From Commodity to Value: A Privacy Preserving e-Business Architecture*

Yitao Duan and John Canny
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720, USA
{duan, jfc}@cs.berkeley.edu

Abstract

Privacy has been recognized as a very important issue in electronic commerce. However, many privacy techniques were not adopted and many online anonymity services failed. In this paper we propose treating privacy as a "value" that is to be added to other services to avoid the adoption pitfall. We present an architecture that anonymizes online transactions and makes them unlinkable to any customer or to each other. In the mean time, our scheme guarantees fair payment to the vendor and provides efficient fraud detection and tracing mechanisms. Not only our scheme anonymizes transactions that involve transmission of digitized messages only, it can also, with minimal change to existing delivery services, protect transactions that require delivery of physical items. Moreover, our system could give the vendor accurate aggregate *information about customers' usage pattern, which can be very valuable for its business operation. Our scheme utilizes existing financial infrastructure thus has very low adoption cost.*

1 Introduction

Automated and networked transaction processing gives the customers much convenience but also creates opportunities for attackers to pry on and misuse user's private information. Studies and surveys showed an interesting dichotomy between users' attitudes towards privacy and their behaviors regarding it. On one hand, most people stated that they value privacy and the fear of exposure of personal information often prevents them from conducting business online [9, 13]. On the other hand, studies such as [1, 15] have found that even those individuals who are concerned about privacy are willing to trade privacy for convenience or small rewards. This was believed to be one of the reasons for the failure of several online anonymity providers. It also led to the widespread belief that "people won't pay for privacy" or "privacy doesn't sell". Yet other studies (e.g. [24]) and observations showed the opposite. The market acceptance of curtains, caller ID services, etc., are all good examples that privacy does sell.

We believe the practice of "selling privacy" created a mismatch between what the market offers and what customers need: costumers need privacy and are willing to pay for it. But it is not offered in the right form. As [1] pointed out, individuals often act irrationally in the economic sense when facing privacy sensitive decisions and sometimes underestimate their privacy risk. This motives us to address the online privacy issue with a different approach: instead of treating privacy as a product or commodity that is to be sold for a price, we follow the perspective of viewing it as a *value*, which is what it really is, and construct schemes that allow an Anonymizing Service Provider (ASP) to *add* it to other existing services. The acceptance of ASP is not dependent on creating new "market for privacy". Rather, it rides on the popularity of other, more tangible services. By adding values to other services or goods, ASP can make them more attractive thus the other business will be willing to participate. This incentive, together with the low adoption cost of the scheme we present in this paper, will make our solution economically more viable.

Treating privacy as a value is not new: privacy has long been recognized as a value by law and the business community. However, this view is rarely followed in commercial practice by the privacy providers who are still trying to sell privacy as a *product*. Our contribution is to provide a concrete, practical technical solution that allows the idea to be implemented in practice. Compared to previous works, our scheme enjoys the following: (1) using cryptographic tools, it provides provable unlinkablity and anonymity for user transactions even to the *vendor* who can see the contents of the transactions. (2) With minimal change to existing delivery services, our scheme can anonymize transactions involving delivery of physical items which require real world identity and physical address. (3) Our scheme fea-

*This is the full version of a paper with the same title that appears in ICEBE '06.

tures a simple architecture involving only the ASP, the vendor and the user. Unlike many e-cash or payment schemes (e.g. [8, 28]), ours does *not* rely on special cooperation of financial institutions. Instead we utilize services in existing financial infrastructure and follow existing payment practice used by today's vendors. This lowers the cost of adoption and makes our scheme easier to implement.

The rest of the paper is organized as follows. In Section 2 we discuss possible business operation models and state our assumptions. Section 3 examines previous and existing privacy solutions and other related research. In Section 4 and 5 we give detailed description of our service architecture and the protocols. Section 6 gives a high-level analysis of our design against a number of possible attacks. Finally we discuss implementation issues in Section 7.

2 Models and Assumptions

We consider the scenario where users purchase goods or services from a vendor. The goods can be either in an electronic format or physical entities. A user initiates a transaction by submitting a query to the vendor. Multiple rounds of messages may be exchanged to complete a transaction. All the messages belonging to the same transaction form a *session*.

There are many models by which an online business can operate. This paper focuses on the following two:

1. Subscription model: Users subscript to a vendor and pay a fee for the service. The billing model can either be a flat monthly/yearly fee or pay-per-use. This model requires strong user authentication each time a user requests service.

2. Random shopper model: Users select a vendor, purchase the goods or services it offers and pay the vendor using either cash or credit/debit cards.

These two models are prevalent in both real world and e-commerce. The subscription model is the most interesting one as it offers opportunities to perform many collaborative tasks and provides infrastructure support for other services such as notification, storage, etc. We foresee these two models will still be prevailing in future e-commerce and securing them is the major focus of this paper.

We assume the communications between any two players are via secure channels and the vendor is separate from the communication provider and cannot monitor and control all the communication links. In this setting traffic analysis and timing attacks can be fended off by techniques such as [4] which are orthogonal to our work. We focus on preventing the vendor, who is allowed to see the content of a transaction, from linking the transactions.

We do not handle the attack where one party terminates the transaction early thus leaving the other with some loss. This can be addressed by techniques such as fair exchange [27] or external mechanisms such as legal action if the fraud is serious. However, we believe that for most applications the legal system and market forces provide enough deterrent so few parties should mount such attack thus the expensive fair exchange protocols are not necessary and the resulting system is much more efficient and scalable.

