P6V2G: A Privacy-Preserving V2G Scheme for Two-Way Payments and Reputation

Rebecca Schwerdt[*]
Karlsruhe Institute of Technology
Department of Informatics
Karlsruhe, Germany
rebecca.schwerdt@kit.edu

Matthias Nagel[†]
Karlsruhe Institute of Technology
Department of Informatics
Karlsruhe, Germany
matthias.nagel@kit.edu

Valerie Fetzer[‡]
Karlsruhe Institute of Technology
Department of Informatics
Karlsruhe, Germany
valerie.fetzer@kit.edu

Tobias Gräf
Karlsruhe Institute of Technology
Department of Informatics
Karlsruhe, Germany
tobias.graef@student.kit.edu

Andy Rupp[†,‡]
Karlsruhe Institute of Technology
Department of Informatics
Karlsruhe, Germany
andy.rupp@kit.edu

ABSTRACT

The number of electric vehicles (EVs) is steadily growing. This provides a promising opportunity for balancing the smart grid of the future, because vehicle-to-grid (V2G) systems can utilize the batteries of plugged-in EVs as much needed distributed energy storage: In times of high production and low demand the excess energy in the grid is stored in the EVs' batteries, while peaks in demand are mitigated by EVs feeding electricity back to the grid. But the data needed for managing individual V2G charging sessions as well as for billing and rewards is of a highly personal and therefore sensitive nature. This causes V2G systems to pose a significant threat to the privacy of their users. Existing cryptographic protocols for this scenario either do not offer adequate privacy protection or fail to provide key features necessary to obtain a practical system.

Based on the recent cryptographic toll collection framework P4TC, this work introduces a privacy-preserving but efficient V2G payment and reward system called P6V2G. Our system facilitates two-way transactions in a semi online and post-payment setting. It provides double-spending detection, an integrated reputation system, contingency traceability and blacklisting features, and is portable between EVs. The aforementioned properties are holistically captured within an established cryptographic security framework. In contrast to existing protocols, this formal model of a V2G payment and reward system allows us to assert all properties through a comprehensive formal proof.

CCS CONCEPTS

•**Security and privacy** →*Cryptography; Formal security models;*
•**Applied computing** →*Digital cash; Electronic funds transfer;*

KEYWORDS

Vehicle-to-grid, Location Privacy, Provable Security, Electronic Payments, Reward System, Double-Spending Detection.

[*]The author is supported by the German Research Foundation (DFG) as part of the Research Training Group GRK 2153: Energy Status Data-Informatics Methods for its Collection, Analysis and Exploitation.
[†]The author is supported by the German Federal Ministry of Education and Research via the Competence Center for Applied Security Technology (KASTEL).
[‡]The author is supported by DFG grant RU 1664/3-1.

1 INTRODUCTION

In an effort to counter global warming and fossil fuel depletion, energy transitions from fossil fuels to renewable energy sources are expedited on a global scale. This change from time-independent, fully controllable power plants to highly volatile, seasonal and distributed renewable power generation poses many challenges. Foremost among them is balancing generation and demand to maintain stability of the energy distribution system and provide uniform power quality. This requires increased storage capacities and load flexibility. One aspect of the energy transition—which has the potential to solve at least part of this problem—is the introduction of electric vehicles (EVs) as a means of private and public transportation. EVs do not require fossil fuels and can significantly reduce air and noise pollution, especially in urban areas. Most industrial nations have started to subsidize EVs as well as set concrete objectives on how many EVs they want to deploy by a certain year [25]. Target dates for complete bans on the sale of new cars with internal combustion engines are as early as 2020 in some countries—with many others planning to have phased them out by 2030 [11, 20, 39]. While EVs themselves pose a substantial and growing demand on the system, they also have the capacity to contribute to a solution. Privately owned vehicles remain parked for more than 95% of the time [35]. During this time the EV's battery can be utilized in a vehicle-to-grid (V2G) scheme, providing storage capacity and flexibility in the recharging process to help balance power generation and demand in the grid [35]: Surpluses from volatile renewable power generation can be stored in parked EV's batteries, which collectively provide substantial distributed energy storage. Conversely, energy stored in EVs can be used to mitigate peaks in demand and drops in generation. In contrast to the benefits such a system provides it can also pose a significant threat to the privacy of its users [26, 36]. The individual location data that may be gathered from identifying or linkable transactions is highly sensitive and can be maliciously exploited. To further participation in a voluntary V2G system, privacy guarantees can be just as important as low effort usability, as concerns arising from loss of personal information as well as any effort required from the user could discourage its use. Facilitating effortless payment for EV charging at a public charging spot (EVSE) as well as rewards for ancillary services during the charging session while keeping transactions privacy-preserving, unidentifying, and unlinkable poses a big challenge.

This paper introduces a novel and privacy-preserving V2G payment system called P6V2G. It was developed by adapting the recently proposed cryptographic toll collection system P4TC [32] to a V2G setting where multiple charging point operators provide publicly accessible EVSEs for EVs. P6V2G facilitates two-way transactions for EV charging, allowing both payment from the user to the operator for having their vehicle charged as well as rewards for providing the EV's battery as a buffer to the grid. P6V2G is a post-payment system where a user accumulates debt and rewards on a wallet and gets periodic, e.g., monthly, bills from their e-mobility operator (eMO). Therefore, P6V2G offers convenient low effort usability, leading more people to make their EV's batteries available for ancillary services. Our protocols are designed to be efficient even when performed on low performance hardware that is realistically present in EVs, EVSEs and wallets.[1] To further take realistic technology restrictions into account, EVSEs are not required to have permanent online capabilities; tasks can be conducted offline without loss of any functionality. An integrated reputation scheme records how reliable the user is, e.g., in leaving their EV plugged in as long as they predicted at the start of a charging session. This enables EVSEs to better plan the charging of vehicles and, e.g., offer cheaper/higher rewarded long term charging to reliable users. To ensure security, P6V2G provides a fraud detection mechanism that not only detects fraudulent behavior and identifies the misbehaving user, but outputs a publicly verifiable proof of guilt to avert the possibility of false accusations. With consent of a designated authority, misbehaving users can have their wallets traced (*contingency traceability*), blacklisted, or their debt recalculated—for instance if the user fails to present their wallet for billing at the end of a billing period. In addition, our system implements a mechanism to blacklist EVs (independently of blacklisted wallets), so that stolen vehicles can be traced or even detained at an EVSE. Since EVs and wallets are modeled as two different entities in P6V2G, they may be combined arbitrarily, resulting in a portable system. This is particularly useful for users with multiple cars as well as for renting or sharing EVs. The whole system is formalized in a comprehensive security model and validated by a rigorous proof asserting system security as well as user privacy.

1.1 Related Work

In this section we firstly give a summary of different payment and reward systems that were proposed for, or can be adapted to a V2G setting. This overview shows that while all of them have their respective merits, none so far provides solutions for *all* the challenges and restrictions posed by realistic V2G interactions. In a second part, we give a more detailed introduction of the two cryptographic systems (BBA+ [29] and P4TC [32]) that our own V2G system P6V2G is based on.

[1]Wallets might be realized on smart phones, but a separate device like a smart card might be more desirable as it can be left in the car. To handle the low performance of currently available smart cards, we would assume wallets to act as a secure store of the users secrets only and be able to outsource more costly computation to the EV.

Other (V2G) Payment and Reward Systems. There are several previously published payment and reward systems that can be deployed in a V2G scenario. Reward-only systems (without two-way payments) include TPP [44], and PRAC [43]. Both systems are not able to aggregate rewards. This results in linkability unless rewards are sufficiently uniform and redeemed individually. Furthermore both systems require permanent online capabilities and very few properties are proven in any formal way. For the remainder of this paragraph we will focus on payment systems that offer two-way transactions. A terse comparison of the discussed systems, the common options of (Chip-and-PIN) *Debit/Credit Card* and *Paper Cash*, and our system is depicted in table 1. In a survey by Han and Xiao [28] another overview of several V2G payment and reward systems, as well as a general summary of challenges and methods associated with privacy in the V2G environment can be found.

System	Location Privacy	Post-payment	Semi Online	Double-spending Detection	Reputation	Contingency Traceability	Blacklisting	Portable	Formal Proof
PRT [4]	✓	✗	✗	n/a[1]	✗	✓	✗	✓	✓/✗
PnC (ISO 15118) [21, 22]	✗	✓	✓	n/a[2]	✗	n/a[2]	✓	✗	✗
E-Cash [7]	✓	✗	✓	✓	✗	✗	✗	✓	✓/✗
Debit/Credit Card	✗	✗/✓	✓	n/a[2]	✗	n/a[2]	✓	✓	✗
Paper Cash	✓	✗	✓	n/a	✗	✗	✗	✓	✗
P6V2G	✓	✓	✓	✓	✓	✓	✓	✓	✓

[1] Double-spending detection is not applicable in online systems.
[2] Double-spending detection/traceability are not applicable in identifying systems.

Table 1: Comparison of Two-way Payment Systems

In 2012, Liu et al. [38] proposed a system designed to provide privacy at the "right" time (PRT), which has been further improved by Au et al. [4]. PRT facilitates two-way transactions between an EV and a single operator running multiple EVSEs. Further parties representing the operator include a grid management and a billing server. User's accounts are pre-paid and have to be topped up regularly. This excludes the possibility of problems arising from users who do not pay their bills (on time) but requires more effort on the user side and leads to users being stuck at EVSEs if they do not have sufficient balance on their account to charge an empty EV. Each EVSE needs a permanent connection to the operator's other parties as PRT is designed as an online system. In particular, all-time access to the central billing server is essential to prevent double-spending. Further assumptions are some read-only memory on the EV that can not be removed from the vehicle. PRT provides traceability that is facilitated by a judge authority and only possible with the user's consent. Via this traceability the system implements lost protection, dispute resolution and tracing of stolen vehicles. While neither [38] nor [4] provide any reputation system, [38] describes the possibility of facilitating a portable mode where the account is managed on some mobile device instead of the EV. This, however, excludes other features like correct tracing and the system still relies on a read-only memory that now has to be available on this mobile device. [4] includes two game-based proofs for cheating prevention and location privacy. Unfortunately these proofs only assert the impossibility of very specific attacks on the system and do not give any guarantees for attacks that were not explicitly considered.

The International Electrotechnical Commission's V2G standard ISO 15118 [21, 22] allows for contract based charging called "Plug & Charge" (PnC). In this system the EV shows a valid contract certificate from its eMO to the EVSE at the start of a charging session. At the end of each charging session, the EV signs the service detail record (SDR) of this session and sends it to the EVSE. In an online phase, the EVSE sends the SDR to the respective eMO who collects and uses them to bill the contract owner at the end of each billing period. Since the certificate includes the contract ID, this system is fully identifying. Furthermore—since the system is not portable between EVs—driver specific billing within EV fleets (like car sharing or company fleets) is not possible while simultaneously preserving the drivers privacy from the operator of this fleet: the corresponding eMO has to provide the fleet operator with a detailed account of all the EV's charging sessions so they can match them to specific drivers. The PnC system facilitates semi online post-payments and provides blacklisting. Please also note that only minimal security is guaranteed in ISO 15118. While off-the-shelf cryptographic methods like encryption and signatures are employed for EV authentication and to guarantee privacy from eavesdropping but non-involved parties, EVSEs are essentially assumed to be trustworthy and very little user security is offered. Furthermore, no resulting privacy or security guarantees are formalized, formally discussed or proven. In addition to the PnC payment method, the ISO 15118 allows for alternative payment methods (like P6V2G) to be added in future. Although the ISO 15118 only deals with direct communication between the EV and SDR—this inherently represents only a small part of our system, namely Part I of Debt Accumulation (cf. figs. 1 and 2)—much of our protocol can be integrated into this standard in an intuitive way. Details of how this might be done can be found in appendix B.

A privacy increasing modification of PnC was proposed under the name POPCORN [31]. POPCORN employs anonymous group signatures and three different trusted third parties to achieve anonymous one-way payments as long as no parties collude. It does not hide individual transactions from the eMO, is not portable, its proof-of-concept implementation is inefficient and no security or privacy proof is given. Some of POPCORN's privacy properties were later verified in ProVerif [9] by Fazouane et al. [24], who simultaneously identified several weaknesses.

Lastly, we consider the option of anonymous e-cash for V2G payments. One general drawback of e-cash is that it is a prepaid and primarily one-way system. Hence parties need to always have coins of exactly the right amount at hand for any transaction they might want to participate in—which limits either pricing flexibility or efficiency. When trying to employ e-cash for two-way payments in a V2G scenario there are several technical roadblocks. In standard offline e-cash (e.g., [14]), coins are not transferable without consulting the bank. When the user receives e-coins from the EVSE—which could be done by letting the EVSE participate in the spending protocol as the payer and the user as the payee—they first need to deposit those coins with the operator who originally issued them. In case this operator and the EVSEs collude they learn all the locations a user got rewards at. We also can not assume that a user receives freshly issued coins directly from the operator as rewards, as the corresponding e-cash withdrawal protocol is not anonymous; only spending a coin protects the privacy of the payer. Hence, the privacy guarantees that standard offline e-cash provides do not fit our needs. Transferable e-cash such as [7] does not achieve our goals either. Transferable e-cash schemes allow to anonymously transfer an e-coin multiple times between parties without consulting the bank. Thus, in our scenario an EVSE could transfer a coin received from the operator or another user to a user who is eligible for a reward. Unfortunately, there is an impossibility result regarding privacy that applies to this scenario. Canard and Gouget's work [15] implies, that if the parties representing the V2G payment infrastructure collude, payment and reward transactions of a user can be linked.

BBA+ and P4TC. At its core, our proposed scheme P6V2G facilitates a black-box accumulator (BBA+) [29]. This building block implements a personal wallet for anonymous, unlinkable point collection and redemption. It ensures that a wallet can only be used by its legitimate owner, protects against manipulation of the wallet's true value, provides double-spending detection and comes with a mix of game-based and simulation-based proofs for user security and privacy as well as system security. Hoffmann et al. [32] utilize BBA+ to build a system called P4TC which applies BBA+ to a toll collection scenario and offers anonymous, unlinkable tolling with post-payments. In order to fulfill all desired properties of a toll collection scenario, the authors augmented BBA+ with additional features like blacklisting, selective tracing of misbehaving users and recalculation of debt in case of a dispute. Moreover, P4TC comes with a full simulation-based security and privacy proof.

1.2 Our Contribution

This work transfers, adjusts and expands P4TC to fit the V2G scenario. The resulting system P6V2G allows for unlinkable and efficient V2G payments, rewards and reputation scores. It facilitates post-payment two-way transactions in a semi online and portable setting and includes double-spending detection as well as features for blacklisting (of both EVs and users), recalculation of debt and contingency traceability. To the best of our knowledge it is the first system offering all these properties simultaneously and for some properties it is the first to provide them at all. Furthermore, P6V2G is privacy-preserving and secure against malicious adversaries. This was asserted in a comprehensive formal model and proof within an established cryptographic framework, the Universal-Composability (UC) setting [16, 17].[2]

To achieve the above qualities we build upon the toll collection scheme P4TC [32]. Transferring this to the V2G scenario already provides several of the required functions and properties, e.g., privacy-preserving accumulation of debt, a verifiable fraud detection feature and blacklisting for wallets. Some requirements

[2]Although formal security proofs are not widely appreciated in application-oriented communities yet, we consider it vital for security (especially of complex and necessarily convoluted systems) that this changes in future.

of our system, like two-way payments, are not present in the toll collection scenario where users only have to pay toll and clear their wallet once, but do not collect rewards. Fortunately this important feature is still achieved by the employed black-box accumulator without any further modifications. There are, however, several characteristics of our setting with either much simpler or no equivalent aspects at all in P4TC, which call for novel solutions. First and most challenging among them is the presence of multiple operators instead of just one operating party in P4TC. This introduces considerable complexity as it yields additional potential for security threats in the system which our protocols have to preclude. Similar challenges arise from the user side of the system. Other than in P4TC—where a user is represented by their car's fixed on-board unit—our system is designed to be portable. Hence, the user's side is divided into two separate parties: user accounts, containing the users' wallets, and EVs represented by their communication controller. While user accounts have a similar role to on-board units in P4TC, EV communication controllers introduce additional complexity. Among other challenges, they require an additional but likewise privacy-preserving blacklisting mechanism to deal with stolen cars. Furthermore, P4TC does not include any reputation scheme for the reliability of users. While our formal model and proof largely follow the structure of [32], they are more complex due to the existence of arbitrarily many operators (instead of just one in P4TC) and EV communication controllers as entirely new parties.

2 SYSTEM DEFINITION

This section introduces the general type of V2G system we consider. In addition to the overall setting this includes details on participating parties, required functions and desired features. An overview can be found in fig. 1.

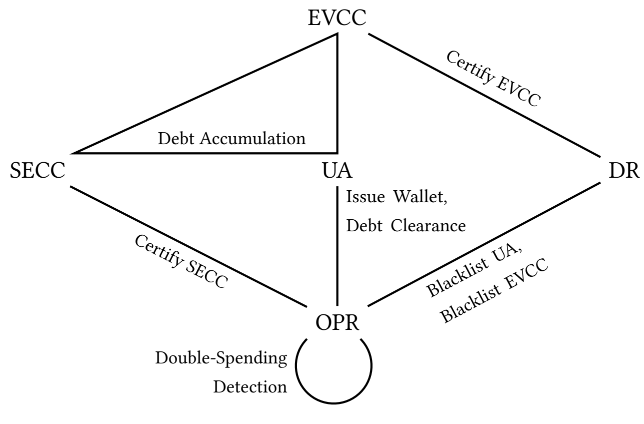


Figure 1: Overview of Parties and Their Interactions

We consider a setting in which multiple charging point operators each provide publicly accessible EVSEs for EVs. Users can charge their EVs at EVSEs as well as offer their EV's battery capacity to provide ancillary services to the grid. For these bidirectional services each user gets an accumulated bill from their own eMO at the end of each billing period. We assume that charging point operators and eMOs cooperate in a way that every user may use any of the EVSEs. While the necessary information to manage operator accounting is included in our system, the financial settlement between operators lies outside the scope of our system.

2.1 Parties

We now give a detailed description of the different parties and their roles. An *operator* (*OPR*) $\mathcal{O}$ is either the eMO of an EV, a charging point operator maintaining EVSEs, or possibly both. Each EVSE is represented by their *supply equipment communication controller* (*SECC*) $\mathcal{C}$ and is assigned to one specific (charging point) OPR but may be used by any (e-mobility) OPR's customers. It is assumed to have the capabilities for occasional database synchronization with its OPR but does not require any permanent internet connection to other parties. Different OPRs are assumed to occasionally synchronize their databases as well in order to catch double-spenders. An EV is represented by its *electric vehicle communication controller* (*EVCC*) $\mathcal{E}$. It needs to be registered to be able to participate in the protocols but is not associated with any other party, enabling multiple users to share the same car or one person using several different cars. A *user account* (*UA*) $\mathcal{A}$ is a low performance electronic device (we tend to think of it as a smart card) that is issued to a user upon registration with their OPR. It contains the user's wallet which is used to collect debt and reputation during charging sessions and is not bound to a specific EV. In addition to these mandatory parties we assume existence of some regulatory *dispute resolver* (*DR*) $\mathcal{D}$. This party mainly handles disputes and gives permission for any exceptional measures that either limit a party's access to the system (like being blacklisted) or detract from their privacy (e.g., recalculating the debt on an uncooperative UA). A *user* is a physical person using an EV, deciding where and when to charge it, owning a UA, picking their eMO and paying the bills. They do not, however, participate as a separate party in our protocols. The only input needed from a user are the charge targets at the start of a charging session. As we assume this choice to be indicated via the EVSE's human machine interface, it is formally made by the SECC in our protocols. Other than this input the user is represented by a UA and an EVCC for the duration of a charging session.

2.2 Functions and Features

The remainder of this section gives an overview of the functions and features we require of a V2G system. In contrast to more cumbersome prepaid and cash options, our aim is a post-payment system where the user is able to charge their vehicle over the course of a billing period and gets a combined bill for all their charging sessions afterwards. Basic functions of a post-payment V2G system include the accumulation of debt on a personal wallet and clearance of this wallet's debt. Both should be conducted in a privacy preserving way, keeping transactions anonymous as well as unlinkable. As an additional feature, we want users to be able to gain (or lose) reputation for, e.g., adhering to their predicted behavior. This way EVSEs might offer special tariffs to reliable users while not trusting the predictions of users who regularly end their charging sessions sooner than promised. All other features of the system deal with fraud and misbehaving users: As explained in section 4.1, not all kinds of fraud can be prevented in the given setting. But any fraud needs to at least be detected after the fact. To protect users from false accusations, this fraud detection mechanism has to provide a publicly verifiable proof of the user's guilt. To further protect the system from misbehaving users, we require the possibility to blacklist UAs as well as EVs (independently of each other) and a mechanism to recalculate the debt accumulated by an uncooperative UA or lost wallet with the consent of the DR. In extreme circumstances we also want the charging sessions of a UA to become traceable.

3 PROTOCOL DESCRIPTION

This section describes the main features and protocols of the P6V2G system on a high level. To make these explanations more accessible, we first give an intuition behind the basic cryptographic building blocks utilized therein. But readers with a cryptographic background may want to skip section 3.1. After going into detail on how wallets work in section 3.2, section 3.3 gives an overview of the different tasks P6V2G is composed of. Lastly, section 3.4 illustrates the efficiency of our system. For the complete protocol we refer the reader to appendix E.

3.1 Cryptographic Background

In order to easily understand the following protocol descriptions, familiarity with a few cryptographic primitives is essential.

Public Key Encryption. Encryption enables two parties to exchange messages such that no other party can read them. For public key encryption, the party who wants to receive messages first has to generate a *public key* and a *secret key.* The public key is then distributed to any prospective senders, while the secret key is kept secret. When a sender wants to send a confidential message, they *encrypt* the message with the public key of the recipient and send the resulting *ciphertext* instead. The recipient uses their secret key to *decrypt* the ciphertext and recover the message.

Digital Signatures. A *digital signature* is a means to verify the authenticity and integrity of a received message. Just like public key encryption, digital signatures use a public key infrastructure where two keys correspond to each party: a secret *signing key* and a public *verification key.* The sender uses their signing key to *sign* the message and sends the resulting *signature* along with the message to the receiver, who *verifies* the signature using the sender's public verification key.

Commitments. A *commitment scheme* enables one party to commit themselves to a value towards another (receiving) party. The receiver is sent a *commitment*, inside which the value itself is safely hidden. At a time of their choosing, the sending party can reveal the value to the receiver by *opening* the commitment. But the sender can only open the commitment to reveal the value they initially committed to and not alter the content in retrospect.

