficient manner. To increase efficiency Merkle tree is proposed instead of hashing all the messages at once [5]. If we hashed all the messages at once we would need all the messages to verify the membership of any single message in a given commitment. On the contrary, to verify a message committed to a Merkle tree we only need to know the hashes or message in the path connecting the message we want to verify to the Merkle root (commitment). Such a sequence of hashes is known as authentication path [5]. For instance, if one needs to verify if Transaction B is included in the commitment given by the Merkle root in Figure. 2.3, they need only Hash A, and Hash CD. Formally, for a tree with n transactions, the authentication path needed to verify the membership of any transaction contains $\log n$ intermediate hashes.

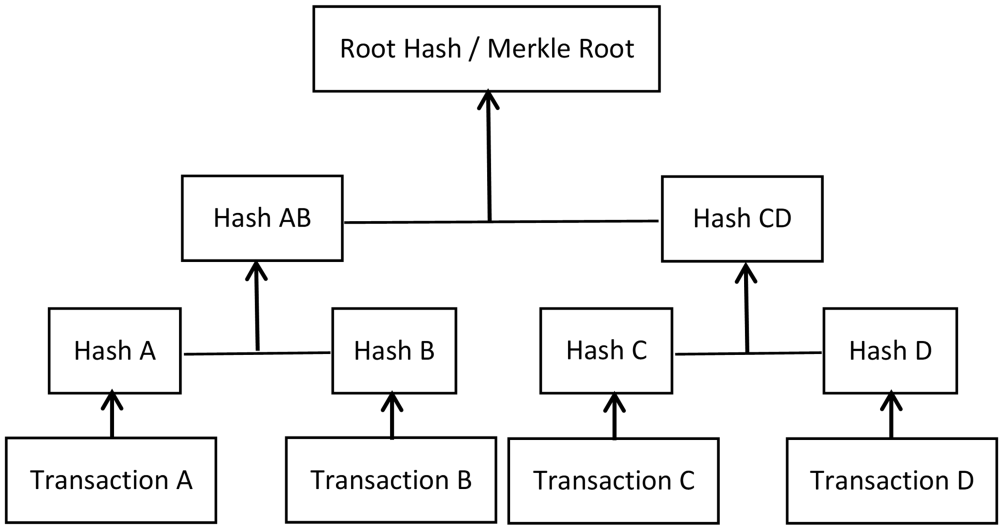

Figure 2.3: Merkle tree typical layout [5].

Linked timestamping Linked timestamping accumulates Merkle trees over time [18]. As depicted in Figure. 2.4, in blockchains, the header in every block consists of the Merkle root hash of all the transactions in that block and the hash of the header of the previous block. Accordingly, one is able to verify the precedence of a transaction over another. Additionally, any attempted alteration of any previous transaction can be spotted because the computed block header of the subsequent blocks will not be the same.

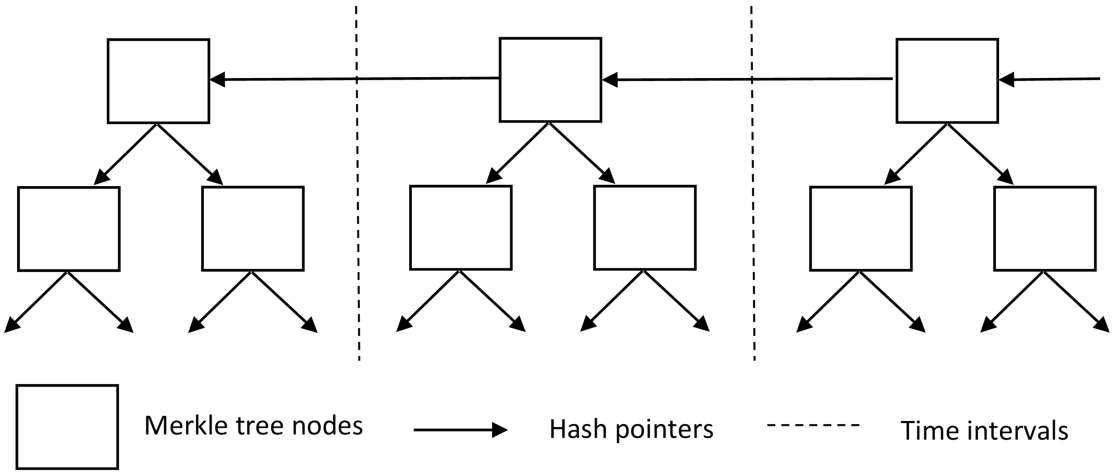

Figure 2.4: The ledger data structure in linked timestamping [6].

2.5 Consensus

Blockchain being a decentralized network any decision within the P2P network needs to be agreed upon by most of the nodes in the network because there is no central authority to make those decisions. To get consensus of the network, a PoW mechanism is employed. PoW enables a P2P network to reach consensus even with the existence of malicious nodes that create fake nodes in an attempt to gain majority control [19].

Proof of work. PoW is a moderately hard algorithm as it has a computational delay and it's energy intensive. In this consensus algorithm, full nodes or miners solve a mathematical problem for each block where they need to guess the correct nonce as the proof. For example, miners need to guess a nonce which will be combined with the data of the block and hashed. The hash would then have to match any predetermined condition such as the hash must start with a specific number of zeroes. The computed hash is called the block hash. Once someone finds the solution, they will broadcast it to the whole network and everyone else will verify it. The verification is very swift because the nodes only need to solve one problem instead of brute force which was used to find the nonce. After verification, miners add the new block to their own blockchain and then the miner who first solved the PoW is rewarded with a specific amount of newly minted cryptocurrency.

Chapter 3

Zero-knowledge Proofs

Zero-knowledge proofs were proposed by Goldwasser, Micali, and Rackoff in 1989 [20]. In a scenario with two parties, one is termed the prover and the other one is termed the verifier. With the help of a zero-knowledge proof protocol, the prover can prove to a verifier that they know a certain secret such that a given statement is true without revealing the secret. A statement in a zero-knowledge proof is a secret solution to a computationally hard problem and the secret cannot be efficiently extracted from the proof. For example, one may prove knowledge of a logarithm x of some point y to the base g, i.e., knowledge of x such that $y = g^x$. Every zero-knowledge proof must satisfy the following three properties.

Completeness. An honest prover who knows the secret can always convince the verifier that the statement is true.

Soundness. With negligible probability, a cheating prover who does not know the secret is unable to convince the verifier of the truthfulness of the statement.

Zero-knowledge The verifier only gets the knowledge that the statement is true or false and no additional details regarding the secret are revealed.

Zero-knowledge proofs are divided into two categories depending on the interaction between the prover and verifier.

Interactive. In this type of zero-knowledge proof, the prover has to answer multiple challenges given by the verifier. Correctly answering the challenges ensures the verifier that the prover is in possession of the correct secret. Abstract examples of zero-knowledge proof such as "The Ali Baba cave" and "Two balls and the colour-blind friend" are interactive because of the interchanges between the prover and verifier [21]. Even though an interactive zero-knowledge proof creates a valid proof that the secret is known by the prover, the process is not always ideal due to its real-time interaction requirement.

Non-interactive. Non-interactive zero-knowledge proofs, sometimes known as one-shot proofs enable the prover to generate random challenges and the proof itself while maintaining soundness [22]. These proofs are adopted in blockchain applications because the verifier is all the miners of the network so any form of interaction is not possible. One of the most prominent types of non-interactive zero-knowledge proofs is Zero-Knowledge Succinct Non-interactive Arguments of Knowledge (zkSNARK) [23].

3.1 ZkSNARK Generic Proof System

Zk-SNARK stands for Zero-Knowledge Succinct Non-Interactive Argument of Knowledge. It is one of the most widely used forms of zero-knowledge cryptography as it seamlessly allows any application to verify the knowledge of secret inputs to generic computations. It translates statements involving the knowledge of inputs to generic computations to statements that prove the knowledge of constrained polynomials. More precisely, the prover is required to prove that they know the coefficients of a degree d polynomial P over a field $\mathbb{F}_p$ that has specific public roots. Since, two different degree d polynomials coincide in at

most d points, then with very low probability $\frac{d}{|\mathbb{F}_p|}$, a cheating prover can find another set of coefficients for a degree d polynomial that has the same roots [24]. In what follows, we provide the description of the main building blocks used in zkSNARK.

Homomorphic Hiding (HH). For any encryption $E(x)$ of an input x, the properties of homomorphic hiding are as follows.

- It should be hard to find input x from the resulting $E(x)$.
- Different inputs will always result in different outputs. So, for inputs where $x \neq y$ the outputs should similarly be $E(x) \neq E(y)$.
- Given the hidings $E(x)$ and $E(y)$, one can compute the hiding $E(x + y)$.