NOTATIONS Let K be a key, and K^{-1} its inverse. Subscripts will be added to distinguish the key's owner if necessary. In symmetric cryptosystems, $K = K^{-1}$. For asymmetric cryptosystems, K and K^{-1} are the public and secret part of a key pair (K, K^{-1}), respectively. Let X be a message or a data file and Y cipher-text, we use $E_K(X)$ to represent X encrypted under K. We use $D_{K^{-1}}(Y)$ to denote decryption of Y using decryption key K^{-1}. Signing a message X with private key K^{-1} is $S_{K^{-1}}(X)$.

3 Related Work

In [8] Chaum et al. introduced the digital cash, or e-cash, scheme. An e-cash scheme mimics physical cash and strives to provide features such as anonymity, unforgeability, uncopyability, etc. [1] Technically, e-cash systems provide good payment solutions for e-commerce, with provable security and anonymity. However, they require substantial prearrangement between parties such as the vendor, the bank and the customers and, to prevent double spending, the bank must be involved in each transfer and therefore is a bottleneck. Besides, many e-cash systems are based on cryptographic tools such as blind signature [6] or zero-knowledge proofs [11] and tend to be computationally expensive. In the end market forces worked against such systems and they were not widely adopted.

Nevertheless the ideas presented in e-cash systems are valuable and can potentially be used in many situations to enhance privacy. For example, an e-cash scheme can be used as an anonymous authentication and/or payment scheme. The idea is, instead of relying on the bank, the vendor sells the coins to the users who later buy services from the vendor with these coins. The vendor acts as *both* the bank and the merchant. Because of the elimination of the bank, this scheme is much simpler and more practical. The Unlinkable Serial Transaction (UST) scheme presented in [26] is a system based on such idea. This system, and other similar schemes, still require much prearrangement between the customer and the vendor. And because they rely on the anonymous tokens for user authentication, once a token is revealed but for some reason the transaction is not granted (the vendor can easily fake system failure), it

[1] See [17] for discussion on the requirements of an e-cash scheme.

either becomes unusable anymore or loses the unlinkability property.

[16] presented an anonymous credit card scheme that uses similar idea as ours, i.e. distributing trust among multiple parties. However, their transactions are more complex, involving the user, the vendor, the communication exchange and multiple banks. While distributing trust in general is a good idea, it is not clear how beneficial it is to scatter it all cross the system. And the trust invested in some of the parties are quite substantial (e.g. the communication exchange also needs to serve as an electronic Federal Reserve to transfer funds between banks) and cost/benefit sharing becomes an issue. Such overly complex architecture is hard to implement in practice, requiring the consent and cooperation of many parties (e.g. the banks) and it is likely that such systems will suffer the same fate as the e-cash systems.

Another area of research that is relevant to us is the work on anonymity and unlinkability. There are several papers that describe protocols for maintaining unlinkability between senders and recipients of a message. Examples include the "MIX-network" [5, 22, 21, 18, 23] and Onion routing [14] etc. Other techniques such as [7, 10] were also proposed to protect unlinkability in communication. Please see [3] for a good bibliography on anonymity. In Sect. 6.1 we will extend the framework established in [25] to define unlinkability and anonymity precisely and analyze our protocols.

4 System Overview

The subscription model requires user authentication while the random model calls for a payment scheme. In both cases a user inevitably has to disclose some of her personal information (e.g. her ID, address, etc) in order to conduct a transaction. The protection our scheme provides is unlinkability, namely a user's identity information is kept unlinkable to the transactions. This is achieved via the involvement of an Anonymizing Service Provider (ASP) that participates in the transactions between a user and the vendor and off-loads some responsibilities (thus information and trust) from the vendor.

The relationship among the ASP, the vendor and the users should be as follows:

- The ASP is a commercial entity providing such anonymizing service (such as what Anonymizer [2] provides for web surfing). It should sign a contract with the vendor guaranteing that the latter receives fair payment for the services or goods the it provides to the users. The contract should also forbid the ASP from misusing the vendor's services. [2]

- The ASP should also sign a contract with its user prohibiting it from colluding with the vendor to pry on user's privacy. Non-collusion is provided not only by this contract, which may have legal consequences if violated, but also other factors. Since the ASP is a separate entity independent of the vendor, there is a mutual *distrust* between the two. Collusion requires sharing of data and both parties will be aware of the cheating. Neither can trust the other not to expose the cheating.

- How the ASP generates revenue is not the focus of this paper but our solution supports some viable arrangements. For example, the ASP could charge the user a small fee for the anonymous service it provides. Another possibility is to charge the vendor instead, like the way credit cards work. Because the ASP adds value (privacy) to the vendor's services, more users will be attracted. So the vendor should have the incentive to cooperate.

The ASP's role in the scheme is twofold. It performs user authentication and anonymizes the communication. The vendor itself is in charge of processing user transactions. The ASP will know users identity, which is necessary for authentication, but not their transactions. The vendor knows the transactions, which are the product of service provision, but not their identity. The vendor may not be comfortable delegating user authentication to the ASP. Our protocol consists of a stage when the vendor is given the opportunity to authenticate users itself, albeit at a later time and without being able to link a user to any transactions, and verify that it receives the correct payment.

We stress that although the ASP is involved in user transactions, it is *not* treated as a trusted party. Our protocol bears some analogy to the secure socket layer (SSL) protocol. The ASP acts as an intermediate router and cannot gain access to the actual transactions. The harm a malicious ASP can cause is minimized.

5 The Protocol

The main transaction protocol consists of two stages: Query Processing and Billing Resolution. All transactions are carried out as Query Processing. Billing Resolution can run periodically, e.g. monthly, to resolve the balance between the participants. A bill or statement is generated for each user after this stage. We first describe our protocol for the subscription model (and will extend it to random shoppers in Sect. 5.2.1) where Alice needs to subscribe to the service first. This step may involve registration of some other information. As far as our protocol is concerned, she only needs to generate a public/private key pair and register

[2] We are only using the contract as a legal bound. Our scheme provides technological mechanisms to detect cheating by any parties and produce non-deniable evidence.

her public key with the vendor who signs her public key and issues her a certificate $CERT_A$.

