A Blockchain-based Energy Trading Scheme for Electric Vehicles

Mohamed Baza*, Ramy Amer†, Amar Rasheed*, Gautam Srivastava‡, Mohamed Mahmoud§, Waleed Alasmary¶
*Department of Computer Science, Sam Houston State University, Huntsville, TX, USA
*Department of Electrical Engineering, CONNECT Centre, Trinity College Dublin, Ireland
‡Department of Mathematics and Computer Science, Brandon University, Manitoba, Canada
§Department of Electrical and Computer Engineering, Tennessee Tech University, Cookeville, TN, USA
¶Department of Computer Engineering, Umm Al-Qura University, Makkah, Saudi Arabia

2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC) | 978-1-7281-9794-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/CCNC49032.2021.9369517

***Abstract*—An energy-trading system is essential for the successful integration of Electric vehicles (EVs) into the smart grid. Existing systems merely focus on making optimal decisions while others depend on anonymization to achieve EVs drivers' privacy which is not enough because they can be identified from visited locations. In this paper, leveraging blockchain technology, we propose a privacy-preserving charging-station-to-vehicle (CS2V) energy trading scheme. To preserve privacy, EVs are anonymous, however, a malicious EV may abuse the anonymity to launch Sybil attacks by pretending as multiple non-exiting EVs to launch powerful attacks such as Denial of Service (DoS) by submitting multiple reservations/offers without committing to them, to prevent other EVs from charging and make the trading system unreliable. To thwart the Sybil attacks, we use a common prefix linkable anonymous authentication scheme, so that if an EV submits multiple reservations/offers at the same timeslot, the blockchain can identify such submissions. To further protect the privacy of EV drivers, we introduce an anonymous and efficient blockchain-based payment system that cannot link individual drivers to specific charging locations. Our experimental results indicate that our schemes are secure and privacy-preserving with low communication and computation overheads.**

***Index Terms*—Security, privacy, blockchains,smart contracts, energy trading, and Electric Vehicles (EVs).**

I. Introduction

Electric Vehicles (EVs) have attracted considerable attention in the past few years because [1], [2] of theri numerous advantages over gasoline cars such as using stored electricity instead of fossil fuels which constitutes better transportation systems, less operation cost, and zero carbon dioxide emissions. EVs can charge at home from the electrical grid and solar panels. Another charging approach is to charge from charging stations (CSs). This form of charging is called charging stations to the vehicle (CS2V) energy trading, where energy traders can offer competitive prices to the EVs [3]. CS2V is useful for providing fast charging or when an EV is traveling for a long distance.

To facilitate the charging of EVs, an energy trading system is needed to match the bids of energy sellers to the requests of buyers and connect them. In these systems, a central system (server) that acts as a service manager is usually used. To organize energy trading, both the energy sellers and buyers need to exchange sensitive information with the service manager about the location and time of energy trading. Without privacy protection, the untrustworthy server and malicious adversaries can infer sensitive information about the EVs' drivers. For example, if the charging stations are installed at medical clinic parks, working place, and hospitals, sensitive information about the EVs' drivers can be revealed such as their habits, workplaces, and health conditions. Moreover, the server can infer if the users are on travel by monitoring their trading activities, e.g., if they do not buy or sell energy for a long time. Furthermore, it has been shown that the server can infer the real identity of EV drivers using the locations visited by them, and therefore, using anonymization alone is not enough to preserve privacy. Besides, running the service by a central server, makes the system vulnerable to the single point of failure and cyber attacks [4], [5]. Also, the centralized platform suffers from a lack of transparency since the server can favor certain buyers to be served first and certain sellers to sell their energy first. Also, relying on existing payment systems (such as credit or debit cards) for energy trading transactions may violate the privacy of the EV drivers, because their charging locations can be known or tracked over long periods. On the other hand, utilizing cryptocurrencies such as Bitcoin and Zcash [6] is not a viable solution, since they are prohibited or restricted in some countries [7].

Instead of using a central architecture for the energy trading system, some works used blockchain to design energy trading systems [8], [9] but they mainly focus on optimization techniques to maximize the sellers' profit. Very few works, such as [3], [8], have studied the security and privacy issues in energy trading. Nevertheless, the proposed scheme in [8] does not consider privacy and charging reservations. The scheme proposed in [3] suffers from several problems and limitations. First of all, the selected charging stations do not know if they are selected since the reservation is made hidden on the blockchain and this may lead to scheduling conflict since two different EVs may select the same bid. Finally, the paper does not develop an anonymous payment method to enable payment while protecting the privacy of the EVs' drivers.