Assume Bonnie sends $E(x)$ and $E(y)$ to Clyde along with a such that $x + y = a$. Now Clyde can easily compute $E(x+y)$ due to the homomorphic hiding property and crosscheck the result with $E(a)$. Thus verifying that Bonnie knows x and y. In practice the process is identical but more complexity is added via introducing modulo operation over a cyclic group. In a cyclic group, if g is a generator of group $\mathbb{G}_p$ then all elements of $\mathbb{G}_p$ can be described as ga where $a = \{0, \ldots, p-2\}$. Thus a more accurate detailed example of the homomorphic hiding would look like the following where g indicates a generator of a group where the discrete log problem is intractable.

$$E(x + y) = g(x + y) \bmod p = (gx + gy) \bmod p = E(x)E(y)$$

Blind Evaluation of Polynomials. Blind evaluation of polynomials is done with the help of homomorphic hiding. For instance, let P be a polynomial of degree d over the field $\mathbb{F}_p$

$$P(X) = a_0 + a_1X + a_2X^2 + \ldots + a_dX^d$$

If we need to evaluate P at any point $s \in \mathbb{F}_p$ then $P(s) = a_0 + a_1 s + a_2 s^2 + \ldots + a_d s^d$. From the equation we can see that for someone with knowledge of P, calculating $P(s)$ is a simple linear combination. Linear combination is also supported by HH, e.g., $E(ax+by) = g(ax+by) = (gx)a + (gy)b = aE(x) + bE(y)$. Here g is the generator of the group. Let us assume Clyde has a polynomial P of degree d and Bonnie chooses a random point $s \in \mathbb{F}_p$. Bonnie needs to verify some statements about $P(s)$ without knowing P. On the other side, Clyde cannot learn s. Firstly, Bonnie sends $E(1), E(s), \ldots, E(s^d)$ to Clyde. Now Clyde can blindly evaluate $P(s)$ by computing $E(P(s)) = a_0 E(1) + a_1 E(s) + \cdots + a_d E(s^d)$ and send it to Bonnie. By making the prover blindly evaluate their polynomial at any random point we can be assured that the prover has the correct polynomial.

The Knowledge of Coefficient Test and Assumption. Clyde is able to compute $E(P(s))$ without knowledge of s in the blind evaluation of polynomials. But there is a small loophole in the protocol as there is no certainty if Clyde sends the correct $E(P(s))$ and not some random number. To ensure that the proper coefficients in P are utilized to compute $E(P(s)$ the following protocol is adopted.

- Bonnie chooses random α and a where $\alpha \in \mathbb{F}_p$ and $a \in \mathbb{G}_p$. These values are used to compute $b = \alpha \cdot a$ where $a, b \neq 0$ and $b \in \mathbb{F}_p$.

- Bonnie sends the α pair (a, b) to Clyde as a challenge.

- Now Clyde needs to respond with a new α pair (a', b') where $b' = \alpha \cdot a'$.

- Bonnie will accept Clyde's response when she sees that the response (a', b') is indeed an α pair.

Since the discrete log problem is hard in $\mathbb{G}_p$, given (a, b), it is not feasible for Clyde to find α. In this case the solution is that Clyde chooses a value γ where $\gamma \in \mathbb{F}_p$ and

multiplies it by the (a, b) pair provided by Bonnie. So, the responding pair from Clyde is $(a', b') = (\gamma \cdot a, \gamma \cdot b)$ and it is an α pair. If the response (a', b') from Clyde is valid then Bonnie can be assured that Clyde knows the coefficient γ. This is called The Knowledge of Coefficient Assumption (KCA) [25].

Verifiable Blind Evaluation of Polynomials. In the previous description of KCA we saw Bonnie sending one α pair to Clyde. For the extended version of KCA, Bonnie will send several α pairs $(a_1, b_1), \ldots, (a_d, b_d)$ which will be made up of the same α. Similarly to last time, Clyde now has to respond with another α pair. Moreover, this time his α pair will also contain all the the α pairs sent by Bonnie. For instance, if Bonnie sent two α pairs then Clyde will multiply those elements with $c_1, c_2 \in \mathbb{F}_p$. The resultant α pair is given by $(a', b') = (c_1 \cdot a_1 + c_2 \cdot a_2, c_1 \cdot b_1 + c_2 \cdot b_2)$. We can confirm that it is an α pair from the following equation where a' is a non-zero element. $b' = c_1 \cdot b_1 + c_2 \cdot b_2 = c_1\alpha \cdot a_1 + c_2\alpha \cdot a_2 = \alpha(c_1 \cdot a_1 + c_2 \cdot a_2) = \alpha \cdot a'$. The d-power Knowledge of Coefficient Assumption (d-KCA) generalizes KCA for α pairs of higher degrees. Let Bonnie choose the random elements $\alpha \in \mathbb{F}_p$ and $s \in \mathbb{F}_p$ and sends the newly modified α pairs $(g, \alpha g), (sg, \alpha sg), \ldots, (s^d g, \alpha s^d g)$, i.e., α pairs of the hidings of $\{1, s, s^2, \ldots, s^d\}$. In response, using the polynomial coefficients $c_0, c_1, , c_d$, Clyde computes $a' = P(s)g$ and $b' = \alpha P(s)g$ and responds to Bonnie. Clyde is able to do this computation because both $P(s)$ and $\alpha P(s)g$ are linear combinations of the hidings sent by Bonnie. Bonnie accepts Clyde's response only if $b' = \alpha a'$ because she can be certain that Clyde knows $c_0, \ldots, c_d \in \mathbb{F}_p$ due to d-KCA.

3.1.1 From Computations to Polynomials

zk-SNARK transform any statement represented in an arithmetic circuit form to a polynomial. For example, assume Clyde has to prove that he knows $c_1, c_2, c_3 \in \mathbb{F}_p$ where $(c_1 \cdot c_2) \cdot (c_1 + c_3) = 9$. Before creating the polynomial this statement has to be represented

in an arithmetic circuit similar to that given in Figure. 3.1.

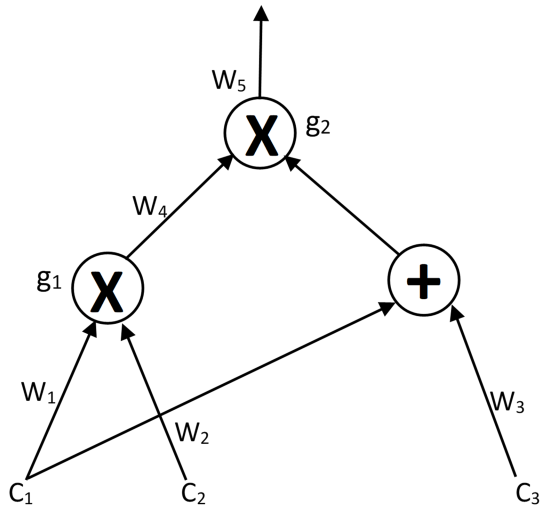

Figure 3.1: zk-SNARK arithmetic circuit example [7].

The rules for creating the circuit from the computation are as follows.

- All wires outgoing from a singular source, c_1 will be labeled as if they are a single wire, similar to w_1.

- The two input wires of the multiplication gates are known as the right and left wire.

- There are no additional labels for the output wire of addition gate. For instance, in the example both w_1 and w_3 are considered as the right input for g_2 gate.

Thus, our circuit created the legal assignment $(c_1, \ldots, c_5)$ where $c_4 = c_1 \cdot c_2$ and $c_5 = c_4 \cdot (c_1 + c_3)$. The next step from this circuit is called the reduction to a Quadratic Arithmetic Program (QAP) [26]. Multiplication gates g_1 and g_2 are given the values 1 and 2 $\in \mathbb{F}_p$, respectively, which are known as the target points. Afterwards, left wire polynomials will be associated with $L_1, \ldots, L_5$, right wire polynomials with $R_1, \ldots, R_5$ and output wire polynomials with $O_1, \ldots, O_5$. Polynomials will generally be zero on target points apart from the ones involved with the target points associated with multiplication

gates. For the output wire g_1; W_1 is left, W_2 is right and W_4 is the output. So, we can say $L_1 = R_2 = O_4 = 2-X$ because $2-X$ results in 1 on the target point 1 and zero on the target point 2. Similarly, for the output wire g_2; W_4 is left, W_1 and W_3 are right and W_5 is the output. So, we say $L_4 = R_1 = R_3 = O_5 = X-1$ because $X-1$ results to one on the target point 2 and zero on the target point 1. With fixed values $(c_1, \ldots, c_5)$, we can define all the left, right, and output sum polynomials along with the relation between them for the polynomial in the following manner.

$$L := \sum_{i=1}^{5} c_i \cdot L_i, R := \sum_{i=1}^{5} c_i \cdot R_i, O := \sum_{i=1}^{5} c_i \cdot O_i$$

$$P := L \cdot R - O$$

In this QAP, $(c_1, \ldots, c_5)$ is considered a legal assignment only if P turns to zero at every target point. For example, in target point 1 the final form of the calculation will be $P(1) = c_1 \cdot c_2 - c_4$ and in target point 2 it will similarly be $P(2) = c_4 \cdot (c_1 + c_3) - c_5$. In both cases P vanishes on the target points which makes $(c_1, \ldots, c_5)$ a legal statement. To prove that the polynomial in question goes to 0 at point $a \in \mathbb{F}_p$ we can use the algebraic fact that if $P(a) = 0$, then $P(a)$ is divisible by $(X - a)$. So, for any polynomial H, $P = (X - a) \cdot H$. Accordingly, a target polynomial $T(X) := (X - 1) \cdot (X - 2)$ can also divide P when $(c_1, \ldots, c_5)$ is a legal assignment. Formally, a QAP with degree d and size m and target polynomial T of degree d results in the following relation for the assignment $(c_1, \ldots, c_m)$.

$$L := \sum_{i=1}^{m} c_i \cdot L_i, R := \sum_{i=1}^{m} c_i \cdot R_i, O := \sum_{i=1}^{m} c_i \cdot O_i$$

$$P := L \cdot R - O = T \cdot H$$

3.2 Pinocchio Protocol

Using QAP, it is possible to convert any statement to a system of polynomials which can be later used to prove the statement. The polynomials should be made up using assignments $(c_1, \ldots, c_m)$. Parno et al. introduced the Pinocchio protocol that utilized QAPs to build zk-SNARK [27]. Particularly, it showed a formally specified protocol that given any generic computation with secret inputs, one may translate it to the knowledge of $P(s) = H(s) \cdot T(s)$. This can be verified at some random point $s \in \mathbb{F}_p$. Given a public computation, with a known target polynomial T, the protocol between Bonnie and Clyde where Clyde is the prover is described as follows.