5.1 Query Processing

When Alice wants to conduct anonymous transaction with the vendor V, she first forms a query

$$Q_0 = (Query, Timestamp, TTL, K_S)$$

where $Query$ specifies the transaction request, K_S is a one-time session key that will be used for communication during this session and TTL is the time this query is supposed to be valid. This is to prevent the system from wasting resources on stale requests. Alice then generates another secret key, K_Q, and encrypts Q_0 using symmetric encryption and assembles the following

$$Q = (E_{K_Q}(Q_0), E_{K_V}(K_Q), E_{K_A}(K_Q)) \quad (1)$$

and computes $h_Q = h(Q)$ where h is a cryptographic hash function. Alice then sends

$$M = (Alice, V, S_{K_A^{-1}}(h_Q), CERT_A, Q)$$

to her ASP, who is denoted Z.

Note that we assume a probabilistic public-key cryptosystem (e.g. El-Gamal) with semantic security which implies that two encryptions of the same message are (almost always) different. So it is impossible for the vendor to attack the unlinkability by encrypting K_Q with the public keys of all its registered users and comparing the ciphertexts with the message field.

The purpose of attaching $E_{K_A}(K_Q)$ in Q is that the message is also intended as a record for the transaction and will be sent back to Alice later as part of her statement (Sect. 5.2.2). It is unrealistic to assume that client machines can be very reliable and can save all the keys. Embedding $E_{K_A}(K_Q)$ in Q guarantees Alice's accessibility to the message yet protects it from all other parties.

Upon receiving and successfully decrypting the message, Z does the following:

1. Verifies the certificate $CERT_A$ and obtains Alice's public key, K_A.
2. Computes $h_Q = h(Q)$ and verifies $S_{K_A^{-1}}(h_Q)$.
3. If 1 and 2 are successful, Z sends $M' = (Z, S_{K_Z^{-1}}(h_Q), Q)$ to V.
4. Z saves M locally as record of the transaction.

Upon receiving M', V decrypts it and verifies Z's signature on h_Q. It recovers K_Q by decrypting $E_{K_V}(K_Q)$ and obtains the query Q_0. The query is processed if it is valid and an answer formed as:

$$A = (SessionID, h_Q, E_{K_S}(h_Q, t_P, R, S_{K_V^{-1}}(h_R)))$$

where $SessionID$ is the identifier the vendor assigns to this session. R is the reply to Alice's query, t_P is the time this query is processed and $h_R = h(h_Q, t_P, R)$ is a hash of the reply. The vendor sends A back to Z and saves the tuple $(t_P, Q, S_{K_Z^{-1}}(h_Q))$ under Z's account.

Using h_Q, Z finds the user who initiated the query, namely Alice, and forwards the answer to her. Once a session key is set up, Alice won't use Equation 1 to compute Q for subsequent messages. Instead Q is computed by

$$Q = (V, SessionID, E_{K_S}(Q_0))$$

and Z uses $(V, SessionID)$ to route messages.

The use of a session key is for efficiency as well as privacy: The vendor does not know who initiated the query so public key encryption is not possible.

5.2 Billing Resolution

During the execution of Query Processing, transaction records of all users are kept under the ASP's account. At the end of a transaction or billing period, users, ASP and the vendor all participate in this stage to resolve the bills and expose any fraud that could have happened. This stage is asynchronous and only involves passing of a few messages. We provide protocols for different billing models. Each has its own system requirements and implication on privacy which we will discuss in Sect. 6.

5.2.1 Proxied Payment Scheme

In this scheme a third party, PP (for Payment Proxy), acts as a proxy for both the customer and the vendor in the process of transferring fund. [3] Alice should have an account with PP. At the end of each transaction, Alice, the vendor V and PP execute the following:

Protocol PPS:

1. V sends PP a "transfer request ticket" (TRT):

$$TRT = ((date, \tilde{A}, V, PP, c, h_t), \\ S_{K_V^{-1}}(date, \tilde{A}, V, PP, c, h_t))$$

where $\tilde{A}$ is a one-time pseudonyme Alice generates randomly. $h_t = H(M_t)$ is a cryptographic hash on M_t, the set of all the messages exchanged so far during this transaction, and c is the price for this transaction.

[3] For generality we assume a separate PP in our description. In many applications the ASP can act as the PP.

2. Alice sends PP a "transfer authorization ticket" (TAT):

$$TAT_A = ((date, Alice, \tilde{A}, V, PP, c, h_t), \\ S_{K_A^{-1}}(date, \tilde{A}, V, PP, c, h_t))$$

3. PP uses the tuple (payer pseudonyme, payee, amount, h_t) to match a TRT to its corresponding TAT. If a match if found, it verifies the tickets and performs the transfer if both are valid. PP then bills Alice for this transfer (maybe with a small fee) or deducts some amount from Alice's pre-deposited fund. PP then notifies both parties of the result.

4. Upon receiving a "transfer success" message from PP, the vendor releases the goods or information. Otherwise it aborts the transaction.

This scheme is realistic and uses only existing financial infrastructure (e.g. the transfer between PP and V can use a regular credit card). There are already similar services (e.g. PayPal) with weaker privacy protection. It is conceivable that some existing ASPs such as Anonymizer [2] team up with some vendors and provide such PP service in the future.

This model can easily support the random shopper purchase behavior: A user can open an account with an ASP/PP and shop anonymously with participating vendors. We discuss the delivery of physical goods in Sect. 5.3.

