A Methodology for Retrofitting Privacy and Its Application to e-Shopping Transactions

Jesus Diaz[1], Seung Geol Choi[2], David Arroyo[3], Angelos D. Keromytis[4], Francisco B. Rodriguez[3], and Moti Yung[5]

[1] Blue Indico - BEEVA
jesus.diaz@beeva.com, jesus.diaz.vico@gmail.com
[2] United States Naval Academy
choi@usna.edu
[3] Universidad Autónoma de Madrid
{david.arroyo,f.rodriguez}@uam.es
[4] Georgia Institute of Technology
angelos@gatech.edu,
[5] Columbia University,
moti@cs.columbia.edu

Abstract. The huge growth of e-shopping has brought convenience to customers and increased revenue to merchants and financial entities. Moreover, e-shopping has evolved to possess many functions, features, and requirements (e.g., regulatory ones). However, customer privacy has been mostly ignored, and while it is easy to add simple privacy to an existing system, this typically causes loss of functions. What is needed is enhanced privacy on one hand, and retaining the critical functions and features on the other hand. This is a dilemma which typifies the "privacy vs. utility" paradigm, especially when it is applied to an established primitive with operational systems, where applying conventional privacy-by-design principles is not possible and completely altering information flows and system topologies is not an option. This dilemma is becoming more problematic with the advent of regulations such as the European GDPR, which requires companies to provide better privacy guarantees whenever and wherever personal information is involved.

In this work, we put forward a methodology for privacy augmentation design that is specially suitable for real world engineering processes that need to adhere to the aforementioned constraints. We call this the "utility, privacy, and then utility again" paradigm. In particular, we start from the state of the art industry systems that we need to adapt; then we add privacy enhancing mechanisms, reducing functionality in order to tighten privacy to the fullest (privacy); and finally, we incorporate tools which add back lost features, carefully relaxing privacy this time (utility again).

Specifically, we apply this process to current e-shopping infrastructures, making them privacy-respectful without losing functionality. This gives an e-shopping system with enhanced privacy features, presents a set of "utility-privacy trade-offs," and showcases a practical approach implementing the notion of "privacy by design" while maintaining as much compatibility as possible with current infrastructures. Finally, we note that we implemented and tested performance of our design, verifying its reasonable added costs.

1 Introduction

Privacy for systems in production? The principle of privacy by design mandates that privacy enhancing mechanisms need to be taken into account early at the design stage of any system. Although important and prevalent, however, this principle is not very helpful when we would like to enhance privacy of the infrastructures and systems that have already been deployed and well-established. One could attempt to apply this principle by re-designing the whole (existing) system from scratch. Even in this case, however, one cannot rule out the existing system completely. In particular, the new design must still be greatly constrained by the main processes and information flows of the existing system. Otherwise, the amount of chain-effect changes that the new design could entail may be too costly and thereby unacceptable.

As an example of the costs of modifying deployed systems, the banking ecosystem comes to mind. IBM's mainframes have been the foundation of most banking infrastructures for decades. In recent years, probably due to the arrival of PSD2[6] and similar initiatives, deep changes have been made in the banking industry. However, even with these years-long efforts, work remains to be done. This reflects the complexity of updating already established systems.

Still, with the advent of another European regulation, the GDPR[7], we are urged to think about how to incorporate privacy into already existing processes. This places privacy engineers in a delicate spot. We need to not only maintain compatibility (i.e., reasonably similar interfaces) with related processes, but also introduce privacy in a context where this very privacy often directly conflicts with the provided services. Think, for instance, of enhancing privacy for fraud prevention and marketing tools, which have become essential parts of the e-commerce ecosystem. To support these services, a record of users' activities seems necessary, but this conflicts privacy.

Privacy vs. utility inspiration: the group signature case. The evolution of privacy primitives in various specific domains often centers around the notion of balancing privacy needs and utility requirements. For example, consider the basic notion of "digital signature" [31,61] whose initial realization as a public key infrastructure [58] mandated that a key owner be certified with its identity and its public verification key. This is done by a certification authority (CA) which signs a record (called certificate) identifying the user and its signature public verification key. Later on, for some applications, it was suggested that CA's sign anonymous certificates which only identify the keys (for example, a bulk of keys from a group of users is sent to the CA via a mix-net and the CA signs and publish the certificates on a bulletin board: only the owner of a key can sign anonymously with its certified key. Alternatively the CA blindly signs certificates). This brings digital signing to the domain of anonymous yet certified action (i.e., the action/message is known to originate from the group that was certified). However, it was noted quite early that under the mask of anonymity users can abuse their power and sign undesired messages, where no one can find the abuser. Therefore, primitives like group signature [22] or traceable signature [41] were designed, assuring that the anonymity

[6] `https://ec.europa.eu/info/law/payment-services-psd-2-directive-eu-2015-2366_en`. Last access on April 17th, 2018.