- Clyde chooses the polynomials L, R, O, H all of whose maximum degree is d.
- Bonnie chooses a random point $s \in \mathbb{F}_p$, uses it to compute $E(T(s))$ and sends $E(1), E(s), \ldots, E(s^d)$ to Clyde.
- Clyde evaluates the HHs of all his polynomials at point s and sends them to Bonnie. The polynomials computed at s are $E(L(s)), E(R(s)), E(O(s)), E(H(s))$.
- Bonnie checks if $E(L(s) \cdot R(s) - O(s)) = E(T(s) \cdot H(s))$ to ensure that the equation holds at the chosen point s.

In this interaction, Clyde chooses the polynomials without knowing s. This problem is solved by implementing the verifiable blind evaluation of polynomials protocol which ensures the polynomials are of the right degree. To ensure that the polynomials are from an assignment we need to combine polynomials L, R, O into one polynomial F where $i \in \{1, \ldots, m\}$ as

$$F_i = L_i + X^{d+1} \cdot R_i + X^{2(d+1)} \cdot O_i$$

R_i is multiplied by X^{d+1} and O_i is multiplied by $X^{2(d+1)}$ so the coefficients of L, R, O stay separate from each other. From this equation it can be ensured that L, R, O were produced from an assignment if F is a linear combination of F_i's. The verifiable blind evaluation of polynomials protocol is used to perform this task. First, Bonny chooses a random value $\beta \in \mathbb{F}_p^*$ and sends $E(\beta \cdot F_1(s)), \ldots, E(\beta \cdot F_m(s))$ to Clyde. Here, $\mathbb{F}_p^*$ represents multiplicative group of finite field. Clyde then sends back $E(\beta \cdot F(s))$. If its valid then it means F is a linear combination of F_i due to the extended version of the KCA [28].

To avoid leakage of information while conducting the verifiable blind evaluation of polynomials protocol, the assignment needs to be masked by adding a random shift T for every polynomial as follows:

$$L_z := L + \delta_1 \cdot T, R_z := R + \delta_2 \cdot T, O_z := O + \delta_3 \cdot T$$

Here, $\delta_1, \delta_2, \delta_3 \in \mathbb{F}_p^*$ are chosen at random. As, all the additions are multiples of T we can say that $L_z \cdot R_z - O_z$ is also divisable by T or $L_z \cdot R_z - O_z = T \cdot H_z$.

Pairings of Elliptic Curves. Elliptic curves are used to introduce multiplication in HH. The new HH enables the transition from an interactive protocol to a non-interactive protocol as the proof system.

For a prime number $p > 3$, let us take two variables $u, v \in \mathbb{F}_p$ where $4u^3 + 27v^2 \neq 0$. Then, the equation $Y^2 = X^3 + u \cdot X + v$ forms an elliptic curve C. Elliptic curve C can be represented with a set of points $(x, y) \in \mathbb{F}_p^2$ which satisfy this equation. Groups can be created from the curve using the $(x, y) \in \mathbb{F}_p^2$ points and a special point O. O is known as the "point at infinity" and works as the zero of the group.

Let's say a vertical line $X = c$ intersected elliptic curve C in two points $P = (x_1, y_1)$ and

$Q := (x_1, -y_1)$. These two points are the only points intersected because the line is vertical and the Y of the curve is of degree two. So, $P + Q = O$ or $P = -Q$. Thus, Q is the inverse of P.

If P and Q have different first coordinates. X of the elliptic curve is of degree three. So, a line intersecting points P and Q will intersect the curve on a third point R. So, $P + Q + R = O$ or $P + Q = -R$. $-R$ can be found by flipping the second coordinate of R.

Let, the group of the elliptic curve be denoted by $C(\mathbb{F}_p)$ or $\mathcal{G}_1$ which consists of r number of elements. Here, r is a prime number which is different from p. Moreover, we need integer k which is known as the embedding degree of the elliptic curve where $p^k - 1$ is divisible by r.

The multiplicative group of $\mathbb{F}_{p^k}$ can be denoted as $C(\mathbb{F}_{p^k})$ or $\mathcal{G}_T$ which is of order r. It contains $r - 1$ subgroups of order r including $\mathcal{G}_1$ and $\mathcal{G}_2$.

Tate reduced pairing is capable of efficiently mapping pair of elements from two groups into an element of a new group [29]. By using this we can take elements from $\mathcal{G}_1$ and $\mathcal{G}_2$ and reduce them to only one element in $\mathcal{G}_T$. Let's say generators for $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_T$ are sequentially g, h and $\mathbf{g}$. By applying Tate reduced pairing we get the following.

$$E_1(x) := x \cdot g, E_2(x) := x \cdot h, E(x) := x \cdot \mathbf{g}$$

This is a weaker version of HH but it supports both addition and multiplication.

To make the total process non-interactive we need three steps. First, we need to publish Bonnie's first message as a common reference string (CRS). This is part of the initial setup process and will be later used to create and verify proofs.

Setup: Publish the CRS as.

$$(E_1(1), E_1(s), \ldots, E_1(s^d), E_2(\alpha), E_2(\alpha s), \ldots, E_2(\alpha s^d))$$

Proof: Using the elements in CRS, Clyde can compute $a = E_1(P(s))$ and $b = E_2(\alpha P(S))$ as they support linear combinations.

Verification: Set $x, y \in \mathbb{F}_r$ so that $a = E_1(x)$ and $b = E_2(y)$. Bonnie computes both $E(\alpha x) = \text{Tate}(E_1(x), E_2(\alpha))$ and $E(y) = \text{Tate}(E_1(1), E_2(y))$. Now she has to check if they are equal. If they are then $\alpha x = y$.

Chapter 4

Blockchain-based Vehicle Passport System

The vehicle industry is starting to embrace blockchain technology for improving various aspects of the industry [1, 2]. Such projects have been able to succeed in some aspects but are still lacking in a lot of areas. This proposed blockchain-based vehicle passport system aims to improve on the current projects. In the case of a private blockchain, the data is shared offchain most of the time due to convenience. Accordingly, as long as the server is trusted and secured it is difficult for outside attacks to access the data. In the case of public blockchain, the data has to be shared onchain because there is no central authority. But everything on top of a public blockchain is public to anyone with an internet connection. Consequently, the data need to be encrypted before uploading it to the blockchain so only individual entities are able to decrypt it.

4.1 System Objectives

The primary objective of the proposed system is to guarantee the fair exchange of authentic confidential vehicle reports between service providers and potential buyers. To achieve this goal, we combine zk-SNARK [30], ElGamal encryption [31] and Poseidon hash function [32] in a smart contract-based application on top of Ethereum blockchain [33]. ElGamal encryption is utilized to transmit the requested report encrypted to the smart contract. Such an encryption can be only decrypted by the potential buyer's private key. Poseidon hash is used by service providers to produce commitments of the vehicle reports which are published on the blockchain. Finally, zkSNARK is used to enable the smart contract to verify that the delivered ElGamal encrypted report encrypts a report under the potential buyer's public key and it is same as the report committed to the blockchain using Posidon hash. Intuitively, zkSNARK allows the smart contract to verify the authenticity of the reports without access to their contents. The two main security properties of our system are as follows.

Fair Exchange. This security property is the most vital part of any transaction. It ensures that both parties receive their desired items without any of them not fulfilling their part of the deal [34]. The easiest way to implement fair exchange is by introducing a trusted third party to mediate between untrusted parties. Fair exchange is inherent in smart contracts on blockchains as their code is autonomously executed by the network nodes and its output is enforced by the consensus mechanism.

Confidentiality. In the available blockchain-based vehicle passport solutions, vehicle reports are delivered offchain by a trusted party after a potential buyer has paid the requested fees. Accordingly, confidentiality is provided by the delivery method, e.g., encrypted email

or authorized download access. In the case of onchain delivery, all nodes and any observer of the blockchain can view the contents of the report. We employ verifiable encryption to provide confidentiality of the report while being able to verify properties about the report itself.

In what follows, we give the notation and specification of the algorithms we used in the proposed blockchain-based vehicle passport (BVP) system. Then we detail the architecture and threat model of the protocol.

Notation. Sampling x uniformly at random from $\mathbb{F}_p$ is denoted by $x \xleftarrow{R} \mathbb{F}_p$. An elliptic curve $\mathcal{EC}$ over $\mathbb{F}_p$ has all its points $(x, y) \in \mathbb{F}_p^2$. The order of the elliptic curve group is p and it is a large prime number and all the equations are modulo p. $\mathcal{EC}$ is defined by the set of solutions (x, y) satisfying $y^2 = x^3 + ux + v$ and $4u^3 + 27v^2 \neq 0$ where u, and $v \in \mathbb{F}_p$. We assume that the discrete logarithm problem is intractable in prime order elliptic curve (EC) groups, i.e., for any point Q and a secret $e \in \mathbb{F}_p$, there is a point P, where $P = eQ$ and can be efficiently computed. On the other hand, when given P and Q, it is infeasible to find e. Thus, e is referred to by the discrete log of P to the base Q.

The blockchain accounts of the authorized wallets to access vehicle data is denoted by $\mathcal{W}_{spj}$ (i.e., a list of auto mechanics and technicians under the same affiliation), where $1 < j < l$ and l is the number of service providers supported by BVP. The offchain server of BVP has a blockchain account $\mathcal{S}_{BVP}$ which can communicate with the onchain smart contract.

4.2 ElGamal Encryption over EC

Elliptic curves cryptosystems require keys smaller than those based on Galois fields due to the lack sub-exponential algorithms that solve the EC discrete log problem [35]. For

comparison, ElGamal 128-bit security requires a Galois field of prime order 3072 and an EC field of order 256 [36].