5.2.2 Direct Payment – Pay-Per-Use Model

In the case where a payment proxy is not available, users have to pay directly to the vendor. The following protocol only applies to the subscription model where users pay for each use. In this case the vendor has the benefit of having aggregate information about each client's usage but users may have to live with lower degree of unlinkability. We will discuss the impact in Sect. 6.

The protocol should be carried out at the end of each billing cycle and has two stages:

Protocol DPS-INIT:

1. The vendor processes ASP's account. For each tuple $(t_P, Q, S_{K_Z^{-1}}(h_Q))$, it generates an item in the statement like $(t_P, Q, S_{K_Z^{-1}}(h_Q), cost, S_{K_V^{-1}}(cost))$ where $cost$ is the charge associated with this transaction. It then sends to the ASP an itemized statement listing all the queries this ASP has made and the associated cost.

2. The ASP validates the statement against its record and its signatures on each item. If they are correct, the ASP then generates statements for each of the users who have used its service. One item of the statement may look like

$$(Alice, V, t_P, Q, S_{K_A^{-1}}(h_Q), cost, S_{K_V^{-1}}(cost))$$

where $cost$ is obtained from the corresponding item on the statement received from the vendor and $S_{K_A^{-1}}(h_Q)$ is retrieved from ASP's own record. The statements are then sent to each user. Enclosed in the statement, the ASP also sends Alice a message containing the subtotal s_A for her:

$$M_{s_A} = (Alice, Date, N, s_A, S_{K_Z^{-1}}(s_A))$$

where N is a nuance is to prevent reuse attack.

3. Alice receives the statement and verifies $S_{K_A^{-1}}(h_Q)$. It then decrypts Q (by recovering the decryption key K_Q first) to obtain the original query Q_0 and verifies that s_A agrees with the sum of the cost of corresponding transactions. A human readable bill is generated and presented to Alice who should authorize payment if she agrees on the bill.

The next stage is to provide an efficient fraud detection mechanism:

Protocol DPS-PPU

1. Upon receiving and verifying a statement from Z, Alice sends to the vendor a message acknowledging the subtotal she has received from the ASP:

$$(M_{s_A}, S_{K_A^{-1}}(M_{s_A}))$$

2. The vendor checks the signatures on the message. If they are correct, it saves the message and sends back Alice *its* signature on the message as a receipt.

3. Alice sends this receipt to Z who verifies the vendor's signature, the nuance and the amount. It marks Alice as done and sends Alice a "done receipt" with its signature if everything is correct.

4. After a grace period, Z sends to the vendor $(\bar{U}, T_{\bar{U}})$ where $\bar{U}$ is the set of users who have used its service during this billing cycle but are not marked as done and $T_{\bar{U}}$ is the set of transaction records of the users in $\bar{U}$.

5. If the vendor finds any $u \in \hat{U} \cap \bar{U}$, where $\hat{U}$ are users who have acknowledged their usage, it contacts u requesting the "done receipt". If u fails to produce a valid "done receipt", then u did not follow the protocol faithfully and is possibly framing Z. Otherwise Z is framing u.

6. The vendor checks if $s = \sum_{u \in \bar{U} \cup \hat{U}} s_u$ where s is the total cost for all users during this billing cycle. If the equality holds, it should deal with users in $\bar{U} \setminus \hat{U}$, and possibly Z in case Z is found cheating in the previous step. The protocol terminates. Otherwise V sends $\hat{U}$ to Z.

7. Z checks $\hat{U}$ and its own records. If it finds any user u who is marked as done but $u \notin \hat{U}$, it concludes that V is cheating and shows its record as evidence. If it fails to do so, V concludes that Z has allowed illegitimate users to use the service.

Alice's signature on the message serves as an authentication and the vendor only issues a receipt for its valid customers. And Alice can't acknowledge a smaller amount to the vendor unless she can fake Z's signature.

5.2.3 Direct Payment – Flat-Rate Model

When the billing model allows the users to enjoy the service for a flat rate, the following protocol can be used and the result is stronger privacy.

This protocol makes novel use of a cryptographic tool called homomorphic commitment scheme. A commitment scheme allows a prover to give a verifier a commitment to a secret number. The verifier should not be able to compute any information about the secret from the commitment. The prover can later open the commitment by revealing the secret, and any auxiliary random bits accompanying this commitment, and the verifier can verify that the commitment indeed "contains" the secret. If the prover lies about the secret, it will be detected. Formally, let q be a prime. A commitment scheme for numbers modulo q consists of a polynomial time probabilistic function $\mathcal{C} : \mathbb{Z}_q \times \mathbb{Z}_q \to G$ where G is the set of all possible values of $\mathcal{C}$. To commit to an integer $x \in \mathbb{Z}_q$, the committer chooses $r \in \mathbb{Z}_q$ at random and computes the commitment X as $X = \mathcal{C}(x, r)$. To open a commitment, the prover sends x' and r' and the verifier can verify whether $X = \mathcal{C}(x', r')$. It should be infeasible for the prover to come up with x' and r' such that $x' \neq x$ but $X = \mathcal{C}(x', r')$. A commitment scheme is homomorphic if for any x_1, x_2, r_1, $r_2 \in \mathbb{Z}_q$, $\mathcal{C}(x_1 + x_2, r_1 + r_2) = \mathcal{C}(x_1, r_1)\mathcal{C}(x_2, r_2)$.

There exist efficient homomorphic commitment schemes such as Pedersen's discrete log based scheme [19]. Our protocol does not depend on any specific implementation and we will use it abstractly.