To tackle the above challenges, in this paper, by leveraging blockchain technology, we propose a privacy-preserving energy trading scheme to enable energy trading between CSs and EVs. In the CS2V scheme, the CSs publish energy bids and EVs select an appropriate bid and reserve it. For efficiency and scalability our the scheme, we choose a *consortium* blockchain

made by the charging stations to run our scheme. Moreover, although anonymity is important to preserve the privacy of the EVs' drivers, some EVs may abuse this anonymity to launch Sybil attacks to pretend as multiple non-existing EVs and then launch powerful attacks on the system such as Denial of Service (DoS) attack. Specifically, the Sybil EVs may launch a DoS attack by making many fake reservations to block other EVs from charging and thus make the energy trading system unavailable. To thwart the Sybil attacks, we leverage a common prefix linkable anonymous authentication [10] so that a vehicle is allowed to authenticate its messages anonymously, however, if it tries to pretend as multiple EVs and submit multiple messages (reservations) with the same prefix, i.e., timeslot, the blockchain can link these submissions and know that they are sent from the same EV. However, no one can link the messages that are sent from the same vehicle at different times to preserve privacy.

Also, we develop an anonymous and efficient blockchain-based payment system that is based on real currency and integrate it into our scheme. In particular, the system leverages a financial institution (FI) to exchange real currency with digital coins that are provably untraceable. During purchasing the coins, FI can know the real identity of the buyer but when the coins are deposited by a charging station, FI can not link coins to the buyers to preserve their location privacy. Also, coin ownership can be transferred to enable EVs to pay for charging without involving FI, and thus our system does not require involving the financial institution in the payment of each transaction for efficiency and scalability. Besides, our system is secure against unauthorized use of coins, by enforcing proof of ownership for a batch of coins using a zero-knowledge proof (ZKP) protocol instead of verifying ownership of individual coins for efficiency. Moreover, our system is secure against double-spending using blockchain which stores the hash of spent coins.

The rest of this paper is organized as follows. In Section II, we discuss some techniques that are used in our schemes. Section III discusses the considered network and threat models. Then, our proposed scheme is presented in detail in Section IV. In Section V, we present the security and privacy analysis of our schemes followed by performance evaluations. Section VI discusses the related works. Finally, we give concluding remarks in Section VII.

II. Preliminaries

In this section, we present the necessary background needed to understand this paper.

A. Blind Elliptic Curve DSA Signatures

In a blind signature cryptosystem, a user (*requester*) obtain a valid signature on a message M from the *signer* without the need to reveal M to the signer. In our payment system, we leverage the blind signature cryptosystem described in [11] to create anonymous coins because it has low computation overhead with significantly shorter signatures. The aforementioned cryptosystem consists of the following steps.

- All parties use the same elliptic curve of order n with generator G. In addition, the signer's public key is $P = d \cdot G$, where d is the corresponding private key and $d \in \mathbb{Z}_n^*$. - The signer selects a uniformly random element $k \in \mathbb{Z}_n^*$ and sends $R = k \cdot G$ to the requester. - The requester selects uniformly random elements $\gamma, \delta \in \mathbb{Z}_n^*$ and computes $A = R + \gamma \cdot G + \delta \cdot P$. Let x be the x-coordinate of point A, and $t = x \bmod n$. The requester computes $c = H(M||t) \bmod n$ and sends $c' = (c - \delta) \bmod n$ to the signer. $H(\cdot)$ is a cryptographically secure hash function, and $H : \{\}^* \rightarrow Z_n^*$.

- The signer computes $s' = (k - c' \cdot \mathrm{d}) \bmod n$ and sends the result back to the requester.

- The requester calculates $s = (s' + \gamma) \bmod n$ and stores M signature as (s, c). - To verify the signature, the *verifier* calculates $A = c \cdot P + s \cdot G$. After that, it computes $t = x \bmod n$, where x is x-coordinate of point A. The verifier checks whether $c \stackrel{?}{=} H(M||t) \bmod n$.

B. Schnorr's Identification Protocol

Schnorr identification protocol enable the owner of a public key (*prover*) to prove to a *verifier*, in zero knowledge, that he knows the value of the underlying secret key [12]. In this scheme, the prover's public key is $P = d \cdot G$, where d is the corresponding private key. Then, the prover chooses a uniformly random element $k \in \mathbb{Z}_n^*$ and sends $R = k \cdot G$ to the verifier (*commitment*). The verifier selects a random element $e \in \mathbb{Z}_n^*$ and sends it to the prover (*challenge*). Finally, the prover computes $s = (k + e \cdot \mathrm{d}) \bmod n$ and sends it to the verifier (*response*). The verifier should accept the proof if $s \cdot G = R + e \cdot P$.

Gennaro et al. [13] proposes higher degree polynomials that enable the execution of multiple Schnorr protocol instances simultaneously with low overhead. The protocol is similar to the one described above, except for the last step. In particular, the prover have m public keys $\{P_1, P_2, \ldots, P_m\}$, corresponding to private keys $\{d_1, d_2, \ldots, d_m\}$. When the prover receives the verifier's challenge (e), it calculates the response $s = (k + \sum_{i=1}^{m} e^i \cdot d_i) \bmod n$. The verifier then accepts the proof by checking if $s \cdot G \stackrel{?}{=} R + \sum_{i=1}^{m} e^i \cdot P_i$.

C. Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK)

A zero-knowledge proof (zk-proof) enable a party (prover) to generate a cryptographic proof to another party (verifier) that certain values are obtained by executing pre-defined operations on some private inputs (witness) without the need to reveal any information about the witness [14]. The zk-SNARK further allows such proof to be generated non-interactively. More importantly, the proof is *succinct*, i.e., the proof size is independent of the complexity of the statement to be proved. More precisely, zk-SNARK has three phases. The setup phase outputs the public parameters to establish a SNARK for an NP-complete language L. In the proof generation phase, the Prover uses an instance x, and witness w to generate a proof for the statement $x \in L$. Finally, in the verification phase, the Verifier can efficiently verify the proof.

D. Common-prefix-linkable anonymous authentication

Recently, Lu et. al, [10] have proposed a common prefix linkable anonymous authentication scheme based on zk-SNARK described in Section. II-C. In this scheme, a user can authenticate messages anonymously and prove the validity of his certificate without being identified and without linking the messages that are sent from the same user. The only exception occurs when the same user authenticates two messages with the same prefix. In this case, anyone can determine that those two messages are sent from the same user. The scheme consists of the following five algorithms:

- $\mathsf{Setup}(1^\lambda) \rightarrow (PP, msk, mpk)$: This algorithm generates the public parameters PP needed for zk-SNARK scheme and the system's master public key mpk and master secret key msk.

- $\mathsf{CertGen}(msk, PK_i) \rightarrow cert_i$: This algorithm generates a certificate $cert_i$ to validate a public key PK_i of a private key SK_i.

- $\mathsf{Auth}(p||m, SK_i, PK_i, cert_i, PP) \rightarrow \pi = (t_1, t_2, \eta)$: This algorithm outputs an attestation π on a message m and a given prefix p that the sender learns the secret key that is corresponding to a valid certificate $cert_i$ without being able to identify the sender. The algorithm first computes two tags, $t_1 = H_1(p, SK_i)$ and $t_2 = H_1(p||m, SK_i)$, where H_1 is a secure hash function, e.g., SHA-256. Then, let $\vec{w} = (SK_i, PK_i, cert_i)$ represents the private witness, and $\vec{x} = (p||m, mpk)$ be all common knowledge, the algorithm runs zk-SNARK proving algorithm $\mathsf{Prover}(\vec{x}, \vec{w}, PP)$ for the following language $\mathcal{L}_T = \{t_1, t_2, \vec{x} = (p||m, mpk) \mid \exists \vec{w} = (SK_i, PK_i, cert_i)$ s.t. $\mathsf{CertVrfy}(cert_i, PK_i, mpk) = 1 \wedge pair(PK_i, SK_i) = 1 \wedge t_1 = H_1(p, SK_i) \wedge t_2 = H_1(p||m, SK_i)\}$, where the CertVrfy algorithm verifies the certificate using a signature verification, and $pair$ algorithm verifies whether two keys are a corresponding public/secret key pairs. Finally, the algorithm outputs $\pi = (t_1, t_2, \eta)$.

- $\mathsf{Verify}(p||m, \pi, mpk, PP) \rightarrow \{0, 1\}$: this algorithm outputs $0/1$ to indicate whether the attestation is valid or not for the attested message by running the zk-SNARK Verifier algorithm on π and PP.

- $\mathsf{Link}(m_1, \pi_1, m_2, \pi_2) \rightarrow \{0, 1\}$: On inputting two attestations $\pi_1 = (t_1^1, t_2^1, \eta_1)$ and $\pi_2 = (t_1^2, t_2^2, \eta_2)$, the algorithm simply checks $t_1^1 \stackrel{?}{=} t_1^2$. If the check is true, the algorithm outputs 1, indicating that π_1 and π_2 are computed from the same user on the same prefix; otherwise, it outputs 0, indicating that π_1 and π_2 are computed by different users.

III. Network and Threat Models

A. Network Models

Figs. 1 illustrates the network models of the CS2V scheme. The main entities are charging stations, charging EVs, discharging EVs, and the financial institution (FI). In addition to these entities that are involved in running our schemes, there is also an offline Key Distribution Center (KDC). The KDC's role is to distribute the public parameters and credentials of our schemes' cryptosystems. It also issues public key certificates to the charging stations and EVs that participate in the energy trading and the financial institution that runs the anonymous payment system. In practice, the KDC can be run by a governmental agency that is interested in the security of the energy trading system [15]. The FI sells *untraceable* digital coins to EVs and exchanges these coins for real currency.

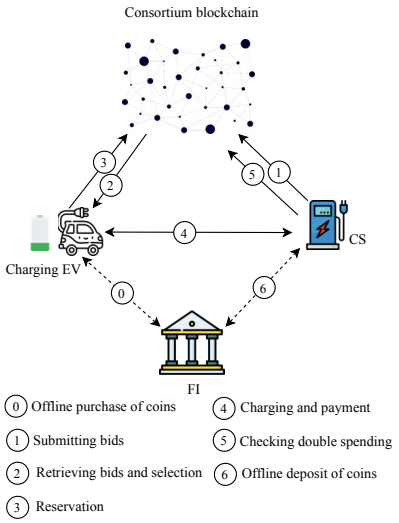


Figure 1: Network model of the CS2V scheme.