Definition 1 (ElGamal over EC [31])**.** *let $f(m) : m \rightarrow P_m$ is a public bijective function which maps any message m to an elliptic curve point P_m. The ElGamal scheme has the following three procedures.*

- *Key generation. $(pk, sk) \leftarrow \mathcal{K}(1^\lambda)$, where $\lambda \in \mathbb{N}$ denotes the security parameter. Given a primitive point $\alpha \in \mathcal{EC}$, a secret value $sk \xleftarrow{R} \mathbb{Z}_q$ is chosen and $y = sk\alpha$ is calculated. Now, the public key $pk = (y, \alpha)$, and the secret key sk.*
- *Encryption. $(c_1, c_2) \leftarrow \mathcal{E}(pk, m)$, $k \xleftarrow{R} \mathbb{Z}_q$ is randomly chosen, then $c_1 = k\alpha$, and $c_2 = f(m) + ky$.*
- *Decryption. $m \leftarrow \mathcal{D}(sk, c_1, c_2)$, $m = f^{-1}(c_2 - skc_1)$.*

4.3 Poseidon Hash

For zero-knowledge proofs, a small circuit is essential because the size of the circuit affects the computation efficiency. Compared to other competitor hash functions like Pederson hash function [37], Posidon's circuit is eight times smaller [32]. This results in less prover and verifier complexity which makes it significantly faster.

Definition 2 (Poseidon Hash [32])**.** *An iterative hash function that outputs a message digest after the following three sets of permutation rounds.*

- *$H_f \leftarrow R_f(ARC, S_b, m)$. For a message m, ARC denotes the addition of round constants, S_b denotes S-boxes and H_f is the state of the hash after R_f rounds are completed. R_f rounds have full substitution-box (S-box) layers. S-box transforms input bits to different bits as output according to its lookup table.*

- $H_p \leftarrow R_p(ARC, S_b, m)$. *$H_p$ is the state of the hash after R_p rounds are completed. R_p rounds have partial S-box layers to improve efficiency.*

- $H \leftarrow R_f(ARC, S_b, m)$. *$H$ is the final state of the hash after R_f rounds are completed.*

4.4 Architecture

The BVP system has two subsystems. One is the offchain subsystem which consists of the BVP server $\mathcal{S}_{BVP}$ and all the service providers which are included in a list known as $spList$ which is a list stored on the blockchain and updated by $\mathcal{S}_{BVP}$. The other is the onchain subsystem which consists of the smart contract C_{BVP}. Figure 4.1 depicts the interaction between the entities in the BVP system.

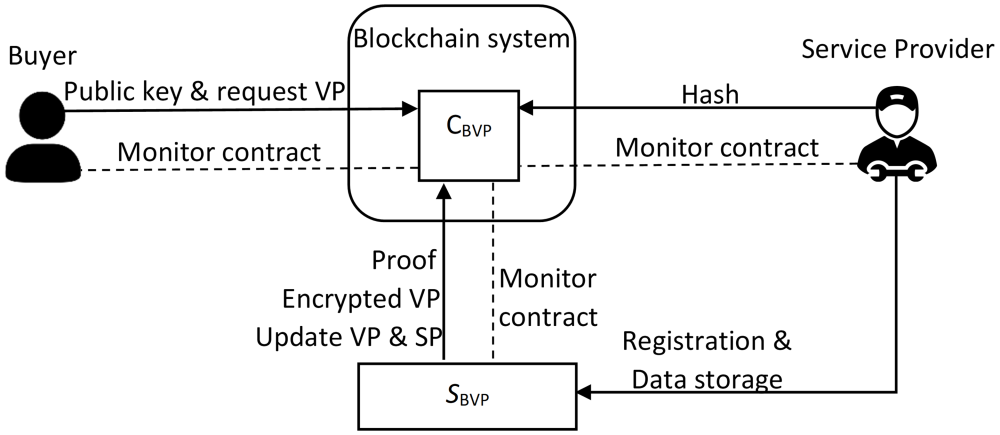

Figure 4.1: BVP system architecture.

4.4.1 Offchain Subsystem

The offchain subsystem is responsible for all the tasks conducted outside the blockchain. Thus $\mathcal{S}_{BVP}$ and all the service providers in $spList$ are part of the offchain subsystem.

BVP server $\mathcal{S}_{BVP}$. This server is responsible for the deployment of the smart contract. $\mathcal{S}_{BVP}$ also registers new vehicles and service providers to the blockchain. Apart from that,

$\mathcal{S}_{BVP}$ is responsible for generating the proof Ψ and encrypting vehicle passports as $\mathcal{E}_g$ when requested by a potential buyer. $\mathcal{S}_{BVP}$ monitors the smart contract all the time to respond to any new requests.

Service Providers in $spList$. Service providers are responsible for collecting the raw data of the vehicles. The raw data is sent to $\mathcal{S}_{BVP}$ by the service providers. To ensure commitment in the system, service providers hash the vehicular passport and add the latest version to the smart contract. Service providers receive a portion of the report fee when a transaction occurs. This incentives them to provide data on a regular basis.

4.4.2 Blockchain Subsystem

The blockchain subsystem is responsible for verification and fair exchange. Smart contract C_{BVP} is the core of this subsystem.

Smart Contract C_{BVP}**.** This contract is responsible for verifying that the encrypted vehicle passport sent by $\mathcal{S}_{BVP}$ is generated using the public key of the potential buyer and that the encryption is performed on a report equal to the report whose hash commitment is associated with the vehicle. The core functionality of C_{BVP} is executing the verifier part of zkSNARK. Only upon successful verification, C_{BVP} transfers the fees to $\mathcal{S}_{BVP}$ thus guaranteeing fair exchange.

4.5 Threat Model

Assumptions regarding the security of the system components and sources of threat are provided in this section.

- ***Smart contract and blockchain communication.*** Code of the smart contract is public, thus, anyone can check whether the protocols are correctly implemented or not as per the

proposal. This creates a trust in the onchain system that it will execute all of its tasks properly. All communications done with the blockchain are visible to everyone, so any confidentiality is lost if sensitive data is shared on the blockchain.

- ***Service providers.*** Only service providers designated by $\mathcal{S}_{BVP}$ will have access to add hashes to the blockchain and sending the raw data to the server. Service providers are trusted entities in the system as there is no protocol in place to check the validity of their tasks.

- ***Server.*** The system assumes that the server is not a trusted component. Particularly $\mathcal{S}_{BVP}$ is not trusted to deliver the correct vehicle passport after receiving the fees. Thus, smart contract is trusted for fair exchange instead.

- ***Potential buyer.*** Anyone has access to request vehicle passport from the system. The system does not trust any buyer. A buyer may send less balance than the required fee. Fair exchange ensures the transaction doesn't take place in this scenario.

4.6 Message Flow

In Figure 4.2, the sequential flow of messages between the smart contract C_{BVP} and buyer $\mathcal{W}_b$, service providers in $spList$ and BVP server $\mathcal{S}_{BVP}$ is depicted. Here, $\mathcal{W}_b$ represents the hashed public key of the buyer's wallet. The update, request and response protocols are highlighted in green, red and blue, respectively. Before any vehicle passport is requested by a potential buyer, the update protocol is initiated with m_u messages from $\mathcal{S}_{BVP}$ and service providers from $spList$ such as $\mathcal{W}_{sp}$. Here, $\mathcal{W}_{sp}$ represents the hashed public key of a service provider's wallet. $\mathcal{S}_{BVP}$ has administrative access to C_{BVP}, so any new registration for vehicle id V_n and service provider $\mathcal{W}_{sp}$ is initiated after $\mathcal{W}_{sp}$ sends $m_{u0} = (\mathcal{W}_{sp}, V_n, R_d)$

to $\mathcal{S}_{BVP}$ where R_d represents the raw data of the vehicle. Afterwards $\mathcal{S}_{BVP}$ registers $\mathcal{W}_{sp}$ and V_n to C_{BVP} with message $m_{u1} = (\mathcal{W}_{sp}, V_n)$. Any service provider included in the $spList$ can send m_{u2} messages to C_{BVP} which includes a newly computed hash H_n for a corresponding V_n. Now, C_{BVP} can take requests from buyer $\mathcal{W}_b$ for V_n with m_{rq1} messages. In the request protocol, an eligible m_{rq1} request contains a valid vehicle id number V_n, buyer's public key pk and p_b which is equivalent to the vehicle passport price and the gas fee. After an eligible vehicle passport request is added to C_{BVP}, $\mathcal{S}_{BVP}$ will be notified as it is continuously monitoring C_{BVP}. Following a request, the response protocol takes place where $\mathcal{S}_{BVP}$ sends m_{rs1} message to C_{BVP} which includes the proof Ψ, ElGamal encryption $\mathcal{E}_g$ of the vehicle data and V_n. Finally, C_{BVP} conducts the proof verification onchain where Ψ proves that $\mathcal{E}_g$ is an encryption under pk and that it encrypts vehicle data equal to the opening of the Poseidon hash commitment associated with V_n. Upon successful verification, C_{BVP} logs $\mathcal{E}_g$ with V_n and transfers p_b. Otherwise the transaction will not take place.