[7] `https://www.eugdpr.org/`. Last access on April 17th, 2018.

property of a signed message usually stays, but there are authorities which can unmask abusers, or unmask certain message signatures in order to keep balance between anonymity of well behaving signers while protecting the community against unacceptable message signing practices. More complex actions can be constructed based on this setting, e.g., checks that are anonymous to the general public, while the clearing house is the revocation authority which revokes anonymity in order to clear checks based on the checks' actual account.

Utility, privacy, and then utility again. The above development on group signatures shows that even in one of the simplest case of anonymity vs. basic message authenticity, there is already certain advantage in providing partial anonymity to perform in a desirable environment which balances various needs. Additionally, the described case of privacy by design for already deployed systems calls out for variants of this methodology. For this, extrapolating from the above staged methodology that gave us the primitives of group signature and traceable signature, we follow a methodology that can be viewed as "utility, privacy, and then utility again": first translating a primitive to an idealized anonymous primitive, but then identifying lost utility which complete anonymity prevents; and, in turn, relaxing privacy for additional utility. In the full circle of e-shopping this gives rise to numerous trade-offs which we unveil and discuss (based on utility needed in various steps of the transaction system). Importantly, our methodology allows us to maintain the same basic information flow and main processes than in current e-shopping systems, thus making it easier to come up with a proposal compatible with the existing complex e-commerce ecosystem.

Our results. We put forward our approach in this work through to the involved case of the real world (compound) process of e-shopping, to which we apply the aforementioned construction framework. As a result, we obtain an e-shopping system that maintains the same infrastructure and information flow than industry e-shopping systems and offers a higher privacy level, while still maintaining added value services such as marketing and fraud prevention tools.

We implemented a prototype of our mechanism. This allows us to check the actual applicability from a software development perspective (beyond paper design of transactions, crypto building blocks, and protocol exchanges), and it enables us to report the measurement results for the overall transaction processes, demonstrating that our system incurs low additional costs.

1.1 Organization

We contextualize our e-shopping proposal among related work in Section 2 and we describe our methodology for maintaining both utility and privacy in Section 3. Then, after some preliminaries in Section 4, we proceed to apply the methodology to the e-shopping case: we begin by modeling the e-shopping ecosystem, pinpointing the core entities and processes, as well as the added value tools it incorporates in Section 5, where we additionally define the privacy threats. Then, we sketch in Section 6 the result of applying privacy to the traditional system (*Privacy*); and finally, we analyze this system to show its shortcomings and proceed to recover utility in Section 7 (*Utility again*). Section 8 formalizes the security properties that our system needs to ensure and provides the necessary proofs. Next, in Section 9 we present the experimental results obtained from a prototype implementing our proposal. In Section 10, we chart a way to endow our infrastructure with additional functionality (typically available in industrial systems). In Section 11, we conclude and outline our future work.

2 Related Work

There is a plenty of work on ensuring privacy in the different sub-processes of e-shopping.

Payment. The most prolific area has been on anonymous payments, e-cash [21] being its main representative. There has been a huge boost since the proposal of Bitcoin [49], creating the so-called crypto-currencies. While Bitcoin itself does not provide robust privacy, more advanced proposals have been made addressing it [46,10,34][8]. These systems address only the payment sub-process, and are typically not concerned with additional functionalities, except [34], where Zerocash is extended to incorporate support for regulatory concerns. Some traditional e-cash proposals also incorporate utility to some extent, mainly enabling tracing (after the payment has been done) [18,27,50] or spending limitation [50,64]. Privacy respectful payment systems out of the e-cash domain also exist, such as [39] which is built on mix networks to prevent linking customers and merchants, and [66] which introduces discounts based on the users' history (users are always pseudonymous).

Purchase. Privacy preserving purchase systems have been divided in two types of systems depending on the information hidden to service providers [59]: private purchase systems hiding the purchased items [60], and anonymous purchase systems hiding buyers' identities [65]. In particular, the system in [65] works by interleaving proxies that contain software removing identifiable information about customers.

Profiling, delivery, and completion. There have been works focusing on privacy respectful user profiling [67,54,26], mostly for affinity programs, although some approaches are also applicable to fraud prevention [26]. Anonymous delivery systems of physical goods have also been proposed [65,5], covering a crucial phase that has received much less attention. Finally, solutions related to the completion phase (leaving feedback, complains, etc.) have been basically ignored, although this phase have been shown to allow de-anonymization attacks [47]. Underlying most of these proposals are, often, cryptographic primitives such as oblivious transfer [2] or anonymous credentials [16,24], which are of natural interest in this domain as core building blocks.