The protocol is also carried out in two stages and the first stage is the same as DPS-INIT in Sect. 5.2.2. Note that since this is a flat rate model the subtotal s_A is simply the amount of resources (e.g. bandwidth or number of transactions) Alice consumed during this billing cycle and s will be the total amount of resources processed by the vendor. The second stage is as follows:

Protocol DPS-FR

1. Upon receiving and verifying a statement from Z, Alice randomly selects $r_A \in \mathbb{Z}_q$ and sends to the vendor $(Alice, Date, N, c_A, P, S_{K_A^{-1}}(c_A))$ where $c_A = \mathcal{C}(s_A, r_A)$ and P is a non-interactive zero-knowledge proof that $s_A < L$ constructed using the protocol described in Appendix B for a given limit L.

2. The vendor checks the signatures and the proof. If they are correct, it saves the message and sends back Alice its signature on the message as a receipt.

3. Alice sends this receipt together with r_A to Z who verifies the message and the commitment $c_A = \mathcal{C}(s_A, r_A)$. If everything is correct it marks Alice as done and sends Alice a "done receipt" with its signature.

4. After a grace period, Z sends to the vendor $(r, \{r_u : u \in \bar{U}\}, \bar{U}, T_{\bar{U}})$ where $\bar{U}$ and $T_{\bar{U}}$ are the same as in Protocol DPS-PPU and $r = \sum_{u \in U} r_u$. U is the set of all the users who have used its service during this billing period.

5. If the vendor finds any $u \in \hat{U} \cap \bar{U}$, it contacts u requesting the "done receipt". If u fails to produce a valid "done receipt", then u did not follow the protocol faithfully and is possibly framing Z. Otherwise Z is framing u.

6. The vendor verifies whether $\mathcal{C}(s', r') = \prod_{u \in \hat{U} \setminus \bar{U}} c_u$ where $s' = s - \sum_{u \in \bar{U}} s_u$ and $r' = r - \sum_{u \in \bar{U}} r_u$. If the equality holds, it should deal with users in $\bar{U}$ and the protocol terminates. Otherwise V sends $\hat{U}$ to Z.

7. The same as the last step of Protocol DPS-PPU.

In this protocol the user "commits" to the vendor but opens the commitment to the ASP. This allows detection of fraud without disclosing s_A to the vendor. The purpose of performing the ZKP in the first step of the protocol is to prevent a cheating ASP from allowing illegitimate users to use the service by colluding with a malicious user. Without this step the ASP can put all the illegal usage to the colluder's account and let the user acknowledge a large amount to the vendor. Since now the number is hidden by the commitment, the vendor cannot detect this anomaly and the homomorphism verification still goes through. This way the ASP can "resell" the service for its own profit. By using the ZKP, the amount of service that the ASP can illegally "resell" is limited by a user's quota L, the sharing of which is in a sense a legitimate right of a user.

5.3 Delivery of Physical Goods

Some online transactions involve delivery of physical goods which requires the disclosure of user's ID and address. Here we propose a trivial change/addition to existing delivery services (e.g. UPS) that allows a sender to push an item to the network *without* knowing its destination or the receiver's ID. This is achieved by the following protocol run at the end of the transaction:

1. Alice generates a one-time public/private key pair $(\tilde{K}_A, \tilde{K}_A^{-1})$ and sends $\tilde{K}_A$ to V.
2. V generates a "label" for the item that is to be delivered to Alice: $L = (h_t, V, \tilde{K}_A, S_{K_V^{-1}}(h_t, V, \tilde{K}_A))$ where h_t is a cryptographic hash on all the messages exchanged between Alice and V during the transaction. It puts this label on the item and ships it to the "discreet deliver" (DD) for delivery.
3. Alice sends her ID and address, together with a "pull request" (PR) to DD: $PR = (h_t, V, S_{K_A^{-1}}(h_t, V))$.
4. DD uses PR to find the label for this request and authenticates Alice using the public key enclosed in the label. If it is successful, it adds Alice's ID and address to the package and ships it. It also notifies both the vendor and Alice. It may charge Alice or the vendor for the shipping cost.

6 Security Analysis

In this section we use the framework developed in [25] to analyze the unlinkability and anonymity our scheme achieves and discuss potential attacks that participants of our protocol can mount.

6.1 Unlinkability

For any billing cycle, let $U = \{u_1, u_2, \ldots, u_n\}$ be the set of active users and $T = \{t_1, t_2, \ldots, t_m\}$ be the set of transactions. According to [25], unlinkability is defined in terms of equivalence relations on one or two sets. Specifically, in our context, let $\sim_{r(T)}$ denote the equivalence relation on set T and $\sim_{r(U,T)}$ denote that on sets $U \times T$. For any $u \in U$ and $t \in T$, $u \sim_{r(U,T)} t$ if u conducted t. For any $t_i, t_j \in T$, $t_i \sim_{r(T)} t_j$ if $\exists u \in U$ such that $u \sim_{r(U,T)} t_i$ and $u \sim_{r(U,T)} t_j$.

The degree of unlinkability is defined as the attacker's a posterior entropy. On transactions:

$$\begin{aligned} d(t_i, t_j) &:= H(t_i, t_j) \\ &= -P(t_i \sim_{r(T)} t_j)\log(P(t_i \sim_{r(T)} t_j)) \\ &\quad -P(t_i \nsim_{r(T)} t_j)\log(P(t_i \nsim_{r(T)} t_j)) \end{aligned}$$

Between users and transactions:

$$\begin{aligned} d(u_i, t_j) &:= H(u_i, t_j) \\ &= -P(u_i \sim_{r(U,T)} t_j)\log(P(u_i \sim_{r(A,T)} t_j)) \\ &\quad -P(u_i \nsim_{r(U,T)} t_j)\log(P(u_i \nsim_{r(A,T)} t_j)) \end{aligned}$$

Clearly $d(t_i, t_j) = 1$ iff $P(t_i \sim_{r(T)} t_j) = P(t_i \nsim_{r(T)} t_j) = 1/2$ and $d(u_i, t_j) = 1$ iff $P(u_i \sim_{r(U,T)} t_j) = P(u_i \nsim_{r(U,T)} t_j) = 1/2$.