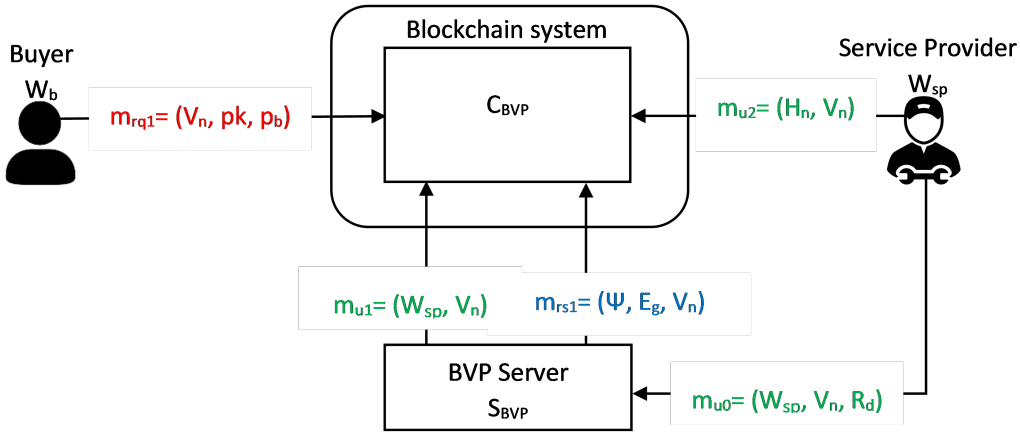

Figure 4.2: Update/Request/Response message flow in the BVP protocol

4.7 BVP Contract C_{BVP}

Smart contract C_{BVP} is the onchain entity which is responsible for managing all activities done on top of the blockchain. In this section, we first focus on how C_{BVP} follows set conditions and log data sent from $\mathcal{S}_{BVP}$, $\mathcal{W}_b$ and $\mathcal{W}_{sp}$ on the blockchain. Afterwards we focus on how it verifies the data when it is responding to the request. Figure 4.3 illustrates a pseudocode representation of the C_{BVP} smart contract. The contract is created by $\mathcal{S}_{BVP}$ in such a way that it has all access. During initialization, list of service provider $spList$, list of vehicle id numbers $\{V\}_1^n$ and a set price p_{vin} is created. In addition to this, common reference string CRS is created during the initial setup. To add to $spList$ and $\{V\}_1^n$, $\mathcal{S}_{BVP}$ calls functions $addserviceprovider(\mathcal{W}_{sp})$ and $addvehiclepassport(V_n)$ respectively. From the service provider side, any wallet account of the $spList$ such as $\mathcal{W}_{sp}$ can add a new hash with function $addhash(V_n, H_n)$. Now that all the updates are finished, any potential buyer such as $\mathcal{W}_b$ is able to request a vehicle passport by invoking the function $requestvehiclepassport(V_n, pk, p_b)$ which checks if V_n is part of $\{V\}_1^n$ and if p_b is equal or greater than p_{vin} after the gas fee has been deducted from p_b. If it meets both criteria only then p_b will be logged in the blockchain otherwise the majority of p_b will be sent back after subtracting all the gas fees. To respond to the request, $\mathcal{S}_{BVP}$ invokes the function $response(\Psi, \mathcal{E}_g, V_n)$. There is another internal function $(Verify(crs, \Psi, \mathcal{E}_g, H_n, pk))$ which implements the zkSNARK verification onchain (see Appendix A.2 for code). If the result is $\top$ then it will log $\mathcal{E}_g$ with V_n. Otherwise the transaction will not take place.

BVP contract C_{BVP}

`Initialize`:
admin = $\mathcal{S}_{BVP}$, $spList$ = [], vehicle id number = $\{V\}_1^n$, price = p_{vin}
Let Q, α and s generate crs in initial setup.

`addserviceprovider`: On receiving $\mathcal{W}_{sp}$ from $\mathcal{W}_{BVP}$
If ($\mathcal{W}_{BVP} = \mathcal{S}_{BVP}$)
Add $\mathcal{W}_{sp} \rightarrow spList$

`addvehiclepassport`: On receiving V_n from $\mathcal{W}_{BVP}$
If ($\mathcal{W}_{BVP} = \mathcal{S}_{BVP}$)
Add $V_n \rightarrow \{V\}_1^n$

`addhash`: On receiving V_n and H_n from $\mathcal{W}_{sp}$
If ($\mathcal{W}_{sp} \in spList$)
Log H_n with V_n

`requestvehiclepassport`: On receiving V_n, pk and p_b from $\mathcal{W}_b$
If ($V_n \in \{V\}_1^n$)
Log pk with V_n
If (p_b - gas fee $\geq p_{vin}$)
Log p_b
else
return p_b - gas fees
else
return p_b - gas fees

`response`: On receiving Ψ, $\mathcal{E}_g$ and V_n from $\mathcal{W}_{BVP}$
$Verify(crs, \Psi, \mathcal{E}_g, H_n, pk))$
If ($\mathcal{W}_{BVP} = \mathcal{S}_{BVP}$) and ($\top \leftarrow Verify(crs, \Psi, \mathcal{E}_g, H_n, pk)$)
Log $\mathcal{E}_g$ with V_n
else
return p_b - gas fees

$Verify(crs, \Psi, \mathcal{E}_g, H_n, pk))$
$(\top/\bot) \leftarrow (crs, \Psi, \mathcal{E}_g, H_n, pk)$
Output $(\top/\bot)$

Figure 4.3: Pseudocode of the smart contracts.

4.8 Security

BVP service ensures its security through two main properties which are confidentiality and fair exchange.

4.8.1 Confidentiality

Confidentiality needs to be added to the service because we are conducting all transactions on top of a blockchain. Thus, anyone with access to the internet also has access to all the data published on the blockchain. With the addition of the confidentiality property to the BVP service we can ensure that only the buyer can read the data. Formally we define it as follows.

Definition 3 (Confidentiality)**.** *We say that a system provides confidentiality when any PPT adversary $\mathcal{A}$ who is not the valid buyer $\mathcal{W}_b$ is unable to know the corresponding content. Formally, given ElGamal encryption $\mathcal{E}_g$ and negligible function $\epsilon(\lambda)$, the success probability of $\mathcal{A}$ is given by* $\mathsf{Pr}[(m) \leftarrow \mathcal{A}(C_{BVP}, \mathcal{E}_g)] \leq \epsilon(\lambda)$.

Theorem 1. *Given ElGamal encryption $\mathcal{E}_g$ under public key p_k. BVP achieves confidentiality of vehicle data under Definition 3.*

Proof. Assuming that the adversary $\mathcal{A}$ does not have access to the secret key corresponding to the encryption key p_k, then the view of $\mathcal{A}$ is similar to the view of a distinguishing adversary $\mathcal{B}$ in a chosen ciphertext indistinguishability attack on ElGamal encryption scheme. Since ElGamal encryption is proven to be semantically secured under the decisional Diffie-Hellman assumption, then there exists no PPT adversary $\mathcal{B}$ that can distinguish whether a given ciphertext is associated with a particular plaintext or not. Since $\mathcal{B}$ does not exist, therefore $\mathcal{A}$ does not exist. □

4.8.2 Fair Exchange

Transactions in the BVP system are not governed by any central authority. Smart contracts moderate the exchange of assets between untrusted parties. In BVP, the contract has to guarantee that either both parties get their expected asset, i.e., the buyer gets a vehicle passport whose hash is equal to the committed value and the BVP server gets the required fees, or no assets are exchanged. Since the contract does not accept a request unless the provided funds are equal to the requested report fees, our definition of fair exchange focuses on the guarantee that the potential buyer receives an authenticated report. Formally, we define it as follows.

Definition 4 (Fair Exchange)**.** *We say that the BVP system provides fair exchange if an adversarial BVP server $\mathcal{A}$ does not provide C_{BVP} with $\mathcal{E}_g$ that encrypts the opening of the commitment associated with the requested V_n. The success probability of $\mathcal{A}$ in breaking the fair exchange is given by*

$$\mathsf{Pr}[(\mathcal{E}_g, \Psi,) \leftarrow \mathcal{A}(C_{BVP}) : C_{BVP}.Verify(crs, \Psi, \mathcal{E}_g, H_n, pk) = \top] \leq \epsilon(\lambda)$$

Theorem 2. *Given a sound zero-knowledge proof Ψ and a collision resistant hash function H, BVP system provides fair exchange under Definition 4.*

Proof. An adversarial BVP server $\mathcal{A}$ is successful in causing $C_{BVP}.Verify(crs, \Psi, \mathcal{E}_g, H_n, pk)$ to return true in two cases. First, if $\mathcal{A}$ is able to find another report X that has the same hash as the commitment of the report which is published on the blockchain, i.e., $H(x) = H_n$. Since H is collision resistant, then the success probability of $\mathcal{A}$ is negligible. In the second case, $\mathcal{A}$ may try to generate a zero-knowledge proof where the report in the encryption $\mathcal{E}_g$ is

not equal to the opening of the commitment H_n. Since, the zero-knowledge proof is proven to be sound in the CRS model, the probability that $\mathcal{A}$ generates such a proof is negligible.

□

Chapter 5

Performance Evaluation

A proof-of-concept of the proposal has been open sourced and implemented on top of Ethereum. In addition to showing that it is practical to deploy on top of Ethereum, this also gives an option to assess the performance and efficiency of the prototype. The offchain section of the prototype can be divided into three sections: zkSNARK for the zero-knowledge proof, ElGamal for report encryption and Poseidon hash function to ensure integrity through commitments published on blockchain. The prototype allows verification onchain through a smart contract that is written in Solidity which is the primary language of Ethereum. The smart contract is given in Appendix A.2.

5.1 Measurements

The implementation has been done on a simulated local Ethereum full node running on an Acer Aspire A515 laptop with 10th generation Intel Core i5 CPU and 12 GB of RAM. To evaluate the performance of the prototype, three measurements were used which are (i) proof size (ii) prover and verifier time (iii) gas cost of proof verification on Ethereum. zk-SNARK utilizes the elliptic curve BN128 where each field element occupies 32 bytes.

Ethereum contract EIP-196 is also precompiled to perform point addition and multiplication operations [38]. Gas used for the deployment of the contract is 1256563 gas.

