Enhancing Location Privacy for Electric Vehicles (at the *right* time) *

Joseph K. Liu[1], Man Ho Au[2], Willy Susilo[2], and Jianying Zhou[1]

[1] Cryptography and Security Department
Institute for Infocomm Research, Singapore
{ksliu, jyzhou}@i2r.a-star.edu.sg
[2] School of Computer Science and Software Engineering
University of Wollongong, Australia
{aau, wsusilo}@uow.edu.au

Abstract. An electric vehicle is a promising and futuristic automobile propelled by electric motor(s), using electrical energy stored in batteries or another energy storage device. Due to the need of battery recharging, the cars will be required to visit recharging infrastructure very frequently. This may disclose the users' private information, such as their location, which may expose users' privacy. In this paper, we provide mechanisms to enhance location privacy of electric vehicles at the right time, by proposing an anonymous payment system with privacy protection support. Our technique further allows traceability in the case where the cars are stolen.

1 Introduction

An electric vehicle (also known as EV) is powered by an electric motor instead of a gasoline engine. The electric motor obtains energy from a controller, which regulates the amount of power based on the driver's use of an accelerator pedal. The electric vehicle uses energy stored in its rechargeable batteries, which can be recharged by the common household electricity for normal charging (slow charging). Electric vehicles have several potential benefits compared to conventional internal combustion automobiles that include a significant reduction of urban air pollution as they do not emit harmful tailpipe pollutants from the onboard source of power at the point of operation (zero tail pipe emissions); reduced greenhouse gas emissions from the onboard source of power depending on the fuel and technology used for electricity generation to charge the batteries; and less dependence on foreign oil, which for many developed and emerging countries is a cause of concern due to its vulnerability to price shocks and supply disruption.

Future electric vehicles may even support Vehicle-to-grid (V2G). The concept allows V2G cars to provide power to help balance loads by "valley filling" (charging at night when the demand is low) and "peak shaving" (sending power back to the grid when the demand is high). It can enable utilizing new ways to provide regulation services (keeping voltage and frequency stable) and provide spinning reserves (meet sudden demands for power). In future development, it has been proposed that such use of electric vehicles could buffer renewable power sources such as wind power, for example, by storing excess energy produced during windy periods and providing it back to the grid during high load periods, thus effectively stabilizing the intermittency of wind power. If the car has installed solar panel, it may further generate additional electricity and sell it back to utilities when it is parked outside under a sunshine. One may also regard this application of vehicle-to-grid technology as a renewable energy approach that can penetrate the baseline electric market.

Despite their potential benefits, widespread adoption of electric vehicles faces several hurdles and limitations. One of the major problems is the driving range. Most electric vehicles can only go about 100 to 150 km before recharging, while gasoline vehicles can go over 500 km before refueling. This may be sufficient for city trips or other short hauls. Nevertheless, people can be concerned that they

* This is the full version of the paper published in ESORICS 2012.

would run out of energy from their battery before reaching their destination, a worry known as the range anxiety.

One of the solutions is to install more fast charging stations with high-speed charging capability so that consumers could recharge the 100 km battery of their electric vehicle to 80 percent in about 30 minutes. electric vehicle drivers may then charge their vehicles at their homes, offices, shopping malls or car parks outside restaurants when they are having dinner.

Location Privacy Concern. Paying for recharging in those infrastructures may disclose the users private information such as their location privacies [16]. Those location privacies include the drivers' living places, working companies, the amusement places they usually go, etc. [24, 25, 23]. Privacy is regarded as a fundamental human right and leaking them is possible to identify at many negative effects [17, 3, 19]. The first one is Location-based "spam", which means that the location information could be used by malicious businesses to bombard an individual with unsolicited marketing for products or services related to that individuals location. Another negative effect is that the location can be used to infer an individual's political views, state of health, or personal preferences. Furthermore, the disclosure of location privacy may also result in safety problems. For example, it may be used by unscrupulous persons such as the robbers for stalking or physical attacks.

The location privacy problem does not exist in gasoline cars. There are two main reasons for that. First, when the car is running out of gasoline, drivers may choose to pay cash instead of credit card when they pay for the gasoline in the gas station. Cash is a form of anonymous payment that cannot be traced. Second, gasoline vehicles do not need to be re-filled within a short distance. Even for daily drivers they may only re-fill the gasoline once a week. Activities within that week will be unknown even if they choose to pay by credit card at the gas station. Another reason is that EV can support V2G charging which is not existed in gasoline cars.

We will examine other payment systems and their impact on location privacy in Section 1.2.

Revocation of Location Privacy at the "RIGHT" time. Yet providing unconditional location privacy is not always good. In the case that when a car is stolen, the car owner definitely wants to know the location of his stolen car. Currently some anti-theft or thief-tracing devices can be installed in the car (e.g. GPS with GSM communication device) so that if the car is stolen, the device will send a signal to the car owner telling about the current location of the stolen car. Although these kind of devices can be used to trace any stolen car, the installation and running cost are very high. It is fine for a luxury car as the cost of the anti-theft device compared to the cost of the car itself is just negligible. However, for some lower-end used cars, it is impractical to install such devices where the price is comparable to the value of the used car.

The short driving range is one of the disadvantages of electric vehicles. Yet on the other side, it provides a cheap solution to trace a stolen vehicle. As the vehicle is required to be re-charged very frequently, charging stations can be used to trace any stolen vehicle. If a stolen car is being re-charged at a charging station, the charging station can report to the police or the car owner about the location of this stolen car. It may also refuse to provide charging service to any stolen cars.

1.1 Contributions

In this paper, we enhance the location privacy of electric vehicles at the right time, by proposing a new payment system that provides the following privacy related features:

Two-way Anonymous Payment: It supports anonymous payment *in both directions*. First, the electric vehicle remains anonymous when it re-charges at any charging station. It further supports V2G system. That is, if the car wants to sell back its stored or solar generated electricity to the grid through the charging station, it will receive its credit anonymously. The location privacy of the car is protected *in normal operations*.

Traceability of Stolen Car: If the electric vehicle is stolen, the owner may provide some secret information to charging stations so that next time when the stolen car is being re-charged at any charging station, its location will be revealed.

We argue that our system is practical, as it also provides some additional features that can be favoured by users or supplier:

1. **Prevention of Cheating User**: Different from e-cash which cannot prevent users from cheating or double-spending (it can only *detect* such behaviour), our payment system supports prevention of any cheating behaviour. If any party does not follow the algorithms, the other party can stop providing service immediately. This protects the supplier from being cheated. (The difference between prevention and detection of cheating user will be further explained in Section 1.2.)
2. **Support Judging Authority (JA)**: In case there are some disputes between two parties (maybe due to some physical factors such as sudden breakdown of electricity supply), the affected party may submit all transactions to a Judging Authority. The authority can reveal the identity and investigate the situation.
3. **Low Implementation Cost**: Our system does not require any special security device (e.g. different from ATM). Our security comes from cryptographic algorithms. Our system is also efficient enough to be implemented into mobile device (e.g. smart phone) for the user side.
   Also different from some of the current theft-tracing anti-stolen devices, we do not require any GPS or GSM communication. Thus the cost is much cheaper than those devices.
4. **Lost Protection of Prepaid Credit**: Since our system is account based, we support lost protection. That is, even if the user has lost his mobile device (used for charging), the credit stored in his accout cannot be used by any party. He can regain his credit by providing some authenticated information.

We further analyze our system in security, efficiency and cost to prove that it is practical to be used.

1.2 Existing Payment Systems

There are many different forms of existing payment systems. We examine some of the most practical ones and explain why they are not suitable for electric vehicles.

- **Paper cash**: Different from gas stations, charging stations for electric vehicles are all machine operated. If they allow cash payment, the installation costs will be very large due to high security requirement of cash machine (similar to those for ATM). Note that currently there are many ticketing machine installed in car parks or automatic selling machines (e.g. selling softdrink) which can accept paper cash or coins. However, as the cost for car park or softdrink is far less than charging electric vehicles, the physical security requirement can be much lower. Thus although paper cash can provide anonymity, the high installation and running cost are the main obstacles that are disfavoured by supplier to adopt paper cash as a kind of payment system in the charging station.
- **E-cash**: Alternatively, e-cash is the electronic form of paper cash which also provides anonymity. However, e-cash is mainly used in small amount transaction (e.g. a few dollars) instead of large amount transaction (e.g. a few hundred dollars) due to security and efficiency concerns. In order to support two-way payments, transferrable e-cash is needed and it has been shown complexity of transferrable e-cash grows linearly in the number of transfer supported [15][3]. Apart from that, off-line e-cash cannot provide double-spending *prevention*. It can only *detect* double-spending and *reveal* the identity of the double-spender when the electronic coins are deposited back to the bank. If a cheating user double-spends many times before going bankrupt, the deceived shops cannot get back the money that they deserved to have. Furthermore, different from credit card, e-cash does not provide lost protection. No one will put a few thousand or even a few hundred

[3] A recent approach achieve constant size transferrable e-cash, at the expense that the user storage is linear to the number of his spent coins[20].

dollars in the e-wallet. Thus e-cash is only suitable for small amount transaction. Charging for an electric vehicle definitely does not belong to the small amount transaction category.
- **Prepaid cash card or cash coupon**: Prepaid cash card or cash coupon is another common way of anonymous e-payment. However, similar to e-cash, it does not support lost protection. Executing large amount transaction may bring inconvenience to user: They may neither want to bring many coupons together, nor buy the coupons or topup everyday. In addition, it also does not fully support 2-ways transactions, which is a nessary requirement for the future Vehicle-to-grid system.
- **Paypal**: Paypal is a kind of most commonly used electronic prepaid system. However, it requires a third party (PayPal company). If the authority colludes with the PayPal company (e.g. by telling the PayPal company the exact time and location of a particular transaction), the user can be traced. Thus we regard PayPal providing *partial location privacy* only.
- **Credit card**: Credit card is a widely adopted payment system for large amount transaction instead. It also supports 2-ways transaction. Nevertheless, credit card is not anonymous. Due to the frequent charging requirement for electric vehicles, location privacy will be lost by tracing the credit card payment easily.

Note that none of the existing payment systems can support traceability of stolen cars. We summarize the comparison of our system with some existing payment systems in Table 1.