NIZK Proofs. NIZK stands for *Non-Interactive Zero-Knowledge.* As with any proof, a *NIZK proof* enables one party (called the prover) to convince a verifying party that a specific statement is true. The *zero-knowledge* property asserts that the receiver does not learn *why* the statement is true, only that it is. In fact, the receiver can infer nothing from the proof except the validity of the claim. *Non-interactive* means only one message has to be send from the prover to the verifier—namely the proof itself. No further (back or forth) interaction between the parties is necessary.

Pseudo-Random Functions. A *pseudo-random function*, intuitively, is a function that is indistinguishable from one with truly random output. At the same time it is efficiently computable for anyone with knowledge of the right input information.

Dynamic Cryptographic Accumulators. The central idea of *cryptographic accumulators* is the representation of a set of elements in a single *accumulation value* of fixed size. On one hand this accumulation value securely hides the elements within it. On the other hand it must be possible to efficiently prove that a given element is indeed contained in the accumulator. Cryptographic accumulators are called *dynamic* if elements can efficiently be added to and removed from the set over time.

3.2 Wallets

The concepts of *wallets* and *wallet states* are central for our P6V2G system. At the beginning of each billing period, the UA and corresponding OPR create a new wallet for the UA. This wallet is used for one billing period and then cleared and exchanged for a new one at the beginning of the next billing period.

A wallet is identified by its wallet ID λ and its current status described by a wallet state τ. This wallet state is altered with each transaction. It does not only store the current balance and reputation of the wallet, but rather all information needed to conduct a successful update after the next charging session. To be more formal, a (simplified[3]) wallet state τ is of the form

$$\tau := \big(\underbrace{s \,\big|\, b, r \,\big|\, x^{\text{next}} \,\big|\, \mathsf{cert}_C}_{\text{updatable part}} \;\big|\; \underbrace{\lambda, \mathsf{a}_\lambda, \mathsf{pk}_{O_\lambda}}_{\text{fixed part}}\big).$$

It consists of an *updatable part* $(s \,|\, b, r \,|\, x^{\text{next}} \,|\, \mathsf{cert}_C)$ that changes with each transaction and a *fixed part* $(\lambda, \mathsf{a}_\lambda, \mathsf{pk}_{O_\lambda})$ that is created once along with the wallet and stays the same for the whole life cycle of this wallet. The components of the updatable part are the last used serial number s, the current balance b and reputation r, the transaction counter x^{next} for the upcoming transaction and the certificate cert_C of the last SECC that updated the wallet (containing the SECC's attributes and OPR ID). The components that pertain to the fixed part are the wallet ID λ, the wallet attributes a_λ and the public key pk_{O_λ} of the OPR that issued this wallet.

At the beginning of a new billing period each UA $\mathcal{A}$ and its OPR O create a new wallet by performing the task Issue Wallet (see section 3.3). The wallet's initial state is

$$\tau := \big(s \,\big|\, 0, r \,\big|\, 1 \,\big|\, \mathsf{cert}_{C_O} \,\big|\, \lambda, \mathsf{a}_\lambda, \mathsf{pk}_O\big).$$

The certificate cert_{C_O} and public key pk_O correspond to the issuing OPR, who also assigns the initial reputation r (probably choosing the reputation the UA had accumulated on their last wallet) and wallet attributes a_λ (for details about the choice of wallet attributes see section 4.2). The wallet ID λ and serial number s are jointly

[3]For a cleaner presentation we omit some variables from the wallet state at this point. The left out variables are only relevant for cryptographic details like creating an anonymous NIZK proof for the validity of the wallet state which we do not explain in detail here. The complete wallet state can be found in appendix E.

computed randomness. Note, however, that λ is only known to the UA and the OPR does not learn it.

Each time the user charges their EV, the wallet is updated via the task Debt Accumulation (see section 3.3). Suppose at the beginning of a charging session with an SECC C the wallet's state was

$$\tau^* := \left(s^* \,\middle|\, b^*, r^* \,\middle|\, x \,\middle|\, \mathrm{cert}_{C^*} \,\middle|\, \lambda, \mathsf{a}_\lambda, \mathsf{pk}_{\mathcal{O}_\lambda}\right).$$

Then a fresh random serial number s is jointly computed by the UA and SECC and the transaction counter increased by one. Balance and reputation are updated by respectively adding the total cost (price minus rewards) p of the conducted charging and reputation gain d. This forms the new wallet state

$$\tau := \left(s \,\middle|\, b^* + p, r^* + d \,\middle|\, x + 1 \,\middle|\, \mathtt{cert}_C \,\middle|\, \lambda, \mathsf{a}_\lambda, \mathsf{pk}_{\mathcal{O}_\lambda}\right),$$

which the UA saves at the end of the charging process.

At the end of each billing period, the wallet is turned over to the UA's OPR for billing purposes. This is handled via the task Debt Clearance (see section 3.3). The OPR learns the final balance and reputation of the wallet and can use this information to bill the user accordingly. When this task is finished, the UA discards the wallet and initiates the task Issue Wallet again to get a new one.

So far we only described the wallet's state and life cycle, but not how they are used to guarantee several of our system's properties. On one hand, whenever a UA communicates with an SECC to update a wallet, they do not want to be identified or disclose the content of their wallet. On the other hand, the SECC has to be assured that the wallet they help updating is actually valid and does not, e.g., contain a balance or attributes that were illicitly manipulated by the user. This is achieved by utilizing commitments, digital signatures and NIZK proofs. Although the actual process is a little more complicated, it generally works like this: Instead of disclosing the old wallet state to the SECC, the UA proves the (hidden) state's validity by showing that the fixed part was created and signed by its OPR (and could therefore not have been altered since the wallet was issued) and that the updatable part was updated and signed by a properly certified SECC. Note that this previous SECC is not identified in the process. After checking these proofs, the SECC provides the UA with the necessary information to update the wallet in exactly the right way and signs the (still hidden) new wallet state so the UA can assure the next SECC of its validity. This way, both privacy and system security can be achieved.

3.3 Tasks

This section illustrates the different tasks (like registering parties or issuing wallets) our V2G system is composed of. While most of our tasks have similar counterparts in P4TC, some are entirely new (like Certify EVCC and Blacklist EVCC), and some tasks (like Debt Accumulation) contain a core part that is similar to the corresponding task in the toll collection setting but have been modified and extended to encompass the additional requirements of our scenario. Due to space restrictions, we describe the main task of Debt Accumulation in detail but sketch the other tasks on a high level only. Figure 1 depicts an overview of the tasks and the parties involved in them. The only tasks missing in fig. 1 are the registration task every party conducts on their own before participating in the system and the task Guilt Verification, which can be performed by anyone (even from outside the system) as our double-spending proofs are publicly verifiable.

Registration. Before participating in the system, every party has to register. This registration entails the generation of their respective cryptographic keys (used for encryption, digital signatures, etc.).

Certification. Each SECC has to get a certificate from its OPR. Main part of this certificate are a digital signature on the public key and the attributes of the SECC. It is renewed each billing period and used to ensure that the SECC is not corrupted. In addition to the actual SECCs, each OPR certifies themselves as an SECC as well.[4]

EVCCs have to also be certified, but by the DR. The core part of their certificate is a revocation value *rev*. This value is drawn randomly and stored by the DR. In case the EV containing the EVCC gets, for example, stolen, the DR adds the corresponding revocation value to a blacklist $\mathrm{bl}_{\mathrm{EVCC}}$ that each SECC has access to. Apart from the revocation value, the certificate contains a digital signature from the DR on the EVCC's revocation value, attributes and public key.

Issue Wallet. The task Issue Wallet is executed by a UA and its OPR at the start of each billing period. The UA obtains a new wallet λ, allowing it to take part in the task Debt Accumulation, and the OPR obtains a hidden trapdoor htd_λ, allowing the DR to blacklist the wallet and recover its debt if necessary.

Debt Accumulation. The Debt Accumulation task is special in that it requires the interaction of three different parties (all other require at most two). An SECC, UA and an EV represented by its EVCC interact to realize the charging process of the EV. Note that the task is only identifying for the SECC while the identities of the UA and EVCC remain hidden from the SECC. Since Debt Accumulation is the main task and in many ways representative of our system, we describe it in some detail.

The task can be split into two parts: The first part, prior to charging the EV, consists of the SECC interacting with the EVCC to determine its authenticity, battery characteristics and the user's charging choices. In the second part—reminiscent of P4TC's Debt Accumulation—the SECC and UA facilitate the billing after charging took place. Overviews of both parts can be found in figs. 2 and 3 respectively.

The first part starts with the SECC sending a list $\mathrm{bl}_{\mathrm{EVCC}}$ of blacklisted EVCC's revocation values to the participating EVCC. The EVCC computes a NIZK proof that shows its own revocation value is not contained in this list. Additionally, it sends its battery's characteristics β, its attributes $\mathsf{a}_{\mathcal{E}}$, and a NIZK proof that these attributes are correct. The SECC verifies both proofs and selects charge targets[5] based on this information. For bookkeeping purposes and dispute cases, information about the selected targets is sent to the EVCC, which marks the end of the first part.

The second part starts with the SECC authenticating itself to the UA by sending its certificate $\mathtt{cert}_C$. The UA checks this and

[4]They need an SECC certificate so wallets that are freshly issued by the OPR can not be distinguished from wallets updated by an SECC.

[5]The possibilities for target setting are intended to be communicated via the human machine interface of the SECC and selected by the (physical) user. Since we do not model the user as an explicit party, the targets are formally set by the SECC in this protocol.

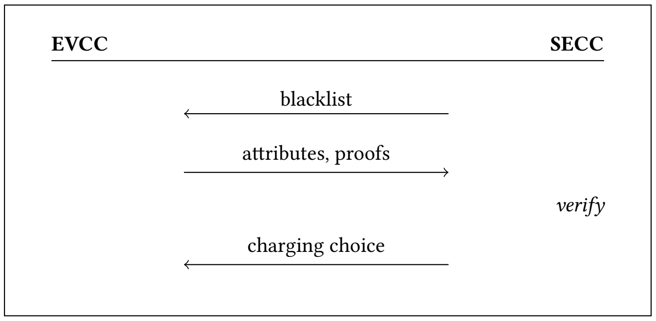


Figure 2: Overview of Task Debt Accumulation Part I

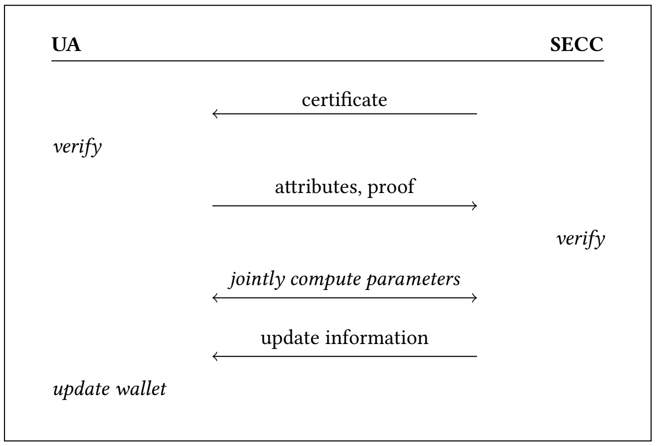


Figure 3: Overview of Task Debt Accumulation Part II

calculates the proper next fraud detection ID φ. Fraud detection IDs are similar to a transaction's serial number and essential for the detection of double-spending in the task Double-Spending Detection. They are computed by applying a pseudo-random function PRF to the UA's wallet ID λ and current transaction counter x, taken from the latest recorded wallet state τ^*, i.e., $\varphi := \mathsf{PRF}(\lambda, x)$. This assures that fraud detection IDs are random and unique if and only if no double-spending is committed. The UA sends φ over to the SECC, together with its wallet attributes a_λ and a NIZK proof that the wallet state which all this information was taken from was valid. The SECC verifies that the UA is not blacklisted by checking φ against the UA blacklist $\mathrm{bl}_{\mathrm{UA}}$ and verifies the NIZK proof. Note that since the UA has to stay anonymous, it can not simply send its wallet state over in the clear. Instead, it proves the wallet state's validity without revealing anything about the state itself. In particular, the so far accumulated debt b^* and reputation r^* stay secret, as this information could infringe upon the UA's anonymity and privacy. Then both parties jointly choose a fresh random serial number s using a Blum coin toss.[6] A double-spending tag t (see task Double-Spending Detection) is jointly computed as well. The SECC then sends all information that the UA needs to correctly update its wallet. This update information includes the price p and reputation gain d. The UA ensures that its new wallet state τ is valid and saves it. The SECC creates two database entries $\omega_{\mathrm{bl}} := (\varphi, p, d)$ and $\omega_{\mathrm{ds}} := (\varphi, t)$, where ω_{bl} is used again in the task Blacklist UA to recompute the overall debt and reputation of a blacklisted and uncooperative UA and ω_{ds} is used again in the task Double-Spending Detection to convict a UA of double-spending. We assume that each SECC periodically sends these database entries to the corresponding OPRs.

Debt Clearance. The task Debt Clearance is executed between a UA and its OPR at the end of each billing period. It is similar to the interaction of an SECC and a UA in Debt Accumulation. The main differences are that the UA is not anonymous in Debt Clearance, no new wallet state is created and the OPR learns the balance and reputation accumulated on the UAs wallet. The goal of this task is for both parties to calculate the debt the user owes its OPR, so that they can be billed out-of-band.

Double-Spending Detection. The task Debt Accumulation and Debt Clearance are always conducted with the assumption that the UA presents its most recent wallet and wallet state. Due to the semi online setting of the SECCs, this can not be enforced during the tasks themselves, but any misbehavior has to be detected afterwards. This kind of fraud that consists of the UA reusing and old wallet state is called *double-spending* (see also the paragraph on security in section 4.1). During the Debt Accumulation task, the SECC learns a unique fraud detection ID φ. If an OPR detects two entries ω_{ds}, ω^*_{ds} in his database (which the SECCs regularly send their collected data to) featuring the same fraud detection ID φ, double-spending occurred. The OPR can now calculate the identity of the UA by combining the two (otherwise non-identifying) database entries that contain the same fraud detection ID φ, because they will contain different double-spending tags t, t^*. The output of this algorithm is the identity of the UA along with a proof of guilt π that shows the UA is indeed guilty of double-spending. After learning the identity of the fraudulent UA, appropriate measures can be taken by the OPR out-of-band.

Guilt Verification. Whether a UA is guilty of double-spending can be verified using the Guilt Verification task. This algorithm may be run by any party. Essentially, the algorithm checks if a proof of guilt π asserts the guilt of a given UA's identity.

Blacklist UA. An OPR and the DR cooperate in the scope of the task Blacklist UA to provide the OPR with the data necessary to blacklist a certain UA. The OPR loads the set HTD_λ of hidden trapdoors from its storage and selects the entries htd_λ for all wallets of the UA that is to be blacklisted. The DR is the only party able to decrypt the ciphertexts contained in hidden trapdoors and is assumed to only cooperate in blacklisting tasks if the OPR has valid reason (such as proven fraud) to suspend the users privacy. The DR verifies that the htd_λ originate from the UA it wants to be blacklisted to ensure the OPR handed over the correct hidden trapdoors. Using htd_λ, the DR can calculate the (as of yet secret) wallet ID λ. Since the UA uses a pseudo-random function PRF to calculate the fraud detection IDs φ from the wallet ID λ in each run of Debt Accumulation, the DR can use its knowledge of λ to precompute all fraud detection IDs the UA might use in the current billing period. This yields fraud detection IDs $\{\varphi_0, \ldots, \varphi_{x_{\mathrm{bl}}}\}$ =

[6] A Blum coin toss is a two-party protocol that—using commitments—results in a mutually known truly random value as long as at least one party is honest.

$\{\mathsf{PRF}(\lambda, 0), \ldots, \mathsf{PRF}(\lambda, x_{\text{bl}})\}$ for each wallet which are collected in a set $\Phi_{\mathcal{A}}$. By adding $\Phi_{\mathcal{A}}$ to the blacklist bl_{UA}, this wallet is not able to partake in a successful run of Debt Accumulation again. The OPR can use the set $\Phi_{\mathcal{A}}$ to identify the corresponding entries of his database Ω_{bl} and recompute the current overall debt and reputation of the blacklisted UA.

Blacklist EVCC. With the help from the DR an OPR is able to revoke the access of a specific EVCC. To this end they run the task Blacklist EVCC. Note that we require our system to be able to blacklist EVCCs independently from any UAs they might conjointly be used with and that the Blacklist UA method—disclosing otherwise unpredictable fraud detection IDs—is not applicable to EVCCs. Hence the design of a separate and different mechanism was required to complete our P6V2G system. Using cryptographic accumulators we were able to achieve this in a way that makes adding EVCCs to the existing blacklist very efficient: The OPR sends the EVCCs public key $\mathsf{pk}_{\mathcal{E}}$ to the DR who assures itself that this EVCC should legitimately be banned from the system. The DR then uses $\mathsf{pk}_{\mathcal{E}}$ to look up the EVCC's revocation value *rev* and returns it to the OPR. This value can now be added to the blacklist bl_{EVCC} on which the cryptographic accumulator is computed during Debt Accumulation.

3.4 Efficiency

To obtain a practical real world system it is paramount protocols perform efficiently under the hardware restrictions of the given setting. In this section we show this to be the case for P6V2G. Instead of discussing each task of the system in turn, we present the runtime analysis for the task Debt Accumulation in detail. Not only is this by far the computationally most extensive task of our system, but it is (partly) conducted in the presence of a waiting user and therefore time-sensitive.

The efficiency of our protocols is heavily determined by the cost of cryptographic operations, i.e., creating commitments, signing messages, computing NIZK proofs and so on. Any other factors only contribute to a negligible fraction of the overall runtime and are therefore not considered in our analysis. The individual cryptographic operations are constructed from atomic operations in the underlying pairing group setting. More precisely, every cryptographic operation is a combination of $\mathbb{G}_1/\mathbb{G}_2$ exponentiations and pairing evaluations. This enables us to computationally determine a reliable upper bound on the runtime of our protocols: We calculate an upper bound on the number of respective atomic operations performed by each party, measure runtimes of single operations on the type of hardware parties would realistically employ, and combine those values to attain an overall runtime estimation. Another relevant aspect for the runtimes of our system are precomputations. From the detailed protocol description in appendix E it is apparent that many of the more complicated operations can easily be precomputed before the start of the actual protocol. We therefore distinguish between online and offline operations: Operations that can be conducted before the start of the actual protocol are denoted as *offline operations*. For Part I of Debt Accumulation, these may be performed at any point before the EV is plugged into an SECC; for Part II the parties compute them during the charging process itself, before the user returns to retrieve their car. All operations that depend on inputs of the protocol and can therefore only be computed at the proper time are denoted as *online operations*. By precomputing as much as possible the online runtime of the actual protocol can be significantly shortened and offline runtimes flexibly scheduled to convenient time slots. Table 2 shows the number of atomic operations in the Debt Accumulation task, differentiated by parties and offline and online computations.

	Party	offline			online		
		$\mathbb{G}_1$	$\mathbb{G}_2$	Pairing	$\mathbb{G}_1$	$\mathbb{G}_2$	Pairing
Part I	EVCC	91	82	0	$42+2v$	33	0
	SECC	v	0	0	0	0	$140+y$
Part II	UA	240	228	0	4	0	$z+9$
	SECC	4	0	$252+2j$	3	9	10

Here, $j := |a_\lambda|$, $y := |a_{\mathcal{E}}|$, $z := |a_C|$ and $v := |\text{bl}_{\text{EVCC}}|$.

Table 2: Upper Bound on $\mathbb{G}_1/\mathbb{G}_2$ Exponentiations and Pairing Evaluations in Debt Accumulation

For the verification of NIZK proofs, we assume batch verification techniques from [30] to significantly speed up the verification process. Also note, that the number of operations needed for verification was generously estimated. Therefore, the values in table 2 are upper bounds for the computational costs rather than exact figures.

To get a more tangible measurement of runtimes, we need to make assumptions about the hardware used in a realization of our system. For an EVCC we consider a conventional on-board unit, like the Savari MobiWAVE-1000 [42], to be a realistic choice. We therefore measured the runtime of $\mathbb{G}_1/\mathbb{G}_2$ exponentiations and pairing evaluations on the type of processor present in this on-board unit: An i.MX6 Dual-Core processor running at 800MHz with 1GB DDR3 RAM and 4GB eMMC Flash. The processor runs an embedded Linux and is ARM Cortex-A9 based (32-bit). For the bilinear group setting, we use the Barreto-Naehrig curves Fp254BNb and Fp254n2BNb [8, 34] (with 254-bit order) and the optimal Ate pairing [40]. The resulting benchmarks for single exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_2$ and for pairing evaluations are 6.07 ms, 15.52 ms and 32.19 ms respectively.

For use in SECCs the Sitara AM335x processor—based on an ARM Cortex-A8 (32-bit) running at up to 1GHz—has been suggested [33]. Since this processor has a similar core to the ones of the i.MX6, we use the same benchmarks to estimate runtimes for the SECC.[7] To realize UAs we propose to use either smart phones or a smart card plugged into the EVCC to utilize its computing power. Since the latter yields slower runtimes, we choose this option to obtain an upper bound on the realistic runtime. We therefore use the above benchmarks to calculate the runtimes for the EVCC and SECC, as well as the UA. Combining the figures from table 2 with the benchmarks we can now calculate upper bounds on the runtime of both parts of the task Debt Accumulation. The results are depicted in table 3.

[7]Note that they were indeed measured on a single thread only.

	Party	Party Runtimes		Total Runtime	
		offline	online	offline	online
Part I	EVCC	1825 ms	1374 ms	**2129 ms**	**5945 ms**
	SECC	304 ms	4571 ms		
Part II	UA	4995 ms	346 ms	**13389 ms**	**826 ms**
	SECC	8394 ms	480 ms		

Here, $j := |\mathsf{a}_{\lambda}| = 4$, $y := |\mathsf{a}_{\mathcal{E}}| = 1$, $z := |\mathsf{a}_{C}| = 1$ and $v := |\mathsf{bl}_{\mathrm{EVCC}}| = 50$.

Table 3: Upper Bound on Runtimes of Debt Accumulation

These results show that the only time-critical part—the online part of Part II—takes significantly less than one second to complete. All other parts do not compel the user to wait for completion and with roughly 2, 6 and 13 seconds they lie well within acceptable timeframes. Although these times are already sufficient to not impede a user's experience with our system, we note that they are still upper bounds in several ways: They were obtained assuming a very naive implementation without any computational optimizations, the number of group operations for proof verification was generously overestimated and runtimes were calculated using a single core only, when realistically multiple cores would be available. Hence, we assume real runtimes to be even better than the figures given in this section.

4 SECURITY AND PRIVACY

This section discusses the security and privacy properties of our P6V2G protocol. After reviewing the inherent limitations implied by the underlying scenario, we individually detail the security and privacy properties our protocol was proven to provide. Finally we sketch how the formal proof was conducted.

4.1 Inherent Privacy and Security Limitations

To assess the level of security and privacy our protocol provides, we first discuss which security limitations and privacy losses are inherent in the chosen setting and what levels of security and privacy an optimal solution might be able to yield.