As shown in Fig. 1, there are charging station(s) owned by private companies that charge EVs. The charging stations can have many charging points that can charge EVs simultaneously, or they can have a few charging points installed in parking lots of clinics, shopping malls, workplaces, etc. Each EV interacts with the system through a Web or mobile application. The blockchain network is a consortium network made by the charging stations that run our scheme by processing all transactions' messages sent by the different entities.

B. Adversary and Threat Model

In this paper, we assume that FI, CSs, EVs, and validators are honest-but-curious, i.e., they execute our schemes correctly, but aim to infer sensitive information about the EVs' drivers by examining the exchanged messages/transactions. Adversaries can eavesdrop on the exchanged messages and recorded transactions on the blockchain so that they try to learn the visited locations and the driving patterns of victim EVs' drivers, guess the locations of drivers at a specific time or even track them over time. Also, attackers may launch Sybil attacks to pretend as multiple non-existing EVs to launch severe attacks such as DoS attacks by submitting many reservations to deprive honest EVs of getting charged or selling their energy which makes the energy trading service unavailable. We also consider attacks against the payment system. Specifically, some attackers may try to create fake coins and double-spend valid coins.

IV. Our Proposed Privacy-Preserving Energy Trading Scheme

A. System Initialization

In this phase, the KDC issues public key certificates to the charging stations, EVs, and the financial institution involved in the anonymous payment system. The KDC first generates

a master public/private key pair (mpk/msk) which is used for a digital signature scheme, such as RSA, for signing the certificates. Then, an EV v_i, having a unique identity (e.g., license plate number), creates a public/private key pair (PK_{v_i}/SK_{v_i}) and KDC generates a certificate $cert_{v_i}$ binding PK_{v_i} to v_i. The KDC also runs the Setup algorithm of zk-SNARK to generate the public parameters (PP) of the zk-SNARK. Finally, the KDC publishes the zk-SNARK parameters PP and the master public key mpk to the system users. These parameters are used by the EVs to authenticate messages using the common-prefix linkable anonymous authentication scheme. Note that the system initialization phase is done off-line and only once.

B. Purchasing Digital Coins

In our schemes, EVs need to purchase untraceable coins from the FI. Specifically, we assume that the energy trading application on the EV driver's smartphone has an *e-wallet* service, which maintains coins digitally signed by the FI. The FI only issues coins with predefined momentary values so that the coin value can not be used to identify the EV that bought the coin because many EVs buy coins with the same value. The purchase of digital coins proceeds as follows. Suppose the EV needs to purchase m digital coins from the FI. After the payment is done using credit card, the client generates m secret and random *elements* $\{s_1, s_2, \ldots, s_m\} \in \mathbb{Z}_n^*$ and computes m coin public keys $\{CP_1, CP_2, \ldots, CP_m\}$, where $CP_i = s_i \cdot G$, $\forall i \in \{1, 2, \ldots, m\}$. Each public key uniquely identifies one coin. The client and the FI then invoke the blind signature protocol discussed in Section II so that the client gets a valid signature $sig_{FI}(CP_i)$ for each purchased coin CP_i without revealing CP_i to FI.

C. Privacy-Preserving CS2V Energy Trading Scheme

As illustrated in Fig. 1, the CS2V scheme consists of the following phases. In the *submitting bids* phase, CSs submit competitive charging bids to the blockchain. In the *reservation* phase, an EV that needs to charge submits a reservation request to the blockchain, and the validators verify the reservation request and store it in the blockchain. In the *charging and payment* phase, the EV charges and pays. Finally, the last phase is the *coin deposit* where the CS deposits the coins in the FI.

1) Submitting Bids: In this phase, the charging stations publish their energy bids on the blockchain as follows. First, the day is divided into predefined timeslots, e.g., each timeslot can be 30 minutes or 1 hour. Then, when a CS i has an available charging point, it composes a *bidding* message msg_1:

$$msg_1 = TS \| bID_i \| PK_i \| \ell_i \| CR_i \| PO_i,$$

where TS is the timeslot of charging, bID_i is bid ID, PK_i is CS's public key, ℓ_i is the location of the CS, CR_i is the charging rate (price of each KWh), and PO_i is the amount of charging energy (KWh) it can provide.

The message is signed with the private key of the CS and is broadcasted on the blockchain network. Note that a charging station can send multiple bids at a specific timeslot and the number of bids depends on the number of EVs the charging station can charge at a certain timeslot. Before storing the bid on the shared ledger, the validators of the blockchain network should verify the signature of the message.

2) Reservation: In this phase, a charging EV cv_i that needs to charge queries the blockchain to retrieve the bids of the charging stations of a certain geographic area. Then, the EV should select the most appropriate bid in terms of distance to the CS, charging rate, and the amount of energy. The selection of the best bid depends on the preference of the charging vehicle. For instance, an EV may prefer a nearby charging station regardless of the charging rate while others may prefer lower charging rates.