Table 1. Comparison of existing payment systems

Scheme	Location privacy	Prevention of cheating	Support JA	Low implementation cost	Lost protection	2-ways transactions	Stolen car traceability
Paper cash	✓	✓	×	×	×	✓	×
Prepaid cash card/ Cash coupon	✓	✓	×	✓	×	✓/×	×
Transferrable e-cash	✓	×	✓/× [a]	✓	×	✓	×
Credit card	×	✓	✓	✓	✓	✓	×
PayPal	✓/×	✓	✓	✓	✓	✓	×
Our payment system	✓	✓	✓	✓	✓	✓	✓

[a] Most of the existing e-cash systems do not support judge, though some of them (e.g. [6, 11, 4, 14]) do support judge.

2 System Architecture

2.1 Entities

We consider a system which is composed of the following entities:

1. **User**: A user refers to an electric vehicle or driver, which depends on the mode of operation (will be defined in next Section).
2. **Supplier**: It refers to the power grid company. It supplies (sells) electricity to the cars, and also collects (buy) electricity back from the cars. It is responsible for account opening. Every user needs to obtain an account from it and deposits some money into this account.
3. **Judging Authority (JA)**: It is responsible to investigate into some disputed transactions between user and power company. It has the power to *open* any transaction in case of any dispute. It maybe the government authority or the court.

2.2 Overall Structure

We briefly describe the overall structure of our system. It can be implemented in two different modes:

1. **Portable Mode**: In the portable mode, the account unit for user is per person. That is, if a person has more than one electric vehicles, he can use one single account to manage all cars. The hardware device for the user interface will be a portable device (e.g. smart phone). Note that data connectivity is *not* required.
   When the user is driving his car to the charging station, his portable device will communicate with the charging station. Thus traceability of stolen car cannot be operated under this mode. This mode maybe suitable for those users who want to manage more than one car in a single account; or if the car is driven by different persons everyday (e.g. taxi).
2. **Embedded Mode**: In the embedded mode, the account unit for user is per car. That is, one single car has an unique account. The hardware device for the user interface will be an In-Car-Unit.
   Traceability of stolen car is supported under this mode. This maybe suitable for those users whose car is used by themselves or their family only. Similar to portable mode, data connectivity is not required. However, a USB storage device (e.g. USB thumb drive) is needed in order to support traceability of stolen car.

Our system contains of the following processes regardless of the running mode:

- Registration: The user contacts the supplier for registration and account opening. He needs to pay a deposit for his account so that the balance should have at least D dollars. The supplier returns a token to the user which stores the current value of this account. The user may store this token into his smart phone or In-Car-Unit. (The supplier may develop a new app, or a new physical device for this.) The token is valid for a period of time (e.g. a month). The user needs to update the token before the expiration date.
- Charging: The user presents his token (from his smart phone) and carries an interactive protocol with the charging station, which first checks with the grid management server to confirm the grid capacity is fine. If the price is dynamic (if it is within peak period the price maybe set higher) it further checks with the grid management server for the updated price. Other than that, ***the charging station works as a front-end terminal and the major (cryptographic) computation (e.g. those involving secret key) is done in the supplier's billing server.*** It communicates with the billing server to make sure the token is valid. If it is valid and the balance of the user account is larger than the price of the requested service, the charging station starts to charge the car. The user obtains an updated token with decremented balance and stores it into the his/her portable device (e.g. smartphone).
  The process is described in Figure 1.
- Discharging or Topup: The process is similar to charging. The only difference is that upon completion of the protocol, the user's updated token contains an incremented balance.
- Statement: Every statement period, the user goes to the supplier to topup the balance in the account if it is less than D dollars and update his token.
- Tracing Stolen Car (Embedded mode): If the user's car has been stolen, he needs to retrieve from his backup token and sends the backup token to the supplier. It checks whether this information is correct. If yes, if a vehicle using this token in any charging station, it will report to the user and the police about the location.
- Report of Lost Token (Optional): If the user has backuped every newly generated token, in case he has lost his token (e.g. if his smart phone is stolen), he needs to retrieve from his backup and sends the backup token to the supplier. It checks whether this information is correct. If yes, it will block any party from using his lost token. The process is similar to the report of lost credit card.
- Open (Optional): If the user has some disputes with the supplier, he may reveal his identity together with the corresponding transaction information (e.g. location, time) to the JA. It may also request the supplier to provide related information and investigate this particular transaction.

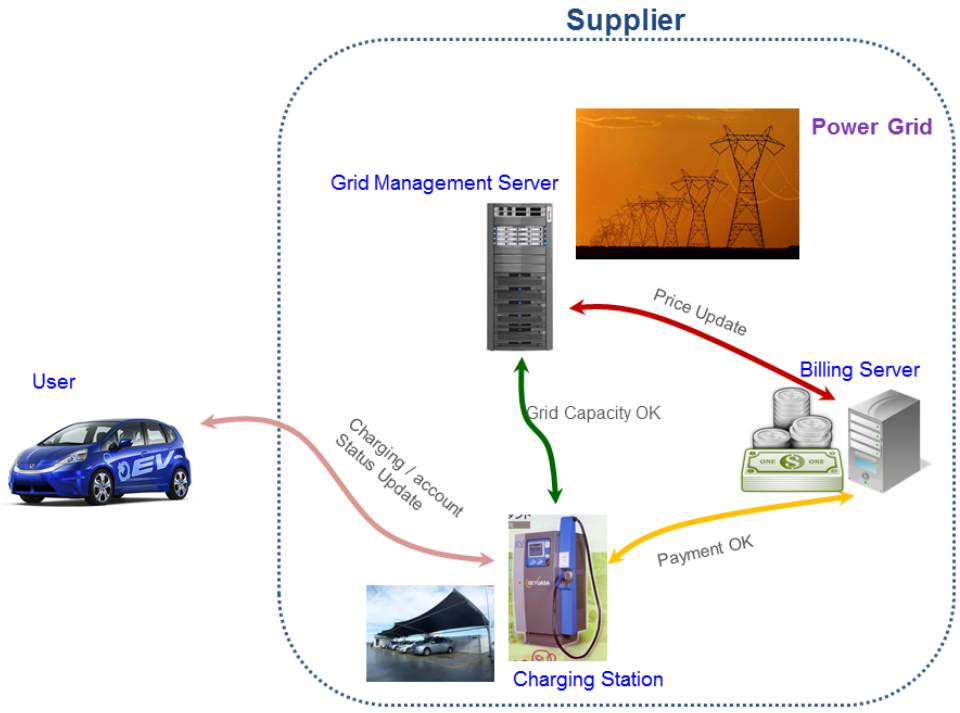

Fig. 1. Charging/Topup Scenario

3 Primitives

In this section we first review some cryptographic primitives that will be used.

Bilinear Pairing. Bilinear pairing (or bilinear map) is a popular building block in public key cryptography. We briefly review its property here. Let $\mathbb{G}, \mathbb{G}_T$ be two cyclic groups of prime order p where p is of λ-bit for some security parameter λ. A function $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is called a bilinear pairing if the following holds:

1. *Bilinearity*: For all $g, h \in \mathbb{G}$, and $a, b \in \mathbb{Z}_p$, $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$.
2. *Non-degeneracy*: There exists $g \in \mathbb{G}$ such that $\hat{e}(g, g)$ has order p in $\mathbb{G}_T$.
3. *Computability*: It is efficient to compute $\hat{e}(g, h)$ for all $g, h \in \mathbb{G}$.

Commitment. Our system uses the well known commitment scheme due to Pedersen [27]. Let $\mathbb{G}$ be a cyclic group of prime order p and g, h be generators of $\mathbb{G}$. On input a value $x \in \mathbb{Z}_p$, the committer randomly chooses $r \in \mathbb{Z}_p$, computes and outputs $C = g^x h^r$ as a commitment of value x. To reveal the value committed in C, the committer outputs (x, r). Everyone can test if $C = g^x h^r$. Sometimes we say r is the opening of C with respect to x. One could extend the commitment scheme to allow committing a tuple of elements $(x_1, \ldots, x_n)$ at the same time by setting $C = g_1^{x_1} \cdots g_n^{x_n} h^r$, where g_i are independent generators of $\mathbb{G}$.

We use $\mathsf{CMT}(x)$ (resp. $\mathsf{CMT}(x_1, \ldots, x_n)$) to denote a Pedersen Commitment of a value x (resp. $(x_1, \ldots, x_n)$). Note that this commitment scheme is homomorphic: $\mathsf{CMT}(a) * \mathsf{CMT}(b)$ gives $\mathsf{CMT}(a+b)$ and the opening of the later is the sum of that of the formers.

BBS+ Signature. We employ the signature scheme proposed by Au et al. [2], which is based on the schemes of Camenisch and Lysyanskaya [10] and of Boneh et al. [5]. Their scheme, called BBS+ signature, is briefly reviewed here. Let $g, g_0, g_1, g_2, g_3 \in \mathbb{G}$ be generators of $\mathbb{G}$. Let $\hat{e}$ be a pairing defined over $(\mathbb{G}, \mathbb{G}_T)$.

The signer's secret is a value $\gamma \in \mathbb{Z}_p$ and the public key is $(w = g^\gamma, g_0, g_1, g_2)$. To create a signature over a tuple of messages (m_1, m_2, m_3), the signer randomly picks $e, y \in_R \mathbb{Z}_p$, computes $A = (gg_0^s g_1^{m_1} g_2^{m_2} g_3^{m_3})^{\frac{1}{\gamma+e}}$. The signer outputs (A, e, y) as the signature on message (m_1, m_2, m_3).

Anyone can verify the signature by testing if the following verification equation holds:

$$\hat{e}(A, wg^e) \stackrel{?}{=} \hat{e}(gg_0^y g_1^{m_1} g_2^{m_2} g_3^{m_3}, g)$$

The BBS+ signature allows the signer to produce signature in a partially blinded way. That is, it allows the signer to sign a tuple of the messages (m_1, m_2, m_3) in a commitment $\mathsf{CMT}(m_1, m_2, m_3)$ without knowing the values.