Privacy. Although ideally we would like OPRs to learn nothing at all, there are things we can not hope to hide from them due to their control over the EVSEs. Information they will always obtain for a charging session are the ID and location of the EVSE, time and duration of charging, charge targets put in by the (anonymous) user, the EV's battery properties, and the actual SDR[8] of the session. A privacy-optimal solution would not give away anything more than those parameters.

Security. Some security limitations are imposed by post-payments paired with the semi online setting of SECCs. Facilitating this in a privacy preserving manner requires users to collect their debt on a wallet instead of trusting the OPR with their information. In this setting it is always possible for a user to reuse an old wallet state for upcoming charging sessions. Without instant synchronization the participating SECC has no way of knowing the old wallet state has already been used before. This kind of misbehavior is called *double-spending* and is a widely accepted drawback of semi online payment systems like bitcoin or e-cash—regardless of application. An optimal solution will detect this unavoidable fraud after the fact, identify the misbehaving user if required, and be able to recalculate the debt missing from the user's wallet. This feature is also required in case a fraudulent user refuses to present their wallet for billing or claims to have lost it, which can not be prevented either. Another inherent limitation is that a maliciously colluding user and SECC are always able to agree on updating the wallet in a way that is not reflective of any charging session that physically took place. Due to this gap between the digital communication captured by our model and the real, physical world, we had to omit the case of collusions between the user side (UA, EVCC) and operator side (OPR, SECC) from our security proof. We suggest mitigating the effects of corrupted SECCs by adding timestamps to their attributes so that keys of corrupted SECCs expire. Note also that the corrupt behavior of a colluding user and SECC (or OPR) has no more consequences for the security and privacy of *honest* users than if only the SECC/OPR were corrupted, which is covered by our proof.

4.2 Proven Privacy Properties

After listing some scenario specific impossibilities, let us discuss how P6V2G can yield nearly optimal privacy in this setting. Our proof of user privacy asserts that in addition to the necessary information listed in section 4.1, the only thing SECCs (and therefore OPRs) learn from each charging session are the wallet's, EV's and previous SECC's attributes as well as the certifying OPRs. Content of these attributes is a subject of choice in implementation and fully determines the level of privacy provided by the system. While empty attributes yield complete privacy, it would also be possible to, e.g., include the users/EVs/EVSEs identities in their respective attributes, and hence implement a fully identifying system. The important point is that this level of privacy is explicit and easy to check upon registration and charging.[9] Realistically, wallet's attributes might contain the billing period (e.g., month) they are valid for and information on whether the user has a business or private contract with his operator. EV's attributes might differentiate between standard cars and busses. An SECC's attributes should consist of the current billing period only, leaving all honest SECCs with the same attributes. Simple properties like these yield that any recorded transaction can only be attributed to a group of k different and indistinguishable pairs of UAs/EVs—with k being the number of UAs with the same wallet attributes and OPR multiplied by the number of EVs of the same type—but not to any specific user in this group.

Assuming this kind of attributes we formally prove the following privacy properties, even under participation of malicious OPRs/SECCs or UAs/EVs in the system:

4.3 Proven Security Properties

We consider static corruption (and collusion) of an arbitrary number of either UAs and EVCCs or of SECCs and OPRs (for an explanation

[8]Of course, this SDR should only contain details of the conducted charging, but no identifying information.

[9]Once the system is implemented with a specific form of attributes, checking the content of attributes is easily automated (and we would advise to do so), saving users the effort of checking the system's privacy level themselves.

(P1) Charging sessions of honest UAs/EVs are anonymous and unlinkable.
(P2) Tracing of an honest UA and recalculation of its accumulated debt are impossible without cooperation of the DR.
(P3) Tracing of a misbehaving UA has no more implications for the privacy of all other UAs/EVs than if the traced UA had never participated in the system at all.

on UAs/EVCCs colluding with SECCs/OPRs, see section 4.1). Any corrupted party may maliciously deviate from the protocol instead of just acting in an honest-but-curious manner.

Under this kind of corruption we prove the following security properties:

(S1) A UA can only use wallets that were legitimately issued to this UA.
(S2) If a UA commits double-spending (see section 4.1), it will be identified.
(S3) A UA which does not commit double-spending can not lie about or modify the debt or reputation on its wallet.
(S4) An OPR is able to blacklist a specific UA with cooperation from the DR. An SECC will know that a UA is blacklisted before its wallet is used for payment.
(S5) An OPR is able to recalculate a specific UA's debt with cooperation from the DR.
(S6) A UA that does not clear its wallet's debt will be detected.
(S7) Only registered EVs can take part in Debt Accumulation.
(S8) An OPR is able to blacklist specific EVs with cooperation from the DR. An SECC will know that an EV is blacklisted at the start of a charging session.
(S9) No party can lie about or modify their attributes.

4.4 Proof Sketch

We conducted a simulation-based proof following the real/ideal paradigm. For this type of proof the execution of the real world protocol is compared to the execution of an ideal model where all computation is done by an uncorruptible ideal functionality. The idea is for the ideal functionality to be designed in a way that it obviously guarantees the desired security and privacy properties. A proof of indistinguishability of the real world and ideal model yields that the protocol provides these properties as well. The advantage over game-based approaches is that all security properties and restrictions are explicit. However, the traditional notion of simulation-based security only captures security requirements in a standalone setting, where a single protocol instance runs in isolation. If multiple protocol instances run concurrently, this approach fails to guarantee security. The Universal Composability (UC) framework [16] on the other hand does not only capture parallel execution of multiple protocol instances, but also the use of protocols in an arbitrary context.

For our protocol π_{P6V2G}, we prove in the UC $\{\mathcal{F}_{\text{CRS}}, \overline{\mathcal{G}}_{\text{BB}}\}$-hybrid model (cp. [18, 19]) that it securely realizes the ideal V2G functionality $\mathcal{F}_{\text{V2G}}$ under common hardness assumptions for cryptographic building blocks. Further information on the ideal functionality is provided in appendix C while additional details of the proof can be found in appendix D.

5 CONCLUSION AND FUTURE WORK

Based on the cryptographic toll collection framework P4TC [32] we were able to develop the privacy-preserving V2G payment and reputation scheme P6V2G. Our system facilitates two-way transactions between EV users and EVSEs as well as reputation scores. In offering post-payment functionality—with, e.g., monthly billing—P6V2G is a highly convenient system with low effort usability. We achieve real world practicality by offering double-spending detection and guilt verification features, contingency traceability (with the consent of a designated authority) as well as blacklisting of EVs and users. All the systems functions are subject to explicit and formally proven privacy and security guarantees. Furthermore SECCs do not require permanent online capabilities, as P6V2G is designed to be operated in a semi online setting. Our runtime analysis demonstrates the P6V2G protocols to be efficient enough for the low performance hardware that is realistically present in the V2G setting. With EVs and users modeled as two separate entities, the system is fully portable between EVs and hence supports users with multiple cars as well as car sharing between multiple users.

Although the P6V2G system is a complete and practical V2G payment scheme, there are several directions where further expansion might prove beneficial. Modelling the EV and the actual charging process in more detail could yield additional features, e.g., an automated pricing check and continuous monitoring of the SECC's charging instructions. Another direction with potential for further development would be the financial settlement between charging point and e-mobility OPRs. At the moment each SECC needs to learn the user's eMO during a charging session to realistically provide OPRs with the means for financial settlement between them. Appending P6V2G with a cryptographic protocol for financial settlement between OPRs could remove the need for this information and thus further improve privacy.

ACKNOWLEDGMENTS

We would like to thank Max Hoffmann (Ruhr Universität Bochum) for providing us with benchmarks for single group exponentiations and pairing evaluations.

REFERENCES

[1] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. 2011. Optimal structure-preserving signatures in asymmetric bilinear groups. In *Annual Cryptology Conference*. Springer, 649–666.

[2] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Takeya Tango. 2014. Converting Cryptographic Schemes from Symmetric to Asymmetric Bilinear Groups. In *Advances in Cryptology – CRYPTO 2014, Part I (Lecture Notes in Computer Science)*, Juan A. Garay and Rosario Gennaro (Eds.), Vol. 8616. Springer, Heidelberg, 241–260.

[3] Masayuki Abe, Markulf Kohlweiss, Miyako Ohkubo, and Mehdi Tibouchi. 2015. Fully structure-preserving signatures and shrinking commitments. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 35–65.

[4] Man Ho Au, Joseph K Liu, Junbin Fang, Zoe L Jiang, Willy Susilo, and Jianying Zhou. 2014. A new payment system for enhancing location privacy of electric vehicles. *IEEE transactions on vehicular technology* 63, 1 (2014), 3–18.

[5] Man Ho Au, Patrick P Tsang, Willy Susilo, and Yi Mu. 2009. Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In *Cryptographers' Track at the RSA Conference*. Springer, 295–308.

[6] Man Ho Au, Patrick P. Tsang, Willy Susilo, and Yi Mu. 2009. Dynamic Universal Accumulators for DDH Groups and Their Application to Attribute-Based Anonymous Credential Systems. In *Topics in Cryptology – CT-RSA 2009 (Lecture Notes in Computer Science)*, Marc Fischlin (Ed.), Vol. 5473. Springer, Heidelberg, 295–308.

[7] Foteini Baldimtsi, Melissa Chase, Georg Fuchsbauer, and Markulf Kohlweiss. 2015. Anonymous Transferable E-Cash. In *Public-Key Cryptography – PKC 2015*, Jonathan Katz (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 101–124.

[8] Paulo S. L. M. Barreto and Michael Naehrig. 2006. Pairing-Friendly Elliptic Curves of Prime Order. In *SAC 2005Annual International Workshop on Selected Areas in Cryptography (Lecture Notes in Computer Science)*, Bart Preneel and Stafford Tavares (Eds.), Vol. 3897. Springer, Heidelberg, 319–331.

[9] Bruno Blanchet. 2016. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. *Foundations and Trends in Privacy and Security* 1, 1-2 (2016), 1–135.

[10] Dan Boneh and Xavier Boyen. 2004. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Advances in Cryptology - EUROCRYPT 2004*, Christian Cachin and Jan L. Camenisch (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 223–238.

[11] Isabella Burch and Jock Gilchrist. 2018. Survey of Global Activity to Phase Out Internal Combustion Engine Vehicles. Center for Climate Protection. (September 2018).

[12] Jan Camenisch, Rafik Chaabouni, and others. 2008. Efficient protocols for set membership and range proofs. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 234–252.

[13] Jan Camenisch, Kristiyan Haralambiev, Markulf Kohlweiss, Jorn Lapon, and Vincent Naessens. 2011. Structure Preserving CCA Secure Encryption and Applications. In *Advances in Cryptology – ASIACRYPT 2011 (Lecture Notes in Computer Science)*, Dong Hoon Lee and Xiaoyun Wang (Eds.), Vol. 7073. Springer, Heidelberg, 89–106.

[14] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. 2005. Compact E-Cash. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*. 302–321.

[15] Sébastien Canard and Aline Gouget. 2008. Anonymity in Transferable E-cash. In *Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings*. 207–223.

[16] Ran Canetti. 2001. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. 136–145.

[17] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. 2007. Universally Composable Security with Global Setup. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*. 61–85.

[18] Ran Canetti and Marc Fischlin. 2001. Universally Composable Commitments. In *Advances in Cryptology — CRYPTO 2001*, Joe Kilian (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 19–40.

[19] Ran Canetti, Daniel Shahaf, and Margarita Vald. 2016. Universally composable authentication and key-exchange with global PKI. In *IACR International Workshop on Public Key Cryptography*. Springer, 265–296.

[20] Department of Public Expenditure and Reform. 2019. Project Ireland 2040: National Development Plan 2018-2027. Government Declaration. (January 2019).

[21] DIN-Normenausschuss Automobiltechnik. 2015. *Road vehicles – Vehicle to grid communication interface – Part 1: General information and use-case definition*. Deutsches Institut für Normung e.V., Berlin. DIN EN ISO 15118-1.

[22] DIN-Normenausschuss Automobiltechnik. 2016. *Road vehicles – Vehicle to grid communication interface – Part 2: Network and application protocol requirements*. Deutsches Institut für Normung e.V., Berlin. DIN EN ISO 15118-2.

[23] Yevgeniy Dodis and Aleksandr Yampolskiy. 2005. A verifiable random function with short proofs and keys. In *International Workshop on Public Key Cryptography*. Springer, 416–431.

[24] Marouane Fazouane, Henning Kopp, Rens Wouter van der Heijden, Daniel Le Métayer, and Frank Kargl. 2015. Formal Verification of Privacy Properties in Electric Vehicle Charging. In *ESSoS (Lecture Notes in Computer Science)*, Vol. 8978. Springer, 17–33.

[25] Aoife Foley, Ian Winning, and Brian O Gallachoir. 2010. State-of-the-art in electric vehicle charging infrastructure. In *2010 IEEE Vehicle Power and Propulsion Conference (VPPC)*. IEEE, 1–6.

[26] Philippe Golle and Kurt Partridge. 2009. On the Anonymity of Home/Work Location Pairs. In *Pervasive Computing*, Hideyuki Tokuda, Michael Beigl, Adrian Friday, A. J. Bernheim Brush, and Yoshito Tobe (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 390–397.

[27] Jens Groth and Amit Sahai. 2008. Efficient non-interactive proof systems for bilinear groups. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 415–432.

[28] Wenlin Han and Yang Xiao. 2016. Privacy preservation for V2G networks in smart grid: A survey. *Computer Communications* 91 (2016), 17–28.

[29] Gunnar Hartung, Max Hoffmann andCryptographic AccumulatorsNg Matthias Nagel, and Andy Rupp. 2017. BBA+: Improving the Security and Applicability of Privacy-Preserving Point Collection. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. 1925–1942.

[30] Gottfried Herold, Max Hoffmann, Michael Klooß, Carla Ràfols, and Andy Rupp. 2017. New Techniques for Structural Batch Verification in Bilinear Groups with Applications to Groth-Sahai Proofs. In *ACM Conference on Computer and Communications Security*. 1547–1564.

[31] Christina Höfer, Jonathan Petit, Robert Karl Schmidt, and Frank Kargl. 2013. POPCORN: privacy-preserving charging for emobility. In *CyCAR@CCS*. ACM, 37–48.

[32] Max Hoffmann, Valerie Fetzer, Matthias Nagel, Andy Rupp, and Rebecca Schwerdt. 2018. P4TC—Provably-Secure yet Practical Privacy-Preserving Toll Collection. Cryptology ePrint Archive, Report 2018/1106. (2018). https://eprint.iacr.org/2018/1106.

[33] Texas Instruments. 2017. TI Designs: TIDEP-0087. Human Machine Interface (HMI) for EV Charging Infrastructure Reference Design. http://www.ti.com/lit/ug/tidude7/tidude7.pdf. (2017). [Online; accessed 09-January-2019].

[34] Yuto Kawahara, Tetsutaro Kobayashi, Michael Scott, and Akihiro Kato. 2016. *Barreto-Naehrig Curves*. Internet Draft. Internet Engineering Task Force. Work in Progress.

[35] Willett Kempton and Steven E. Letendre. 1997. Electric vehicles as a new power source for electric utilities. *Transportation Research Part D: Transport and Environment* 2, 3 (1997), 157 – 175.

[36] John Krumm. 2007. Inference Attacks on Location Tracks. In *Pervasive Computing*, Anthony LaMarca, Marc Langheinrich, and Khai N. Truong (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 127–143.

[37] Zi Lin and Nicholas Hopper. 2010. Jack: Scalable accumulator-based nymble system. *Proceedings of the ACM Conference on Computer and Communications Security* (01 2010), 53–62.

[38] Joseph K Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. 2012. Enhancing location privacy for electric vehicles (at the right time). In *European Symposium on Research in Computer Security*. Springer, 397–414.

[39] Stefan Löfven. 2019. Regeringsförklaring. Government Declaration. (January 2019).

[40] Dustin Moody, Rene C. Peralta, Ray A. Perlner, Andrew R. Regenscheid, Allen L. Roginsky, and Lidong Chen. 2015. Report on Pairing-based Cryptography. In *Journal of Research of the National Institute of Standards and Technology*, Vol. 120. National Insititute of Standards and Technology, Gaithersburg, MD, USA, 11–27.

[41] Lan Nguyen. 2005. Accumulators from Bilinear Pairings and Applications. In *Topics in Cryptology – CT-RSA 2005 (Lecture Notes in Computer Science)*, Alfred Menezes (Ed.), Vol. 3376. Springer, Heidelberg, 275–292.

[42] Savari.net. 2017. MobiWAVE On-Board-Unit (OBU). http://savari.net/wp-content/uploads/2017/05/MW-1000_April2017.pdf. (2017). [Online; accessed 05-February-2018].

[43] Zhiguo Wan, Wen-Tao Zhu, and Guilin Wang. 2016. PRAC: Efficient privacy protection for vehicle-to-grid communications in the smart grid. *Computers & Security* 62 (2016), 246–256.

[44] Huaqun Wang, Bo Qin, Qianhong Wu, Li Xu, and Josep Domingo-Ferrer. 2015. TPP: Traceable privacy-preserving communication and precise reward for vehicle-to-grid networks in smart grids. *IEEE Transactions on Information Forensics and Security* 10, 11 (2015), 2340–2351.

A NOTATION

Tables 4 and 5 provide an alphabetically ordered look-up table for the notation used in our protocol and ideal model.

Notation	Description
$\mathbb{1}_{\mathbb{G}}$	Neutral element of a group $\mathbb{G}$
$\mathcal{A}$	A UA
ACC	Cryptographic accumulator scheme
$\mathsf{a}_{\mathcal{P}}$	Attributes of a party $\mathcal{P}$
$base$	Base used to represent a wallet ID
b	Balance, i.e., amount of debt on wallet
b^{bill}	Debt of a user at the end of a billing period
bl_{EVCC}	List of revocation values, used to blacklist EVCCs
bl_{UA}	List of fraud detection IDs, used to blacklist UAs
β	Battery attributes of an EV
C	Commitment scheme
$\mathcal{C}$	A SECC
$\mathcal{C}_{\mathcal{O}}$	The SECC played by the OPR $\mathcal{O}$ itself
$\mathsf{c}_{(\mathcal{P})}$	Commitment (made by party $\mathcal{P}$)
$\mathtt{cert}_{\mathcal{C}}$	$= (\mathsf{pk}^{\text{sig}}_{\mathcal{C}}, \mathsf{a}_{\mathcal{C}}, \mathsf{pk}^{\text{cert}}_{\mathcal{O}_{\mathcal{C}}}, \sigma^{\text{cert}}_{\mathcal{O}_{\mathcal{C}}})$ Certificate of a SECC issued by $\mathcal{O}_{\mathcal{C}}$
$\mathtt{cert}_{\mathcal{E}}$	$= (rev, \mathsf{a}_{\mathcal{E}}, \mathsf{c}, \mathsf{d}, \sigma)$ Certificate of an EVCC
CRS	Common reference string
$\mathcal{D}$	The DR
d	Reputation gained during the charging session
$\mathsf{d}_{(\mathcal{P})}$	Decommitment (made by party $\mathcal{P}$)
E	Encryption scheme
$\mathcal{E}$	An EVCC
e	Encryption of secrets under the key of the DR
$\mathcal{F}$	Ideal functionality
$\mathcal{F}_{\text{CRS}}$	Ideal CRS functionality
f_{Φ}	Mapping of wallet IDs and transaction counters to fraud detection IDs
f_{Π}	Mapping of double-spending proofs to UA IDs
f_{rev}	Mapping of EVCC IDs to their revocation value rev
Φ	Space of fraud detection IDs
φ	Fraud detection ID
$\mathbb{G}$	Group
g	Generator of group $\mathbb{G}$
$\mathcal{G}_{\text{BB}}$	Ideal bulletin board functionality
HTD_{λ}	Set of hidden trapdoors
htd_{λ}	Hidden trapdoor to blacklist wallet λ
$\text{id}_{\mathcal{P}}$	ID of a party $\mathcal{P}$
$\mathsf{L}^{(i)}$	Language used in proof scheme $\mathsf{P}i$
λ	Wallet ID
λ_i	i^{th} digit of the base-$base$ representation of λ
λ', λ''	Shares of wallet IDs, created by parties to jointly chose a wallet ID
$\mathtt{m}_{(\mathcal{P})}$	Message content (from party $\mathcal{P}$)
μ	Choice of charging program

Table 4: Notation Used in this Paper

Notation	Description
$\mathcal{O}$	An OPR
$\mathcal{O}_{\mathcal{P}}$	The OPR of a party $\mathcal{P}$ (or wallet λ)
out	Output
P	Proof scheme
$\mathcal{P}$	A party
p	Price of a charging session
$\mathsf{pk}^{\text{pur}}_{\mathcal{P}}$	Public key of party $\mathcal{P}$ for purpose pur
PRF	Pseudo-random function
π	NIZK proof
r	Reputation accumulated on a wallet
r^{bill}	Reputation on a wallet at the end of a billing period
$rand$	Randomness used to encrypt secrets for the DR
rev	Revocation value of an an EVCC
S	Signature scheme
$\mathcal{S}$	Simulator
S	Space of serial numbers
s	Serial number of a transaction
s', s''	Shares of serial numbers, created by parties to jointly chose a serial number
$\mathsf{sk}^{\text{pur}}_{\mathcal{P}}$	Secret key of party $\mathcal{P}$ for purpose pur
$stmnt$	Statement
$\sigma_{(\mathcal{P})}$	Signature (of party $\mathcal{P}$)
t	Double-spending tag
$\mathtt{td}$	Trapdoor
$TRDB$	Database of transaction records $trdb$
$trdb$	$= (s^{\text{prev}}, s, \varphi, x, \lambda, \text{id}_{\mathcal{A}}, \text{id}_{\mathcal{C}}, b, r, p, d)$ Transaction record in the transaction database
τ	$= \big(s \big\vert b, r \big\vert x^{\text{next}}, u_1^{\text{next}} \big\vert (\mathsf{c}, \mathsf{d}, \sigma)_{\mathcal{C}}, \mathtt{cert}_{\mathcal{C}} \big\vert \lambda, \mathsf{a}_{\lambda}, (\mathsf{c}, \mathsf{d}, \sigma)_{\mathcal{O}_{\lambda}}, \mathsf{pk}^{\text{sig}}_{\mathcal{O}_{\lambda}}\big)$ Wallet state at the end of the transaction with serial number s
u_i	Randomness to mask secrets in double-spending tags
v	Accumulator value
w	Witness needed to prove a revocation value is not blacklisted
wit	Witness for a statement $stmnt$
wl	Whitelist of UA public keys
Ω_{bl}	Database of blacklisting information ω_{bl}
ω_{bl}	$= (\varphi, p, d)$ Database entry for debt and reputation recalculation
Ω_{ds}	Database of double-spending information ω_{ds}
ω_{ds}	$= (\varphi, t, u_2)$ Database entry used to detect double-spending
x	Transaction counter of a wallet
x_{bl}	Upper bound of transactions in one billing period
$\mathcal{Z}$	Environment

Table 5: Continuation of Notation Used in this Paper

B INTEGRATION OF P6V2G INTO ISO 15118

In addition to the PnC payment method discussed in section 1.1, the international standard ISO 15118 [21, 22] allows for alternative payment methods to be added in future. In this section we will discuss the integration of our P6V2G system into this standard, following its communication structure of functional groups A-H. The elementary use cases needed for P6V2G within those functional groups follow the examples of use cases defined for a similar purpose in the post-payment method PnC.