5.2 Prototype Performance

The utilized zkSNARK follows Groth's proposal [39] where the proof size is two points in $\mathbb{G}_1$ and one point in $\mathbb{G}_2$ and the verifier only needs to check three pairings to decide on accepting or rejecting the proof. For the proof generation, the prototype needs to process all the public inputs first which are (i) four field elements for ElGamal encryption (ii) two field elements for the public key and (iii) one field element for the Poseidon hash. All the public inputs are encoded as $\mathbb{F}_p$ elements. Size of $\mathbb{F}_p$ is 128 bit as zkSNARK use BN128 elliptic curve. The public inputs are processed by the verifier as one point addition and one point multiplication for each field element. The gas cost of point addition, multiplication and three pairing checks is 150, 6000 and $3 \times 34000 + 45000$ respectively [40]. Accordingly, the gas cost for the zkSNARK proof verification is:

$$cost = 3 \times 34000 + 45000 + 7 \times (150 + 6000) = 190050$$

5.3 Comparison to Generic Proof Systems

In order to compare different proof systems, we have to compare their performance for the same number of circuit gates denoted by N. Such a comparison is depicted in Table 5.1. The generic comparison is done with the following proof systems.

- Bulletproof [41]. The system is EC based on secp256k1 curve and has 128-bit security. It is implemented on a typical desktop system.

- zkSTRAK [42]. The system is not EC based and has 80-bit security. The security is dependent on the collision resistance of hash functions. The proving algorithm is run on a server with 32 AMD cores running at clock speed of 3.2GHz, with 512GB of DDR3 RAM. So, on a typical desktop system the proving times are expected to be considerably longer.

- Bootle et al. [43]. The system uses an OpenSSL curve and has 128-bit security. The exact curve is not mentioned in the paper but it uses one with 256-bit prime field. It is implemented on a typical desktop system.

Table 5.1: Comparison to Generic Proof Systems

System	Gates (N)	Proof Size (KB)	Timing		Trusted Setup	Security Assumption
			Prover	Verifier		
Bulletproof [41][b]	4361	1.2	2723 ms	125.5 ms	No	DLP
zkSTRAK [42][c]	4361	>200	100 ms	50 ms	No	CSH
Bootle et al. [43][d]	4361	1.8	1183 ms	394 ms	No	DLP
zk-SNARK	4361	0.12	698 ms	5 ms	Yes	KOE

[b] Prove and verify times are linear in N and are approximately evaluated from Table 3 in [41].
[c] All timings are approximated from the figures in [42].
[d] Logarithmic communication of $6log(N)+13$ elements, prove computation takes $12N$ exponentiations and verify computation takes $4N$ exponentiations. Every exponentiation requires around $22.6\mu s$ [43].

In Table 5.1 security for two of the proof systems Bulletproof and Bootle et al. relies on Discrete Log Problem (DLP). They do not require any trusted setup and have comparable proof size. Their timing for proving and verifying are by far the slowest among all the proof systems. Security of zkSTRAK relies on Cryptographically Strong Hash (CSH). It also does not require any trusted setup. The proving and verifying time is considerably better

than Bulletproof and Bootle et al. But zkSTRAK is not as secure as others due to its 80-bit security and it also has the largest proof size at more than 200 KB. Security of zkSNARK relies on the Knowledge of Exponent/Coefficient (KOE) [44] assumption. Among all the parameters in Table 5.1, verifier timing is the most vital part because verification takes place on top of the blockchain. In this crucial parameter, zkSNARK is by far the most efficient with a verification time of just 5 ms. Moreover, proof size of zkSNARK is considerably smaller then every other proof at 0.12 kB. The difference in proving time is not drastically better but it still performed faster than all but zkSTRAK. Unlike other systems, zkSNARK requires a trusted setup where the public parameters which construct proofs and verify them are generated. Hence, if someone gets these parameters they will be able to prove any statement. Thus the random numbers need to be destroyed. With multi-party computation, all the parties collaborate together to assemble the parameters [45]. By implementing multi-party computation, final parameters are only compromised when all the parties collude. Thus, the chance of compromise is greatly reduced.

5.4 Comparison of Blockchain Based Vehicle Passport Systems

Blockchain based vehicle passport systems BVP, VINChain [1] and Xdev [2] mainly differ in the method to provide the service and their security guarantees as shown in Table 5.2. Blockchain is at the core of all three systems and the differences in approach start there. BVP is built on top of Ethereum which makes it a public blockchain. VINChain has a different approach and is built on top of a hybrid blockchain called Graphene. In Graphene, VINChain has the central authority and other service providers have limited access to it. Xdev on the other hand is built on top of a consortium blockchain called Hyperledger

Fabric [46]. Among the three, only BVP delivers the report onchain. As it's published on a public blockchain, the report is encrypted before being published. Cryptographic algorithms like zk-SNARK and hash function Poseidon are used in BVP to provide integrity and fair exchange of the reports without trusting any central authority. VINChain only uses hashing to provide a way for buyers to check the integrity of the report after buying it. Xdev's main purpose of using blockchain is to allow its stakeholders to collaborate by accessing each other's data and provide new services. This require constant update of a car's report. Both VINChain and Xdev have central authorities to conduct their transactions and don't provide fair exchange. Because they will loose control over the system if they do it on blockchain. BVP transactions are done with a smart contract which has a fair exchange protocol in place. This ensures that transaction will only go through when both party demands are met without involvement of an authority. The price of a full vehicle passport is 15 USD for VINChain and for Xdev it has not been selected yet. Price for BVP's service with the current prototype is approximately 30 USD. However, we expect that this system will be supported by its own dedicated public blockchain where the service fees are significantly reduced.

Table 5.2: Comparison of Blockchain Based Vehicle Passport

Traits	BVP	VINChain [1]	Xdev [2]
Blockchain Type	Public	Hybrid	Consortium
Blockchain Technology	Ethereum	Graphene	Hyperledger Fabric
Crypto Algorithm	zk-SNARK and Hash	Hash	No
Report Encryption	ElGamal	No	No
Report Delivery	Onchain	Offchain	Offchain
Fair Exchange	Yes	No	No
Price	30 USD*	15 USD	NA

* Gas used for Zk-SNARK = 190050. Approximate gas used for logging encrypted vehicle passport onchain is 10000. So total fee is 0.0100025 ETH (29.6 USD) at gas price 50 Gwei (0.00000005 ETH). 1 ETH = 2,960.61 USD as per March 18, 2022. Additional small gas fees for the queries will make the final price approximately 30 USD.

Chapter 6

Final Discussion and Conclusion

In this chapter we discuss some potential points regarding our design choices as well as areas for future work, and finally present our conclusion.

- *Why a public blockchain-based system?* Public blockchain provides transparency unlike any other centralized system. Due to using a public blockchain, the users do not need to trust any centralized authority. Fair exchange and verification are done on-chain and the process is transparent to all. Sensitive information is also encrypted before logging them onto the blockchain because it is visible to everyone.

- *Does BVP require a governing body?* Because of fair exchange and zero knowledge proof, BVP does not require a governing body. Encryption and generation of the proof is done on the $\mathcal{S}_{BVP}$ server. If $\mathcal{S}_{BVP}$ is compromised, the smart contract C_{BVP} stops any transaction from going through because the encrypted report needs to be generated from the same report whose hash is stored in the blockchain.

- *Does BVP's transactions take a long time to get confirmed?* The proof generation is done off-chain in the $\mathcal{S}_{BVP}$ server because if its done onchain then the unencrypted report will

be visible to everyone accessing the blockchain network. The verification and the fair exchange does not require a lot of computation which makes the process almost as quick as a generic transaction on the blockchain.

- *How can this prototype be implemented fully and what other improvements can be made to the existing model?* To implement the *BVP* system, it requires a designated blockchain of its own. There are two main reasons for implementing it on its own blockchain. Firstly, it will make the system efficient as there will not be any bloat. Secondly, it makes the system very cheap to use by providing a cheap token for the blockchain while maintaining the incentive to make profits for its nodes. Moreover, a designated blockchain would also make it easy to add new features to the existing service such as tracking any vehicle. It will be hard to implement new features in the existing system efficiently because it already has high traffic and workload. So any new feature on top of Ethereum will be slower compared to a designated blockchain.

6.1 Conclusion

In this work we proposed a system to share vehicle passport in a blockchain and zero-knowledge based system while providing fair exchange and confidentiality. The prototype showcased how we can implement the system in a public blockchain to make it as decentralized as possible without compromising on confidentiality. In search of providing a decentralized solution we used ElGamal encryption and zero-knowledge proof to provide similar securities that would be provided in a centralized system. Thus we made no compromise to the security of the system and removed the necessity to have trust in any authority to provide fair exchange and confidentiality. Future direction of the system is implementing it on a new designated blockchain to make the service cheaper and easy to add new services.