Once cv_i determines the best bid, it then creates a new public/private key ($\mathcal{PK}_{cv_i}/\mathcal{SK}_{cv_i}$) used for once and the corresponding address $\mathcal{AD}_{cv_i}$. Then, it uses`sameprefix`-linkable anonymous authentication scheme to generate an attestation π_{cv_i} by running Auth algorithm descried in Section II-D and using the zk-SNARK's public parameters PP and the timeslot TS included in the bid as the prefix as follows:

$$\pi_{cv_i} = \mathsf{Auth}(TS || M_{cv_i}, PK_{cv_i}, SK_{cv_i}, cert_{cv_i}, PP),$$

where $M_{cv_i} = bID_j \| \mathcal{PK}_{cv_i}$. Then, cv_i creates a *reservation* message msg_2:

$$msg_2 = bID_j \| \mathcal{PK}_{cv_i} \| \pi_{cv_i}$$

where bID_j is the selected bid ID, π_{cv_i} is the authentication proof and $\mathcal{PK}_{cv_i} = k \cdot G$ is the one-time public key computed by the cv_i where $k \in \mathbb{Z}_n^*$. Then, the message is broadcasted on the blockchain network. Note that $\mathcal{PK}_{cv_i}$ will be used later by the cv_i to prove to the CS that it is the one that indeed made the reservation.

Then, the reservation request is verified and stored in the blockchain. To do so, the validators verify the attestation proof π_{cv_i} by running Verify algorithm of `sameprefix`-linkable anonymous authentication scheme described in Section II-D as follows:

$$\mathsf{Verify}(TS || M_{cv_i}, \pi_{cv_i}, mpk, PP),$$

Also, the validators check for multiple reservations by executing $\mathsf{Link}(\pi_{cv_i}, \pi_{cv_*})$ discussed in Section II-D for each valid authentication attestation π_{cv_*} that was received in the timeslot TS. The objective here is that it is not allowed that one EV makes more than one reservation in one timeslot. Once all these verifications succeed, the reservation transaction is stored on the ledger. Note that running the *Link()* algorithm is very efficient because it does not require cryptographic operations from the validators and it is just checking the equality of the two authentication attestations as described in Section II-D, and this makes our scheme scalable.

3) EV Charging and Payment: After reserving a charging point, in this phase, an EV charges and pays. First, when the EV arrives at the charging station, it has to prove that it has reserved a charging point. In particular, the EV and the CS engage together in Schnorr's identification protocol (Section II), for the EV to prove to the CS that it indeed knows the private key that is corresponding to the public key that used in the reservation. Following successful authentication, the EV

charges the amount of power in the reservation and then initiates the payment procedure.

The CS chooses m secret random *elements* $\{k_1, k_2, \ldots, k_m\} \in \mathbb{Z}_n^*$ and computes m public keys $\{P_1, P_2, \ldots, P_m\}$, where $P_j = k_j \cdot G$, $\forall j \in \{1, 2, \ldots, m\}$ assuming that cv_i pays m coins to CS. The EV transfers the ownership of each coin to the CS as follows.

The EV first selects a random element $a \in \mathbb{Z}_n^*$, then it computes $a \cdot G$, $r = x_1 \bmod n$, where x_1 is the x-coordinate of the point $a \cdot G$. Then, it computes $a^{-1} \bmod n$ and $b = H(TC_j)$, where $TC_j = CP_i \| sig_{FI}(CP_i) \| P_j$, $CP_i \| sig_{FI}(CP_i)$ is the coin to be transfered, and $H : \{\}^* \rightarrow Z_n^*$. Then, it uses the secret key s_i of the coin $CP_i \| sig_{FI}(CP_i)$ to compute $s = a^{-1}(b + s_i \cdot r) \bmod n$. The value (s, r) is the signature $Sig_{CP_i}(TC_j)$ on TC_j. Thus, $TC_j \| Sig_{CP_i}(TC_j)$ is the transferred coin. Then, the EV sends the transferred coin to the blockchain. The blockchain should do the following: (i) verify the FI signature on the original coin to be transfered, (ii) check if the original coin has not been spent before by checking the list of spent coins on the blockchain and (iii) verify the signature of the transferred coin as follows. It computes $b = H(TC_j)$, $u_1 = b \cdot s^{-1} \bmod n$, and $u_2 = r \cdot s^{-1} \bmod n$. Then, it computes $(x_1, y_1) = u_1 \cdot G + u_2 \cdot CP_j$, if $x_1 = r$, then the signature is valid. If all the verifications succeed, the original coin is stored in the list of spent coins to prevent double spending the coin and the blockchain send transferred coins to the CS.