Zero-knowledge Proof. A zero-knowledge proof [22] is an interactive protocol for one party, the prover, to prove to another party, the verifier, that some statement is true, without revealing anything other than the veracity of the statement. In [21], it has been shown that, assuming the existence of one-way function, one can create a zero-knowledge proof system for the NP-complete graph coloring problem with three colors. Since every problem in NP can be efficiently reduced to this problem, it means that all problems in NP have zero-knowledge proofs. In practice, various efficient construction of zero-knowledge proof for statements regarding relationship about discrete logarithms in cyclic group of known order has been proposed [7]. We follow the notation notation introduced by Camenisch and Stadler [13]. For example, $\mathcal{PK}\{(x) : y = g^x\}$ denotes a zero-knowledge proof that the prover knows an integer x such that the statement $y = g^x$ holds. Symbols appearing on the left of the colon denote values whose knowledge are being proved while symbols appearing on the right, but not the left, of the colon denote public values.

4 Our Proposed System

4.1 Assumptions

As discussed, our system is constructed using cryptographic techniques and hence, it does not depend on any proprietary hardware. Nonetheless, we would like to re-state that security of any cryptographic algorithms depends on the confidentiality of the secret key. Implicitly, when we state some values are to be kept secret, we assume they are stored privately inside the user's device. For example, the secret value stored inside the user's device (e.g. the smartphone) should be kept away from the adversary. This is assumed to be achieved by external means, such as keeping the device to be always in possession or set it to be password-protected.

Our system can only protect location privacy of the payment system and when considering its physical security, it is out of the scope of this paper. For instance, suppose a physical camera is installed in each charging station and it records the physical identifier of the vehicle (e.g. registration plate number), and therefore, it is obvious that location privacy cannot be maintained. This is analogous to the use of physical money. Suppose the cash register records the image of the payer, then it is always possible to link the payment from the user across different locations, and therefore anonymity is no longer preserved.

We further assume that all communication channels are encrypted and authenticated. When considering some attacks such as IP hijacking, distributed denial-of-service attack, man-in-the-middle attack etc., it is out of the scope of this paper.

4.2 High Level Description

Our construction is motivated from the reputation-based blacklistable anonymous authentication system [1]. Authentication in their system results in an increase or decrease in the user reputation,

which is stored at the user side. We adapt their idea and view the reputation as the user's balance. A top-up transaction is an authentication that leads to an increase in reputation. Likewise, a charging transaction is an authentication that leads to a decrease in reputation.

- Registration: User pays a deposit D for registration. The balance B of the user is the value D. Supplier assigns a unique identifier I to the user. User chooses a random number s. User sends $\mathsf{CMT}(I, B, s)$ to the supplier and obtains σ_s which is a BBS+ signature on (I, B, s). Due to the property of the commitment scheme, the value s remains hidden to the supplier. User stores (σ_s, I, B, s) as his/her secret.
- Charging: The user charges his/her vehicle with fee v as follow. User is in possession of (σ_s, I, B, s). He/she first checks if the balance $B > v$. If yes, the user randomly chooses a number s' and sends $\mathsf{CMT}(I)$, $\mathsf{CMT}(B)$, $\mathsf{CMT}(B - v)$, $\mathsf{CMT}(s')$, s to the supplier. The user proves to the supplier, in zero-knowledge, that he/she knows four values (σ_s, I, B, s) such that the following statements are true:
  1. σ_s is a valid signature on (I, B, s)
  2. $\mathsf{CMT}(I)$, $\mathsf{CMT}(B)$, $\mathsf{CMT}(B - v)$, $\mathsf{CMT}(s')$ are formed correctly
  3. $B - v > 0$

  The supplier further checks that s has never been shown by anybody. Note that this check is necessary as it ensures the user cannot use his/her previous balance after making a payment. After that, the supplier creates a new signature $\sigma_{s'}$ on the tuple (I, B', s') where $B' = B - v$ for the user.

  We stress again that in the process, all that the supplier can infer are the commitments, but not the actual values, of I, B, $B - v$, s', and thus the user's identity remains hidden.
- Discharging or Topup: Discharging or topup the balance is similar to the Charging process. User is in possession of (σ_s, I, B, s). Let say the topup amount is v. The user chooses a new random number s' and sends $\mathsf{CMT}(I)$, $\mathsf{CMT}(B)$, $\mathsf{CMT}(s')$, s to the supplier. The user proves to the supplier, in zero-knowledge, that he/she knows four values (σ_s, I, B, s) such that the following statements are true:
  1. σ_s is a valid signature on (I, B, s)
  2. $\mathsf{CMT}(I)$, $\mathsf{CMT}(B)$, $\mathsf{CMT}(s')$ are formed correctly

  The supplier further checks that s has never been shown by anybody. After that, the supplier creates $\mathsf{CMT}(B + v)$ from $\mathsf{CMT}(B)$ and v and issues a new signature $\sigma_{s'}$ on the tuple (I, B', s') where $B' = B + v$ for the user.
- Statement: Every statement period, the user sends (I, B, s) to the supplier, along with a proof that he has a signature σ_s on the tuple (I, B, s). The supplier further checks s has never been shown by anybody. The user pays the amount d such that $D = B + d$. The user then sends $\mathsf{CMT}(s')$ to the supplier and obtains a new signature $\sigma_{s'}$ from the supplier which is a signature on (I, D, s').
- Tracing Stolen Car: (Embedded mode) Assume the user has done a backup for every newly generated token. In case of his car being stolen, the user could report to the supplier immediately and present the value s so that the use of the stolen device could be identified. Any charging station receiving this s in the future will terminate the service and report to the user and the police.
- Report of Lost Token: (Optional) Assume the user has done a backup for every newly generated token. In case of losing the token (e.g. iPod has been lost or stolen), the user could report to the supplier immediately and present the value s so that the use of the stolen device could be identified. Then, a new token containing the same balance could be issued to the user easily. Further details are discussed in Section 5.
- Open: (Optional) Observe that the user secret after each operations contains the same identifier I and that I is either sent in plain (in the statement protocol) or in the commitment $\mathsf{CMT}(I)$ (in all other protocols). Suppose we replace the function $\mathsf{CMT}(I)$ with an encryption of I under the public key of a trusted party called judge, that party would be capable of revealing the identifier of the user in any transactions. Further details are discussed in Section 5.

4.3 Detailed Description

We first describe our system under the portable mode. Later we will show how to modify it into the embedded mode.

Portable Mode:

- System Setup: Let $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map as discussed. In practice, we could use asymmetric pairing (such as type D pairing) for better space efficiency. $\mathbb{G}$ will be chosen so that it is of prime order p where p is of length λ, the security parameter. Let $g, g_0, g_1, g_2, g_3 \in_R \mathbb{G}$. The supplier randomly picks $\gamma \in_R \mathbb{Z}_p$ and computes $w = g^\gamma$. The system parameter is

$$\mathsf{param} = (\mathbb{G}, \mathbb{G}_T, \hat{e}, g, g_0, g_1, g_2, g_3, w)$$

  and the secret key of the supplier is γ.
- Registration: Each user is assigned a unique identity I in the system. In practice, this could be his/her driver license number. Let D be the deposit. The user engages the supplier and enrolls into the system as follow.
  1. The user randomly picks $y', s \in_R \mathbb{Z}_p$, computes and sends $C = g_0^{y'} g_3^s$ to the supplier, along with the following proof:
$$\mathcal{PK}_1\{(y', s) : C = g_0^{y'} g_3^s\}.$$
     $\mathcal{PK}_1$ assures the supplier that the value C is computed correctly. Precise description of the proof (and subsequent proofs) will be given in Appendix A.
  2. The supplier randomly picks $y'', e \in_R \mathbb{Z}_p$, computes $A = (Cgg_0^{y''} g_1^I g_2^D)^{\frac{1}{e+\gamma}}$ and returns (A, y'', e) to the user.
  3. The user computes $y = y' + y''$ and checks if
$$\hat{e}(A, wg^e) \stackrel{?}{=} \hat{e}(gg_0^y g_1^I g_2^D g_3^s, g).$$
     User parses $\sigma_s = (A, e, y)$ and stores a four tuple (σ_s, I, D, s). Note that σ_s is a BBS+ signature on the tuple (I, D, s).

  The registration protocol is shown in figure 2.
- Charging: Let v be the value of the transaction. The user parses his/her storage as $\left(\widetilde{\sigma_s} := (\tilde{A}, \tilde{e}, \tilde{y}), I, \tilde{B}, \tilde{s}\right)$ and checks if $\tilde{B} - v \geq 0$. Next, they engages in the following protocol.
  1. The user randomly picks $y', s \in_R \mathbb{Z}_p$, computes and sends $C = g_0^{y'} g_1^I g_2^{\tilde{B}} g_3^s$ as well as $\tilde{s}$ to the supplier, along with the following proof:
$$\mathcal{PK}_2\{(\tilde{A}, \tilde{e}, \tilde{y}, I, \tilde{B}, y', s) : C = g_0^{y'} g_1^I g_2^{\tilde{B}} g_3^s \wedge \hat{e}(\tilde{A}, wg^{\tilde{e}}) = \hat{e}(gg_0^{\tilde{y}} g_1^I g_2^{\tilde{B}} g_3^{\tilde{s}}, g) \wedge D \geq \tilde{B} - v \geq 0\}.$$
  2. The supplier checks that $\tilde{s}$ has never been used[4] and randomly picks $y'', e \in_R \mathbb{Z}_p$, computes $A = (Cgg_0^{y''} g_2^{-v})^{\frac{1}{e+\gamma}}$ and returns (A, y'', e) to the user.
  3. The user computes $y = y' + y''$, $B = \tilde{B} - v$ and checks if
$$\hat{e}(A, wg^e) \stackrel{?}{=} \hat{e}(gg_0^y g_1^I g_2^B g_3^s, g).$$
     User parses $\sigma_s = (A, e, y)$ and stores a four tuple (σ_s, I, B, s). Note that σ_s is a BBS+ signature on the tuple (I, B, s).

  The charging protocol is shown in figure 3.
- Topup: Let v be the topup value. The user parses his/her storage as $\left(\widetilde{\sigma_s} := (\tilde{A}, \tilde{e}, \tilde{y}), I, \tilde{B}, \tilde{s}\right)$ and checks if $\tilde{B} + v \leq D$. We assume D is the maximum account balance. Next, they engages in the following protocol.