As ISO 15118 only deals with communication between EVCC and SECC, the only part of our system it inherently applies to is Debt Accumulation Part I—which would naturally be implemented as an elementary use case of functional group D "Identification, Authentication and Authorisation". It is, however, possible to integrate the complete tasks of Certify EVCC, Issue Wallet, Debt Accumulation and Debt Clearance into the ISO 15118 in an intuitive way. Note that apart from EVCCs and SECCs, these tasks are conducted with participation of the DR, OPRs and UAs, so there is a party mismatch to be considered.

Certify EVCC. This task is executed by the EVCC and DR to obtain a certificate containing vehicle attributes and a revocation value for blacklisting. It is conducted only once and has to take place prior to the first charging session that is to be payed for with a P6V2G contract. We therefore propose this to be done at the original equipment manufacturer (OEM), who should be in direct contact with the DR to have all their vehicles certified. In this case it would behave like the bootstrap or OEM provisioning certificate, but without the need to be replaced by a contract certificate later on. Alternatively—to remove the need for trusting the OEM with anything more than provisioning certification—we propose to conduct the protocol for this task within an elementary use case C3 "EVCC Certificate Installation" in functional group C "Certificate Handling", resembling the current example of C2 "Certificate Installation". This elementary use case C3 would only apply to the situation where the EVCC has not been certified before and is connected to an SECC controlled by the DR.

Debt Accumulation. This task is the most complicated to integrate into ISO 15118, because parts of it are conducted prior to and other parts after charging. Our system's portability, with UAs separated from EVCCs, further complicates the matter. While ISO 15118 provides the possibility for external identification methods like our UA in functional group D "Identification, Authentication and Authorisation", it does not allow for keeping this identification method physically fixed to interact with again after charging. By plugging the UA (as a smart card) directly into the EVCC, however, and having the EVCC relay (encrypted) messages between SECC and UA, this shortcoming could be bypassed without loosing portability of our system. Hence the UA could be construed as part of the EVCC for the duration of the charging session. For Part I of the Debt Accumulation protocol (communication between EVCC and SECC), we propose an elementary use case D5 "Authorization of EVCC performed at the EVSE" in functional group D "Identification, Authentication and Authorisation" following the example of D1 "Authorization using Contract Certificates performed at the EVSE". Part II of the Debt Accumulation protocol (communication between UA and SECC) needs to be split up between two functional groups: Authorization of the UA[10] should be conducted within an elementary use case D6 "Authorization of UA performed at the EVSE", again resembling D1. The remainder of the protocol would then form an elementary use case G3 "Wallet Update" within functional group G "Value-added Services" resembling the elementary use case G2 "Charging Details" which is used for creating the SDR for PnC.

Issue Wallet and Debt Clearance. In addition to an external method (e.g., direct communication between the UA smart card and the user's eMO via an NFC reader at the users home computer), the tasks Issue Wallet and Debt Clearance could be conducted in the scope of a charging session.[11] Even though they are more complicated and provide more functionality, wallets in P6V2G fulfill a similar role to contract certificates in PnC. We therefore propose to realize the Issue Wallet and Debt Clearance protocols as use cases C4 "Issue Wallet" and C5 "Debt Clearance" within functional group C "Certificate Handling"—resembling the use case C1 "Certificate Update" for PnC. C4 and C5 should be conducted automatically within the first charging session of a new billing period.

C IDEAL FUNCTIONALITY

For a simulation-based proof following the real/ideal paradigm we need an ideal model to compare our protocol to. This type of ideal model comes in the form an ideal functionality $\mathcal{F}$, which can be thought of as an imaginary trusted third party incorruptibly performing all functions we desire in the given scenario. If a protocol π is proven to be indistinguishable from the ideal functionality $\mathcal{F}$, it necessarily yields the same properties and guarantees. Before explaining the extensive and more complicated ideal V2G payment and reputation functionality $\mathcal{F}_{\text{V2G}}$, we want to illustrate the workings of an ideal functionality with a small example.

Example. Imagine a scenario where two mutually distrusting parties $\mathcal{P}_1$ and $\mathcal{P}_2$ want to jointly compute the result of a function f. Each party $\mathcal{P}_i$ is allowed to choose a part x_i of the input that f is supposed to be evaluated on. While both parties want to learn the output $f(x_1, x_2)$ and be sure that what they learn is actually the correct result (security), neither wants the other to learn anything more about their respective input than the result itself discloses (privacy). The ideal functionality $\mathcal{F}_f$ in this context would know the function f, get the inputs x_1 and x_2 from $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively, compute $f(x_1, x_2)$, and send this result back to both parties.

$\mathcal{F}_f$ guarantees security since it is uncorruptible and f is fixed within its description. No party can influence the computation of the result $f(x_1, x_2)$ in any other way than by choosing their own input. Due to its outputs, $\mathcal{F}_f$ also yields obvious privacy. All any party learns in an execution of $\mathcal{F}_f$ is the result $f(x_1, x_2)$ as there are no other outputs the ideal functionality makes (note that in the ideal model no party can read, intercept or otherwise meddle with message between the ideal functionality and another party). In particular, no further information about the other parties input is obtained.

[10]This includes everything up to the blacklisting check $\varphi \in \text{bl}_{\text{UA}}$ in fig. 32.
[11]Please note, that this requires an online connection from the participating SECC to the eMO.

This example illustrates three main ways an ideal functionality is able to yield security and privacy guarantees: Firstly, any information that is needed, but that no party should be able to choose, lie about or unduly influence (e.g., the function f) is stored within the ideal functionality. Secondly, All computations are correctly performed by the ideal functionality, not by any party. And thirdly, the ideal functionality only outputs information to a party that this specific party is supposed to learn and nothing else.

In the rest of this section we illustrate how the above principles can be applied to get an ideal functionality $\mathcal{F}_{\text{V2G}}$ for the scenario described in section 2.

We divide the description of $\mathcal{F}_{\text{V2G}}$ up into two parts. The first part consists of the state of the ideal functionality (containing all information parties should not be able to lie about), the second details its behavior in all the different tasks of a V2G payment system. An overview can be found in fig. 4 and we will explain both in more detail below.

Functionality $\mathcal{F}_{\text{V2G}}$

I. State

The ideal functionality $\mathcal{F}_{\text{V2G}}$ records:

- (Partial) Mapping a that maps a parties ID[a] $\text{id}_{\mathcal{P}}$ to their respective attributes $\mathsf{a}_{\mathcal{P}}$.
- (Partial) Mapping O that maps a UA's or SECC's ID $\text{id}_{\mathcal{P}}$ to their respective OPR's public key $O_{\mathcal{P}}$.
- (Partial) Mapping f_{rev} that maps an EVCCs ID $\text{id}_{\mathcal{E}}$ to a revocation value.
- Transaction database $TRDB = \{trdb\}$ with entries

$$trdb = (s^{\text{prev}},\ s,\ \varphi,\ x,\ \lambda,\ \text{id}_{\mathcal{A}},\ \text{id}_{C},\ b,\ r,\ p,\ d).$$

- (Partial) Mapping f_{Φ} that maps a wallet ID λ and a counter x to a fraud detection ID φ.
- (Partial) Mapping f_{Π} that maps a proof π to a UAs ID $\text{id}_{\mathcal{A}}$.

[a]For UAs, potentially different attributes are assigned to each wallet rather than the UA.

II. Behavior – Tasks

- Register DR
- Register OPR
- Register SECC
- Register EVCC
- Register UA
- Certify SECC
- Certify EVCC
- Issue Wallet
- Debt Accumulation
- Debt Clearance
- Double-Spending Detection
- Guilt Verification
- Blacklist UA
- Blacklist EVCC

Figure 4: The Ideal Functionality $\mathcal{F}_{\text{V2G}}$

C.1 State

To assure the right information (e.g., a parties' attributes or previous amount of debt) is used in computations, the ideal functionality $\mathcal{F}_{\text{V2G}}$ stores everything parties should not be able to freely choose or lie about during the tasks of this system. It keeps track of all conducted charging sessions by maintaining a comprehensive database *TRDB* and several (partial) mappings for parties' attributes, OPRs, revocation values, fraud detection IDs and double-spending proofs (cp. fig. 4). This global state is used and amended when conducting individual tasks with participating parties.

Information on all conducted charging sessions is kept in the database *TRDB*. Its entries

$$trdb = (s^{\text{prev}}, s, \varphi, x, \lambda, \text{id}_{\mathcal{A}}, \text{id}_{C}, b, r, p, d)$$

are uniquely identified by a serial number s. Another serial number s^{prev} links back to the logically previous charging session that was payed for with the same wallet. Each database entry contains all information pertinent to one charging session. It notes the identities $\text{id}_{\mathcal{A}}$ and id_{C} of participating parties as well as the wallet ID λ. Balance b and reputation r give the state of the wallet λ after charging, price p and reputation gain d indicate by how much these values changed during this charging session. The counter x indicates the number of charging sessions conducted to arrive at the current state of the wallet. Lastly *trdb* contains a fraud detection ID φ which is a random number assigned to each pair (λ, x) of wallet ID and counter and is used to detect double spending by misbehaving users. This value is (slightly redundantly) stored in the mapping f_{Φ} as well: For every entry *trdb* with values λ, x and φ the equation $\varphi = f_{\Phi}(\lambda, x)$ holds.

In addition to this basic information, $\mathcal{F}_{\text{V2G}}$ keeps track of some information that is only relevant to feature tasks: The mapping f_{Π} stores proofs π that indicate double spending by a UA $\text{id}_{\mathcal{A}} = f_{\Pi}(\pi)$ has been detected. The mapping f_{rev} stores a revocation value for every registered EVCC that can be used to blacklist this car, e.g., because is was reported stolen.

C.2 Behavior

An overview of the tasks provided by our system was given in section 3. The ideal functionality $\mathcal{F}_{\text{V2G}}$ offers the same tasks with the same interfaces (otherwise the protocols could not be indistinguishable from the ideal world): A number of setup tasks, three basic tasks (Issue Wallet, Debt Accumulation and Debt Clearance), and four additional feature tasks. In this section we explain the ideal version of them by mostly summarizing what $\mathcal{F}_{\text{V2G}}$ does, but to better illustrate the inner workings of the ideal functionality, the central task of Debt Accumulation is shown in more detail.

Setup Tasks. Setup tasks include registration and certification of parties. Upon registration, the ideal functionality checks a party has not been registered before and supplies it with the necessary keys to participate in the system. Registered EVCCs and SECCs have to be certified as well. Certification of an EVCC by the DR entails a check if this EVCC has already been certified or even blacklisted before and otherwise assigns a revocation value to the EVCCs ID to enable blacklisting in the future. Each SECC is certified by the OPR maintaining it to provide it with the credentials necessary to participate in charging sessions and update balance and reputation of wallets in the name of this OPR.

Issue Wallet. In the task Issue Wallet $\mathcal{F}_{\text{V2G}}$ creates a new wallet for a UA and OPR. It checks the UA against the OPRs whitelist and initializes a new wallet λ with an entry

$$trdb = (\bot, s, \varphi, x, \lambda, \text{id}_{\mathcal{A}}, \text{id}_{O}, 0, r, 0, r).$$

The mappings a and f_Φ are appended as well. Attributes and initial reputation are provided by the OPR. The OPR only receives the serial number s as output, the UA obtains its new wallets attributes and reputation and the OPR's ID as well.

Debt Accumulation. The ideal functionality's behavior during the Debt Accumulation task can be found in fig. 5.[12] Again, $\mathcal{F}_{\text{V2G}}$ checks blacklisting and looks up the EVCC's attributes before facilitating the exchange of the EV's battery attributes β and the charging choice μ between the SECC and EVCC. In the second part, $\mathcal{F}_{\text{V2G}}$ looks up the transaction $trdb^{\text{prev}}$ corresponding to the wallet state s^{prev} that the UA indicates they want to continue from.[13] It correctly computes s, φ, x, b, and r for the new transaction entry

$$trdb = (s^{\text{prev}}, s, \varphi, x, \lambda, \text{id}_{\mathcal{A}}, \text{id}_{C}, b, r, p, d),$$

which is inserted into *TRDB*. Price p and reputation gain d are provided by the SECC.

As output the SECC receives the transactions serial number s, fraud detection ID φ and the attributes and OPR IDs of the wallet and the previous SECC. The UA on the other hand learns the serial number s, its wallet's current balance b and reputation r as well as the price p and reputation gain d of this charging session.

Debt Clearance. The Debt Clearance task in the ideal model works very much like the second part of Debt Accumulation. Differences are that the price of the new transaction entry is set to $p = -b^{\text{bill}}$ and there is no reputation gain. The OPR additionally learns the UA's ID and the final balance b^{bill} and reputation r^{bill} of the cleared wallet. The UA in turn only learns this balance and reputation, but does not get any further information. This ensures the UA can not use it's wallet again (without committing double-spending) after it has been cleared.

Feature Tasks. The most important feature task is Double-Spending Detection. On request of an OPR the ideal functionality checks its database *TRDB* for double-spending, i.e., for two separate entries with the fraud detection ID φ. If such entries exist, the corresponding UAs ID and a proof of its guilt are appended to f_Π as well as output to the OPR. If any party wants to check the validity of such a proof, they can do so by means of the task Guilt Verification. On input $(\text{id}_{\mathcal{A}}, \pi)$ the ideal functionality checks if it previously recorded $f_\Pi(\pi) = \text{id}_{\mathcal{A}}$ and returns the result.

The other two feature tasks pertain to blacklisting and are joint tasks of an OPR and the DR. In both cases the DR only has to give its permission while the OPR has to supply the ID of the party that is supposed to be blacklisted. For EVCCs the ideal functionality returns its recorded revocation value, for UAs all used and upcoming fraud detection IDs are returned as well as the current balance and reputation of the UAs wallet. This information can then be added to the respective blacklists as well as being used for billing the UA in question.

[12]Please note that for ease of presentation, some special behavior in case of various corruption cases has been omitted from this figure.
[13]Cp. double-spending in section 4.1. Without the offline setting, it would be sufficient to look up the last transaction of the UA's wallet.

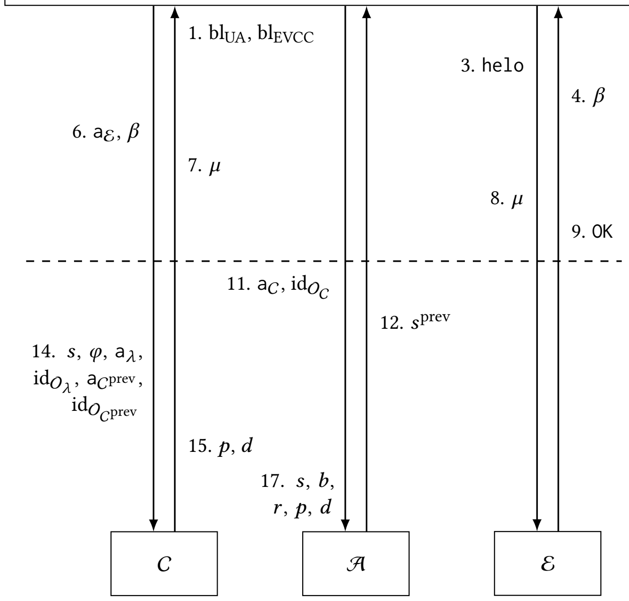
Functionality $\mathcal{F}_{\text{V2G}}$

II. Behavior – Task Debt Accumulation

2. Upon receiving $(\text{bl}_{\text{UA}}, \text{bl}_{\text{EVCC}})$ from C:
   - Leak bl_{EVCC} to the adversary.
5. Upon receiving (β) from $\mathcal{E}$:
   - If $f_{\text{rev}}(\text{id}_{\mathcal{E}}) \in \text{bl}_{\text{EVCC}}$,
     Output `blacklisted` to both parties and abort.
   - Look up the EVCCs attributes $\mathsf{a}_{\mathcal{E}}$.

- -

10. Upon receiving (`OK`) from $\mathcal{E}$:
    - Look up the SECC's attributes a_C and OPR id_{O_C}.
13. Upon receiving (s^{prev}) from $\mathcal{A}$:
    - Select entry
      $$(\cdot, s^{\text{prev}}, \cdot, x^{\text{prev}}, \lambda, \text{id}_{\mathcal{A}}, \text{id}_{C^{\text{prev}}}, b^{\text{prev}}, r^{\text{prev}}, \cdot, \cdot)$$
      from the database *TRDB*.
    - Pick previously unused serial number $s \xleftarrow{R} S$.
    - Increase counter $x := x^{\text{prev}} + 1$.
    - If $f_\Phi(\lambda, x)$ is already defined
      Set $\varphi := f_\Phi(\lambda, x)$.
      Else
      Pick previously unused fraud detection ID $\varphi \xleftarrow{R} \Phi$.
      Append $f_\Phi(\lambda, x) := \varphi$ to f_Φ.
    - If $\varphi \in \text{bl}_{\text{UA}}$,
      Output `blacklisted` to both parties and abort.
    - Look up the wallet's attributes a_λ and OPR id_{O_λ}.
    - Look up the previous SECC's attributes $\mathsf{a}_{C^{\text{prev}}}$ and OPR $\text{id}_{O_{C^{\text{prev}}}}$.
16. Upon receiving (p, d) from C:
    - $b := b^{\text{prev}} + p$.
    - $r := r^{\text{prev}} + d$.
    - Append entry
      $$(s^{\text{prev}}, s, \varphi, x, \lambda, \text{id}_{\mathcal{A}}, \text{id}_{C}, b, r, p, d)$$
      to *TRDB*.



1. $\text{bl}_{\text{UA}}, \text{bl}_{\text{EVCC}}$
3. helo
4. β
6. $\mathsf{a}_{\mathcal{E}}, \beta$
7. μ
8. μ
9. OK
11. $\mathsf{a}_C, \text{id}_{O_C}$
12. s^{prev}
14. $s, \varphi, \mathsf{a}_\lambda, \text{id}_{O_\lambda}, \mathsf{a}_{C^{\text{prev}}}, \text{id}_{O_{C^{\text{prev}}}}$
15. p, d
17. s, b, r, p, d
C
$\mathcal{A}$
$\mathcal{E}$

Figure 5: Task Debt Accumulation of $\mathcal{F}_{\text{V2G}}$

D PROOF

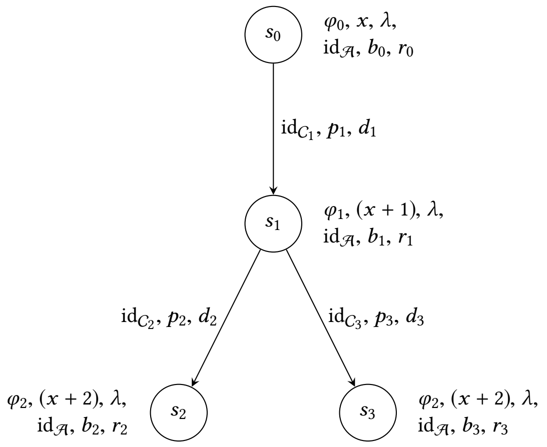
Following the example of [32], we conducted our proof in three different stages. Firstly, the database maintained by the ideal functionality is construed as a transaction graph and several structural properties of this graph are shown. Secondly, for each corruption case a simulator is constructed with which we can, thirdly, prove indistinguishability of executions of the real protocol and ideal functionality.

D.1 Transaction Graphs

The database *TRDB* maintained by the ideal functionality $\mathcal{F}_{\mathrm{V2G}}$ can be visualized as a directed graph. We call this graph the *ideal transaction graph* of the system. Each node in the graph corresponds to the state of the participating wallet after a transaction. Therefore nodes are labeled with the tuple

$$(s, \varphi, x, \lambda, \mathrm{id}_{\mathcal{A}}, b, r),$$

containing all information relevant to the wallet's state. The serial number s serves as a unique identifier of the node, so we often think of the nodes being the set of all serial numbers with the other information attached to it. The directed edges (s^{prev}, s) of the Ideal Transaction Graph describe the link between a wallet's state before and after a single transaction. Hence edges correspond to transactions themselves and, are labeled with the information (id_C, p, d). An example for an ideal transaction graph can be seen in fig. 6; node s_2 and s_3 illustrate what happens if double-spending occurs.



s_0
φ_0, x, λ,
$\mathrm{id}_{\mathcal{A}}$, b_0, r_0
id_{C_1}, p_1, d_1
s_1
φ_1, $(x+1)$, λ,
$\mathrm{id}_{\mathcal{A}}$, b_1, r_1
id_{C_2}, p_2, d_2
id_{C_3}, p_3, d_3
φ_2, $(x+2)$, λ,
$\mathrm{id}_{\mathcal{A}}$, b_2, r_2
s_2
s_3
φ_2, $(x+2)$, λ,
$\mathrm{id}_{\mathcal{A}}$, b_3, r_3

Figure 6: Visualization of an Ideal Transaction Graph

The first step towards our proof is to show different structural lemmas about the ideal transaction graph. As an example we give the first of these lemmas and its proof in full detail.

LEMMA D.1. *The ideal transaction graph TRDB is a directed forest.*

PROOF. A directed graph is a forest if and only if it is cycle-free and every node has in-degree at most one. We prove these properties by induction. On initialization of the system the graph is empty and the statement trivially satisfied. Now consider how each task modifies the graph. Tasks that insert a new entry into *TRDB* create a new node. The only tasks to do so are Issue Wallet, Debt Accumulation and Debt Clearance. Assuming that *TRDB* is a forest before inserting a new node, we can assert that *TRDB* is still a forest afterwards by looking at these tasks in detail. Issue Wallet adds an entry $trdb = (\bot, s, \ldots)$ to *TRDB* and hence adds a node s but no new edge. Note that s is indeed a new node as it is chosen from the set of idle serial numbers. Therefore s is the root of a new tree and *TRDB* is still a forest. Debt Accumulation and Debt Clearance insert a new entry $trdb = (s^{\mathrm{prev}}, s, \ldots)$ each. Again s is chosen from the set of idle serial numbers and hence a new node in our graph. The only edge inserted is (s^{prev}, s) which gives s in-degree one, does not change the in-degree of any other node and can not close a cycle as s has no outgoing edges yet. Hence *TRDB* remains a forest. Every other task only queries the ideal transaction graph, but does not change it. □

There are several other statements we prove about the ideal transaction graph. At this point we will only list them and omit the proofs.