Please refer to [25] for the intuition behind these definitions.

These definitions are good for measuring the *state* of a system, but not suitable for assessing how a protocol or a process affects the unlinkability. Because they do not consider a priori information on structure of $\sim_{r(T)}$ or $\sim_{r(U,T)}$, it is not clear how the attacker acquires the a posterior knowledge: Does it learn the information by observing/participating in the protocol or does it know it already prior to the protocol?

For this purpose we extend the definitions and introduce the notions of a priori and a posteriori unlinkability. Degrees of a priori and a posteriori unlinkability (for both within a set and between sets) are defined as the degrees of unlinkability calculated with the above definitions using a priori and a posteriori probabilities, respectively. Let $d_0(T), d(T)$ be the a priori and a posteriori unlinkabilities within the set T respectively and $d_0(U,T), d(U,T)$ be the a priori and a posteriori unlinkabilities between sets U and T, respectively. We define the advantage of an attacker, $\mathcal{A}$, on protocol Π's unlinkability as

$$\mathrm{ADV}^{\mathcal{A}}_{\Pi,U} = \max_{\Pi}\Big(\max_{u\in U, t, t_i, t_j \in T}\big(|d(t_i,t_j) - d_0(t_i,t_j)|, \\ |d(u,t) - d_0(u,t)|\big)\Big)$$

This definition captures how much information $\mathcal{A}$ acquires from running Π that can be used to attack unlinkability and we believe it is a suitable definition for measuring the *protocol's* role in the system's overall unlinkability.

In addition to unlinkability, anonymity is also a frequently used term to quantify privacy. In [20] anonymity is defined as the state of being not identifiable within a set of subjects (the anonymity set). Given a transaction $t \in T$ and a user $u \in U$, the degree of anonymity for u can be measured as the probability that an attacker cannot associates t to her: $d(u,t) = 1 - P(u \sim_{r(U,T)} t)$ [12]. Again we extend this definition to include a priori probability as well to measure how a protocol affects an attacker's ability to identify any user for any given transaction. An attacker $\mathcal{A}$'s advantage on protocol Π's anonymity is defined as

$$\mathrm{ADV}^{\mathcal{A}}_{\Pi,A} = \max_{\Pi}\Big(\max_{u\in U, t\in T}|P(u \sim_{r(U,T)} t|\Pi) - P(u \sim_{r(U,T)} t)|\Big)$$

Unlinkability is a sufficient but not necessary condition of anonymity [20] which provides weaker privacy protection. Please see [20] for detailed discussion on their relation. We will show that one of billing models cannot not achieve full unlinkability but, with the right dynamics, can have good anonymity.

Using these definitions, we define unlinkable/anonymous protocols under a given adversary $\mathcal{A}$ as:

Definition 1 (Unlinkable/Anonymous Protocol) *Given the sets U and T and two equivalence relations $\sim_{r(T)}$ and $\sim_{r(U,T)}$, Let $\Pi(u, P_1, \ldots, P_k)$ be a protocol involving any user $u \in U$ and k other parties. For any party $P \in \{P_1, \ldots, P_k\}$, if either U or T is disclosed to P after the protocol, it is considered P's a priori knowledge.* [4] *Π is d-unlinkable if $1 - \mathrm{ADV}^{\mathcal{A}}_{\Pi,U} \geq d$. A protocol is unlinkable if it is 1-unlinkable. Furthermore, Π is d-anonymous if $1 - \mathrm{ADV}^{\mathcal{A}}_{\Pi,A} \geq d$. A protocol is anonymous if it is 1-anonymous.*

Note that a protocol being unlinkable or anonymous does not guarantee that the system is unlinkable or anonymous: it is possible that information is leaked elsewhere. But securing the protocol is a necessary condition for the system to be privacy preserving. And since our scheme provides a comprehensive solution for transaction processing, billing resolution and fraud detection, at least for some applications, the vendor won't need external mechanisms to collect additional user information. In this case the system provides fairly good privacy protection.

In the following we analyze the unlinkability of our protocols under the assumptions that all participants follow the protocol. In Sect. 6.2 we discuss the consequences of malicious players who can deviate arbitrarily from the protocol.

Assuming the cryptographic tools are secure, we have the following results:

Theorem 1 *Protocols PPS, DPS-FR and DPG are all unlinkable.*

Theorem 2 *Let $\pi(u)$ be the number of transactions conducted by user $u \in U$. We assume that, $\forall u \in U$, $\pi(u) \geq 1$. Let $\bar{\pi}$ be the average number of transactions conducted by one user and $\delta = \max_{u \in U}(\frac{\pi(u)}{\bar{\pi}})$. Then DPS-PPU is not unlinkable but can be $(1 - |\frac{\delta - 1}{n}|)$-anonymous.*

The proof of theorem 1 follows straightforward examination of the equivalence relations $\sim_{r(T)}$ and $\sim_{r(U,T)}$ throughout the protocols and is omitted due to lack of space. The proof of theorem 2 is given in the appendix.

[4] Knowing U or T does not give an adversary any more advantage on attacking the system's unlinkability/anonymity. It is the equivalence relations $\sim_{r(T)}$ and $\sim_{r(U,T)}$ that matter.

REMARKS As the above results show, among all the schemes, DPS-PPU does not provide full unlinkability. This is an inherent problem with this billing model. Essentially since the users have to pay directly to the vendor, there is no way to prevent the latter from learning how many transactions a user conducted which reduces the scheme's unlinkability. However, as Theorem 2 shows, users can still remain anonymous (albeit with weaker protection) even in protocol DPS-PPU. The best anonymity is reached when all users conduct the same number of transactions. DPS-PPU gives the vendor anonymous aggregate information about each user which may be useful in optimizing its operation and even be beneficial to the customers. For example, the vendor may offer bargain price to those who buy in bulk. The choice of protocols is left as an application decision.