[4] The practical issue of the checking process will be described in Section 6.1.

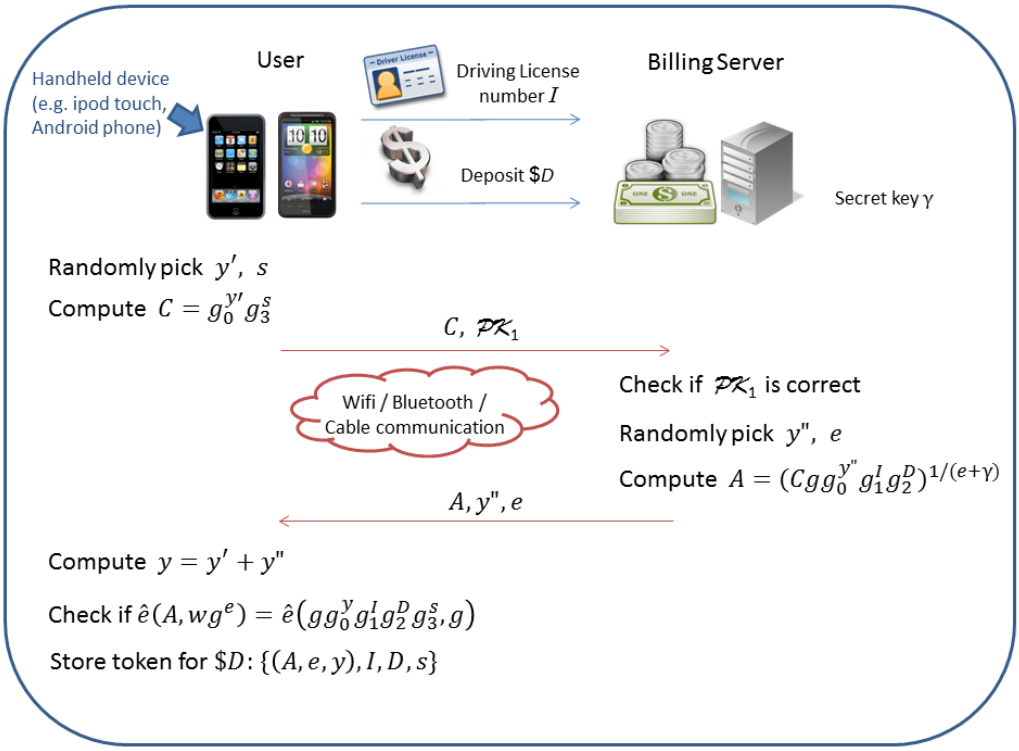

Fig. 2. Registration

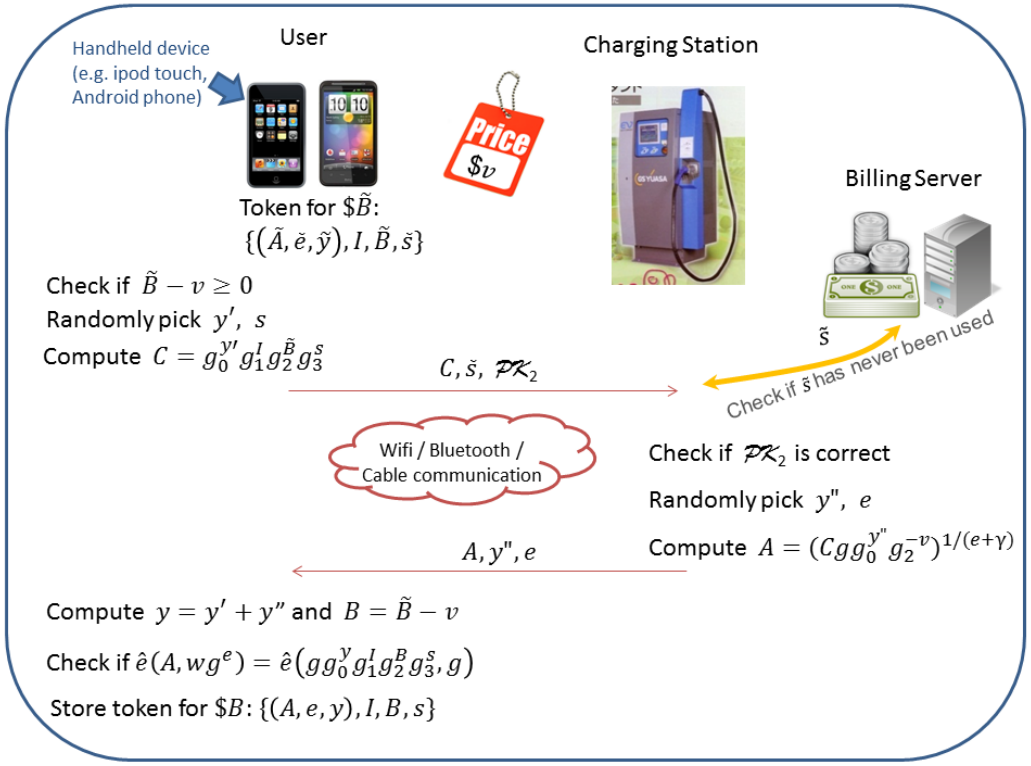

Fig. 3. Charging

1. The user randomly picks $y', s \in_R \mathbb{Z}_p$, computes and sends $C = g_0^{y'} g_1^I g_2^{\tilde{B}} g_3^s$ as well as $\tilde{s}$ to the supplier, along with the following proof:

$$\mathcal{PK}_3\{(\tilde{A}, \tilde{e}, \tilde{y}, I, \tilde{B}, y', s) : C = g_0^{y'} g_1^I g_2^{\tilde{B}} g_3^s \wedge \hat{e}(\tilde{A}, wg^{\tilde{e}}) = \hat{e}(gg_0^{\tilde{y}} g_1^I g_2^{\tilde{B}} g_3^{\tilde{s}}, g) \wedge D \geq \tilde{B} + v \geq 0\}.$$

2. The supplier checks that $\tilde{s}$ has never been used and randomly picks $y'', e \in_R \mathbb{Z}_p$, computes $A = (Cgg_0^{y''} g_2^v)^{\frac{1}{e+\gamma}}$ and returns (A, y'', e) to the user.
3. The user computes $y = y' + y''$, $B = \tilde{B} + v$ and checks if

$$\hat{e}(A, wg^e) \stackrel{?}{=} \hat{e}(gg_0^y g_1^I g_2^B g_3^s, g).$$

User parses $\sigma_s = (A, e, y)$ and stores a four tuple (σ_s, I, B, s). Note that σ_s is a BBS+ signature on the tuple (I, B, s).

– Statement: The user parses his/her storage as $\left(\widetilde{\sigma_s} := (\tilde{A}, \tilde{e}, \tilde{y}), I, \tilde{B}, \tilde{s}\right)$ and pays $v = D - \tilde{B}$ to settle his account. Next, they engages in the following protocol.

1. The user randomly picks $y', s \in_R \mathbb{Z}_p$, computes and sends $C = g_0^{y'} g_3^s$ as well as $\tilde{s}$, I, $\tilde{B}$ to the supplier, along with the following proof:

$$\mathcal{PK}_4\{(\tilde{A}, \tilde{e}, \tilde{y}, y', s) : C = g_0^{y'} g_3^s \wedge \hat{e}(\tilde{A}, wg^{\tilde{e}}) = \hat{e}(gg_0^{\tilde{y}} g_1^I g_2^{\tilde{B}} g_3^{\tilde{s}}, g)\}.$$

2. The supplier checks that $\tilde{s}$ has never been used and randomly picks $y'', e \in_R \mathbb{Z}_p$, computes $A = (Cgg_0^{y''} g_1^I g_2^D)^{\frac{1}{e+\gamma}}$ and returns (A, y'', e) to the user.
3. The user computes $y = y' + y''$ and checks if

$$\hat{e}(A, wg^e) \stackrel{?}{=} \hat{e}(gg_0^y g_1^I g_2^D g_3^s, g).$$

User parses $\sigma_s = (A, e, y)$ and stores a four tuple (σ_s, I, D, s). Note that σ_s is a BBS+ signature on the tuple (I, D, s).

The statement protocol is shown in figure 4.

Embedded Mode: In embedded mode, all operations are the same, except that on the user side operations are run in the In-Car-Unit instead of the portable device. We assume this In-Car-Unit is tamper resistance in order to support tracing of stolen car.

In order to support traceability of stolen car, the user needs to backup the newly generated token after each operation (including Registration, Charging, Topup, Statement) into his backup device (e.g. USB thumb drive).

Tracing Stolen Car: In case the car is stolen and the user wants to trace his stolen car, he can reveal his token $\{(A, e, y), I, D, s\}$ to the supplier (and all charging stations). Any charging station receiving the token containing s in the future will refuse to provide service and report to the user and the police immediately since that means the stolen car is at that charging station requesting a service.

Note that this requires the user to report the lost before the thief makes a recharge. In this aspect it is similar to credit card. In our extension described in Section 5.3, we discuss how the thief can be traced even if he/she makes a recharge before the user's report.

5 Extensions

We discuss some useful extensions for our system.