LEMMA D.2.

(1) On each tree of TRDB, the UA $\mathrm{id}_{\mathcal{A}}$ is constant.
(2) There is a one-to-one and onto correspondence between trees in TRDB and wallets.
(3) Let $(s^*, \ldots, b^*, r^*)$ and $(s, \ldots, b, r)$ be two nodes in TRDB with an edge (id_C, p, d) from s^* to s. Then

$$b = b^* + p \quad \text{and} \quad r = r^* + d.$$

(4) Let s be a node in TRDB with counter x attached to it. Then s has depth x with respect to its tree in TRDB.
(5) Every node with the same depth in the same tree of TRDB has the same fraud detection ID φ. Conversely, nodes with the same fraud detection ID are in the same tree and have the same depth.

In a next step, we add in and out commitments, decommitments and commitment contents

$$(\mathtt{c}, \mathtt{d}, \mathtt{m})^{\mathrm{in}}_{O_\lambda}, (\mathtt{c}, \mathtt{d}, \mathtt{m})^{\mathrm{in}}_{C}$$
$$(\mathtt{c}, \mathtt{d}, \mathtt{m})^{\mathrm{out}}_{O_\lambda}, (\mathtt{c}, \mathtt{d}, \mathtt{m})^{\mathrm{out}}_{C}$$

from the real protocols to each node in the ideal transaction graph. These commitments are from the fixed and updateable part of the wallet state before and after the transaction that created this wallet state (cp. section 3). This information gives a second set of edges where two nodes s^* and s are connected if and only if the out-information of s^* matches the in-information of s:

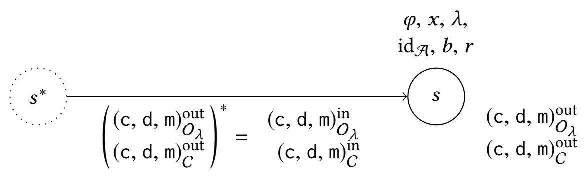
$$\left((\mathtt{c}, \mathtt{d}, \mathtt{m})^{\mathrm{out}}_{O_\lambda}\right)^* = (\mathtt{c}, \mathtt{d}, \mathtt{m})^{\mathrm{in}}_{O_\lambda}$$
$$\left((\mathtt{c}, \mathtt{d}, \mathtt{m})^{\mathrm{out}}_{C}\right)^* = (\mathtt{c}, \mathtt{d}, \mathtt{m})^{\mathrm{in}}_{C}.$$

We call the resulting graph the *augmented transaction graph* (cp. fig. 7).

LEMMA D.3. *The graph structures of the ideal and the augmented transaction graph coincide with overwhelming probability.*

D.2 Simulator

In this section we first explain the function of a simulator in the ideal/real paradigm, before showing the simulator for the task of



$\varphi, x, \lambda,$
$\mathrm{id}_{\mathcal{A}}, b, r$
s^*
s
$\begin{pmatrix}(\mathtt{c},\mathtt{d},\mathtt{m})^{\mathrm{out}}_{O_\lambda}\\(\mathtt{c},\mathtt{d},\mathtt{m})^{\mathrm{out}}_{C}\end{pmatrix}^* = \begin{matrix}(\mathtt{c},\mathtt{d},\mathtt{m})^{\mathrm{in}}_{O_\lambda}\\(\mathtt{c},\mathtt{d},\mathtt{m})^{\mathrm{in}}_{C}\end{matrix}$
$(\mathtt{c},\mathtt{d},\mathtt{m})^{\mathrm{out}}_{O_\lambda}$
$(\mathtt{c},\mathtt{d},\mathtt{m})^{\mathrm{out}}_{C}$

Figure 7: Entry of an Augmented Transaction Graph

Debt Accumulation with corrupted OPRside as an example of the simulators in our proof.

The general idea of a simulator is that it functions as the ideal world counterpart of a malicious real world adversary. We show that for every real world adversary attacking the real protocol, there exists a simulator in the ideal world achieving the same things in an execution of the ideal model. But since the ideal model is trivially secure and privacy preserving, the simulator can not gain any advantage and so neither can a real world adversary. "Achieving the same things" in this case means that an outside party can not distinguish between a protocol execution in the real world with the real adversary and the ideal model with the simulator, not even if it may choose parties inputs and learns their outputs. This rather complicated setup is simplified by combining the real world adversary and the distinguishing outside party to form one entity, the so-called *environment* $\mathcal{Z}$.[14] This environment controls all corrupted parties completely—including malicious deviations from the protocol—and chooses inputs for all honest parties as well as learning their outputs. Now either all parties run an instance of the real world protocol π_{P6V2G} (see fig. 8), or all honest parties participate in the ideal model and the simulator has to provide the interface between the ideal functionality $\mathcal{F}_{\mathrm{V2G}}$ expecting inputs from the corrupted parties and the corrupted parties who still run the real protocol and expect protocol messages from the honest parties in return (see fig. 9). Now the protocol π_{P6V2G} securely realizes the ideal functionality $\mathcal{F}_{\mathrm{V2G}}$, if we can construct a simulator $\mathcal{S}$ in such a way that the environment $\mathcal{Z}$ can not distinguish between the two cases, i.e., between the real world and the ideal model in figs. 8 and 9 respectively.

To do this we need to define a separate simulator for every corruption case and their behavior in each of our systems tasks. As an example we take a closer look at the simulator for user security and privacy performing the task Debt Accumulation (see fig. 10). In this case, the participating EVCC and UA are honest, while the SECC may be corrupted. Therefore the simulator $\mathcal{S}$ has to provide a SECC's input to the ideal functionality $\mathcal{F}_{\mathrm{V2G}}$ and protocol messages the SECC expects from an EVCC and UA running the real protocol π_{P6V2G}. To achieve this, it utilizes the SECC's output it gets from the ideal functionality, the real protocol messages the SECC sends to the EVCC and UA as well as its ability to, e.g., extract commitments or simulate proofs with the trapdoor `td` it knows from choosing the `CRS`.

[14]Note that this simplification gives the adversary more power and therefore only strengthens our security and privacy statement.

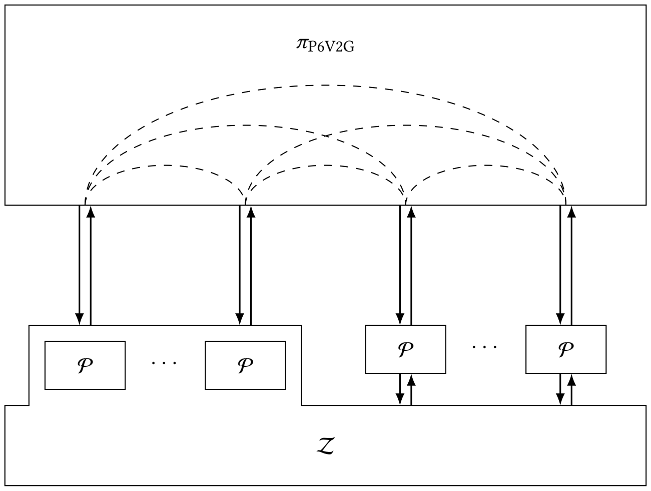


Figure 8: Real World

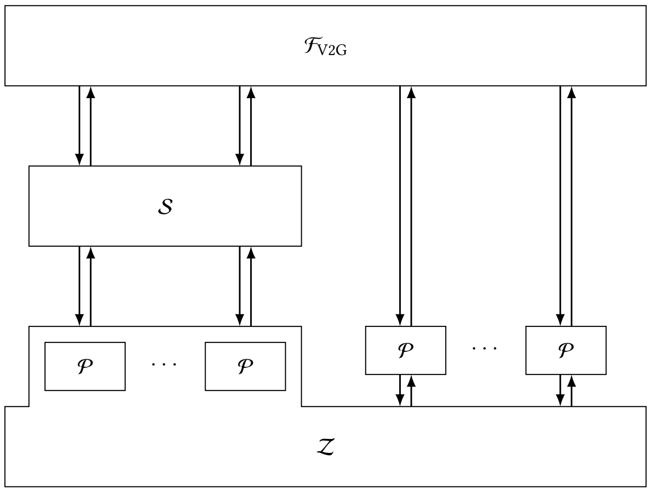


Figure 9: Ideal Model

D.3 Indistinguishability

As a last step we have to prove that the environment's views in the real world and ideal model (cp. figs. 8 and 9) are actually indistinguishable. This is done by defining a series of hybrids between those two worlds. The first hybrid $\mathcal{H}_0$ is equal to executing the real protocol while the last hybrid $\mathcal{H}_{\max}$ equals an execution of the ideal model. Each hybrid is of the form

$$\mathcal{H}_i = \mathsf{Exec}(\pi_i, \mathcal{S}_i, \mathcal{Z}),$$

where π_i and $\mathcal{S}_i$ are incremental modifications of π_{i-1} and $\mathcal{S}_{i-1}$ respectively. While $\pi_0 = \pi_{\mathrm{P6V2G}}$ is our real world protocol, the last version $\pi_{\max}$ equals the ideal functionality $\mathcal{F}_{\mathrm{V2G}}$. The simulator progresses in the other direction with $\mathcal{S}_{\max} = \mathcal{S}$ being the original simulator defined in appendix D.2 and $\mathcal{S}_0$ being a dummy simulator

Simulator $\mathcal{S}$

Task Debt Accumulation

If the SECC is honest, do nothing, else do:

- Load the recorded $\mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}$, $\mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}$ for $\mathrm{id}_{\mathcal{D}}$.
- Upon receiving $(\mathrm{bl}_{\mathrm{EVCC}})$ from $\mathcal{Z}$ in the name of C:
  - Call $\mathcal{F}_{\mathrm{V2G}}$ in the name of C with input $(\emptyset, \mathrm{bl}_{\mathrm{EVCC}})$.
  - Obtain SECC output $(\mathsf{a}_{\mathcal{E}}, \beta)$ from $\mathcal{F}_{\mathrm{V2G}}$.
  - $(\mathsf{c}_{rev}, \mathsf{d}_{rev}) := \mathtt{C6.Com}(\mathtt{CRS}, 0)$.
  - $stmnt_1 := (\mathsf{a}_{\mathcal{E}}, \mathsf{c}_{rev}, \mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}})$.
  - $\pi_1 := \mathtt{P31.SimProof}(\mathtt{CRS}, \mathtt{td}, stmnt_1)$.
  - Compute $v := \mathtt{ACC.Evaluate}(\mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}, \mathrm{bl}_{\mathrm{EVCC}})$.
  - $stmnt_2 := (\mathsf{c}_{rev}, v)$.
  - $\pi_2 := \mathtt{P32.SimProof}(\mathtt{CRS}, \mathtt{td}, stmnt_2)$.
  - Output $(\beta, \mathsf{a}_{\mathcal{E}}, \mathsf{c}_{rev}, \pi_1, \pi_2)$ to $\mathcal{Z}$ as message from $\mathcal{E}$ to C.
- Upon receiving (μ) from $\mathcal{Z}$ in the name of C:
  - Output $(\mathtt{OK})$ to $\mathcal{Z}$ as message from $\mathcal{E}$ to C.
  - Call $\mathcal{F}_{\mathrm{V2G}}$ in the name of C with input $(\mu, \mathsf{pk}^{\mathrm{cert}}_{\mathcal{O}_C})$.
  - Obtain SECC output $(s, \varphi, \mathsf{a}_{\lambda}, \mathrm{id}_{\mathcal{O}_{\lambda}}, \mathsf{a}_{C^{\mathrm{prev}}}, \mathrm{id}_{\mathcal{O}_{C^{\mathrm{prev}}}})$.
- Upon receiving $(\mathtt{cert}_C, \mathsf{c}''_{\mathrm{ser}}, u_2)$ from $\mathcal{Z}$ in the name of C:
  - Parse $(\mathsf{pk}^{\mathrm{sig}}_{C}, \mathsf{a}_C, \mathsf{pk}^{\mathrm{cert}}_{\mathcal{O}_C}, \sigma^{\mathrm{cert}}_{\mathcal{O}_C}) := \mathtt{cert}_C$.
  - If $\mathtt{S4.Vfy}(\mathsf{pk}^{\mathrm{cert}}_{\mathcal{O}_C}, \sigma^{\mathrm{cert}}_{\mathcal{O}_C}, (\mathsf{pk}^{\mathrm{sig}}_{C}, \mathsf{a}_C)) = 0$, let $\mathcal{F}_{\mathrm{V2G}}$ abort.
  - $s'' \leftarrow \mathtt{C4.Extract}(\mathtt{CRS}, \mathsf{c}''_{\mathrm{ser}})$.
  - Set $s' := s \cdot s''^{-1}$.
  - If a record $(\varphi, (\mathsf{pk}^{\mathrm{cert}}_{\mathcal{O}_C})^*, t^*, u_2^*) \in \Omega_{\mathrm{ds}}$ exists, set
    $$t := t^* + \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}} \cdot (u_2 - u_2^*).$$
    Else set $t \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q$.
  - Insert $(\varphi, \mathsf{pk}^{\mathrm{cert}}_{\mathcal{O}_C}, t, u_2)$ into Ω_{ds}.
  - $(\mathsf{c}'_C, \mathsf{d}'_C) := \mathtt{C2.Com}(\mathtt{CRS}, (0, 0, 0, 0, 0))$.
  - $stmnt := (\varphi, t, u_2, \mathsf{a}_{\lambda}, \mathsf{a}_{C^{\mathrm{prev}}}, \mathsf{c}'_C, \mathsf{pk}^{\mathrm{sig}}_{\mathcal{O}_{\lambda}}, \mathsf{pk}^{\mathrm{cert}}_{\mathcal{O}_{C^{\mathrm{prev}}}})$.
  - $\pi := \mathtt{P4.SimProof}(\mathtt{CRS}, \mathtt{td}, stmnt)$.
  - Output $(\pi, s', \varphi, t, \mathsf{c}'_C, \mathsf{a}_{\lambda}, \mathsf{a}_{C^{\mathrm{prev}}}, \mathsf{pk}^{\mathrm{sig}}_{\mathcal{O}_{\lambda}}, \mathsf{pk}^{\mathrm{cert}}_{\mathcal{O}_{C^{\mathrm{prev}}}})$ to $\mathcal{Z}$ as message from $\mathcal{A}$ to C.
- Upon receiving $(s'', \mathsf{d}''_{\mathrm{ser}}, \mathsf{c}_C, \mathsf{d}''_C, \sigma_C, p, d)$ from $\mathcal{Z}$ in the name of C:
  - Set $\mathsf{d}_C := \mathsf{d}'_C \cdot \mathsf{d}''_C$.
  - If $\mathtt{C2.Open}(\mathtt{CRS}, (\mathbb{1}_{\mathbb{G}}, \mathsf{g}_1^p, \mathsf{g}_1^d, \mathbb{1}_{\mathbb{G}}, \mathsf{g}_1), \mathsf{c}_C, \mathsf{d}_C) = 0$, let $\mathcal{F}_{\mathrm{V2G}}$ abort.
  - If $\mathtt{S1.Vfy}(\mathsf{pk}^{\mathrm{sig}}_{C}, \sigma_C, (\mathsf{c}_C, s)) = 0$, let $\mathcal{F}_{\mathrm{V2G}}$ abort.
  - Call $\mathcal{F}_{\mathrm{V2G}}$ in the name of C with input (p, d).

Figure 10: User Security Simulator for Debt Accumulation

that does nothing but relay all messages. Instead of proving indistinguishability between the real protocol and ideal model in one go, it is now possible to do this stepwise by proving indistinguishability between each pair of consecutive hybrids.

As an example lets look at one of the hybrid hops in our proof of user security. The incremental difference between the hybrids $\mathcal{H}_9$ and $\mathcal{H}_{10}$ in this case is the following: $\mathcal{H}_{10}$ modifies the task DebtAccumulation. $\mathcal{S}_{10}$ replaces c'_C in the message to the SECC (cp. fig. 32) by a commitment containing only zeros. Everything else remains the same. The proof of indistinguishability between the hybrids $\mathcal{H}_9$ and $\mathcal{H}_{10}$ is a reduction to the cryptographic properties of the commitment scheme used for c'_C and its underlying hardness assumption; assuming we had an environment $\mathcal{Z}$ that could distinguish between $\mathcal{H}_9$ and $\mathcal{H}_{10}$, we construct an adversary who would be able to effectively violate the hiding property of the commitment scheme.

E P6V2G PROTOCOL

This section contains our concrete instantiation and complete protocol π_{P6V2G}. Firstly, we show how the abstractly used cryptographic building blocks (e.g., encryption and commitments) may be instantiated to implement our system. Secondly, our protocol π_{P6V2G} is given.

E.1 Instantiation

Our P6V2G protocol π_{P6V2G} uses cryptographic primitives like commitments and digital signatures in an abstract fashion. Our security proofs state what properties they need to have, but another requirement is only stated implicitly. The languages for which NIZK proofs are generated contain statements about primitives. Therefore these statements need to be compatible with the NIZK proof system that is used. In the following paragraphs we provide an example for an instantiation of the protocol π_{P6V2G}. The security of the instantiations relies on the *SXDH assumption* [27], the *q-strong DH assumption* [5] and the *n-DDHI assumption* [10].

NIZK Proof System. The proof systems P1, P2, P31, P32, P4 and P5 can be instantiated with the SXDH-based variant of the Groth-Sahai (GS) proof system [27]. It is defined for languages L_{gp} that contain statements described by the conjunction of pairing-product equations, multi-scalar equations over $\mathbb{G}_1$, multi-scalar equations over $\mathbb{G}_2$ and quadratic equations over $\mathbb{Z}_q$. The GS proof system is perfectly complete, perfectly sound and F_{gp}-extractable for the language F_{gp} that maps group elements to group elements and elements $x \in \mathbb{Z}_q$ to g_i^x. Moreover, it is known to be composable zero-knowledge under certain restrictions. These are met by the language considered in the protocol.

The language $\mathsf{L}^{(1)}$ of P1 contains range proofs to show that

$$\lambda'_i \in \{0, \ldots, base - 1\}.$$

They can be implemented using the signature-based technique of Camenisch and Chabounni [12].

Commitment Schemes. Two different commitment schemes are used throughout our protocol. The shrinking l-message-commitment scheme from Abe et al. [3] with message space $\mathbb{Z}_q^l$, commitment space $\mathbb{G}_2$ and opening value space $\mathbb{G}_1$ is correct, statistically hiding, additively homomorphic, equivocal and $\mathsf{F}_{\mathrm{gp}}{}'$-Binding for

$$\mathsf{F}_{\mathrm{gp}}{}'(m_1, \ldots, m_l) := (\mathsf{g}_1^{m_1}, \ldots, \mathsf{g}_1^{m_l})$$

under the SXDH-assumption. It has to be $\mathsf{F}_{\mathrm{gp}}{}'$-binding, because statements about commitments from this scheme are proven and the GS proof system is only F_{gp}-extractable. We use instantiations of this scheme for C1, C2, C3 and C6 with l equal to two, five, two and one respectively.

The extractable commitment scheme introduced by Groth and Sahai [27] has message space $\mathbb{G}_1$ commitment space $\mathbb{G}_1^2$ and opening value space $\mathbb{Z}_q^2$. It is correct, hiding, equivocal, extractable and binding under the SXDH assumption. We use this to instantiate $\mathsf{C4}$.

Cryptographic Accumulator. We instantiate the cryptographic accumulator with a construction that has originally been proposed by Nguyen [41] for symmetric pairings. It can accumulate up to k_{ACC} elements, with k_{ACC} being a public, pre-determined system parameter. Au et al. [6] extended this construction with proofs of non-memberships and Lin and Hopper [37] adopted it to asymmetric pairings. Security holds under the q-SDH assumption. The accumulator is sound for any choice $k_{\mathsf{ACC}} \leq q$. The space of accumulatable elements is $\mathbb{Z}_q \setminus \{-\mathsf{sk}^{\mathrm{acc}}\}$ with $\mathsf{sk}^{\mathrm{acc}} \in \mathbb{Z}_q$ denoting the accumulator's trapdoor. The value space of the accumulator equals $\mathbb{G}_1$.

Digital Signatures. The signature schemes $\mathsf{S1}$ to $\mathsf{S4}$ can be instantiated with the structure-preserving signature scheme of Abe et al. [1]. It is EUF-CMA secure in the generic group model. The message space $M = \mathbb{G}_1^\nu \times \mathbb{G}_2^\mu$ is defined by the two parameters $\nu, \mu \in \mathbb{N}_0$. Then $\sigma \in \mathbb{G}_1 \times \mathbb{G}_2^2$, $\mathsf{sk} \in \mathbb{Z}_q^{\nu+\mu+2}$ and $\mathsf{pk} \in \mathbb{G}_1^{\mu+2} \times \mathbb{G}_2^\nu$ holds. We use instantiations of this scheme for $\mathsf{S1}$, $\mathsf{S2}$, $\mathsf{S3}$, and $\mathsf{S4}$ with (ν, μ) equal to $(1, 1)$, $(0, y + 1)$, $(0, j + 1)$, $(2l + 4, 0)$ and $(3, z + 1)$, respectively. Here, $y := |\mathsf{a}_{\mathcal{E}}|$, $j := |\mathsf{a}_\lambda|$ and $z := |\mathsf{a}_C|$.

Pseudo-Random Function. The PRF used to generate the fraud detection IDs can be instantiated with the PRF introduced by Dodis and Yampolsky [23]. It is an algebraic, group-based construction and allows to prove that the function was evaluated correctly. This function, defined by

$$\mathsf{PRF}(\lambda, x) : \mathbb{Z}_q^2 \to \mathbb{G}_1, (\lambda, x) \mapsto \mathrm{g}_1^{\frac{1}{\lambda+x}}$$

with key $\lambda \in \mathbb{Z}_q$, is secure for inputs $x \in \{0, \ldots, n\} \subset \mathbb{Z}_q$ under the n-DDHI assumption.