6.2 Security Against Possible Attacks

The involvement of the ASP introduces some risk for the vendor. Essentially the vendor delegates the task of user authentication to the ASP. Perhaps from the vendor's perspective, the biggest fear is that the ASP may misuse this power and grant illegitimate users access to its services. The ASP may even act as a "reselling proxy" and resell the services for a profit. We show that this is not possible without being detected.

First, the power of ASP is restricted. All messages passing through the ASP are encrypted with keys to which it does not have access. As far as the transactions are concerned, the ASP is acting as an oblivious router. Second, in PPS, the vendor won't complete the transaction (e.g. start shipping the goods) until the customer authorizes the fund transfer. This effectively protects it from misuse. In DPS-PPU, each legitimate user will acknowledge to the vendor the amount she owns it. This gives the vendor a chance to authenticate the users itself and allows it to tally its balance by summing up the acknowledgements and comparing against its own record. If there is inconsistency between s, s' and the number reported by the ASP, then there could be potential fraud by either a user or the ASP. The final steps in Protocol DPS-PPU is to track the faulty party. If a user u tries to evade payment for the services she receives, her ID will be in the list $\bar{U}$ which is sent to the vendor. If she is framing ASP by not forwarding the acknowledge receipt to it, she will fail to produce a "done receipt" when asked. If the ASP allows illegal access to the system, or if it tries to frame a user, either it will fail to produce a user not in the vendor's list ($\hat{U}$) who has a valid acknowledgement receipt from the vendor or the user can show a valid "done receipt". In either case it can be detected. DPS-FR is similar but uses homomorphic bit commitment scheme so that the vendor does not even see the actual number a user uses its system.

A user can cheat by not acknowledging her subtotal to

the vendor or acknowledging a smaller amount. Both will cause imbalance in the vendor's account and trigger investigation. In either DPS-PPU or DPS-FR, the ASP will report her to the vendor otherwise the ASP has to bear the consequences.

The vendor can cheat by overcharging users. This can be achieved by not processing queries but still save them as proof of service to charge the user. One solution to this problem is to require that each charge be supported by proof that the user actually receives the reply. But then user can pretend that she never receives the rely and refuses to pay for the service she actually uses. A complete solution must be based on two party secret exchange protocols which are expensive. However, the vendor cannot overcharge the customer without being detected. Recall that each month the user will be presented an itemized statement and any forged charge should be noticed by the user and generate a complain. Further, user will experience delay or denial of service if the vendor deliberately drops queries. This will affect customers satisfaction and push them to the competitors. Market force should prevent the vendor from doing so.

7 Discussion

The success of a design depends on many factors, not all of them technical. But efficiency is a very important pragmatic issue. We estimated the running time of our scheme based on recent benchmarks for the Botan cryptographic library (http://botan.randombit.net/bmarks.html). In terms of computation, Protocol DPS-FR is the most expensive one, involving ZKP and commitment. The ZKP requires $\log L$ steps. For each step, showing in zero-knowledge that each bit has value 0 or 1 using Cramer and Damgård [11] requires constant (actually 7) long (mod p) integers. So the total complexity of this ZKP for one user is $O(\log L)$. Using Pedersen's commitment scheme [19], the protocol involves $14 \log L$ exponentiations. Even with $L \sim 10^8$ and large q (e.g. 1024 bits), computing time for each user is around a few seconds. This means the vendor (as well as the ASP) can easily process over 24,000 users in one day using even a single machine (The benchmarks are computed on a 1.6 GHz AMD Opteron running Linux). Since the protocol for each user is independent of each other, the scheme can easily scale to larger number of users with a cluster.

Our scheme also features the ability to utilize existing services and standards. First of all, it only uses existing services (e.g. fund transfer methods) provided by today's financial infrastructure. This is in contrast to many electronic payment schemes (such as [8, 28]) that require special cooperations of some financial institutions. Moreover, the query processing protocol bears remarkable similarity to the Secure Sockets Layer (SSL) protocol. This is not all by coincidence. This design feature allows us to leverage a widely used standard for web communication: HTTP over SSL sockets. The protocol described in Sect. 5 exposes all its internal mechanisms to the ASP. On the vendor side, however, it is easy to implement this step using HTTPS and SSL *without* any custom code, apart from what implements the service logic. The adoption cost thus can be made low and our scheme provides a viable solution for preserving user privacy in e-commerce.

A Proof of Theorem 2

Proof: In this protocol, each user's subtotal is made available to the vendor. If the transactions are not equally priced, the vendor can mount the following attack: It can associate a different amount c_i with each transaction i. When the user u acknowledges with a subtotal s_u, the vendor can try to compute a subset Λ of all transactions such that $\sum_{i\in\Lambda} c_i = s_u$ and link the user's transactions to her identity. This is a classic Knapsack problem and known to be NP-complete. But if the scale of the system is not large, it can be to solved easily. In this case, the protocol is neither unlinkable nor anonymous.

If the transactions are equally priced, then from the subtotal s_u, the vendor can compute $\pi(u)$. The equivalence relation $\sim_{r(U,T)}$ defines equivalence classes on T: $\forall t \in T, t \in T_u$ if $u \sim_{r(U,T)} t$. These equivalence classes divide T into subsets. The sizes of these sunsets (i.e. the $\pi(u)$'s) are known to the vendor. By Theorem 1 in [25], for the vendor the a posteriori unlinkability cannot be 1. However, given only U and T, for the vendor, the a priori unlinkability is 1. This means the vendor's advantage must be greater than 0 and the protocol is not unlinkable.