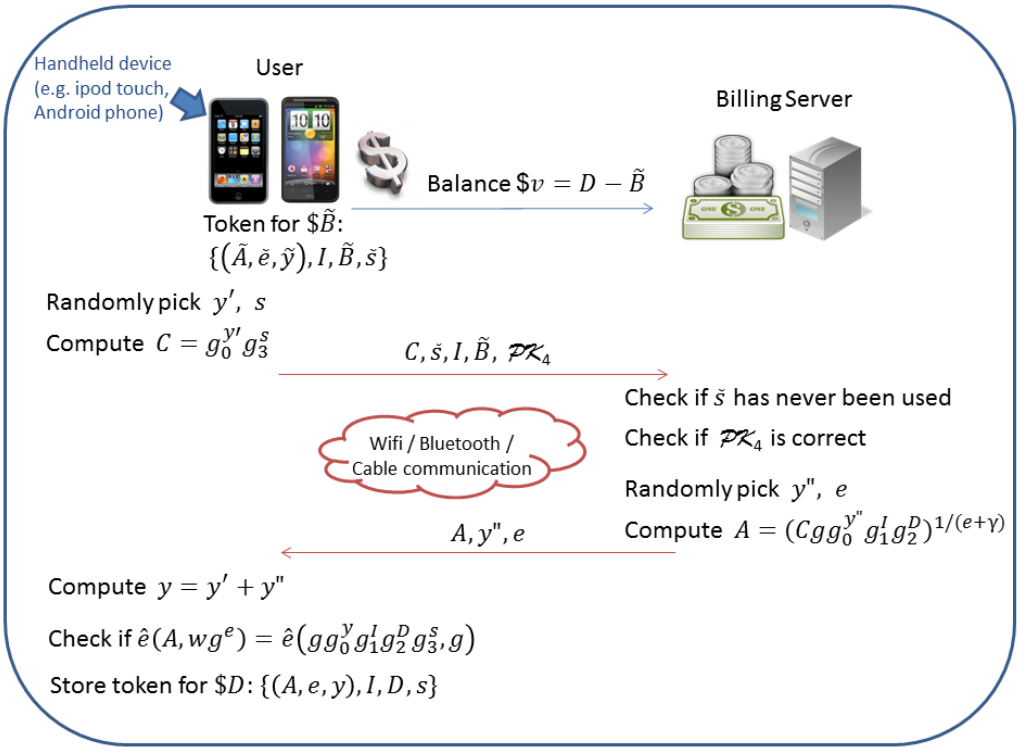

Fig. 4. Statement

5.1 Incorporating Token Expiry

In our basic construction, token never expires and the supplier needs to store all the s forever. An expiration mechanism can be incorporated easily. Let $H : \{0,1\}^* \to \mathbb{G}$ be a collision-resistant hash function. Let $T \in \{0,1\}^*$ be the identifier of the current time period. In practice, T could be the bit string `Jan2012`, `Feb2012`, etc. The public parameter g_1, g_2, g_3 in param is replaced with the hash function H.

Let T_j be the current period and T_{j+1} be the next period. For example, T_j = `Jan2012` and T_{j+1} = `Feb2012`. In the protocols, the value g_i will be replaced with $H(T, i)$ for $i = 1$ to 3. At the end of period T_j, all users will contact the supplier in the statement protocol. During the execution of the protocol, $g_i = H(T_{j+1}, i)$ will be used in the computation of the value A. Thus, in period T_{j+1}, the user will be using $g_i = H(T_{j+1}, i)$ for charging and topup and the previous token will not be usable.

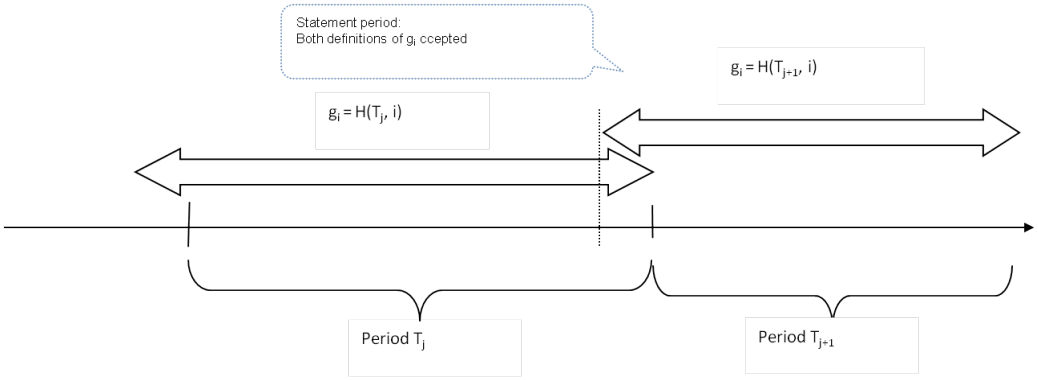

Fig. 5. Timeline demonstrating the expiration mechanism.

Of course, to accommodate the user who executes the statement protocol before the end of T_j, both $g_i = H(T_j, i)$ and $g_i = H(T_{j+1}, i)$ will be accepted at the end of period T_j. Fig.5 illustrates our idea. This extension does not alter the efficiency of our system.

5.2 Incorporating Judge (Open Operation)

Sometimes giving user too much privacy is not preferable. Thus, it is natural to introduce an external entity to the system, called judge, which is capable of identifying the user in all transactions. The judge would be trusted to exhibit its power in appropriate situation only, for example, under the court order. To introduce this additional feature, we review another cryptographic tool called verifiable encryption.

Verifiable Encryption. A verifiable encryption scheme is a public key encryption scheme with an additional feature. In its basic form, it allows a prover to prove to a verifier that the plaintext PT encrypted in a known ciphertext CT under the public key of a third party PKE satisfies some binary relation R. The concept of verifiable encryption was introduce in [28]. In [8], it has been shown that any public key encryption scheme can be turned into a verifiable encryption scheme for all relation having a 3-move proof-of-knowledge protocol (as known as honest verifier zero-knowledge protocol). An efficient construction of such primitive has been proposed in [12].

Introducing Judge in our system. Let PKE be the public key of the judge. Recall that in all our protocols, user is required to send $C = g_0^{y'} g_1^I g_2^{\tilde{B}} g_3^s$ (or $C = g_0^{y'} g_3^s$ in case $\tilde{B}$ and I are sent in plain in registration and statement). In our extension, a user is required to produce a ciphertext CT which

is the encryption of $(y', I, \tilde{B}, s)$ under the public key PKE, and produces a proof that CT is the encryption of the correct values with respect to C. In this case, the judge can always decrypt CT and obtains the values $(y', I, \tilde{B}, s)$ and traces the action of the user.

The most efficient verifiable encryption due to [12] has a message space of $\mathbb{Z}_n$, where n is the product of two primes. One subtlety arises since the values $(y', I, \tilde{B}, s)$ in our basic construction are treated as elements of $\mathbb{Z}_p$. Direct combination of the two would not be secure since a cheating user can encrypt his identity as $I + kp$ for some integer k and produce a proof that he has encrypted his identity. The decrypted value from the Judge would be $I + kp \bmod n$, which may not be $I \bmod p$. To make it compatible with our system, we can change our groups of prime order p to groups of composite number n. This change, however, would make the pairing operation rather inefficient. The reason is that $|p| = 170$ would offer a similar security compared with $|n| = 1024$.

A more effective alternative is to employ the signature scheme [9] instead of BBS+ if judge is introduced in our system[5], which works in a cyclic group of unknown order. In that case, $(y', I, \tilde{B}, s)$ are all treated as integers within a specific range and the above attack is not possible.

5.3 Report of Lost Token/ Tracing Stolen Car

The user token is completely software-based. The user should backup his secret (σ_s, I, B, s) in another USB storage device after each recharge or topup. He sent the token to the supplier to report his lost. The supplier checks whether it is a valid one. If yes, it extracts the value s and blocks any future transaction involving s. The supplier also issues a new token to the user using Statement operation associated with his remaining balance. This process is similar to the traceability of stolen car described in the Section 4.3.

In the case of the lost token (or the lost car in the embedded mode) that has been used by the thief already for recharge, it still could be located. In this situation, the judge will open all transactions within the range of the electric vehicle and look for the identity of the lost token/vehicle. Hence, the lost token can still be traced.

6 Practicality Analysis

In this section, we show that our scheme is practical by giving analysis data in three aspects: efficiency, cost and security.

6.1 Efficiency Analysis

We analyze the efficiency of our scheme using the simulation result from jPBC [26] for the following devices:

- User: We use a smart phone HTC Desire HD as the simulation device (portable mode). It is running Android 2.2, equiped with Qualcomm Snapdragon QSD8255 1GHz as the CPU and with 1.5GB ROM.
- Supplier: We use a desktop equiped with Intel(R) Core(TM)2 Quad CPU Q6600 2.40GHz, 3 GB RAM, Ubuntu 10.04 as the simulation device.

We only count the time required for exponentiation and pairing. Other operations such as hashing, group addition, integer addition/multiplication etc. are insignificant compared with exponentiation and pairing.

For exponentiation, we further optimize for those bases which are constant. It allows the use of some pre-processed data for faster computation. For pairing, we also optimize for those such that one of the pairing elements is a constant. We put our analyzed result in table 2:

Table 2. Operations required for user and supplier

	Registration		Charging/Topup		Statement	
	User	Supplier	User	Supplier	User	Supplier
Group $\mathbb{G}$ exponentiation (pre-processed)	8	5	27	16	17	11
Group $\mathbb{G}$ exponentiation (no pre-processed)	0	2	2	7	1	4
Group $\mathbb{G}_T$ exponentiation (pre-processed)	0	0	7	7	3	6
Group $\mathbb{G}_T$ exponentiation (no pre-processed)	0	0	0	4	0	2
Pairing (1 element is a constant)	1	0	3	3	2	2
Pairing (both elements are not constant)	1	0	1	1	1	0

Table 3. Running time and communication overhead

	Registration		Charging/Topup		Statement	
	User	Supplier	User	Supplier	User	Supplier
Overall Running Time	1.02 s	0.048 s	2.74 s	0.236 s	1.84 s	0.133 s
Communication Overhead	256 bytes	256 bytes	1792 bytes	256 bytes	1152 bytes	256 bytes

We also include the communication overhead. That is, the data sent from the user to the supplier and from the supplier to the user, respectively. The overall result is summarized in table 3.

For storage, the supplier may need to store all transactions for the charging and top processes within a time period. For each process, it may take about 2k bytes. Assume the user charges or topup twice a day. Let each period last for a month. So the supplier may need to store about 120k bytes for every user in each period. Assume there are 1 million users in the system. There will be about 120G bytes transaction data. These stored data are mainly for the judge operation. When the next period comes, the data from the previous period can be deleted. 120G data should be easily stored within a normal harddisk.

During the normal charging/topup operation, the supplier (at the backend) needs to search for a number (selected by the user) in the database to check whether it has already been used within a time period. Using the above assumption (1 month as a time period, there are 1 million users in the system and each user charges/topups twice per day), there will be at most 60 million entries in the database. In practice, this operation will be delegated to the database server (e.g. MySQL) and it is equivalent to a single database query. Modern database server supports 100 Transaction Per Second (TPS) and thus it is reasonable to assume this checking requires less than 0.1 second.