Asymmetric Encryption. The protocol uses an adopted variant of the structure-preserving, IND-CCA secure encryption scheme by Camenish et al. [13]. The original encryption scheme is formalized for a symmetric Type-1 pairing, but we need a scheme that is secure in the asymmetric Type-3 case. For the conversion we followed a transformation procedure as proposed by Abe et al. [2] with some additional, manual optimizations. The transformed scheme can encrypt vectors in $\mathbb{G}_1$ and is secure under the DLIN assumption. The scheme is used as E in the task Issue Wallet to encrypt the hidden trapdoor of the wallet. Some explanations are in order on this choice. Ideally, one would want to encrypt the wallet ID $\lambda \in \mathbb{Z}_q$ in order to enable blacklisting. Moreover, the wallet must prove to the OPRthat it honestly encrypted the correct wallet ID. For practical reasons the Groth-Sahai NIZK is used (see fig. 39). Therefore an encryption scheme with message space $\mathbb{Z}_q$ that is compatible with the Groth-Sahai NIZK-scheme is required. As we are not aware of such a scheme, the encryption scheme with message space $\mathbb{G}_1$ is used instead. But if the wallet would only encrypt g_1^λ, the DR would not be able to recover λ from the decryption of e, because the CDH-assumption holds in $\mathbb{G}_1$. To get around this obstacle, the wallet picks its own share of the seed λ' by randomly picking a "digit representation" λ'_i in the base-*base* system:

$$\lambda' = \sum_{i=0}^{\ell} \lambda'_i \cdot base^i$$

If *base* is chosen in a way that it is efficiently possible to compute the discrete logarithm for the elements $\lambda'_i < base$, it is possible to recover λ' (see fig. 36). The wallet encrypts all λ'_i chunks, the OPR's share λ'' and its own public key $\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}$ inside a single vector. As the OPR's share λ'' is known to the OPRanyway, it can additionally be stored in the clear alongside the encryption and thus does not need to be split into chunks. Nonetheless, it is still required that the OPR's share λ'' is part of the encryption such that none of the components of the wallet ID is malleable. Otherwise an malicious OPRcould try to evaluate the PRF at ineligible points and thus blacklist a different (innocent) user. For the same reason, the wallet's public key must be bound to the encryption.

E.2 Full Protocol

Finally, we include our complete P6V2G protocol π_{P6V2G} in this section. An overview of each party's locally saved state as well as all tasks supported by the system can be found in fig. 11. For readability purposes most tasks are then given in two parts: a wrapper and a core protocol. In the wrapper protocol the participating parties mainly load and save internally stored information needed for the current task. All interaction between parties is conducted in the core protocol which is invoked by the wrapper. Lastly, if a protocol includes a NIZK proof, the corresponding language (i.e., properties this proof asserts) is given at the very end.

UC-Protocol $\pi_{\mathbf{P6V2G}}$

I. Local State

The DR $\mathcal{D}$ internally records:

- Its public and private key $(\mathsf{sk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{D}})$.
- A mapping $\mathsf{pk}_{\mathcal{E}} \mapsto (\mathsf{a}_{\mathcal{E}}, \mathit{rev})$.

An OPR O internally records:

- Its public and private key $(\mathsf{sk}_{O}, \mathsf{pk}_{O})$.
- A self-signed certificate $\mathtt{cert}_{C_O}$.
- A mapping $\mathsf{pk}_{C} \mapsto \mathsf{a}_{C}$.
- A set HTD_{λ} of hidden trapdoors for wallets.
- Sets Ω_{bl} and Ω_{ds} containing blacklisting information and double-spending detection information.

A SECC C internally records:

- Its public and private key $(\mathsf{sk}_{C}, \mathsf{pk}_{C})$.
- Its certificate $\mathtt{cert}_{C}$ validated by an OPR.
- Sets Ω_{bl} and Ω_{ds} containing blacklisting information and double-spending detection information.

An EVCC $\mathcal{E}$ internally records:

- Its public and private key $(\mathsf{sk}_{\mathcal{E}}, \mathsf{pk}_{\mathcal{E}})$.
- Its certificate $\mathtt{cert}_{\mathcal{E}}$ validated by the DR.

A UA $\mathcal{A}$ internally records:

- Its public and private key $(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}})$.
- A set $\{\tau\}$ of all its recorded tokens.

II. Behavior – Tasks

- Register DR (fig. 12)
- Register OPR (fig. 12)
- Register SECC (fig. 12)
- Register EVCC (fig. 12)
- Register UA (fig. 12)
- Certify SECC (fig. 13)
- Certify EVCC (fig. 14)
- Issue Wallet (fig. 15)
- Debt Accumulation (fig. 17)
- Debt Clearance (fig. 16)
- Double-Spending Detection (fig. 18)
- Guilt Verification (fig. 19)
- Blacklist UA (fig. 20)
- Blacklist EVCC (fig. 21)

Figure 11: Protocol $\pi_{\mathbf{P6V2G}}$

UC-Protocol $\pi_{\mathbf{P6V2G}}$ – Task Register Party

Party input: $(\mathtt{register})$

(1) If a key pair $(\mathsf{pk}_{\mathcal{P}}, \mathsf{sk}_{\mathcal{P}})$ has already been recorded, output $\perp$ and abort.
(2) Obtain CRS from $\mathcal{F}_{\mathrm{CRS}}$.
(3) Run $(\mathsf{pk}_{\mathcal{P}}, \mathsf{sk}_{\mathcal{P}}) \leftarrow \mathtt{RegisterParty}(\mathsf{CRS})$ (see figs. 22 to 26).
(4) Record $(\mathsf{pk}_{\mathcal{P}}, \mathsf{sk}_{\mathcal{P}})$ internally and call $\overline{\mathcal{G}}_{\mathrm{BB}}$ with input $(\mathtt{register}, \mathsf{pk}_{\mathcal{P}})$.

Party output: $(\mathsf{pk}_{\mathcal{P}})$

Figure 12: Protocol for Task Register Party

UC-Protocol $\pi_{\mathbf{P6V2G}}$ – Task Certify SECC

SECC input: $(\mathtt{certify})$
OPR input: $(\mathtt{certify}, \mathsf{a}_{C})$

(1) For the SECC side:
  - Load the internally recorded $\mathsf{pk}^{\mathrm{sig}}_{C}$.
  - Retrieve $\mathsf{pk}^{\mathrm{cert}}_{O}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for ID id_{O}.
(2) For the OPR side:
  - Load the internally recorded $(\mathsf{pk}^{\mathrm{cert}}_{O}, \mathsf{sk}^{\mathrm{cert}}_{O})$.
  - Retrieve $\mathsf{pk}^{\mathrm{sig}}_{C}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for ID id_{C}.
  - Check that no mapping $\mathsf{pk}^{\mathrm{sig}}_{C} \mapsto \mathsf{a}^{*}_{C}$ has been registered before, else ouput $\perp$ and abort.
(3) Run $\mathtt{CertifySECC}$ (see fig. 27) for the SECC and the OPR:

$$\begin{pmatrix}\mathtt{cert}_{C}\\ (\mathsf{OK})\end{pmatrix} \leftarrow \mathtt{CertifySECC}\left\langle \begin{matrix} C\left(\mathsf{pk}^{\mathrm{sig}}_{C}, \mathsf{pk}^{\mathrm{cert}}_{O}\right) \\ O\left(\mathsf{pk}^{\mathrm{cert}}_{O}, \mathsf{sk}^{\mathrm{cert}}_{O}, \mathsf{pk}^{\mathrm{sig}}_{C}, \mathsf{a}_{C}\right)\end{matrix}\right\rangle$$

(4) For the SECC side:
  - Record $\mathtt{cert}_{C}$ internally.
  - Parse $(\mathsf{pk}^{\mathrm{sig}}_{C}, \mathsf{a}_{C}, \mathsf{pk}^{\mathrm{cert}}_{O}, \sigma^{\mathrm{cert}}_{O}) := \mathtt{cert}_{C}$.
  - Retrieve id_{O} from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for public key $\mathsf{pk}^{\mathrm{cert}}_{O}$.
(5) For the OPR side:
  - Record $\mathsf{pk}^{\mathrm{sig}}_{C} \mapsto \mathsf{a}_{C}$ internally.

SECC output: $((\mathsf{a}_{C}, \mathrm{id}_{O}))$
OPR output: (OK)

Figure 13: Protocol for Task Certify SECC

UC-Protocol $\pi_{\mathbf{P6V2G}}$ – Task Certify EVCC

EVCC input: $(\mathtt{certify})$
DR input: $(\mathtt{certify}, \mathsf{a}_{\mathcal{E}})$

(1) For the EVCC side:
  - Load the internally recorded $(\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{E}})$.
  - Retrieve $\mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for ID $\mathrm{id}_{\mathcal{D}}$.
(2) For the DR side:
  - Load the internally recorded $(\mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}, \mathsf{sk}^{\mathrm{sig}}_{\mathcal{D}})$.
  - Check that no mapping $\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}} \mapsto (\mathsf{a}'_{\mathcal{E}}, \mathit{rev}')$ has been registered before, else ouput $\perp$ and abort.
(3) Run $\mathtt{CertifyEVCC}$ (see fig. 28) for the EVCC and the DR:

$$\begin{pmatrix}\mathtt{cert}_{\mathcal{E}}\\ (\mathit{rev})\end{pmatrix} \leftarrow \mathtt{CertifyEVCC}\left\langle \begin{matrix} \mathcal{E}\left(\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{E}}, \mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}\right) \\ \mathcal{D}\left(\mathsf{sk}^{\mathrm{sig}}_{\mathcal{D}}, \mathsf{a}_{\mathcal{E}}\right)\end{matrix}\right\rangle$$

(4) For the EVCC side:
  - Record $\mathtt{cert}_{\mathcal{E}}$ internally.
  - Parse $(\mathit{rev}, \mathsf{a}_{\mathcal{E}}, \mathsf{c}, \mathsf{d}, \sigma) := \mathtt{cert}_{\mathcal{E}}$.
(5) For the DR side:
  - Record $\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}} \mapsto (\mathsf{a}_{\mathcal{E}}, \mathit{rev})$ internally.

EVCC output: $(\mathsf{a}_{\mathcal{E}})$
DR output: (OK)

Figure 14: Protocol for Task Certify EVCC

UC-Protocol $\pi_{\mathbf{P6V2G}}$ – Task Issue Wallet

UA input: $(\mathtt{issue})$

OPR input: $(\mathtt{issue}, \mathsf{a}, r, \mathrm{wl})$

(1) For the UA side:
- Load the internally recorded $(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}})$.
- Retrieve $\mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for ID $\mathrm{id}_{\mathcal{D}}$.
- Retrieve $\mathsf{pk}^{\mathrm{sig}}_{\mathcal{O}}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for ID $\mathrm{id}_{\mathcal{O}}$.

(2) For the OPR side:
- Load the internally recorded $\mathsf{sk}^{\mathrm{sig}}_{\mathcal{O}}$ and $\mathsf{sk}^{\mathrm{sig}}_{\mathcal{C}_{\mathcal{O}}}$.
- Load the internally recorded $\mathtt{cert}_{\mathcal{C}_{\mathcal{O}}}$.
- Retrieve $\mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for ID $\mathrm{id}_{\mathcal{D}}$.

(3) Run $\mathtt{IssueWallet}$ (see fig. 29) for the UA and the OPR:

$$\begin{pmatrix} (\tau) \\ (s, htd_{\lambda}) \end{pmatrix} \leftarrow \mathtt{IssueWallet}\left\langle \begin{matrix} \mathcal{A}\left(\mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{pk}^{\mathrm{sig}}_{\mathcal{O}}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}\right) \\ \mathcal{O}\left(\mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{sk}^{\mathrm{sig}}_{\mathcal{O}}, \mathsf{sk}^{\mathrm{sig}}_{\mathcal{C}_{\mathcal{O}}}, \mathtt{cert}_{\mathcal{C}_{\mathcal{O}}}, \mathsf{a}, r, \mathrm{wl}\right) \end{matrix} \right\rangle$$

(4) For the UA side:
- If $\mathtt{VerifyWallet}(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \tau)$ (see fig. 35) returns NOK, output $\perp$ and abort.
- Record τ internally.
- Parse $(s, \mathsf{a}_{\lambda}, r)$ from τ.

(5) For the OPR side:
- Insert htd_{λ} into HTD_{λ}.

UA output: $(s, \mathsf{a}_{\lambda}, \mathrm{id}_{\mathcal{O}}, r)$

OPR output: (s)

Figure 15: Protocol for Task Issue Wallet

UC-Protocol $\pi_{\mathbf{P6V2G}}$ – Task Debt Clearance

UA input: $(\mathtt{clear}, s^{\mathrm{prev}})$

OPR input: $(\mathtt{clear})$

(1) For the UA side:
- Load the internally recorded $(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}})$.
- Load the internally recorded τ^{prev} for s^{prev}.

(2) For the OPR side:
- Load the internally recorded $\mathsf{pk}^{\mathrm{sig}}_{\mathcal{O}}$.

(3) Run $\mathtt{DebtClearance}$ (see fig. 34) for the UA and OPR:

$$\begin{pmatrix} (b_{\mathrm{bill}}, r_{\mathrm{bill}}) \\ \big(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{a}_{\lambda}, \omega_{\mathrm{bl}}, \omega_{\mathrm{ds}}, r^{\mathrm{bill}}, \\ \mathsf{a}_{\mathcal{C}^{\mathrm{prev}}}, \mathsf{pk}^{\mathrm{cert}}_{\mathcal{O}_{\mathcal{C}^{\mathrm{prev}}}}\big) \end{pmatrix} \leftarrow \mathtt{DebtClearance}\left\langle \begin{matrix} \mathcal{A}\left(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}, \tau^{\mathrm{prev}}\right) \\ \mathcal{O}\left(\mathsf{pk}^{\mathrm{sig}}_{\mathcal{O}}\right) \end{matrix} \right\rangle$$

(4) For the OPR side:
- Record ω_{bl} and ω_{ds} internally.
- Parse $(\varphi, -b_{\mathrm{bill}}, 0) := \omega_{\mathrm{bl}}$.
- Retrieve $\mathrm{id}_{\mathcal{A}}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for $\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}$.
- Retrieve $\mathrm{id}_{\mathcal{O}_{\mathcal{C}^{\mathrm{prev}}}}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for $\mathsf{pk}^{\mathrm{cert}}_{\mathcal{O}_{\mathcal{C}^{\mathrm{prev}}}}$.

UA output: $(b_{\mathrm{bill}}, r_{\mathrm{bill}})$

OPR output: $(\varphi, \mathrm{id}_{\mathcal{A}}, \mathsf{a}_{\lambda}, b_{\mathrm{bill}}, r_{\mathrm{bill}}, \mathsf{a}_{\mathcal{C}^{\mathrm{prev}}}, \mathrm{id}_{\mathcal{O}_{\mathcal{C}^{\mathrm{prev}}}})$

Figure 16: Protocol for Task Debt Clearance

UC-Protocol $\pi_{\mathbf{P6V2G}}$ – Task Debt Accumulation

SECC input: $(\mathtt{charge}, \mathrm{bl}_{\mathrm{UA}}, \mathrm{bl}_{\mathrm{EVCC}}, \mu)$

EVCC input: $(\mathtt{charge}, \beta)$

UA input: $(\mathtt{charge}, s^{\mathrm{prev}})$

(1) For the SECC side:
- Retrieve $\mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}$, $\mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}$ for ID $\mathrm{id}_{\mathcal{D}}$.

(2) For the EVCC side:
- Load the internally recorded $(\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{E}})$.
- Load the internally recorded $\mathtt{cert}_{\mathcal{E}}$.
- Retrieve $\mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}$, $\mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for ID $\mathrm{id}_{\mathcal{D}}$.

(3) Run $\mathsf{CheckEVCC}$ (see fig. 31) for the EVCC and SECC:

$$\begin{pmatrix} (\mu) \\ (\mathsf{a}_{\mathcal{E}}, \beta) \end{pmatrix} \leftarrow \mathsf{CheckEVCC}\left\langle \begin{matrix} \mathcal{E}\left(\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{E}}, \mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}, \mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}, \mathtt{cert}_{\mathcal{E}}, \beta\right) \\ \mathcal{C}\left(\mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}, \mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}, \mathrm{bl}_{\mathrm{EVCC}}, \mu\right) \end{matrix} \right\rangle$$

(4) For the SECC side:
- Conduct charging.
- Determine p and d.
- Load the internally recorded $(\mathsf{pk}_{\mathcal{C}}, \mathsf{sk}_{\mathcal{C}})$.
- Load the internally recorded $\mathtt{cert}_{\mathcal{C}}$.

(5) For the UA side:
- Load the internally recorded $(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}})$.
- Load the internally recorded τ^{prev} for s^{prev}.

(4) Run $\mathtt{DebtAccum}$ (see fig. 32) for SECC and UA:

$$\begin{pmatrix} (\tau, p, d) \\ \left(\mathsf{a}_{\lambda}, \mathsf{pk}^{\mathrm{sig}}_{\mathcal{O}_{\lambda}}, s, \omega_{\mathrm{bl}}, \omega_{\mathrm{ds}}\right) \end{pmatrix} \leftarrow \mathtt{DebtAccum}\left\langle \begin{matrix} \mathcal{A}\left(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}, \tau^{\mathrm{prev}}\right) \\ \mathcal{C}\left(\mathsf{sk}^{\mathrm{sig}}_{\mathcal{C}}, \mathtt{cert}_{\mathcal{C}}, \mathrm{bl}_{\mathrm{UA}}, p, d\right) \end{matrix} \right\rangle$$

(6) For the SECC side:
- Record ω_{bl} and ω_{ds} internally.
- Parse φ from ω_{bl}.
- Retrieve $\mathrm{id}_{\mathcal{O}_{\lambda}}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for public key $\mathsf{pk}^{\mathrm{sig}}_{\mathcal{O}_{\lambda}}$.

(7) For the UA side:
- If $\mathtt{VerifyWallet}(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \tau)$ (see fig. 35) returns NOK, output $\perp$ and abort.
- Record τ internally.
- Parse $(\mathsf{a}_{\mathcal{C}}, s, b, r)$ from τ.

SECC output: $(\mathsf{a}_{\lambda}, \mathrm{id}_{\mathcal{O}_{\lambda}}, \mathsf{a}_{\mathcal{E}}, \beta, s, \varphi)$

EVCC output: $(\mathsf{a}_{\lambda}, \mathrm{id}_{\mathcal{O}_{\lambda}}, \mu)$

UA output: $(\mathsf{a}_{\mathcal{C}}, s, b, r, p, d)$

Figure 17: Protocol for Task Debt Accumulation

UC-Protocol $\pi_{\mathbf{P6V2G}}$ – Task Double-Spending Detection

OPR input: $(\mathtt{detect}, \varphi)$

(1) Load the recorded set Ω_{ds} of all double spending transaction information.

(2) Pick $\omega_{\mathrm{ds}}, \omega^{*}_{\mathrm{ds}} \in \Omega_{\mathrm{ds}}$ with $\omega_{\mathrm{ds}} = (\varphi, t, u_2)$ and $\omega^{*}_{\mathrm{ds}} = (\varphi, t^{*}, u^{*}_{2})$, such that $u_2 \neq u^{*}_{2}$.

(3) Set $\mathsf{sk}^{\mathrm{id}}_{\mathcal{A}} := (t - t^{*}) \cdot (u_2 - u^{*}_{2})^{-1} \mod q$.

(4) Set $\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}} := \mathrm{g}_1^{\mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}}$.

(5) Retrieve $\mathrm{id}_{\mathcal{A}}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for $\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}$.

(6) Set $\pi := \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}$.

OPR output: $(\mathrm{id}_{\mathcal{A}}, \pi)$

Figure 18: Protocol for Task Double-Spending Detection

UC-Protocol π_{P6V2G} – Task Guilt Verification

Party input: ($\mathtt{verify}$, $\mathrm{id}_{\mathcal{A}}$, π)

(1) Retrieve $\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for $\mathrm{id}_{\mathcal{A}}$.

(2) If $g_1^{\pi} = \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}$ then $out := \mathtt{OK}$, else $out := \mathtt{NOK}$.

Party output: (out)

Figure 19: Protocol for Task Guilt Verification

UC-Protocol π_{P6V2G} – Task Blacklist UA

DR input: ($\mathtt{blacklistUA}$, $\mathrm{id}_{\mathcal{A}_{\mathcal{D}}}$)

OPR input: ($\mathtt{blacklistUA}$, $\mathrm{id}_{\mathcal{A}_{O}}$)

(1) For the DR side:
- Load the internally recorded $\mathsf{sk}^{\mathrm{enc}}_{\mathcal{D}}$.
- Retrieve $\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}_{\mathcal{D}}}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for $\mathrm{id}_{\mathcal{A}_{\mathcal{D}}}$.

(2) For the OPR side:
- Load internally recorded set HTD_{λ} and set $HTD^{\mathcal{A}}_{\lambda} = \{htd_{\lambda} \mid (\mathrm{id}_{\mathcal{A}}, htd_{\lambda}) \in HTD_{\lambda}\}$.

(3) Run $\mathtt{BlacklistUA}$ (see fig. 36) for OPR and DR:

$$\begin{pmatrix} (\mathtt{OK}) \\ (\Phi_{\mathcal{A}}) \end{pmatrix} \leftarrow \mathtt{BlacklistUA} \left\langle \begin{matrix} \mathcal{D}\left(\mathsf{sk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}_{\mathcal{D}}}\right) \\ O\left(HTD^{\mathcal{A}}_{\lambda}\right) \end{matrix} \right\rangle$$

(4) For the OPR side:
- Load the internally recorded set Ω_{bl}.
- Let $\Omega^{\mathcal{A}}_{\mathrm{bl}}$ be the subset of blacklist database entries $(\varphi, p, d) \in \Omega_{\mathrm{bl}}$ with fraud detection ID $\varphi \in \Phi_{\mathcal{A}}$.
- $(b^{\mathrm{bill}}, r^{\mathrm{bill}}) := \sum_{(\cdot,\cdot,p,d) \in \Omega^{\mathcal{A}}_{\mathrm{bl}}} (p, d)$

DR output: ($\mathtt{OK}$)

OPR output: (b^{bill}, r^{bill}, $\Phi_{\mathcal{A}}$)

Figure 20: Protocol for Task Blacklist UA

UC-Protocol π_{P6V2G} – Task Blacklist EVCC

DR input: ($\mathtt{blacklistEVCC}$)

OPR input: ($\mathtt{blacklistEVCC}$, $\mathrm{id}_{\mathcal{E}}$)

(1) For the DR side:
- Load the mapping $\{\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}^*}\} \to (\{a_{\mathcal{E}^*}\} \times \{rev\})$ and call it f_{rev}.

(2) For the OPR side:
- Retrieve $\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}$ from $\overline{\mathcal{G}}_{\mathrm{BB}}$ for $\mathrm{id}_{\mathcal{E}}$.