Appendix A

A.1 Circuit for zk-SNARK Script

In what follows, we provide the code used to create the circuit for zk-SNARK.

```
include "comparators.circom"
include "babyjub.circom"
include "gates.circom"
include "bitify.circom"
include "escalarmulany.circom"
include "smthash_poseidon.circom"

template DiscreteLogAny() {
    signal private input in;
    signal input G[2];
    signal output out[2];

    component pvkBits = Num2Bits(256);
    pvkBits.in <== in;
    component mulAny = EscalarMulAny(256);
    mulAny.p[0] <== G[0];
    mulAny.p[1] <== G[1];

    var i;
    for (i=0; i<256; i++) {
        mulAny.e[i] <== pvkBits.out[i];
    }
    out[0] <== mulAny.out[0];
    out[1] <== mulAny.out[1];
}

template DiscreteLogFixed() {
    signal private input in;
    signal output out[2];
```

```
    component pvkBits = Num2Bits(256);
    pvkBits.in <== in;
    component mulAny = EscalarMulAny(256);
    mulAny.p[0] <== 995203441582195749578291179787384436505546430278305826713579947235728471134;
    mulAny.p[1] <== 5472060717959818805561601436314318772137091100104008585924551046643952123905;

    var i;
    for (i=0; i<256; i++) {
        mulAny.e[i] <== pvkBits.out[i];
    }
    out[0] <== mulAny.out[0];
    out[1] <== mulAny.out[1];
}


template ElGamalEncryption() {

    signal private input m[2];
    signal private input r;
    signal input pk[2];
    signal output c[4];


    //rG C1
    component dl1 = DiscreteLogFixed();
    dl1.in <== r;
    c[0] <== dl1.out[0];
    c[1] <== dl1.out[1];
    //rPK C2
    component dl2 = DiscreteLogAny();
    dl2.in <== r;
    dl2.G[0] <== pk[0];
    dl2.G[1] <== pk[1];

    component add = BabyAdd();
    add.x1 <== m[0];
    add.y1 <== m[1];
    add.x2 <== dl2.out[0];
    add.y2 <== dl2.out[1];
    c[2] <== add.xout;
    c[3] <== add.yout;

}

template hashing() {

    signal private input m[2];
    signal output out;

    component s2 = SMTHash2();
```

```
    s2.R <== m[0];
    s2.L <== m[1];

    out <== s2.out;

}

template EnS() {
    signal private input m[2];
    signal private input r;
    signal input pk[2];
    signal output c[4];
    signal output out;

    component E = ElGamalEncryption();
    E.m[0] <== m[0];
    E.m[1] <== m[1];
    E.r <== r;
    E.pk[0] <== pk[0];
    E.pk[1] <== pk[1];
    c[0] <== E.c[0];
    c[1] <== E.c[1];
    c[2] <== E.c[2];
    c[3] <== E.c[3];

    component s1 = hashing();
    s1.m[0] <== m[0];
    s1.m[1] <== m[1];

    out <== s1.out;
}

component main = EnS()
```

A.2 Solidity Verifier

In what follows, we provide the solidity code generated from the zk-SNARK circuit for verifying the data onchain.

```
pragma solidity ^0.5.0;
library Pairing {
    struct G1Point {
        uint X;
        uint Y;
    }
    // Encoding of field elements is: X[0] * z + X[1]
    struct G2Point {
        uint[2] X;
        uint[2] Y;
    }
```

```
    /// @return the generator of G1
    function P1() internal pure returns (G1Point memory) {
        return G1Point(1, 2);
    }
    /// @return the generator of G2
    function P2() internal pure returns (G2Point memory) {
        // Original code point
        return G2Point(
            [11559732032986387107991004021392285783925812861821192530917403151452391805634,
             10857046999023057135944570762232829481370756359578518086990519993285655852781],
            [4082367875863433681332203403145435568316851327593401208105741076214120093531,
             8495653923123431417604973247489272438418190587263600148770280649306958101930]
        );


    }
    /// @return the negation of p, i.e. p.addition(p.negate()) should be zero. function negate(G1Point memory p) internal pure returns (G1Point memory)
    {
        // The prime q in the base field F_q for G1
        uint q = 21888242871839275222246405745257275088696311157297823662689037894645226208583;
        if (p.X == 0 && p.Y == 0)
            return G1Point(0, 0);
        return G1Point(p.X, q - (p.Y % q));
    }
    /// @return the sum of two points of G1
    function addition(G1Point memory p1, G1Point memory p2) internal
                                        view returns (G1Point memory r)
                                        {
        uint[4] memory input;
        input[0] = p1.X;
        input[1] = p1.Y;
        input[2] = p2.X;
        input[3] = p2.Y;
        bool success;
        // solium-disable-next-line security/no-inline-assembly
        assembly {
            success := staticcall(sub(gas, 2000), 6, input, 0xc0, r,
                                                    0x60)
            // Use "invalid" to make gas estimation work
            switch success case 0 { invalid() }
        }
        require(success,"pairing-add-failed");
    }
    /// @return the product of a point on G1 and a scalar, i.e.
    /// p == p.scalar_mul(1) and p.addition(p) == p.scalar_mul(2) for
```

```
                                        all points p.
function scalar_mul(G1Point memory p, uint s) internal view returns
                                        (G1Point memory r) {
    uint[3] memory input;
    input[0] = p.X;
    input[1] = p.Y;
    input[2] = s;
    bool success;
    // solium-disable-next-line security/no-inline-assembly
    assembly {
        success := staticcall(sub(gas, 2000), 7, input, 0x80, r,
                                                0x60)
        // Use "invalid" to make gas estimation work
        switch success case 0 { invalid() }
    }
    require (success,"pairing-mul-failed");
}
/// @return the result of computing the pairing check
/// e(p1[0], p2[0]) *  .... * e(p1[n], p2[n]) == 1
/// For example pairing([P1(), P1().negate()], [P2(), P2()]) should
/// return true.
function pairing(G1Point[] memory p1, G2Point[] memory p2) internal
                                        view returns (bool) {
    require(p1.length == p2.length,"pairing-lengths-failed");
    uint elements = p1.length;
    uint inputSize = elements * 6;
    uint[] memory input = new uint[](inputSize);
    for (uint i = 0; i < elements; i++)
    {
        input[i * 6 + 0] = p1[i].X;
        input[i * 6 + 1] = p1[i].Y;
        input[i * 6 + 2] = p2[i].X[0];
        input[i * 6 + 3] = p2[i].X[1];
        input[i * 6 + 4] = p2[i].Y[0];
        input[i * 6 + 5] = p2[i].Y[1];
    }
    uint[1] memory out;
    bool success;
    // solium-disable-next-line security/no-inline-assembly
    assembly {
        success := staticcall(sub(gas, 2000), 8, add(input, 0x20),
                                            mul(inputSize, 0x20),
                                            out, 0x20)
        // Use "invalid" to make gas estimation work
        switch success case 0 { invalid() }
    }
    require(success,"pairing-opcode-failed");
    return out[0] != 0;
}
/// Convenience method for a pairing check for two pairs.
function pairingProd2(G1Point memory a1, G2Point memory a2, G1Point
                                        memory b1, G2Point memory b2)
```

```
                                        internal view returns (bool) {
        G1Point[] memory p1 = new G1Point[](2);
        G2Point[] memory p2 = new G2Point[](2);
        p1[0] = a1;
        p1[1] = b1;
        p2[0] = a2;
        p2[1] = b2;
        return pairing(p1, p2);
    }
    /// Convenience method for a pairing check for three pairs.
    function pairingProd3(
            G1Point memory a1, G2Point memory a2,
            G1Point memory b1, G2Point memory b2,
            G1Point memory c1, G2Point memory c2
    ) internal view returns (bool) {
        G1Point[] memory p1 = new G1Point[](3);
        G2Point[] memory p2 = new G2Point[](3);
        p1[0] = a1;
        p1[1] = b1;
        p1[2] = c1;
        p2[0] = a2;
        p2[1] = b2;
        p2[2] = c2;
        return pairing(p1, p2);
    }
    /// Convenience method for a pairing check for four pairs.
    function pairingProd4(
            G1Point memory a1, G2Point memory a2,
            G1Point memory b1, G2Point memory b2,
            G1Point memory c1, G2Point memory c2,
            G1Point memory d1, G2Point memory d2
    ) internal view returns (bool) {
        G1Point[] memory p1 = new G1Point[](4);
        G2Point[] memory p2 = new G2Point[](4);
        p1[0] = a1;
        p1[1] = b1;
        p1[2] = c1;
        p1[3] = d1;
        p2[0] = a2;
        p2[1] = b2;
        p2[2] = c2;
        p2[3] = d2;
        return pairing(p1, p2);
    }
}
contract Verifier {
    using Pairing for *;
    struct VerifyingKey {
        Pairing.G1Point alfa1;
        Pairing.G2Point beta2;
        Pairing.G2Point gamma2;
        Pairing.G2Point delta2;
```

```
        Pairing.G1Point[] IC;
    }
    struct Proof {
        Pairing.G1Point A;
        Pairing.G2Point B;
        Pairing.G1Point C;
    }
    function verifyingKey() internal pure returns (VerifyingKey memory
                                                   vk) {
        vk.alfa1 = Pairing.G1Point(982399375118323431539727360531427184599556431453901890982453568452092372689,52452714463216704955347517892740314973836656615892467655273551406817328213870,2068611308731315171595258908034254803492797513854402417767325801988233398556810,2796976432538325727096131592764526784668941330909951518900095537544368158134,3701792475337380758903842124615304610334429020714105626018175908737797976711);
        vk.gamma2 = Pairing.G2Point([6785578125151640612691520646852998972810268820458039615692966508095374991428,1741469548933322216583800349249764063106631491241627711786211666329364082918610,[2006676343143890508274177441553338198754797973485700492227373227703031562729115256086221779514356666225630260364187358926755287026224101625001796919844806]);
        vk.delta2 = Pairing.G2Point([3446568175145577813432939572460819702685672154640495980607464560728445686979,1617144191628385355191883804552320615758168267547558204905129891787326502475410,[1121645220029769290891829024937520734825553343502513952777223786915391318522519916001460776640582057239301730670520089432156236173631608346021054796122205]);
        vk.IC = new Pairing.G1Point[](8);
        vk.IC[0] = Pairing.G1Point(8889481907928357886351900990394933414553316232977280844849228988877353478690,11571595821822143047397181576358264266048434772286076517486845782071023597545);
        vk.IC[1] = Pairing.G1Point(1761853006109834052533870097600108084837832794201716011207204050078230402631810,2142827664501030113338023518424681414468859209025582243140148549276315606503810);
        vk.IC[2] = Pairing.G1Point(1387659778136275271025663323690896695245632810820114794617415480462313133802110,2966895691726972017844393460026187602571035293446493368996077335115881174053);
        vk.IC[3] = Pairing.G1Point(1825036389300784421968132886243656837875832113048344950931034269529943678376710,1499143436010333204373294161472209532187895668623802868319931232556914273591110);
        vk.IC[4] = Pairing.G1Point(4925412162781199320284388708885622589494261880635275804561394356455982033243,3750871416789172827422026767636335035397856287546238916657391879433300141357);
        vk.IC[5] = Pairing.G1Point(6480587925214950131718489691493294269169410138954587980496192703992311605532,1746339169327571437743198468029719667896327437724582855644110016010330116861210);
        vk.IC[6] = Pairing.G1Point(1951454987994607026795963566466426308377092335516604134872540461384254101741610,192678602057172700
```

```
        011188489025780743697166812496768007744936007841895840816 95);
        vk.IC[7] = Pairing.G1Point(120762910701248047215621328801201392 1
        282439560814877072060313337177120874178 9,159766459665700759
        052575965875692301652989316688394069699512629666612081935 83);

    }
    function verify(uint[] memory input, Proof memory proof) internal
                                        view returns (uint) {
        uint256 snark_scalar_field = 218882428718392752222464057452572 75
        088548364400416034343698204186575808495617;
        VerifyingKey memory vk = verifyingKey();
        require(input.length + 1 == vk.IC.length,"verifier-bad-input");
        // Compute the linear combination vk_x
        Pairing.G1Point memory vk_x = Pairing.G1Point(0, 0);
        for (uint i = 0; i < input.length; i++) {
            require(input[i] < snark_scalar_field,"verifier-gte-snark-
                                                scalar-field");
            vk_x = Pairing.addition(vk_x, Pairing.scalar_mul(vk.IC[i + 1
                                                ], input[i]));
        }
        vk_x = Pairing.addition(vk_x, vk.IC[0]);
        if (!Pairing.pairingProd4(
            Pairing.negate(proof.A), proof.B,
            vk.alfa1, vk.beta2,
            vk_x, vk.gamma2,
            proof.C, vk.delta2
        )) return 1;
        return 0;
    }
    function verifyProof(
            uint[2] memory a,
            uint[2][2] memory b,
            uint[2] memory c,
            uint[7] memory input
        ) public view returns (bool r) {
        Proof memory proof;
        proof.A = Pairing.G1Point(a[0], a[1]);
        proof.B = Pairing.G2Point([b[0][0], b[0][1]], [b[1][0], b[1][1]]
                                            );
        proof.C = Pairing.G1Point(c[0], c[1]);
        uint[] memory inputValues = new uint[](input.length);
        for(uint i = 0; i < input.length; i++){
            inputValues[i] = input[i];
        }
        if (verify(inputValues, proof) == 0) {
            return true;
        } else {
            return false;
        }
    }
}
```