4) Coin Deposit: In this phase, the CS deposits the coins it collects in a period (e.g., several days) in the FI. The CS sends a group of coins to the FI. The FI verifies the following for each coin: (i) the coin has not been spent before by checking if the hash of the coin is in the list of spent coins, (ii) the FI signature of the original coin is valid, (iii) the signature that transfers the ownership of the coin from an EV to the CS is valid, and (iv) finally, FI checks if the CS owns the coins; to do so, the CS and the FI should invoke the batch version of Schnorr's protocol described in Section II-B so that the CS can prove that it knows the underlying secrets of the submitted coins. If all these verifications succeed, the FI deposits the coins to the CS account and the CS can exchange coins with real currency if it wants. Finally, the FI sends the coins to the blockchain to be stored in the list of spent coins to avoid re-depositing or re-spending them.

V. Evaluations

A. Security and Privacy Analysis

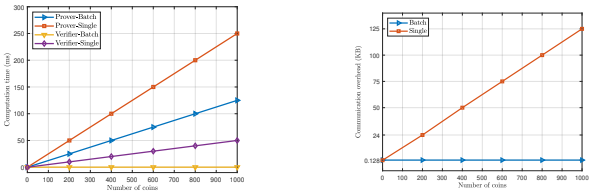
Preserving the privacy of EVs' drivers. In our schemes, we have used several techniques to preserve the privacy of EVs' drivers such as cloaking technique, anonymous authentication scheme, one-time public/private key pair, and anonymous payment. To ensure the anonymity of EVs' drivers, the EV drivers use a random blockchain address that acts as a *pseudonym* and the generation of that address is unlinkable to the EV driver's real identity. Also, the underlying common-prefix-linkable anonymous authentication scheme ensures that an EV can authenticate messages without being able to identify the sender of the messages or link them. Finally, the anonymous payment system ensures that the coins used for payment are untraceable and are not linked to a specific EV.

Unlinkabilty. In our schemes, no one can link a charging reservation or bid to a specific EV that sent it. This is because EVs interact with the blockchain with a randomly generated blockchain address. The blockchain address is a one-time *pseudonym* generated by the EVs, and it can not reveal the EV's real identity. Moreover, the underlying common-prefix-linkable anonymous authentication scheme ensures that an EV can authenticate messages without being linked if an EV sends one message with each prefix so the EV is anonymous and no one can link its messages.

Anonymous and secure payment system. In our schemes, blockchain is used for storing the hash values of spent coins so that double spending can be detected. Thus, due to the use of blockchain, all spent coins are stored in a tamper-resistant and non-repudiable ledger. Also, once the coin's ownership is transferred, no one can re-spend or re-transfer the same coin because the spent coins are stored in the blockchain. Moreover, during the purchase of digital coins, the EV's real identity is used but because of using the blind signature scheme, the FI can not link the coins it signed to the EV that bought them. The payment is also secure against stolen coins. This is because if an attacker steals coins, he/she can not use them in other transactions. After all, the digital coin spender has to prove with a ZKP protocol that it indeed knows the coin's private key without revealing it.

Anonymity-yet-resistance to Sybil attacks. Our schemes ensure the anonymity of EVs drivers during the energy trading because the real identity of each vehicle is not used and no one can link the messages sent from the same EV. However, a malicious EV may try to launch a Sybil attack by pretending to multiple non-existing EVs to launch more powerful attacks such as DoS on the trading system by submitting a large number of reservations/requests to the blockchain to make the system unreliable and the service is unavailable. In our schemes, the blockchain can detect such submissions using the linkable anonymous authentication scheme. This is because, for an EV to successfully authenticate messages, it needs to include a *prefix* to them. If an EV authenticates reservations/requests more than once for the same transaction (i.e, with the same prefix), the blockchain can link such submissions. In this way, Sybil submissions can be discarded to ensure that the system is reliable and the service is available.

Mutual authentication. In our schemes, an EV, either charging or discharging, needs to authenticate messages using their secret credentials by running the *Auth* algorithm of the common linkable anonymous authentication scheme. Besides, before starting the energy trading, a charging EV needs to prove to the CS (or the discharging EV) that it is the one that indeed made the reservation. In the CS2V, the EV and the CS engage in an instance of Schnorr's identification protocol for the EV to prove to the CS that it knows the private key corresponding to the public key that is computed by the EV that made the reservation.



(a) on computation time of verifying the coins ownership.

(b) on coins ownership verification overhead.

Figure 2: Effect of Schnorr's batch protocol.

B. Performance Evaluation

In this section, we evaluate the performance of the proposed schemes in terms of communication and computation overheads.

1) Experiment Setup: We have implemented all cryptosystems used in our schemes with the C programming language, using OpenSSL's library for elliptic curve cryptography, and a 2.8 GHz Intel Core i7 CPU. For 128-bit security, we have selected ANSI's *X9.62 Prime 256v1* curve, whose order n is a 256-bit prime. We have implemented the proposed CS2V scheme and deployed it in Ganache blockchain [16] platform, which is a private blockchain.