(3) Run $\mathtt{BlacklistEVCC}$ (see fig. 37) for the OPR and DR:

$$\begin{pmatrix} (\mathtt{OK}) \\ (rev) \end{pmatrix} \leftarrow \mathtt{BlacklistEVCC} \left\langle \begin{matrix} \mathcal{D}(f_{\mathrm{rev}}) \\ O\left(\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}\right) \end{matrix} \right\rangle$$

DR output: ($\mathtt{OK}$)

OPR output: (rev)

Figure 21: Protocol for Task Blacklist EVCC

```
RegisterDR(CRS)
--------------------------------
$(\mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}, \mathsf{sk}^{\mathrm{sig}}_{\mathcal{D}}) \leftarrow \mathtt{S2.Gen(CRS)}$
$(\mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{sk}^{\mathrm{enc}}_{\mathcal{D}}) \leftarrow \mathtt{E.Gen(CRS)}$
$(\mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}, \mathsf{sk}^{\mathrm{acc}}_{\mathcal{D}}) \leftarrow \mathtt{ACC.Gen(CRS)}$
$(\mathsf{pk}_{\mathcal{D}}, \mathsf{sk}_{\mathcal{D}}) := ((\mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}, \mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}), (\mathsf{sk}^{\mathrm{sig}}_{\mathcal{D}}, \mathsf{sk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{sk}^{\mathrm{acc}}_{\mathcal{D}}))$
return $(\mathsf{pk}_{\mathcal{D}}, \mathsf{sk}_{\mathcal{D}})$
```

Figure 22: Core Protocol `RegisterDR`

```
RegisterOPR(CRS)
--------------------------------
$(\mathsf{pk}^{\mathrm{sig}}_{O}, \mathsf{sk}^{\mathrm{sig}}_{O}) \leftarrow \mathtt{S3.Gen(CRS)}$
$(\mathsf{pk}^{\mathrm{cert}}_{O}, \mathsf{sk}^{\mathrm{cert}}_{O}) \leftarrow \mathtt{S4.Gen(CRS)}$
$(\mathsf{pk}^{\mathrm{sig}}_{C_O}, \mathsf{sk}^{\mathrm{sig}}_{C_O}) \leftarrow \mathtt{S1.Gen(CRS)}$
$(\mathsf{pk}_{O}, \mathsf{sk}_{O}) := ((\mathsf{pk}^{\mathrm{sig}}_{O}, \mathsf{pk}^{\mathrm{cert}}_{O}, \mathsf{pk}^{\mathrm{sig}}_{C_O}), (\mathsf{sk}^{\mathrm{sig}}_{O}, \mathsf{sk}^{\mathrm{cert}}_{O}, \mathsf{sk}^{\mathrm{sig}}_{C_O}))$
return $(\mathsf{pk}_{O}, \mathsf{sk}_{O})$
```

Figure 23: Core Protocol `RegisterOPR`

```
RegisterSECC(CRS)
--------------------------------
$(\mathsf{pk}^{\mathrm{sig}}_{C}, \mathsf{sk}^{\mathrm{sig}}_{C}) \leftarrow \mathtt{S1.Gen(CRS)}$
return $(\mathsf{pk}^{\mathrm{sig}}_{C}, \mathsf{sk}^{\mathrm{sig}}_{C})$
```

Figure 24: Core Protocol `RegisterSECC`

```
RegisterEVCC(CRS)
--------------------------------
$x \xleftarrow{R} \mathbb{Z}_q$
$(\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{E}}) := (g_1^{x}, x)$
return $(\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{E}})$
```

Figure 25: Core Protocol `RegisterEVCC`

```
RegisterUA(CRS)
--------------------------------
$x \xleftarrow{R} \mathbb{Z}_q$
$(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}) := (g_1^{x}, x)$
return $(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}})$
```

Figure 26: Core Protocol `RegisterUA`

```
$\mathcal{C}(\mathsf{pk}_{\mathcal{C}}^{\text{sig}}, \mathsf{pk}_{\mathcal{O}}^{\text{cert}})$                                   $\mathcal{O}(\mathsf{pk}_{\mathcal{O}}^{\text{cert}}, \mathsf{sk}_{\mathcal{O}}^{\text{cert}}, \mathsf{pk}_{\mathcal{C}}^{\text{sig}}, \mathsf{a}_{\mathcal{C}})$
                                                        $\sigma_{\mathcal{O}}^{\text{cert}} := \mathtt{S4.Sign}(\mathsf{sk}_{\mathcal{O}}^{\text{cert}}, (\mathsf{pk}_{\mathcal{C}}^{\text{sig}}, \mathsf{a}_{\mathcal{C}}))$
                                                        $\mathtt{cert}_{\mathcal{C}} := (\mathsf{pk}_{\mathcal{C}}^{\text{sig}}, \mathsf{a}_{\mathcal{C}}, \mathsf{pk}_{\mathcal{O}}^{\text{cert}}, \sigma_{\mathcal{O}}^{\text{cert}})$
                    $\xleftarrow{\mathtt{cert}_{\mathcal{C}}}$
$\mathtt{parse}\ ((\mathsf{pk}_{\mathcal{C}}^{\text{sig}})^*, \mathsf{a}_{\mathcal{C}}, (\mathsf{pk}_{\mathcal{O}}^{\text{cert}})^*, \sigma_{\mathcal{O}}^{\text{cert}}) := \mathtt{cert}_{\mathcal{C}}$
**if** $\mathtt{S4.Vfy}(\mathsf{pk}_{\mathcal{O}}^{\text{cert}}, \sigma_{\mathcal{O}}^{\text{cert}}, (\mathsf{pk}_{\mathcal{C}}^{\text{sig}}, \mathsf{a}_{\mathcal{C}})) = 0$
    **return** $(\bot)$
**return** $(\mathtt{cert}_{\mathcal{C}})$                                    **return** $(\mathtt{OK})$
```

Figure 27: Core Protocol `CertifySECC`

```
$\mathcal{E}(\mathsf{pk}_{\mathcal{E}}^{\text{id}}, \mathsf{sk}_{\mathcal{E}}^{\text{id}}, \mathsf{pk}_{\mathcal{D}}^{\text{sig}})$                                   $\mathcal{D}(\mathsf{sk}_{\mathcal{D}}^{\text{sig}}, \mathsf{a}_{\mathcal{E}})$
                    $\xleftarrow{\mathsf{a}_{\mathcal{E}}}$
$(\mathsf{c}', \mathsf{d}') := \mathtt{C3.Com}(\mathsf{CRS}, (0, \mathsf{sk}_{\mathcal{E}}^{\text{id}}))$
$\mathit{stmnt} := (\mathsf{c}', \mathsf{pk}_{\mathcal{E}}^{\text{id}})$
$\mathit{wit} := (\mathsf{d}', g_2^{\mathsf{sk}_{\mathcal{E}}^{\text{id}}})$
$\pi := \mathtt{P2.Prove}(\mathsf{CRS}, \mathit{stmnt}, \mathit{wit})$
                    $\xrightarrow{\pi, \mathsf{c}', \mathsf{pk}_{\mathcal{E}}^{\text{id}}}$
                                                        $\mathit{stmnt} := (\mathsf{c}', \mathsf{pk}_{\mathcal{E}}^{\text{id}})$
                                                        **if** $\mathtt{P2.Vfy}(\mathsf{CRS}, \mathit{stmnt}, \pi) = 0$
                                                            **return** $(\bot)$
                                                        $\mathit{rev} \overset{\text{R}}{\leftarrow} \mathbb{Z}_q^*$
                                                        $(\mathsf{c}'', \mathsf{d}'') := \mathtt{C3.Com}(\mathsf{CRS}, (\mathit{rev}, 0))$
                                                        $\mathsf{c} := \mathsf{c}' \cdot \mathsf{c}''$
                                                        $\sigma := \mathtt{S2.Sign}(\mathsf{sk}_{\mathcal{D}}^{\text{sig}}, (\mathsf{c}, \mathsf{a}_{\mathcal{E}}))$
                    $\xleftarrow{\mathsf{d}'', \mathsf{c}, \sigma, \mathit{rev}}$
$\mathsf{d} := \mathsf{d}' \cdot \mathsf{d}''$
**if** $\mathtt{C3.Open}(\mathsf{CRS}, (\mathit{Rev}, \mathsf{pk}_{\mathcal{E}}^{\text{id}}), \mathsf{c}, \mathsf{d}) = 0$
    $\vee\ \mathtt{S2.Vfy}(\mathsf{pk}_{\mathcal{D}}^{\text{sig}}, \sigma, (\mathsf{c}, \mathsf{a}_{\mathcal{E}})) = 0$
    **return** $(\bot)$
$\mathtt{cert}_{\mathcal{E}} := (\mathit{rev}, \mathsf{a}_{\mathcal{E}}, \mathsf{c}, \mathsf{d}, \sigma)$
**return** $(\mathtt{cert}_{\mathcal{E}})$                                    **return** $(\mathit{rev})$
```

Figure 28: Core Protocol `CertifyEVCC`

```
$\mathcal{A}(\mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{pk}^{\mathrm{sig}}_{O}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}})$
$s' \overset{R}{\leftarrow} S$
for $i \in \{0, \ldots, \ell\}$
    $\lambda'_i \overset{R}{\leftarrow} \{0, \ldots, base - 1\}$
    $\Lambda'_i := \mathrm{g}_1^{\lambda'_i}$
$\lambda' = \sum_{i=0}^{\ell} \lambda'_i \cdot base^i$
$\Lambda' := \mathrm{g}_1^{\lambda'}$
$(\mathsf{c}'_{\mathrm{ser}}, \mathsf{d}'_{\mathrm{ser}}) \leftarrow \mathtt{C4.Com}(\mathtt{CRS}, s')$
$(\mathsf{c}'_{\text{pre-seed}}, \mathsf{d}'_{\text{pre-seed}}) \leftarrow \mathtt{C5.Com}(\mathtt{CRS}, \lambda')$

$O(\mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{sk}^{\mathrm{sig}}_{O}, \mathsf{sk}^{\mathrm{sig}}_{C_O}, \mathtt{cert}_{C_O}, \mathsf{a}, r, \mathrm{wl})$
$s'' \overset{R}{\leftarrow} S$
$\lambda'' \overset{R}{\leftarrow} \mathbb{Z}_q$
$\Lambda'' := \mathrm{g}_1^{\lambda''}$

$\xrightarrow{\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}},\ \mathsf{c}'_{\mathrm{ser}},\ \mathsf{c}'_{\text{pre-seed}}}$

if $\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}} \notin \mathrm{wl}$
    return ($\mathtt{notwhitelisted}$)

$\xleftarrow{\mathsf{a},\ r,\ s'',\ \lambda'',\ \mathtt{cert}_{C_O}}$

parse $(\mathsf{pk}^{\mathrm{sig}}_{C_O}, \mathsf{a}_O, \mathsf{pk}^{\mathrm{cert}}_{O}, \sigma^{\mathrm{cert}}_{O}) := \mathtt{cert}_{C_O}$
if $\mathtt{S4.Vfy}(\mathsf{pk}^{\mathrm{cert}}_{O}, \sigma^{\mathrm{cert}}_{O}, (\mathsf{pk}^{\mathrm{sig}}_{C_O}, \mathsf{a}_O)) = 0$
    return $(\bot)$
$s := s' \cdot s''$
$\lambda := \lambda' + \lambda''$
$\Lambda'' := \mathrm{g}_1^{\lambda''}$
$u_1^{\mathrm{next}}, rand_1, rand_2 \overset{R}{\leftarrow} \mathbb{Z}_q$
$\mathsf{e} := \mathtt{E.Enc}(\mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}, (\Lambda'_0, \ldots, \Lambda'_\ell, \Lambda'', \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}); rand_1, rand_2)$
$(\mathsf{c}_O, \mathsf{d}_O) := \mathtt{C1.Com}(\mathtt{CRS}, (\lambda, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}))$
$(\mathsf{c}_{C_O}, \mathsf{d}_{C_O}) := \mathtt{C2.Com}(\mathtt{CRS}, (\lambda, 0, r, u_1^{\mathrm{next}}, 1))$
$stmnt := (\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{e}, \mathsf{c}_O, \mathsf{c}_{C_O}, \mathsf{c}'_{\text{pre-seed}}, \Lambda'', r)$
$wit := (rand_1, rand_2, \lambda, \lambda', \lambda'_0, \ldots, \lambda'_\ell, \mathrm{g}_1^{u_1^{\mathrm{next}}},$
    $\mathrm{g}_1^{\lambda}, \mathrm{g}_1^{\lambda'}, \mathrm{g}_1^{\lambda'_0}, \ldots, \mathrm{g}_1^{\lambda'_\ell}, \mathsf{d}_O, \mathsf{d}_{C_O}, \mathsf{d}'_{\text{pre-seed}}, \mathrm{g}_2^{\mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}})$
$\pi := \mathtt{P1.Prove}(\mathtt{CRS}, stmnt, wit)$

$\xrightarrow{\pi,\ \mathsf{c}_O,\ \mathsf{c}_{C_O},\ \mathsf{d}'_{\mathrm{ser}},\ s',\ \mathsf{e}}$

if $\mathtt{C4.Open}(\mathtt{CRS}, s', \mathsf{c}'_{\mathrm{ser}}, \mathsf{d}'_{\mathrm{ser}}) = 0$
    return $(\bot)$
$s := s' \cdot s''$
$stmnt := (\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{e}, \mathsf{c}_O, \mathsf{c}_{C_O}, \mathsf{c}'_{\text{pre-seed}}, \Lambda'', r)$
if $\mathtt{P1.Vfy}(\mathtt{CRS}, stmnt, \pi) = 0$
    return $(\bot)$

$\vdots$
```

Figure 29: First Part of Core Protocol `IssueWallet`

```
$\mathcal{A}(\mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{pk}^{\mathrm{sig}}_{O}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}})$                                   $O(\mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{sk}^{\mathrm{sig}}_{O}, \mathsf{sk}^{\mathrm{sig}}_{C_O}, \mathsf{cert}_{C_O}, \mathsf{a}, r, \mathrm{wl})$

                              ⋮                        $\sigma_O := \mathtt{S3.Sign}(\mathsf{sk}^{\mathrm{sig}}_{O}, (\mathsf{c}_O, \mathsf{a}))$
                                                       $\sigma_{C_O} := \mathtt{S1.Sign}(\mathsf{sk}^{\mathrm{sig}}_{C_O}, (\mathsf{c}_{C_O}, s))$

                              $\xleftarrow{\sigma_{C_O},\ \sigma_O}$

$\varphi := \mathsf{PRF}(\lambda, 0)$
$\tau := \big(s \,\big|\, 0, r \,\big|\, 1, u_1^{\mathrm{next}} \,\big|\, (\mathsf{c}, \mathsf{d}, \sigma)_{C_O}, \mathsf{cert}_{C_O} \,\big|\, \lambda, \mathsf{a}, (\mathsf{c}, \mathsf{d}, \sigma)_O, \mathsf{pk}^{\mathrm{sig}}_{O}\big)$        $htd_\lambda := (\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, s, \lambda'', \mathsf{e})$
return $(\tau)$                                        return $(s, htd_\lambda)$
```

Figure 30: Continuation of Core Protocol `IssueWallet`

```
$\mathcal{E}(\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{E}}, \mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}, \mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}, \mathsf{cert}_{\mathcal{E}}, \beta)$                       $C(\mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}, \mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}, \mathrm{bl}_{\mathrm{EVCC}}, \mu)$

                              $\xleftarrow{\mathrm{bl}_{\mathrm{EVCC}}}$

parse $(rev, \mathsf{a}_{\mathcal{E}}, \mathsf{c}, \mathsf{d}, \sigma) := \mathsf{cert}_{\mathcal{E}}$
if $rev \in \mathrm{bl}_{\mathrm{EVCC}}$
    return (blacklisted)
$v := \mathtt{ACC.Evaluate}(\mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}, \mathrm{bl}_{\mathrm{EVCC}})$
$w := \mathtt{ACC.InitWit}(\mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}, rev, \mathrm{bl}_{\mathrm{EVCC}})$
$(\mathsf{c}_{rev}, \mathsf{d}_{rev}) := \mathtt{C6.Com}(\mathsf{CRS}, rev)$
$stmnt_1 := (\mathsf{a}_{\mathcal{E}}, \mathsf{c}_{rev}, \mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}})$
$wit_1 := (\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}, \mathsf{g}_1^{rev}, \mathsf{d}, \mathsf{d}_{rev}, \mathsf{g}_2^{\mathsf{sk}^{\mathrm{id}}_{\mathcal{E}}}, \mathsf{c}, \sigma)$
$\pi_1 = \mathtt{P31.Prove}(\mathsf{CRS}, stmnt_1, wit_1)$
$stmnt_2 := (\mathsf{c}_{rev}, v)$
$wit_2 = (rev, \mathsf{g}_1^{rev}, \mathsf{d}_{rev}, w)$
$\pi_2 = \mathtt{P32.Prove}(\mathsf{CRS}, stmnt_2, wit_2)$

                              $\xrightarrow{\beta,\ \mathsf{a}_{\mathcal{E}},\ \mathsf{c}_{rev},\ \pi_1,\ \pi_2}$

                                                       $stmnt_1 := (\mathsf{a}_{\mathcal{E}}, \mathsf{c}_{rev}, \mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}})$
                                                       if $\mathtt{P31.Vfy}(\mathsf{CRS}, stmnt_1, \pi_1) = 0$
                                                           return $(\perp)$
                                                       $v := \mathtt{ACC.Evaluate}(\mathsf{pk}^{\mathrm{acc}}_{\mathcal{D}}, \mathrm{bl}_{\mathrm{EVCC}})$
                                                       $stmnt_2 := (\mathsf{c}_{rev}, v)$
                                                       if $\mathtt{P32.Vfy}(\mathsf{CRS}, stmnt_2, \pi_2) = 0$
                                                           return (blacklisted)

                              $\xleftarrow{\mu}$

                              $\xrightarrow{\mathtt{OK}}$

return $(\mu)$                                         return $(\mathsf{a}_{\mathcal{E}}, \beta)$
```

Figure 31: Core Protocol `CheckEVCC`

```
$\mathcal{A}(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}, \tau^{\mathrm{prev}})$                                   $C(\mathsf{sk}^{\mathrm{sig}}_{C}, \mathtt{cert}_C, \mathrm{bl}_{\mathrm{UA}}, p, d)$

                                                                    $s'' \overset{\mathrm{R}}{\leftarrow} S$
                                                                    $u_2 \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q$
                                                                    $(\mathsf{c}''_{\mathrm{ser}}, \mathsf{d}''_{\mathrm{ser}}) := \mathtt{C4.Com}(\mathtt{CRS}, s'')$

                              $\xleftarrow{\mathtt{cert}_C,\ \mathsf{c}''_{\mathrm{ser}},\ u_2}$

$s' \overset{\mathrm{R}}{\leftarrow} S$
parse $(\mathsf{pk}^{\mathrm{sig}}_{C}, \mathsf{a}_C, \mathsf{pk}^{\mathrm{cert}}_{O_C}, \sigma^{\mathrm{cert}}_{O_C}) := \mathtt{cert}_C$
if $\mathtt{S4.Vfy}(\mathsf{pk}^{\mathrm{cert}}_{O_C}, \sigma^{\mathrm{cert}}_{O_C}, (\mathsf{pk}^{\mathrm{sig}}_{C}, \mathsf{a}_C)) = 0$
    return $(\bot)$
parse $(s^{\mathrm{prev}} | b^{\mathrm{prev}}, r^{\mathrm{prev}} | x, u_1 | (\mathsf{c}, \mathsf{d}, \sigma)_{C^{\mathrm{prev}}}, \mathtt{cert}_{C^{\mathrm{prev}}} |$
    $\lambda, \mathsf{a}_\lambda, (\mathsf{c}, \mathsf{d}, \sigma)_{O_\lambda}, \mathsf{pk}^{\mathrm{sig}}_{O_\lambda}) := \tau^{\mathrm{prev}}$
$t := \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}} \cdot u_2 + u_1 \mod q$
$u_1^{\mathrm{next}} \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q$
$(\mathsf{c}'_C, \mathsf{d}'_C) := \mathtt{C2.Com}(\mathtt{CRS}, (\lambda, b^{\mathrm{prev}}, r^{\mathrm{prev}}, u_1^{\mathrm{next}}, x))$
$\varphi^{\mathrm{prev}} := \mathtt{PRF}(\lambda, x-1)$
$\varphi := \mathtt{PRF}(\lambda, x)$
parse $(\mathsf{pk}^{\mathrm{sig}}_{C^{\mathrm{prev}}}, \mathsf{a}_{C^{\mathrm{prev}}}, \mathsf{pk}^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}}, \sigma^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}}) := \mathtt{cert}_{C^{\mathrm{prev}}}$
$stmnt := (\varphi, t, u_2, \mathsf{a}_\lambda, \mathsf{a}_{C^{\mathrm{prev}}}, \mathsf{c}'_C, \mathsf{pk}^{\mathrm{sig}}_{O_\lambda}, \mathsf{pk}^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}})$
$wit := (\lambda, u_1, x, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}, s^{\mathrm{prev}}, \varphi^{\mathrm{prev}}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathrm{g}_1^{\lambda}, \mathrm{g}_1^{u_1}, \mathrm{g}_1^{u_1^{\mathrm{next}}},$
    $\mathrm{g}_1^{x}, \mathrm{g}_1^{b^{\mathrm{prev}}}, \mathrm{g}_1^{r^{\mathrm{prev}}}, \mathsf{d}'_C, (\mathsf{c}, \mathsf{d}, \sigma)_{C^{\mathrm{prev}}}, \sigma^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}}, \mathsf{pk}^{\mathrm{sig}}_{C^{\mathrm{prev}}},$
    $(\mathsf{c}, \mathsf{d}, \sigma)_{O_\lambda})$
$\pi := \mathtt{P4.Prove}(\mathtt{CRS}, stmnt, wit)$

                              $\pi, s', \varphi, t, \mathsf{c}'_C, \mathsf{a}_\lambda,$
                              $\xrightarrow{\mathsf{a}_{C^{\mathrm{prev}}},\ \mathsf{pk}^{\mathrm{sig}}_{O_\lambda},\ \mathsf{pk}^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}}}$

                                                                    $stmnt := (\varphi, t, u_2, \mathsf{a}_\lambda, \mathsf{a}_{C^{\mathrm{prev}}}, \mathsf{c}'_C, \mathsf{pk}^{\mathrm{sig}}_{O_\lambda}, \mathsf{pk}^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}})$
                                                                    if $\mathtt{P4.Vfy}(\mathtt{CRS}, stmnt, \pi) = 0$
                                                                        return $(\bot)$
                                                                    if $\varphi \in \mathrm{bl}_{\mathrm{UA}}$
                                                                        return $(\mathtt{blacklisted})$
                                                                    $s := s' \cdot s''$
                                                                    $(\mathsf{c}''_C, \mathsf{d}''_C) := \mathtt{C2.Com}(\mathtt{CRS}, (0, p, d, 0, 1))$
                                                                    $\mathsf{c}_C := \mathsf{c}'_C \cdot \mathsf{c}''_C$
                                                                    $\sigma_C := \mathtt{S1.Sign}(\mathsf{sk}^{\mathrm{sig}}_{C}, (\mathsf{c}_C, s))$

                              $\xleftarrow{s'',\ \mathsf{d}''_{\mathrm{ser}},\ \mathsf{c}_C,\ \mathsf{d}''_C,\ \sigma_C,\ p,\ d}$

if $\mathtt{C4.Open}(\mathtt{CRS}, s'', \mathsf{c}''_{\mathrm{ser}}, \mathsf{d}''_{\mathrm{ser}}) = 0$
    return $(\bot)$                                 ⋮
```