Comparison with our work. One common aspect among these previous works is that they deal only with specific sub-processes of the overall e-shopping process. They either focus solely on the purchase, payment (checkout), delivery or completion phase. As noted in [30], a holistic approach must be adopted if we aim to protect users throughout all the process. Moreover, not doing so entails a single point of failure since privacy

[8] As well as many proposals in non-academic forums. See, for instance, `https://z.cash/` (a modified implementation of Zerocash) and `https://cryptonote.org/`. Last access on March 21st, 2018.

violations in any of the remaining unprotected phases erode the overall privacy protection.

Some proposals introduce extensive changes into the infrastructure and information flow [39] or require modifications that conflict with regulations or other practical concerns, like requiring the outsourcing of information that would probably be proprietary in many scenarios [67,26].

To conclude, at present, the utility-privacy trade-off is clearly leaning towards utility in the industry and towards full privacy in the academic literature. And, importantly, in any case, there does not seem to exist a wide enough approach that covers all the phases of the e-shopping process. In this work, we aim at addressing both these issues by proposing a novel system that, we believe, typifies a methodology for introducing privacy to real-world processes.

3 Methodology

Well established wisdom in the field states that, if instead of assuming privacy as a requirement from the beginning, it is just added to the system after it has been built, the achieved privacy is very weak. Indeed privacy by design is a necessary principle when architecting new information systems. However, in the real world, users' acceptance is a must to achieve effective and efficient privacy respectful products. Therefore, besides privacy, we have to bear in mind two additional requirements: compatibility and utility.

Compatibility. It has been now several decades since the first electronic communication and information systems were built. A lot of standardization efforts have taken place, relating to data representation, processes, interoperatiblity, etc. Similarly, a vast amount of infrastructures have been deployed. Also, while initially these were standalone systems, with the improved capacity and quality of communications, we now live in a hyperconnected world, where even *things* talk to *things*. Consequently, if we have to change one of these connected components, we need to think of the side effects this change will have to the related systems.

Utility. Current widely deployed information systems have been built with the purpose of enhancing convenience to users and service providers, either by enabling a new use case, or by improving an existing one through the introduction of added value services. Social media platforms recommend new friends by examining the contact list in your mobile phone, second-level connections, common hobbies, etc. Advertising systems track your browsing habits to better filter your interests and offer you goods and services that will most probably be of your interest. Online shopping sites keep record of your purchases to offer you promotions, improve your shopping experience, etc. On the one hand, users of these (and many other) platforms find these enhanced services handy as they make their lifes easier and many times (seemingly) cheaper. On the other hand, service providers and third party services which build on them, find them very useful for their business cases.

Bringing Compatiblity, Utility and Privacy together. Applying privacy by design is, by itself, a hard task that requires advanced knowledge of cryptographic primitives and information security tools. If, additionally, compatibility and utility have to be taken into account, the task can be daunting. As we have mentioned in Section 2, academic proposals usually lean towards the privacy side of this tradeoff in detriment of compatibility and utility. On the other hand, industry systems favor compatibility and utility, and typically cover the privacy counterpart through legal means (terms of service, etc.) which are barely guarantees against bad business practices [62] or security incidents.

In this work, we put forward our approach to reach a balance between these properties. We follow a methodology featuring three steps, which we referred to as *utility*, *privacy* and *utility again* in the introduction:

1. *(Utility)* First, we model the entities and processes of the context under analysis. Specifically, we differentiate the main process from added value processes. Additionally, we identify the privacy threats that we need to address.
2. *(Privacy)* Second, while keeping the same entities and main process, we apply privacy enhancing tools. This allows us to create a system that maintains the same topology than the original one, at the cost of losing added value processes.
3. *(Utility again)* Third, we extend the system obtained in the previous step, reincorporating the lost functionality by building additional cryptographic primitives on top of the ones used in the core process.

As a result of the previous steps, we end up with a system that adheres to the established compatibility restrictions, since we impose from the beginning the need to maintain entities and processes. It also maintains similar utility, due to the expansion made in the third step. Also, privacy is integrated from the first stage of the methodology, allowing us to reach high levels of privacy.

4 Preliminaries

Notation. For an algorithm A, we let $A(x_1, \dots, x_n; r)$ denote the output of A on inputs $x_1, \dots x_n$ and random coins r; in addition, $y \leftarrow A(x_1, \dots, x_n)$ means choosing r uniformly at random and setting $y \leftarrow A(x_1, \dots x_n; r)$. For a set S, we let $x \leftarrow S$ denote choosing x uniformly at random from S. We let $\langle O_A, O_B \rangle \leftarrow P(I_C)[A(I_A), B(I_B)]$ denote a two-party process P between parties A and B, where O_A (resp. O_B) is the output to party A (resp. B), I_C is the common input, and I_A (resp. I_B) is A's (resp. B's) private input; when party B does not have output, we sometimes write $O_A \leftarrow P(I_C)[A(I_A), B(I_B)]$. When a single party algorithm P uses a public key pk, we sometimes write $O \leftarrow P_{pk}(I)$ (although we may omit it when it is clear from the context). For readability, we assume that if any internal step fails, the overall process also fails and stops.