References

[1] "DECENTRALIZED VEHICLE HISTORY - VINchain.io," https://vinchain.io/files/vinchainwhitepaperen.pdf, accessed: 2022-3-18.

[2] B. De La Blockchain, "Passeport du véhicule connecté," https://www.irt-systemx.fr/wp-content/uploads/2020/09/PCC-WhitePaper.pdf, accessed: 2022-3-18.

[3] Z. Zheng, S. Xie, H. N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: a survey," *Int. j. web grid serv.*, vol. 14, no. 4, p. 352, 2018.

[4] B. Preneel, "Cryptographic hash functions," *Eur. trans. telecommun.*, vol. 5, no. 4, pp. 431–448, 2010.

[5] C. Pungila and V. Negru, "Improving blockchain security validation and transaction processing through heterogeneous computing," in *Advances in Intelligent Systems and Computing*. Cham: Springer International Publishing, 2020, pp. 132–140.

[6] A. Narayanan and J. Clark, "Bitcoin's academic pedigree: The concept of cryptocurrencies is built from forgotten ideas in research literature," *ACM Queue*, vol. 15, no. 4, pp. 20–49, 2017.

[7] A. Gabizon, "Explaining SNARKs part v: From computations to polynomials," https://electriccoin.co/blog/snark-explain5/, Apr. 2017, accessed: 2021-2-12.

[8] "How blockchain allows car manufacturers to create digital passports," https://utimaco.com/current-topics/blog/how-blockchain-allows-car-manufacturers-create-digital-passports, accessed: 2022-5-13.

[9] "Canadian vehicle history reports," https://www.carfax.ca/, accessed: 2022-2-23.

[10] "AutoCheck.com," https://www.autocheck.com/, accessed: 2022-2-23.

[11] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," *J. Cryptology*, vol. 3, no. 2, pp. 99–111, 1991.

[12] Satoshi, "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf, 2009, accessed: 2020-12-21.

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf, accessed: 2022-3-18.

[14] S. Tikhomirov, "Ethereum: State of knowledge and research perspectives," in *Foundations and Practice of Security*. Cham: Springer International Publishing, 2018, pp. 206–221.

[15] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *Fast Software Encryption*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 371–388.

[16] B. Kaliski and R. Laboratories, "The mathematics of the RSA public-key cryptosystem," https://www.nku.edu/~christensen/the%20mathematics%20of%20the%20RSA%20cryptosystem.pdf, accessed: 2022-3-18.

[17] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, 2001.

[18] A. Narayanan and J. Clark, "Bitcoin's academic pedigree," *Commun. ACM*, vol. 60, no. 12, pp. 36–45, 2017.

[19] J. R. Douceur, "The Sybil attack," https://www.microsoft.com/en-us/research/wp-content/uploads/2002/01/IPTPS2002.pdf, accessed: 2022-3-18.

[20] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, 1989.

[21] J. Hasan, "Overview and applications of zero knowledge proof (ZKP)," http://ijcsn.org/IJCSN-2019/8-5/Overview-and-Applications-of-Zero-Knowledge-Proof-ZKP.pdf, accessed: 2022-3-18.

[22] D. Lapidot and A. Shamir, "Publicly verifiable non-interactive zero-knowledge proofs," in *Advances in Cryptology-CRYPT0' 90*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 353–365.

[23] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 459–474.

[24] M. Levent Dogan, A. A. Ergür, J. D. Mundo, and E. Tsigaridas, "The multivariate Schwartz-Zippel lemma," https://arxiv.org/pdf/1910.01095.pdf, 2021, accessed: 2022-3-18.

[25] A. Gabizon, "Explaining SNARKs part III: The knowledge of coefficient test and assumption," https://electriccoin.co/blog/snark-explain3/, Mar. 2017, accessed: 2022-3-18.

[26] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct NIZKs without PCPs," in *Advances in Cryptology – EUROCRYPT 2013*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 626–645.

[27] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *IEEE Symposium on Security and Privacy*, 2013.

[28] A. Gabizon, "Explaining SNARKs part VI: The Pinocchio protocol," https://electriccoin.co/blog/snark-explain6/, May 2017, accessed: 2022-3-18.

[29] T. Kobayashi, "Efficient algorithms for tate pairing," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E89-A, no. 1, pp. 134–143, 2006.

[30] C. Reitwießner, "zkSNARKs in a nutshell," https://chriseth.github.io/notes/articles/zksnarks/zksnarks.pdf, accessed: 2022-3-18.

[31] Y. Tsiounis and M. Yung, "On the security of ElGamal based encryption," in *Public Key Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 117–134.

[32] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "POSEIDON: A new hash function for zero-knowledge proof systems," https://eprint.iacr.org/2019/458.pdf, accessed: 2021-5-18.

[33] "Ethereum whitepaper," https://ethereum.org/en/whitepaper/, accessed: 2022-3-18.

[34] H. Pagnia, "Fair exchange," *Comput. J.*, vol. 46, no. 1, pp. 55–75, 2003.

[35] M. Subhashini and R. Gopinath, "Mapreduce methodology for elliptical curve discrete logarithmic problems -securing telecom networks."

[36] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–203, 1987.

[37] M. Barbara, L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, M. Schofnegger, R. Walch, and T. Graz, "Reinforced concrete: Fast hash function for zero knowledge proofs and verifiable computation," https://eprint.iacr.org/2021/1038.pdf, 2021, accessed: 2022-3-18.

[38] Ethereum Improvement Proposals, "EIP-196: Precompiled contracts for addition and scalar multiplication on the elliptic curve alt_bn128," https://eips.ethereum.org/EIPS/eip-196, accessed: 2021-4-13.

[39] J. Groth, *Advances in cryptology - EUROCRYPT 2016: international conference on the theory and applications of cryptographic techniques, Vienna, Austria*, M. Fischlin and J.-S. Coron, Eds. Berlin, Germany: Springer, 2016.

[40] Ethereum Improvement Proposals, "EIP-1108: Reduce alt_bn128 precompile gas costs," https://eips.ethereum.org/EIPS/eip-1108, accessed: 2021-4-17.

[41] B. Bunz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018.

[42] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," https://eprint.iacr.org/2018/046.pdf, accessed: 2021-6-29.

[43] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting," in *Advances in Cryptology – EUROCRYPT 2016*, M. Fischlin and J. S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, vol. 9666, pp. 327–357.

[44] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable zero knowledge with no trusted setup," in *Advances in Cryptology – CRYPTO 2019*. Cham: Springer International Publishing, 2019, pp. 701–732.

[45] A. Rahimi and M. A. Maddah-Ali, "Multi-Party proof generation in QAP-based zk-SNARKs," *arXiv [cs.CR]*, 2021.

[46] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*. New York, NY, USA: ACM, 2018.