Figure 32: First Part of Core Protocol `DebtAccum`

```
$\mathcal{A}(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}, \tau^{\mathrm{prev}})$                    $C(\mathsf{sk}^{\mathrm{sig}}_{C}, \mathtt{cert}_C, \mathrm{bl}_{\mathrm{UA}}, p, d)$

                              $\vdots$
$s := s' \cdot s''$
$\mathsf{d}_C := \mathsf{d}'_C \cdot \mathsf{d}''_C$
$b := b^{\mathrm{prev}} + p$
$r := r^{\mathrm{prev}} + d$
$\tau := \big(s \big| b, r \big| (x+1), u_1^{\mathrm{next}} \big| (\mathsf{c}, \mathsf{d}, \sigma)_C, \mathtt{cert}_C \big|$        $\omega_{\mathrm{bl}} := (\varphi, p, d)$
    $\lambda, \mathsf{a}_\lambda, (\mathsf{c}, \mathsf{d}, \sigma)_{O_\lambda}, \mathsf{pk}^{\mathrm{sig}}_{O_\lambda}\big)$        $\omega_{\mathrm{ds}} := (\varphi, t, u_2)$
return $(\tau, p, d)$                    return $(\mathsf{a}_\lambda, \mathsf{pk}^{\mathrm{sig}}_{O_\lambda}, s, \omega_{\mathrm{bl}}, \omega_{\mathrm{ds}})$
```

Figure 33: Continuation of Core Protocol `DebtAccum`

```
$\mathcal{A}(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}, \tau^{\mathrm{prev}})$                    $O(\mathsf{pk}^{\mathrm{sig}}_{O})$

                                        $u_2 \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q$
                    $\xleftarrow{\quad u_2 \quad}$
parse $\big(s^{\mathrm{prev}} \big| b^{\mathrm{prev}}, r^{\mathrm{prev}} \big| x, u_1 \big| (\mathsf{c}, \mathsf{d}, \sigma)_{C^{\mathrm{prev}}}, \mathtt{cert}_{C^{\mathrm{prev}}} \big|$
    $\lambda, \mathsf{a}_\lambda, (\mathsf{c}, \mathsf{d}, \sigma)_{O_\lambda}, \mathsf{pk}^{\mathrm{sig}}_{O_\lambda}\big) := \tau^{\mathrm{prev}}$
$t := \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}} \cdot u_2 + u_1 \mod q$
$\varphi^{\mathrm{prev}} := \mathsf{PRF}(\lambda, x-1)$
$\varphi := \mathsf{PRF}(\lambda, x)$
parse $(\mathsf{pk}^{\mathrm{sig}}_{C^{\mathrm{prev}}}, \mathsf{a}_{C^{\mathrm{prev}}}, \mathsf{pk}^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}}, \sigma^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}}) := \mathtt{cert}_{C^{\mathrm{prev}}}$

$stmnt := (\varphi, t, u_2, \mathsf{a}_\lambda, \mathsf{a}_{C^{\mathrm{prev}}}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{pk}^{\mathrm{sig}}_{O_\lambda}, \mathsf{pk}^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}},$
    $\mathrm{g}_1^{b^{\mathrm{prev}}}, \mathrm{g}_1^{r^{\mathrm{prev}}})$
$wit := (\lambda, u_1, x, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}, \varphi^{\mathrm{prev}}, \Lambda, U_1, X, s^{\mathrm{prev}},$
    $(\mathsf{c}, \mathsf{d}, \sigma)_{C^{\mathrm{prev}}}, \sigma^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}}, \mathsf{pk}^{\mathrm{sig}}_{C^{\mathrm{prev}}}, (\mathsf{c}, \mathsf{d}, \sigma)_{O_\lambda})$
$\pi := \mathsf{P5.Prove}(\mathsf{CRS}, stmnt, wit)$
                    $\pi, \varphi, t, b^{\mathrm{prev}}, r^{\mathrm{prev}}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}},$
                    $\mathsf{a}_\lambda, \mathsf{a}_{C^{\mathrm{prev}}}, \mathsf{pk}^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}}$
                    $\xrightarrow{\qquad\qquad}$
                                        $stmnt := (\varphi, t, u_2, \mathsf{a}_\lambda, \mathsf{a}_{C^{\mathrm{prev}}}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{pk}^{\mathrm{sig}}_{O}, \mathsf{pk}^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}},$
                                            $\mathrm{g}_1^{b^{\mathrm{prev}}}, \mathrm{g}_1^{r^{\mathrm{prev}}})$
                                        if $\mathsf{P5.Vfy}(\mathsf{CRS}, stmnt, \pi) = 0$
                                            return $(\bot)$
                    $\xleftarrow{\quad \mathsf{OK} \quad}$
                                        $\omega_{\mathrm{bl}} := (\varphi, -b^{\mathrm{prev}}, 0)$
                                        $\omega_{\mathrm{ds}} := (\varphi, t, u_2)$
return $(b^{\mathrm{prev}}, r^{\mathrm{prev}})$                    return $(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathsf{a}_\lambda, \omega_{\mathrm{bl}}, \omega_{\mathrm{ds}}, r^{\mathrm{prev}}, \mathsf{a}_{C^{\mathrm{prev}}}, \mathsf{pk}^{\mathrm{cert}}_{O_{C^{\mathrm{prev}}}})$
```

Figure 34: Core Protocol `DebtClearance`

```
$\mathcal{A}(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \tau)$

parse $(s|b, r|x^{\mathrm{next}}, u_1^{\mathrm{next}}|(\mathsf{c}, \mathsf{d}, \sigma)_C, \mathsf{cert}_C|\lambda, \mathsf{a}_\lambda, (\mathsf{c}, \mathsf{d}, \sigma)_{O_\lambda}, \mathsf{pk}^{\mathrm{sig}}_{O_\lambda}) := \tau$
parse $(\mathsf{pk}^{\mathrm{sig}}_C, \mathsf{a}_C, \mathsf{pk}^{\mathrm{cert}}_{O_C}, \sigma^{\mathrm{cert}}_{O_C}) := \mathsf{cert}_C$
if $\big(\mathsf{C1.Open}(\mathsf{CRS}, (\mathrm{g}_1^{\lambda}, \mathrm{g}_1^{\mathsf{a}_\lambda}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}), \mathsf{c}_{O_\lambda}, \mathsf{d}_{O_\lambda}) = 1 \quad \wedge \quad \mathsf{C2.Open}(\mathsf{CRS}, (\mathrm{g}_1^{\lambda}, \mathrm{g}_1^{b}, \mathrm{g}_1^{r}, \mathrm{g}_1^{u_1}, \mathrm{g}_1^{x}), \mathsf{c}_C, \mathsf{d}_C) = 1$
    $\wedge \quad \mathsf{S3.Vfy}(\mathsf{pk}^{\mathrm{sig}}_{O_\lambda}, \sigma_{O_\lambda}, (\mathsf{c}_{O_\lambda}, \mathsf{a}_\lambda)) = 1 \quad \wedge \quad \mathsf{S1.Vfy}(\mathsf{pk}^{\mathrm{sig}}_C, \sigma_C, (\mathsf{c}_C, s)) = 1 \quad \wedge \quad \mathsf{S4.Vfy}(\mathsf{pk}^{\mathrm{cert}}_{O_C}, \sigma^{\mathrm{cert}}_{O_C}, (\mathsf{pk}^{\mathrm{sig}}_C, \mathsf{a}_C)) = 1\big)$
  return (OK)
else
  return (NOK)
```

Figure 35: Core Protocol `VerifyWallet`

```
$\mathcal{D}(\mathsf{sk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}_{\mathcal{D}}})$                                                  $O(HTD_\lambda)$

                                        $\xleftarrow{\quad HTD_\lambda \quad}$

$\Phi_{\mathcal{A}} := \emptyset$
for $htd_\lambda \in HTD_\lambda$
  parse $(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}_O}, s, \lambda'', \mathsf{e}) := htd_\lambda$
  $(\Lambda'_0, \ldots, \Lambda'_\ell, \Lambda'', \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}_O}) \leftarrow \mathsf{E.Dec}(\mathsf{sk}^{\mathrm{enc}}_{\mathcal{D}}, \mathsf{e})$
  if encryption fails $\vee\ \Lambda'' \neq \mathrm{g}_1^{\lambda''} \vee \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}_{\mathcal{D}}} \neq \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}_O}$
    return $(\perp)$
  $\lambda := \lambda'' + \sum_{i=0}^{\ell} \mathrm{DLOG}(\Lambda'_i) \cdot base^i$
  $\Phi_{\mathcal{A}} := \Phi_{\mathcal{A}} \cup \{\mathsf{PRF}(\lambda, 0), \ldots, \mathsf{PRF}(\lambda, x_{\mathrm{bl}})\}$

                                        $\xrightarrow{\quad \Phi_{\mathcal{A}} \quad}$

return (OK)                                                                      return $(\Phi_{\mathcal{A}})$
```

Figure 36: Core Protocol `BlacklistUA`

```
$\mathcal{D}(f_{\mathrm{rev}})$                                                  $O(\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}})$

                                   $\xleftarrow{\quad \mathsf{pk}^{\mathrm{id}}_{\mathcal{E}} \quad}$

$(\mathsf{a}_{\mathcal{E}}, rev) := f_{\mathrm{rev}}(\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}})$

                                   $\xrightarrow{\quad rev \quad}$

return (OK)                                                          return $(rev)$
```

Figure 37: Core Protocol `BlacklistEVCC`

$$\mathsf{L}^{(2)} := \left\{ \begin{pmatrix} \mathsf{c} \\ \mathsf{pk}^{\mathrm{id}}_{\mathcal{E}} \end{pmatrix}^{\top} \left| \begin{array}{l} \exists\, \mathsf{d} \in \mathbb{G}_1; Sk^{\mathrm{id}}_{\mathcal{E}} \in \mathbb{G}_2 : \\ \mathtt{C3.Open}(\mathsf{CRS}, (\mathbb{1}_{\mathbb{G}_1}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}), \mathsf{c}, \mathsf{d}) = 1, \\ e(\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}, \mathrm{g}_2) = e(\mathrm{g}_1, Sk^{\mathrm{id}}_{\mathcal{E}}) \end{array} \right. \right\}$$

Figure 38: Language used in Core Protocol `CertifyEVCC`

$$\mathsf{L}^{(1)} := \left\{ \begin{pmatrix} \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}} \\ \mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}} \\ \mathsf{e} \\ \mathsf{c}_{\mathcal{O}} \\ \mathsf{c}_{\mathcal{C}_{\mathcal{O}}} \\ \mathsf{c}'_{\text{pre-seed}} \\ \Lambda'', \\ R \end{pmatrix}^{\top} \left| \begin{array}{l} \exists\, rand_1, rand_2, \lambda, \lambda', \lambda'_0, \ldots, \lambda'_{\ell} \in \mathbb{Z}_q; \\ U_1^{\mathrm{next}}, \Lambda, \Lambda', \Lambda'_0, \ldots, \Lambda'_{\ell}, \mathsf{d}_{\mathcal{O}}, \mathsf{d}_{\mathcal{C}_{\mathcal{O}}}, \mathsf{d}'_{\text{pre-seed}} \in \mathbb{G}_1; \\ Sk^{\mathrm{id}}_{\mathcal{A}} \in \mathbb{G}_2 : \\ \mathtt{C1.Open}(\mathsf{CRS}, (\Lambda, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}), \mathsf{c}_{\mathcal{O}}, \mathsf{d}_{\mathcal{O}}) = 1, \\ \mathtt{C2.Open}(\mathsf{CRS}, (\Lambda, \mathbb{1}_{\mathbb{G}_1}, R, U_1^{\mathrm{next}}, \mathbb{1}_{\mathbb{G}_1}), \mathsf{c}_{\mathcal{C}_{\mathcal{O}}}, \mathsf{d}_{\mathcal{C}_{\mathcal{O}}}) = 1, \\ \mathtt{C5.Open}(\mathsf{CRS}, \Lambda', \mathsf{c}_{\text{pre-seed}}, \mathsf{d}_{\text{pre-seed}}) = 1, \\ e(\mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \mathrm{g}_2) = e(\mathrm{g}_1, Sk^{\mathrm{id}}_{\mathcal{A}}), \\ \Lambda = \Lambda' \cdot \Lambda'', \Lambda = \mathrm{g}_1^{\lambda}, \Lambda' = \mathrm{g}_1^{\lambda'}, \\ \lambda' = \sum_{i=0}^{\ell} \lambda'_i \cdot base^i, \\ \mathsf{e} = \mathsf{E.Enc}(\mathsf{pk}^{\mathrm{enc}}_{\mathcal{D}}, (\Lambda'_0, \ldots, \Lambda'_{\ell}, \Lambda'', \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}); rand_1, rand_2), \\ \forall i \in \{0, \ldots, \ell\} : \\ \quad \lambda'_i \in \{0, \ldots, base - 1\}, \\ \quad \Lambda'_i = \mathrm{g}_1^{\lambda'_i} \end{array} \right. \right\}$$

Figure 39: Language used in Core Protocol `IssueWallet`

$$\mathsf{L}^{(31)} := \left\{ \begin{pmatrix} \mathsf{a}_{\mathcal{E}} \\ \mathsf{c}_{rev} \\ \mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}} \end{pmatrix}^{\top} \left| \begin{array}{l} \exists\, \mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}, Rev, \mathsf{d}, \mathsf{d}_{rev} \in \mathbb{G}_1; Sk^{\mathrm{id}}_{\mathcal{E}}, \mathsf{c} \in \mathbb{G}_2; \sigma \in \mathbb{G}_1 \times \mathbb{G}_2^2 : \\ \mathtt{C3.Open}(\mathsf{CRS}, (Rev, \mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}), \mathsf{c}, \mathsf{d}) = 1, \\ \mathtt{C6.Open}(\mathsf{CRS}, Rev, \mathsf{c}_{rev}, \mathsf{d}_{rev}) = 1, \\ \mathtt{S2.Vfy}(\mathsf{pk}^{\mathrm{sig}}_{\mathcal{D}}, \sigma, (\mathsf{c}, \mathsf{a}_{\mathcal{E}})) = 1, \\ e(\mathsf{pk}^{\mathrm{id}}_{\mathcal{E}}, \mathrm{g}_2) = e(\mathrm{g}_1, Sk^{\mathrm{id}}_{\mathcal{E}}) \end{array} \right. \right\}$$

$$\mathsf{L}^{(32)} := \left\{ \begin{pmatrix} \mathsf{c}_{rev} \\ v \end{pmatrix}^{\top} \left| \begin{array}{l} \exists\, rev \in \mathbb{Z}_q; Rev, \mathsf{d}_{rev} \in \mathbb{G}_1; w \in \mathbb{G}_1 \times \mathbb{Z}_q \\ \mathtt{C6.Open}(\mathsf{CRS}, Rev, \mathsf{c}_{rev}, \mathsf{d}_{rev}) = 1, \\ Rev = \mathrm{g}_1^{rev}, \\ \Omega(w, rev, v) = 1 \end{array} \right. \right\}$$

Figure 40: Languages used in Core Protocol `CheckEVCC`

$$\mathsf{L}^{(4)} := \left\{ \begin{pmatrix} \varphi \\ t \\ u_2 \\ \mathsf{a}_{\lambda} \\ \mathsf{a}_{\mathcal{C}^{\mathrm{prev}}} \\ \mathsf{c}'_{\mathcal{C}} \\ \mathsf{pk}^{\mathrm{sig}}_{\mathcal{O}_{\lambda}} \\ \mathsf{pk}^{\mathrm{cert}}_{\mathcal{O}_{\mathcal{C}^{\mathrm{prev}}}} \end{pmatrix}^{\top} \left| \begin{array}{l} \exists\, \lambda, u_1, x, \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}} \in \mathbb{Z}_q; \\ s^{\mathrm{prev}}, \varphi^{\mathrm{prev}}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}, \Lambda, U_1, U_1^{\mathrm{next}}, X, B^{\mathrm{prev}}, R^{\mathrm{prev}}, \mathsf{d}'_{\mathcal{C}}, \mathsf{d}_{\mathcal{O}}, \mathsf{d}_{\mathcal{C}^{\mathrm{prev}}} \in \mathbb{G}_1; \\ \mathsf{c}_{\mathcal{O}_{\lambda}}, \mathsf{c}_{\mathcal{C}^{\mathrm{prev}}} \in \mathbb{G}_2; \\ \sigma_{\mathcal{O}_{\lambda}}, \sigma_{\mathcal{C}^{\mathrm{prev}}}, \sigma^{\mathrm{cert}}_{\mathcal{O}_{\mathcal{C}^{\mathrm{prev}}}} \in \mathbb{G}_1 \times \mathbb{G}_2^2; \\ \mathsf{pk}^{\mathrm{sig}}_{\mathcal{C}^{\mathrm{prev}}} \in \mathbb{G}_1^3 \times \mathbb{G}_2 : \\ \mathtt{C2.Open}(\mathsf{CRS}, (\Lambda, B^{\mathrm{prev}}, R^{\mathrm{prev}}, U_1^{\mathrm{next}}, X), \mathsf{c}'_{\mathcal{C}}, \mathsf{d}'_{\mathcal{C}}) = 1, \\ \mathtt{C1.Open}(\mathsf{CRS}, (\Lambda, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}}), \mathsf{c}_{\mathcal{O}_{\lambda}}, \mathsf{d}_{\mathcal{O}_{\lambda}}) = 1, \\ \mathtt{C2.Open}(\mathsf{CRS}, (\Lambda, B^{\mathrm{prev}}, R^{\mathrm{prev}}, U_1, X), \mathsf{c}_{\mathcal{C}^{\mathrm{prev}}}, \mathsf{d}_{\mathcal{C}^{\mathrm{prev}}}) = 1, \\ \mathtt{S3.Vfy}(\mathsf{pk}^{\mathrm{sig}}_{\mathcal{O}_{\lambda}}, \sigma_{\mathcal{O}_{\lambda}}, (\mathsf{c}_{\mathcal{O}_{\lambda}}, \mathsf{a}_{\lambda})) = 1, \\ \mathtt{S1.Vfy}(\mathsf{pk}^{\mathrm{sig}}_{\mathcal{C}^{\mathrm{prev}}}, \sigma_{\mathcal{C}^{\mathrm{prev}}}, (\mathsf{c}^{\mathrm{prev}}_{\mathcal{C}}, s^{\mathrm{prev}})) = 1, \\ \mathtt{S4.Vfy}(\mathsf{pk}^{\mathrm{cert}}_{\mathcal{O}_{\mathcal{C}^{\mathrm{prev}}}}, \sigma^{\mathrm{cert}}_{\mathcal{O}_{\mathcal{C}^{\mathrm{prev}}}}, (\mathsf{pk}^{\mathrm{sig}}_{\mathcal{C}^{\mathrm{prev}}}, \mathsf{a}_{\mathcal{C}^{\mathrm{prev}}})) = 1, \\ \varphi^{\mathrm{prev}} = \mathsf{PRF}(\lambda, x - 1), \varphi = \mathsf{PRF}(\lambda, x), \\ t = \mathsf{sk}^{\mathrm{id}}_{\mathcal{A}} \cdot u_2 + u_1 \mod q, \\ \Lambda = \mathrm{g}_1^{\lambda}, U_1 = \mathrm{g}_1^{u_1}, X = \mathrm{g}_1^{x}, \mathsf{pk}^{\mathrm{id}}_{\mathcal{A}} = \mathrm{g}_1^{\mathsf{sk}^{\mathrm{id}}_{\mathcal{A}}} \end{array} \right. \right\}$$

Figure 41: Language used in Core Protocol `DebtAccum`

$$\mathsf{L}^{(5)} := \left\{ \begin{pmatrix} \varphi \\ t \\ u_2 \\ \mathsf{a}_\lambda \\ \mathsf{a}_{C^{\text{prev}}} \\ \mathsf{pk}^{\text{id}}_{\mathcal{A}} \\ \mathsf{pk}^{\text{sig}}_{O_\lambda} \\ \mathsf{pk}^{\text{cert}}_{O_{C^{\text{prev}}}} \\ B^{\text{prev}} \\ R^{\text{prev}} \end{pmatrix}^{\top} \middle| \begin{array}{l} \exists\, \lambda, u_1, x, \mathsf{sk}^{\text{id}}_{\mathcal{A}} \in \mathbb{Z}_q; \\ \varphi^{\text{prev}}, \Lambda, U_1, X, \mathsf{d}_{O_\lambda}, \mathsf{d}_{C^{\text{prev}}} \in \mathbb{G}_1; \\ \mathsf{c}_{O_\lambda}, \mathsf{c}_{C^{\text{prev}}}, s^{\text{prev}} \in \mathbb{G}_2; \\ \sigma_{O_\lambda}, \sigma_{C^{\text{prev}}}, \sigma^{\text{cert}}_{O_{C^{\text{prev}}}} \in \mathbb{G}_1 \times \mathbb{G}_2^2; \\ \mathsf{pk}^{\text{sig}}_{C^{\text{prev}}} \in \mathbb{G}_1^3 \times \mathbb{G}_2 : \\ \mathtt{C1.Open}(\mathsf{CRS}, (\Lambda, \mathsf{pk}^{\text{id}}_{\mathcal{A}}), \mathsf{c}_{O_\lambda}, \mathsf{d}_{O_\lambda}) = 1, \\ \mathtt{C2.Open}(\mathsf{CRS}, (\Lambda, B^{\text{prev}}, R^{\text{prev}}, U_1, X), \mathsf{c}_{C^{\text{prev}}}, \mathsf{d}_{C^{\text{prev}}}) = 1, \\ \mathtt{S3.Vfy}(\mathsf{pk}^{\text{sig}}_{O_\lambda}, \sigma_{O_\lambda}, (\mathsf{c}_{O_\lambda}, \mathsf{a}_\lambda)) = 1, \\ \mathtt{S1.Vfy}(\mathsf{pk}^{\text{sig}}_{C^{\text{prev}}}, \sigma_{C^{\text{prev}}}, (\mathsf{c}_{C^{\text{prev}}}, s^{\text{prev}})) = 1, \\ \mathtt{S4.Vfy}(\mathsf{pk}^{\text{cert}}_{O_{C^{\text{prev}}}}, \sigma^{\text{cert}}_{O_{C^{\text{prev}}}}, (\mathsf{pk}^{\text{sig}}_{C^{\text{prev}}}, \mathsf{a}_{C^{\text{prev}}})) = 1, \\ \varphi^{\text{prev}} = \mathsf{PRF}(\lambda, x-1), \varphi = \mathsf{PRF}(\lambda, x), \\ t = \mathsf{sk}^{\text{id}}_{\mathcal{A}} \cdot u_2 + u_1 \mod q, \\ \Lambda = \mathrm{g}_1^{\lambda}, U_1 = \mathrm{g}_1^{u_1}, X = \mathrm{g}_1^{x}, \mathsf{pk}^{\text{id}}_{\mathcal{A}} = \mathrm{g}_1^{\mathsf{sk}^{\text{id}}_{\mathcal{A}}}, \end{array} \right\}$$

Figure 42: Language used in Core Protocol `DebtClearance`