The charging station and the backend server can be connected through internet connection. As the data transmitted per transaction is just a few thousand bytes (refer to table 3), the cost is very low and the transmisson time should be very fast.

After adding up all operations, the overall running time can be approximated as:

- Registration: around 1 second.
- Charging/Topup: around 3 seconds.
- Statement: around 2 seconds.

6.2 Cost Analysis

We analyze the cost required for our system. We divide the analysis into user and supplier.

[5] In other words, BBS+ is more efficient in systems without the need of a Judge. With judge, [9] will be more efficient.

User: For portable mode, each user is only required to have a smart phone or mobile device running Android or iOS. If we use HTC Desire HD (the one mentioned in the above simulation, running Android), it costs about US$400. If we use an Apple iPod Touch (running iOS), it costs just US$200.

For embedded mode, each user is required to install an In-Car-Unit inside the car. It costs about US$400.

Supplier: For each charging station, a normal desktop computer should be enough for carring out all required operations in acceptable time (e.g. as described in table 3). Cost should be around US$500. Including other hardware installing cost, a charging station (payment part) should cost no more than US$800. This is comparable to a credit card terminal which costs around US$500 - US$800 [18].

For the backend server (payment server), as it requires to store all transaction records and perform a real time search operation, a more powerful computer is recommended. A server costs about US$2000 should be more than enough to perform all required operations in an acceptable time as mentioned above. A database server such as MySQL can be freely used.

6.3 Security Analysis

We analyze the security of our scheme using a game-based approach. Each security requirement is modeled as a game played between a probabilistic polynomial time adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The game are defined so that it captures the capabilities and behavior of an adversary. The adversary winning the game would imply it is possible to break a security requirement. Using reduction argument, we would then show any adversary winning the game could be used to break some hardness assumptions.

The details of the analysis are presented in Appendix B.

7 Conclusion

In this paper, we presented a mechanism to enhance location privacy for electric vehicles. Our proposed solution provides an anonymous payment system with privacy protection support. In the case where traceability is required, such as when the electric vehicle is stolen, this feature can also be provided. Hence, our solution provides location privacy enhancement at the right time, which will make the adoption of electric vehicles practical.

We note that the scheme described in this paper is specifically designed for electric vehicles. However, we do not eliminate the possibility to apply our scheme (or modified version) in other environments if they find it suitable.

References

1. M. H. Au, A. Kapadia, and W. Susilo. BLACR: TTP-Free Blacklistable Anonymous Credentials with Reputation. In *NDSS*. The Internet Society, 2012.
2. M. H. Au, W. Susilo, and Y. Mu. Constant-Size Dynamic k-TAA. In R. D. Prisco and M. Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2006.
3. I. Bilogrevic, M. Jadliwala, K. Kalkan, J.-P. Hubaux, and I. Aad. Privacy in mobile computing for location-sharing-based services. In *PETS*, volume 6794 of *Lecture Notes in Computer Science*, pages 77–96. Springer, 2011.
4. O. Blazy, S. Canard, G. Fuchsbauer, A. Gouget, H. Sibert, and J. Traoré. Achieving optimal anonymity in transferable e-cash with a judge. In *AFRICACRYPT*, volume 6737 of *Lecture Notes in Computer Science*, pages 206–223. Springer, 2011.
5. D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.
6. E. F. Brickell, P. Gemmell, and D. W. Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In *SODA*, pages 457–466. ACM/SIAM, 1995.

7. J. Camenisch. Group signature schemes and payment systems based on the discrete logarithm problem. *PhD thesis*, 1998.
8. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In T. Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345. Springer, 2000.
9. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In S. Cimato, C. Galdi, and G. Persiano, editors, *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer, 2002.
10. J. Camenisch and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004.
11. J. Camenisch, J.-M. Piveteau, and M. Stadler. An efficient fair payment system. In *ACM Conference on Computer and Communications Security*, pages 88–94. ACM, 1996.
12. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003.
13. J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In *CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997.
14. B. Carbunar, W. Shi, and R. Sion. Conditional e-payments with transferability. *J. Parallel Distrib. Comput.*, 71(1):16–26, 2011.
15. D. Chaum and T. P. Pedersen. Transferred cash grows in size. In *EUROCRYPT*, pages 390–407, 1992.
16. M. Chia, S. Krishnan, and J. Zhou. Challenges and Opportunities in Infrastructure Support for Electric Vehicles and Smart Grid in a Dense Urban Environment. To appear in IEEE International Electric Vehicle Conference 2012, 2012.
17. M. Duckham. Moving forward: location privacy and location awareness. In *SPRINGL*, pages 1–3. ACM, 2010.
18. M. Express. Credit Card Terminal - Credit Card Machines - Merchant Express. http://www.merchantexpress.com/credit-card-terminals.htm.
19. J. Freudiger, R. Shokri, and J.-P. Hubaux. Evaluating the privacy risk of location-based services. In *Financial Cryptography*, volume 7035 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2011.
20. G. Fuchsbauer, D. Pointcheval, and D. Vergnaud. Transferable constant-size fair e-cash. In *CANS*, pages 226–247, 2009.
21. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38:690–728, July 1991.
22. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
23. P. Golle and K. Partridge. On the anonymity of home/work location pairs. In *Pervasive*, volume 5538 of *Lecture Notes in Computer Science*, pages 390–397. Springer, 2009.
24. B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Enhancing security and privacy in traffic-monitoring systems. *IEEE Pervasive Computing*, 5(4):38–46, 2006.
25. L. Liao, D. J. Patterson, D. Fox, and H. A. Kautz. Learning and inferring transportation routines. *Artif. Intell.*, 171(5-6):311–331, 2007.
26. B. Lynn. The Java Pairing Based Cryptography Library (jPBC), 2010. http://http://libeccio.dia.unisa.it/projects/jpbc/.
27. T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In J. Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.
28. M. Stadler. Publicly verifiable secret sharing. In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199. Springer, 1996.

A Details of $\mathcal{PK}_1$ - $\mathcal{PK}_4$

To efficiently instantiate the zero-knowledge proof $\mathcal{PK}_1$ to $\mathcal{PK}_4$, the supplier adds the following auxiliary parameters: $h, h_1, h_2 \in_R \mathbb{G}$, $t = h^{\delta}$ for some randomly generated $\delta \in_R \mathbb{Z}_p$. For $i = 0$ to D, $\varsigma_i = h^{\frac{1}{\delta+i}}$. For efficiency considerations, set $\hat{E} = \hat{e}(g,g)$, $\hat{E}_i = \hat{e}(g_i, g)$ for $i = 0$ to 3, $\hat{H} = \hat{e}(h,h)$, $\hat{H}_0 = \hat{e}(h_1, w)$, $\hat{H}_1 = \hat{e}(h_1, g)$, $\hat{H}_2 = \hat{e}(h_1, h)$ and $\hat{H}_3 = \hat{e}(h_1, t)$. They will be included in the public parameter to speed up the protocol. Set $\mathsf{param} := \mathsf{param} \cup \{h, h_1, h_2, t, \varsigma_0, \ldots, \varsigma_D, \hat{E}, \hat{E}_0, \hat{E}_1, \hat{E}_2, \hat{E}_3, \hat{H}, \hat{H}_0, \hat{H}_1, \hat{H}_2, \hat{H}_3\}$ and

the value of δ should be be deleted or kept secret. To reduce the number of rounds and for better space-efficiency, we use the well-known Fiat-Shamir transformation where the function H is modeled as a random oracle.

A.1 $\mathcal{PK}_1\{(y', s) : C = g_0^{y'} g_3^{s}\}$

1. The supplier sends a random challenge R.
2. The user randomly chooses $\rho_{y'}, \rho_s \in_R \mathbb{Z}_p$, computes $T = g_0^{\rho_{y'}} g_3^{\rho_s}$.
3. The user computes $c = H(T, R) \in_R \mathbb{Z}_p$.
4. The user computes $z_{y'} = \rho_{y'} - cy'$, $z_s = \rho_s - cs$ and sends $c, z_{y'}, z_s$ to the supplier.
5. The supplier accepts the proof if and only if

$$c \stackrel{?}{=} H(C^c g_0^{z_{y'}} g_3^{z_s}, R).$$

A.2 $\mathcal{PK}_2\{(\tilde{A}, \tilde{e}, \tilde{y}, I, \tilde{B}, y', s) : C = g_0^{y'} g_1^{I} g_2^{\tilde{B}} g_3^{s} \wedge \hat{e}(\tilde{A}, wg^{\tilde{e}}) = \hat{e}(gg_0^{\tilde{y}} g_1^{I} g_2^{\tilde{B}} g_3^{\tilde{s}}, g) \wedge D \geq \tilde{B} - v \geq 0\}$

1. The supplier sends a random challenge R.
2. The user randomly chooses $k_1, k_2, k_3, k_4 \in_R \mathbb{Z}_p$, computes $F_1 = h_1^{k_1} h_2^{k_2}$, $F_2 = \tilde{A} h_1^{k_2}$, $F_3 = h_1^{k_3} h_2^{k_4}$, $F_4 = \varsigma_{\tilde{B}-v} h_1^{k_4}$. Next, the user randomly chooses ρ_{k_1}, ρ_{k_2}, ρ_{k_3}, ρ_{k_4}, $\rho_{y'}$, ρ_I, $\rho_{\tilde{B}}$, ρ_s, $\rho_{\tilde{e}}$, ρ_{β_1}, ρ_{β_2}, ρ_{β_3}, $\rho_{\beta_4} \in_R \mathbb{Z}_p$ and computes the following:

$$\begin{aligned}
T_1 &= g_0^{\rho_{y'}} g_1^{\rho_I} g_2^{\rho_{\tilde{B}}} g_3^{\rho_s} \\
T_2 &= h_1^{\rho_{k_1}} h_2^{\rho_{k_2}} \\
T_3 &= F_1^{-\rho_{\tilde{e}}} h_1^{\rho_{\beta_1}} h_2^{\rho_{\beta_2}} \\
T_4 &= \hat{H}_0^{\rho_{k_2}} \hat{H}_1^{\rho_{\beta_2}} \hat{E}_0^{\rho_{\tilde{y}}} \hat{E}_1^{\rho_I} \hat{E}_2^{\rho_{\tilde{B}}} \hat{e}(F_2, g)^{-\rho_{\tilde{e}}} \\
T_5 &= h_1^{\rho_{k_3}} h_2^{\rho_{k_4}} \\
T_6 &= F_3^{-\rho_{\tilde{B}}} h_1^{\rho_{\beta_3}} h_2^{\rho_{\beta_4}} \\
T_7 &= \hat{H}_2^{\rho_{\beta_4}} \hat{H}_3^{\rho_{k_4}} \hat{e}(F_4, h)^{-\rho_\beta}
\end{aligned}$$