As for anonymity, let Π = DPS-PPU. For any $u \in U$ and $t \in T$, $P(u \sim_{r(U,T)} t|) = \frac{1}{n}$ and $P(u \sim_{r(U,T)} t|\Pi) = \frac{\pi(u)}{\sum_{v\in U}\pi(v)}$. The vendor's advantage on attacking anonymity is $\mathrm{ADV}^{\mathcal{V}}_{\Pi,A} = \max_\Pi(\max_{u\in U, t\in T} |P(u \sim_{r(U,T)} t|\Pi) - P(u \sim_{r(U,T)} t)|) = \max_\Pi(|\frac{\delta\bar{\pi}}{m} - \frac{1}{n}|) = \max_\Pi(|\frac{\delta-1}{n}|) = |\frac{\delta-1}{n}|$.

Therefore DPS-PPU is $1 - |\frac{\delta-1}{n}|$-anonymous. In particular, when $\delta = 1$, i.e., every user conducts the same number of transactions, DPS-PPU is anonymous. ■

B ZK Proof of Limit on Committed Number

To simplify notation, let $S = \mathcal{C}(s, r)$ be the commitment to s using randomness r. Here we present a ZKP that $s < L$. Let $s_1 s_2 \ldots s_l$ be the binary representation of s where $s_i \in \{0, 1\}$, $i = 1, 2, \ldots, l$. The prover generates $l - 1$ random numbers $r_1, \ldots, r_{l-1} \in \mathbb{Z}_q$ and computes $r_l = (r - \sum_{i=1}^{l-1} r_i 2^{i-1})/2^{l-1} \mod q$. The prover then commits to s_i using r_i: $S_i = \mathcal{C}(s_i, r_i)$. She sends the S_i's to the verifier.

For $i = 1, \ldots, l$, the prover gives a zero-knowledge proof that S_i encodes a bit (i.e. it is either 0 or 1) using the protocol in [11]. The verifier then verifies that $S = \prod_{i=1}^{l} S_i^{2^{i-1}}$. This shows that s is at most l-bit thus $s < L = 2^l$, but gives no other information. The proof is honest-verifier zero-knowledge, following the security of the bit commitment protocol. It can be made non-interactive by using a secure hash function to compute the verifier's response.

References

[1] A. Acquisti. Privacy in electronic commerce and the economics of immediate gratification. In *ACMEC '04*, pages 21–29. ACM Press, 2004.

[2] Anonymizer Inc. Anonymizer. http://www.anonymizer.com, 2003.

[3] Anonymous. Anonymity bibliography. http://freehaven.net/anonbib/.

[4] R. Berman, A. Fiat, and A. Ta-Shma. Provable unlinkability against traffic analysis. In *FC '04*. Springer-Verlag, LNCS 3110, 2004.

[5] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[6] D. Chaum. Blind signature systems. In *CRYPTO '83*, page 153, 1983.

[7] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[8] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proceedings of Advances in cryptology*, pages 319–327. Springer-Verlag, 1990.

[9] F. T. Commission. Privacy online: Fair information practices in the electronic marketplace, 2000.

[10] D. A. Cooper and K. Birman. Preserving privacy in a network of mobile computers. In *IEEE Symposium on Security and Privacy*, pages 26–38, 1995.

[11] R. Cramer and I. Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In *CRYPTO '98*, pages 424–441. Springer-Verlag, 1998.

[12] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *PET 2002*. Springer-Verlag, LNCS 2482, April 2002.

[13] ebusinessforum.com. eMarketer: The great online privacy debate, 2000.

[14] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42(2), February 1999.

[15] I.-H. Harn, K.-L. Hui, T. S. Lee, and I. P. L. Png. Online information privacy: Measuring the cost-benefit trade-off. In *23rd International Conference on Information Systems*, 2002.

[16] S. H. Low, S. Paul, and N. F. Maxemchuk. Anonymous credit cards. In *CCS '94*, pages 108–117. ACM Press, 1994.

[17] T. Okamoto and K. Ohta. Universal electronic cash. In *CRYPTO '92*, pages 324–337. Springer-Verlag, 1992.

[18] C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Proceedings of EUROCRYPT 1993*, pages 248–259. Springer-Verlag, LNCS 765, 1993.

[19] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91*, volume 576 of *LNCS*, pages 129–140. Springer-Verlag, 1991.

[20] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity: A proposal for terminology. Draft, version 0.17, July 2001.

[21] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, February 1991.

[22] A. Pfitzmann and M. Waidner. Networks without user observability – design options. In *EUROCRYPT 1985*. Springer-Verlag, LNCS 219, 1985.

[23] C. Rackoff and D. R. Simon. Cryptographic defense against traffic analysis. In *ACM STOC 93*, pages 672–681, 1993.

[24] A. Shostack. "people wont pay for privacy," reconsidered. In *2nd Workshop on Security and Economics*, May 2003.

[25] S. Steinbrecher and S. Köpsell. Modelling unlinkability. In *PET 2003*. Springer-Verlag, LNCS 2760, March 2003.

[26] S. G. Stubblebine, P. F. Syverson, and D. M. Goldschlag. Unlinkable serial transactions: protocols and applications. *ACM Trans. Inf. Syst. Secur.*, 2(4):354–389, 1999.

[27] T. Tedrick. Fair exchange of secrets. In *CRYPTO '84*, volume 196 of *LNCS*, pages 434–438. Springer-Verlag, 1984.

[28] B. Yang and H. Garcia-Molina. PPay: micropayments for peer-to-peer systems. In *CCS '03*, pages 300–310. ACM Press, 2003.