We have used Jsnark library [17] for building the circuits of zk-SNARKs used in the underlying common-prefix linkable anonymous authentication scheme. The library uses libsnark [18] as a backend and has built-in gadgets, i.e., small boolean circuits including a gadget of SHA-256 and gadgets for encryption algorithms such as AES. Then, we wrote the statements of zk-SNARKs in Jsnark using the underlying gadgets. The public parameters of zk-SNARK are generated using the libsnark generator library. Finally, to verify the proofs of the underlying common-prefix linkable anonymous authentication scheme, we have used the modified Ethereum client [19] that is written in Java 1.8 with Spring framework [20].

The metrics that are used in our evaluations are communication and computation overheads. The computation overhead is defined as the processing time required by each entity in our schemes. The communication overhead is measured by the size of messages transmitted between the entities in bytes.

2) Communication and storage overhead: To measure the communication overhead, our schemes use a signature scheme that uses elliptic curve cryptography (ECC). Using an elliptic curve additive group of order 256 bits, the signature's size is 64 bytes.

To purchase a digital coin, the EV and the FI need to run the blind signature scheme as discussed in Section IV-B. The EV sends a message that contains the number and value of coins it wants to purchase. Also, it sends a message that has one field element. Thus, the size of these messages is 70 and 32 bytes respectively. The FI reply with a message that has one group point and a signature. It also needs to send a message that has one field element. Thus, the size of these messages is 128 and 32 bytes, respectively. Then, during the payment phase, the EV sends the transferred coin to the blockchain that contains two public keys, FI's signature, and the EV's signature. Thus, the total size of the transferred coin is 256 bytes. To deposit the coins that the CS collects in a period, it needs to send 256 bytes for each transferred coin to the FI. Thus, to send n coins, the total overhead is $n \times 256$ bytes. Finally, the CS needs to prove to the FI that it owns the coin by proving that it knows the secret that corresponds to the public key in the coin using Schorr's batch protocol. To do so, the CS sends a commitment, i.e., a group element, then the FI sends a challenge, i.e., a field element and the CS sends a response, i.e., a field element. Thus, the total communication overhead is 128 bytes. Fig. 2.b evaluates the effect of Schorr's batch protocol on verifying a group of coins instead of individual ones. As described in Section II, the batch protocol involves the transmission of the same size of messages as the original protocol, so the communication cost is constant (128 bytes) concerning the number of coins. On the other hand, the cost of executing individual instances of Schnorr's protocol incurs a cost that grows linearly with the number of coins.

In the *submitting bids* phase, a charging station needs to send a bid that contains the following: timeslot, bid ID, location, charging rate, amount of energy, CS's public key, and a signature. The total size of the packet is 150 bytes. In the *reservation* phase, a charging EV needs to send a reservation request that contains bid ID, CS's public key, and authentication proof. The total size of the packet is 795 bytes.

Storage overhead. The blockchain stores the charging bids (150 bytes per each bid) and hashes of the spent coins (256 bytes). Thus, the total storage overhead for storing $\mathcal{M}$ bids and $\mathcal{N}$ spent coins is $(150 \times \mathcal{M})$ + $(256 \times \mathcal{N})$ bytes.

3) Computation Overhead: Our measurements indicate that the multiplication and exponentiation operations take 0.005 ms and 4.4 ms, respectively. Note that the addition operation takes a very short time so it can be neglected. For symmetric key encryption using AES-128, the encryption operation takes 0.0203 ms while the decryption operation takes 0.0078 ms. For the signature algorithm, the signing operation takes 3 ms while the verification takes 1,5 ms.

To purchase a digital coin, an EV needs to compute two multiplication operations and one signature. Thus, the computation overhead is 3.01 ms. The FI computes two multiplication operations and one signature. Thus, the computation overhead is 3.01 ms. To transfer a coin, the EV needs to compute a signature that takes 3 ms. Then, the validators verify two signatures, i.e., the FI's signature and the signature of the old coin's owner on the transferred coin. Thus, the computation overhead is 6 ms. Then, to deposit a coin, the FI needs 6 ms to verify two signatures on the transferred coin. Finally, the CS needs to prove the ownership of the digital coins using Schnorr's batch protocol. In Fig. 2b, we show Schnorr's protocol efficiency in verifying the ownership of a huge number of coins. It can be seen that verifying 1000 coins individually (without Schnorr's batch protocol) requires 125 ms at the prover and 250 ms at the verifier, whereas the batch protocol reduces these times to 0.1 ms and 54 ms, respectively. This indicates that Schnorr's batch protocol makes the payment process fast with less computation overhead.

In the *submitting bids* phase, the computation overhead on

the charging station is 3 ms to compute a signature on the message and 1,5 ms for the validators to verify the signature. In the *reservation* phase, the charging EV computes authentication proof using the common-prefix linkable anonymous authentication scheme that needs fifty seconds. Then, the validators verify the authentication proof that needs 2.5 ms. Verifying 10 proofs requires 17.9 ms and thus the on-chain computation overhead is in the range of *milliseconds which is acceptable*.