3. The user computes $c = H(\{F_i\}_{i=1}^4, \{T_i\}_{i=1}^7, R) \in_R \mathbb{Z}_p$.
4. The user computes and sends c, F_1, F_2, F_3, F_4, $z_{k_1} = \rho_{k_1} - ck_1$, $z_{k_2} = \rho_{k_2} - ck_2$, $z_{k_3} = \rho_{k_3} - ck_3$, $z_{k_4} = \rho_{k_4} - ck_4$, $z_{y'} = \rho_{y'} - cy'$, $z_I = \rho_I - cI$, $z_{\tilde{B}} = \rho_{\tilde{B}} - c\tilde{B}$, $z_s = \rho_s - cs$, $z_{\tilde{e}} = \rho_{\tilde{e}} - c\tilde{e}$, $z_{\beta_1} = \rho_{\beta_1} - ck_1\tilde{e}$, $z_{\beta_2} = \rho_{\beta_2} - ck_2\tilde{e}$, $z_{\beta_3} = \rho_{\beta_3} - c(\tilde{B} - v)k_3$, $z_{\beta_4} = \rho_{\beta_4} - c(\tilde{B} - v)k_4$ to the supplier.
5. The supplier computes T_1 to T_7 as follows:

$$\begin{aligned}
T_1 &= C^c g_0^{z_{y'}} g_1^{z_I} g_2^{z_{\tilde{B}}} g_3^{z_s} \\
T_2 &= F_1^c h_1^{z_{k_1}} h_2^{z_{k_2}} \\
T_3 &= F_1^{-z_{\tilde{e}}} h_1^{z_{\beta_1}} h_2^{z_{\beta_2}} \\
T_4 &= \left(\hat{e}(F_2, w)\hat{E}^{-1}\hat{E}_3^{-\tilde{s}}\right)^c \hat{H}_0^{z_{k_2}} \hat{H}_1^{z_{\beta_2}} \hat{E}_0^{z_{\tilde{y}}} \hat{E}_1^{z_I} \hat{E}_2^{z_{\tilde{B}}} \hat{e}(F_2, g)^{-z_{\tilde{e}}} \\
T_5 &= F_3^c h_1^{z_{k_3}} h_2^{z_{k_4}} \\
T_6 &= F_3^{-vc} F_3^{-z_{\tilde{B}}} h_1^{z_{\beta_3}} h_2^{z_{\beta_4}} \\
T_7 &= \left(\hat{e}(F_4, th^{-v})\hat{H}^{-1}\right)^c \hat{H}_2^{z_{\beta_4}} \hat{H}_3^{z_{k_4}} \hat{e}(F_4, h)^{-z_\beta}
\end{aligned}$$

   and accepts the proof if and only if

$$c \stackrel{?}{=} H(\{F_i\}_{i=1}^4, \{T_i\}_{i=1}^7, R)$$

A.3 $\mathcal{PK}_3\{(\tilde{A}, \tilde{e}, \tilde{y}, I, \tilde{B}, y', s) : C = g_0^{y'} g_1^{I} g_2^{\tilde{B}} g_3^{s} \wedge \hat{e}(\tilde{A}, wg^{\tilde{e}}) = \hat{e}(gg_0^{\tilde{y}} g_1^{I} g_2^{\tilde{B}} g_3^{\tilde{s}}, g) \wedge D \geq \tilde{B} + v \geq 0\}$

Same as $\mathcal{PK}_2$ except the value of $-v$ is replaced with $+v$.

A.4 $\mathcal{PK}_4\{(\tilde{A}, \tilde{e}, \tilde{y}, y', s) : C = g_0^{y'} g_3^s \wedge \hat{e}(\tilde{A}, wg^{\tilde{e}}) = \hat{e}(gg_0^{\tilde{y}} g_1^I g_2^{\tilde{B}} g_3^{\tilde{s}}, g)\}$

1. The supplier sends a random challenge R.
2. The user randomly chooses $k_1, k_2 \in_R \mathbb{Z}_p$, computes $F_1 = h_1^{k_1} h_2^{k_2}, F_2 = \tilde{A} h_1^{k_2}$. Next, the user randomly chooses ρ_{k_1}, ρ_{k_2}, $\rho_{y'}$, ρ_s, $\rho_{\tilde{e}}$, ρ_{β_1}, $\rho_{\beta_2} \in_R \mathbb{Z}_p$ and computes the following:

$$\begin{aligned} T_1 &= g_0^{\rho_{y'}} g_3^{\rho_s} \\ T_2 &= h_1^{\rho_{k_1}} h_2^{\rho_{k_2}} \\ T_3 &= F_1^{-\rho_{\tilde{e}}} h_1^{\rho_{\beta_1}} h_2^{\rho_{\beta_2}} \\ T_4 &= \hat{H}_0^{\rho_{k_2}} \hat{H}_1^{\rho_{\beta_2}} \hat{E}_0^{\rho_{\tilde{y}}} \hat{e}(F_2, g)^{-\rho_{\tilde{e}}} \end{aligned}$$

3. The user computes $c = H(\{F_i\}_{i=1}^2, \{T_i\}_{i=1}^4, R) \in_R \mathbb{Z}_p$.
4. The user computes and sends c, F_1, F_2, $z_{k_1} = \rho_{k_1} - ck_1$, $z_{k_2} = \rho_{k_2} - ck_2$, $z_{y'} = \rho_{y'} - cy'$, $z_s = \rho_s - cs$, $z_{\tilde{e}} = \rho_{\tilde{e}} - c\tilde{e}$, $z_{\beta_1} = \rho_{\beta_1} - ck_1\tilde{e}$, $z_{\beta_2} = \rho_{\beta_2} - ck_2\tilde{e}$ to the supplier.
5. The supplier computes T_1 to T_4 as follows:

$$\begin{aligned} T_1 &= C^c g_0^{z_{y'}} g_1^{z_I} g_2^{z_{\tilde{B}}} g_3^{z_s} \\ T_2 &= F_1^c h_1^{z_{k_1}} h_2^{z_{k_2}} \\ T_3 &= F_1^{-z_{\tilde{e}}} h_1^{z_{\beta_1}} h_2^{z_{\beta_2}} \\ T_4 &= \left(\hat{e}(F_2, w) \hat{E}^{-1} \hat{E}_1^{-I} \hat{E}_2^{-\tilde{B}} \hat{E}_3^{-\tilde{s}}\right)^c \hat{H}_0^{z_{k_2}} \hat{H}_1^{z_{\beta_2}} \hat{E}_0^{z_{\tilde{y}}} \hat{e}(F_2, g)^{-z_{\tilde{e}}} \end{aligned}$$

   and accepts the proof if and only if

$$c \stackrel{?}{=} H(\{F_i\}_{i=1}^2, \{T_i\}_{i=1}^4, R)$$

B Security Analysis

Prevention of Cheating User: The following game models the interaction between a cheating user $\mathcal{A}$ and an honest supplier $\mathcal{C}$. $\mathcal{C}$ keeps a running balance W possesses by $\mathcal{A}$. $\mathcal{A}$ wins the game if it can make W to be negative. Note that in this game we allow $\mathcal{A}$ to register multiple times. This models the situation when several users collude together.

System Parameter. $\mathcal{C}$ creates and publishes the system parameter param and keeps the secret key γ private. $\mathcal{C}$ initializes a counter W which is 0.

Interactions. $\mathcal{A}$ can make the following four types of interaction freely with $\mathcal{C}$.

1. *Registration.* $\mathcal{A}$ interacts with $\mathcal{C}$ in the registration protocol. Upon successful completion of the protocol, W is increased by the value D.
2. *Charging.* $\mathcal{A}$ interacts with $\mathcal{C}$ in the charging protocol of value v. Upon successful completion of the protocol, W is decreased by the value v.
3. *Discharging.* $\mathcal{A}$ interacts with $\mathcal{C}$ in the topup protocol of value v. Upon successful completion of the protocol, W is increased by the value v.
4. *Statement.* $\mathcal{A}$ interacts with $\mathcal{C}$ in the statement protocol of value d. Upon successful completion of the protocol, W is increased by the value d.

Winning. $\mathcal{A}$ wins the game if there exists a sequence of interaction query so that W becomes negative.

Proof. Our security proof is by reduction. Specifically, assume there exists $\mathcal{A}$, we show how to construct a forgery attack against the underlying BBS+ signature [2]. Since BBS+ signature is known to be unforgeable, this means no PPT adversary $\mathcal{A}$ can win in the above game. That is, our system supports prevention of cheating user.

Before stating our proof, let us assume the zero-knowledge proof-of-knowledge $\mathcal{PK}_1$, $\mathcal{PK}_2$, $\mathcal{PK}_3$, $\mathcal{PK}_4$ are *sound*. That is, given blackbox access to the prover that makes these zero-knowledge proofs, there exists extractor algorithms $\mathcal{EX}_1$, $\mathcal{EX}_2$, $\mathcal{EX}_3$, $\mathcal{EX}_4$ which are capable of outputting the witnesses

used by the prover. Indeed, the protocols described in Appendix A are *sound* in the random oracle model.

Next, we describe an algorithm, called simulator, $\mathcal{S}$, which provides the view to $\mathcal{A}$ as the honest supplier and at that same time forges a BBS+ signature. $\mathcal{S}$ is given the public key of the BBS+ signature in the form of $(\hat{e}, \mathbb{G}, \mathbb{G}_T, g, g_0, g_1, g_2, g_3, w)$, together with a black-box $\mathcal{SO}$, normally referred to as *signing oracle*. $\mathcal{SO}$ outputs a BBS+ signature (A, e, y) on input (m_1, m_2, m_3). $\mathcal{S}$ successfully forges a BBS+ signature if it can output a valid signature (A^*, e^*, y^*) on message (m_1^*, m_2^*, m_3^*) such that the former is not the output of $\mathcal{SO}$[6].

Now we describe the behavior of $\mathcal{S}$. It sets $\mathsf{param} = (\hat{e}, \mathbb{G}, \mathbb{G}_T, g, g_0, g_1, g_2, g_3, w)$ and gives it to $\mathcal{A}$. Note that $\mathcal{S}$ does not know the secret key of the supplier but param is distributed correctly. Below we show how $\mathcal{S}$ interacts with $\mathcal{A}$ in each of the possible interactions. The value W is set to 0.

1. *Registration.* Upon executing $\mathcal{PK}_1$ with $\mathcal{A}$, $\mathcal{S}$ uses $\mathcal{EX}_1$ to extracts the witness (y', s). $\mathcal{S}$ assigns the unique identity I to this user and issues a signature query with input (I, D, s) to $\mathcal{SO}$. $\mathcal{S}$ receives (A, e, y) and computes $y'' = y - y'$. It returns (A, y'', e) to $\mathcal{A}$. $\mathcal{S}$ sets $W = W + D$.
2. *Charging.* Upon executing $\mathcal{PK}_2$ with $\mathcal{A}$, $\mathcal{S}$ uses $\mathcal{EX}_2$ to extracts the witness $(\tilde{A}, \tilde{e}, \tilde{y}, I, \tilde{B}, y', s)$. If $\tilde{A}, \tilde{e}, \tilde{y}$ is not the output of $\mathcal{SO}$, $\mathcal{S}$ outputs them as the forgery on $(I, \tilde{B}, \tilde{s})$. Otherwise, it checks if $\tilde{s}$ is fresh and issues a signature query with input $(I, \tilde{B} - v, s)$ to $\mathcal{SO}$. $\mathcal{S}$ receives (A, e, y) and computes $y'' = y - y'$. It returns (A, y'', e) to $\mathcal{A}$. $\mathcal{S}$ sets $W = W - v$.
3. *Discharging.* Upon executing $\mathcal{PK}_3$ with $\mathcal{A}$, $\mathcal{S}$ uses $\mathcal{EX}_3$ to extracts the witness $(\tilde{A}, \tilde{e}, \tilde{y}, I, \tilde{B}, y', s)$. If $\tilde{A}, \tilde{e}, \tilde{y}$ is not the output of $\mathcal{SO}$, $\mathcal{S}$ outputs them as the forgery on $(I, \tilde{B}, \tilde{s})$. Otherwise, it checks if $\tilde{s}$ is fresh and issues a signature query with input $(I, \tilde{B} + v, s)$ to $\mathcal{SO}$. $\mathcal{S}$ receives (A, e, y) and computes $y'' = y - y'$. It returns (A, y'', e) to $\mathcal{A}$. $\mathcal{S}$ sets $W = W + v$.
4. *Statement.* Upon executing $\mathcal{PK}_4$ with $\mathcal{A}$, $\mathcal{S}$ uses $\mathcal{EX}_4$ to extracts the witness $(\tilde{A}, \tilde{e}, \tilde{y}, y', s)$. If $\tilde{A}, \tilde{e}, \tilde{y}$ is not the output of $\mathcal{SO}$, $\mathcal{S}$ outputs them as the forgery on $(I, \tilde{B}, \tilde{s})$. Otherwise, it checks if $\tilde{s}$ is fresh and issues a signature query with input (I, D, s) to $\mathcal{SO}$. $\mathcal{S}$ receives (A, e, y) and computes $y'' = y - y'$. It returns (A, y'', e) to $\mathcal{A}$. $\mathcal{S}$ sets $W = W + D - \tilde{B}$.

Due to the setting of the game, the value W remains positive if $\mathcal{S}$ never aborts. This is because in order to reduce the value of W $\mathcal{A}$ has to interact with $\mathcal{S}$ in the discharge protocol and the number of signatures given to $\mathcal{A}$ via $\mathcal{S}$ is limited and that $\mathcal{PK}_3$ assures $\mathcal{S}$ will not accept on message of the form $(\cdot, B, \cdot)$ with $B < v$. Thus, in order for $\mathcal{A}$ to win the game, $\mathcal{S}$ will abort and obtain a forgery to the underlying BBS+ signature.

Location Privacy: Location privacy is defined via the following game. The rationale is that the malicious supplier cannot tell if a particular interaction is due to one out of two possible honest users under the extreme condition that all other interaction sequences are specified by the malicious supplier. Of course, the particular interaction could only be charging or discharging since identity of the actual user is to be known in registration and statement. Our definition also guarantee that the charging or discharging interaction are not linkable.

System Parameter. The malicious adversary $\mathcal{A}$ creates and publishes the system parameter param.

Interactions. $\mathcal{A}$ can make the following four types of interaction freely with $\mathcal{C}$, who acts on behalf of two honest users.

1. *Registration($b \in \{0,1\}$).* $\mathcal{A}$ interacts with $\mathcal{C}$ who acts on behalf of U_b in the registration protocol. The value b is specified by $\mathcal{A}$.
2. *Charging($b \in \{0,1\}$).* $\mathcal{A}$ interacts with $\mathcal{C}$ who acts on behalf of U_b in the charging protocol of value v for user . The value b is specified by $\mathcal{A}$.
3. *Discharging($b \in \{0,1\}$).* $\mathcal{A}$ interacts with $\mathcal{C}$ who acts on behalf of U_b in the topup protocol of value v. The value b is specified by $\mathcal{A}$.

[6] Note that this is formally called strong existential forgery under adaptive chosen message attack, one of the strongest possible attack on digital signature of which BBS+ has been proven to be immune of.

4. *Statement($b \in \{0,1\}$).* $\mathcal{A}$ interacts with $\mathcal{C}$ who acts on behalf of U_b in the statement protocol of value d. The value b is specified by $\mathcal{A}$.

Challenge. $\mathcal{A}$ chooses a type of interaction, either charging or discharging provided that both U_0 and U_1 has sufficient balance in case it is charging. $\mathcal{C}$ flips a fair coin $\hat{b} \in \{0,1\}$ and interacts with $\mathcal{A}$ on behalf of user $U_{\hat{b}}$.

Winning. $\mathcal{A}$ outputs a guess bit b and wins the game if $b = \hat{b}$.

Proof. Our security proof is to show that probability of $\mathcal{A}$ winning the game is always 0.5. That is, the action of two honest users are completely indistinguishable. The view of $\mathcal{A}$ is provided by a simulator $\mathcal{S}$ who has control over the random oracle used. Next we describe the behavior of $\mathcal{S}$.

1. *Registration.* $\mathcal{S}$ acts on behalf of the two users honestly.
2. *Charging.* $\mathcal{S}$ acts on behalf of the two users honestly.
3. *Discharging.* $\mathcal{S}$ acts on behalf of the two users honestly.
4. *Statement.* $\mathcal{S}$ acts on behalf of the two users honestly.

In the *Challenge Phase*, $\mathcal{S}$ first checks if both users are eligible to participate in the transaction. That is, they are having sufficient balance if the interaction is charging. Then $\mathcal{S}$ flips a fair coin $\hat{b} \in \{0,1\}$. Next, $\mathcal{S}$ randomly picks $C \in_R \mathbb{Z}_p$, $\tilde{s} \in_R \mathbb{Z}_p$ and sends them to the supplier. Upon receiving the random challenge R from $\mathcal{A}$ for $\mathcal{PK}_2$ or $\mathcal{PK}_3$, it randomly chooses $F_i \in_R \mathbb{G}$, c, z_{k_1}, z_{k_2}, z_{k_3}, z_{k_4}, $z_{y'}$, z_I, $z_{\tilde{B}}$, z_s, $z_{\tilde{e}}$, z_{β_1}, z_{β_2}, z_{β_3}, $z_{\beta_4} \in_R \mathbb{Z}_p$ and computes

$$
\begin{aligned}
T_1 &= C^c g_0^{z_{y'}} g_1^{z_I} g_2^{z_{\tilde{B}}} g_3^{z_s} \\
T_2 &= F_1^c h_1^{z_{k_1}} h_2^{z_{k_2}} \\
T_3 &= F_1^{-z_{\tilde{e}}} h_1^{z_{\beta_1}} h_2^{z_{\beta_2}} \\
T_4 &= \left(\hat{e}(F_2, w)\hat{E}^{-1}\hat{E}_3^{-\tilde{s}}\right)^c \hat{H}_0^{z_{k_2}} \hat{H}_1^{z_{\beta_2}} \hat{E}_0^{z_{\tilde{y}}} \hat{E}_1^{z_I} \hat{E}_2^{z_{\tilde{B}}} \hat{e}(F_2, g)^{-z_{\tilde{e}}} \\
T_5 &= F_3^c h_1^{z_{k_3}} h_2^{z_{k_4}} \\
T_6 &= F_3^{-vc} F_3^{-z_{\tilde{B}}} h_1^{z_{\beta_3}} h_2^{z_{\beta_4}} \\
T_7 &= \left(\hat{e}(F_4, th^{-v})\hat{H}^{-1}\right)^c \hat{H}_2^{z_{\beta_4}} \hat{H}_3^{z_{k_4}} \hat{e}(F_4, h)^{-z_\beta}
\end{aligned}
$$

Finally, $\mathcal{S}$ sets $c = H(\{F_i\}_{i=1}^4, \{T_i\}_{i=1}^7, R)$. This is possible since $\mathcal{S}$ is in control of the random oracle. $\mathcal{S}$ sends those values to $\mathcal{A}$ as $\mathcal{PK}_2$ or $\mathcal{PK}_3$ in the interaction. Note that the values are correctly distributed and can be based on the storage of U_0 or U_1. For any valid storage $(\tilde{\sigma_s}, I, \tilde{B}, \tilde{s})$, there exists a set of randomness that maps it to the view of the above protocol and that the value $\tilde{s}$ is completely hidden from $\mathcal{A}$. Thus the value $\hat{b}$ is completely hidden from the view of $\mathcal{A}$ and the probability that $\mathcal{A}$ guess correctly is always 1/2.