VI. Related Work

In [8], the authors use smart contracts to select the best bids from other charging stations. The EVs send their planned routes and battery status to the blockchain and then charging stations offer their prices so that the EVs select the best-offered price. Nevertheless, the proposed scheme does not consider charging reservation, or the underlying payment mechanism. In [3], charging stations make bids to EV drivers in response to their charging requests. To select their preferred charging bids, EVs send hidden commitments to the blockchain. These commitments are used later so that the EV can prove to the charging station that it made the reservation. However, the charging station does not know if the charging point has been reserved or not until the EV drives to it and this may lead to poor management of the service by the CS. Furthermore, the paper does not propose an anonymous payment method to protect the privacy of the EVs. Also, it does not consider the Sybil attacks in which attackers may pretend as multiple non-existing EVs and launch severe attacks such as denial of service (DoS) attack by sending a large number of reservation requests to block charging stations from getting customers to use the service.

VII. conclusion

In this paper, we have proposed privacy-preserving schemes for energy trading. The schemes are built using blockchain technology that provides security and transparency without a trusted third party. Moreover, launching Sybil attack by pretending as multiple non-existing EVs to launch powerful attacks such as DoS by submitting a large number of reservations/offers is thwarted to ensure that the energy trading system is reliable and the service is available. To preserve the privacy of EV drivers, we have developed an anonymous and efficient payment system using blockchain to allow EVs to pay their charging fees with untraceable digital coins. Also, the payment system is secure against stolen coins attack and fake payments. Analysis and experiments are conducted to evaluate the proposed schemes and the results indicate that our schemes are secure and can preserve the privacy of EVs' drivers, and the communication and computation overheads are acceptable.

VIII. Acknowledgement

The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number 589

References

[1] M. Baza, M. M. Fouda, A. S. T. Eldien, and H. A. Mansour, "An efficient distributed approach for key management in microgrids," *Proc. of the Computer Engineering Conference (ICENCO), Egypt*, pp. 19–24, 2015.

[2] M. Baza, M. Fouda, M. Nabil, A. S. Tag, H. Mansour, and M. Mahmoud, "Blockchain-based distributed key management approach tailored for smart grid," in *Combating Security Challenges in the Age of Big Data*. Springer, 2019.

[3] F. Knirsch, A. Unterweger, and D. Engel, "Privacy-preserving blockchain-based electric vehicle charging with dynamic tariff decisions," *Computer Science - R&D*, vol. 33, no. 1-2, pp. 71–79, 2018.

[4] M. Baza, M. Mahmoud, G. Srivastava, W. Alasmary, and M. Younis, "A light blockchain-powered privacy-preserving organization scheme for ride sharing services," *Proc. of the IEEE Vehicular Technology Conference (VTC2020-Spring), Belgium*, pp. 1–6, 2020.

[5] M. Baza, N. Lasla, M. Mahmoud, G. Srivastava, and M. Abdallah, "B-ride: Ride sharing with privacy-preservation, trust and fair payment atop public blockchain," *IEEE Transactions on Network Science and Engineering*, 2019.

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

[7] Countries that have banned cryptocurrency. [Online]. Available: https://www.escapeartist.com/blog/countries-banned-bitcoin/

[8] M. Pustisek, A. Kos, and U. Sedlar, "Blockchain based autonomous selection of electric vehicle charging station," *Proc. of the international conference on identification, information and knowledge in the Internet of Things (IIKI)*, pp. 217–222, 2016.

[9] X. Huang, Y. Zhang, D. Li, and L. Han, "An optimal scheduling algorithm for hybrid EV charging scenario using consortium blockchains," *Future Generation Computer Systems*, vol. 91, pp. 555–562, 2019.

[10] Y. Lu, Q. Tang, and G. Wang, "Zebralancer: Private and anonymous crowdsourcing system atop open blockchain," *Proc. of the IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 853–865, 2018.

[11] Q. ShenTu and J. Yu, "A blind-mixing scheme for bitcoin based on an elliptic curve cryptography blind digital signature algorithm," *CoRR*, vol. abs/1510.05833, 2015. [Online]. Available: http://arxiv.org/abs/1510.05833

[12] C. Schnorr, "Efficient signature generation by smart cards," *J. Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[13] R. Gennaro, D. Leigh, R. Sundaram, and W. S. Yerazunis, "Batching schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices," *Proc. of International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pp. 276–292, 2004.

[14] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture." *Proc. of USENIX Security Symposium*, pp. 781–796, 2014.

[15] Y. Zhou, M. Wang, H. Hao, L. Johnson, and H. Wang, "Plug-in electric vehicle market penetration and incentives: a global review," *Mitigation and Adaptation Strategies for Global Change*, vol. 20, no. 5, pp. 777–795, 2015.

[16] Ganache. [Online]. Available: https://www.trufflesuite.com/ganache

[17] [Online]. Available: https://github.com/akosba/jsnark

[18] [Online]. Available: https://github.com/scipr-lab/libsnark

[19] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "Snarks for c: Verifying program executions succinctly and in zero knowledge," *Proc. of the annual cryptology conference*, pp. 90–108, 2013.

[20] [Online]. Available: github.com/maxilbert/ethereumj