Basic cryptographic primitives. We assume readers are familiar with public-key encryption [31,61], digital signature and commitment schemes [15], and zero-knowledge proofs of knowledge (ZK-PoKs) [36]. Let $(\mathtt{EGen}, \mathtt{Enc}, \mathtt{Dec})$ denote a public-key encryption scheme, and $(\mathtt{SGen}, \mathtt{Sign}, \mathtt{SVer})$ denote a digital signature scheme. For readability, we assume that it is possible to extract the signed message from the corresponding signature. We let $\mathsf{com}_m \leftarrow \mathsf{Com}(m; r_m)$ denote a commitment to a message m, where the sender uses uniform random coins r_m; the sender can open the commitment by sending (m, r_m) to the receiver. We use $\pi \leftarrow \texttt{ProveZK}_L(x; w)$ and $\texttt{VerifyZK}_L(x, \pi)$ to refer to creating non-interactive proof π showing that the statement x is in language L (which will sometimes omit if obvious from the context) with the witness w, and to verifying the statement x based on the proof π.

Group signatures and traceable signatures. Group signatures [22,17,41,44,43] provide anonymity. In particular, a public key is set up with respect to a specific group consisting of multiple members. Any member of the group can create a signature ϱ, but signature ϱ reveals no more information about the signer than the fact that some member of the group created ϱ. Group signatures also provide accountability; the group manager (GM) can open signature ϱ and identify the actual signer.

- $(pk_G, sk_G) \leftarrow \texttt{GS.Setup}(1^k)$ sets up a key pair; GM holds sk_G.
- $\langle mk_i, \ell' \rangle \leftarrow \texttt{GS.Join}(pk_G)[M(s_i), GM(\ell, sk_G)]$ allows member M with secret s_i to join group G, generating the private member key mk_i and updating the Group Membership List ℓ to ℓ'.
- $\varrho \leftarrow \texttt{GS.Sign}_{mk_i}(msg)$ issues a group signature ϱ.
- $\texttt{GS.Ver}_{pk_G}(\varrho, msg)$ verifies whether ϱ is a valid group signature.
- $i \leftarrow \texttt{GS.Open}_{pk_G}(sk_G, \varrho)$ returns the identity i having issued the signature ϱ.
- $\pi \leftarrow \texttt{GS.Claim}_{mk_i}(\varrho)$ creates a claim π of the ownership of ϱ.
- $\texttt{GS.ClaimVer}_{pk_G}(\pi, \varrho)$ verifies if π is a valid claim over ϱ.

Traceable signatures [41] are essentially group signatures with additional support of tracing (when we use the previous group signature operations, but with a traceable signature scheme, we use the prefix TS instead of GS).

- $t_i \leftarrow \texttt{TS.Reveal}_{sk_G}(i)$. The GM outputs the tracing trapdoor of identity i.
- $b \leftarrow \texttt{TS.Trace}(t_i, \varrho)$. Given the tracing trapdoor t_i, this algorithm checks if ϱ is issued by the identity i and outputs a boolean value b reflecting the check.

Partially blind signatures. A blind signature scheme [21] allows a user U to have a signer S blindly sign the user's message m. Partially blind signatures [1] are a refinement of blind signatures in which, besides the blinded message m, a common public message is included in the final signature.

- $(pk_S, sk_S) \leftarrow \texttt{PBS.KeyGen}(1^k)$ sets up a key pair.
- $(\tilde{m}, \pi) \leftarrow \texttt{PBS.Blind}_{pk_S}(m, r)$. Run by a user U, it blinds the message m using a secret value r. It produces the blinded message $\tilde{m}$ and a correctness proof π of $\tilde{m}$.
- $\tilde{\varrho} \leftarrow \texttt{PBS.Sign}_{sk_S}(cm, \tilde{m}, \pi)$. Signer S verifies proof π and issues a partially blind signature $\tilde{\varrho}$ on $(cm, \tilde{m})$, where cm is the common message.
- $\varrho \leftarrow \texttt{PBS.Unblind}_{pk_S}(\tilde{\varrho}, \tilde{m}, r)$. Run by the user U, who verifies $\tilde{\varrho}$ and then uses the secret value r to produce a final partially blind signature ϱ.
- $\texttt{PBS.Ver}_{pk_S}(\varrho, cm, m)$ checks if ϱ is valid.

5 The reference e-shopping process

Following [30], we first identify the core functionalities of the existing e-shopping system as follows, ruling out other various features in order to achieve a very high level of privacy.

Entities. The involved parties are:

Customers (`C`*)* Individuals buying goods or services.
Merchants (`M`*)* Selling products and offering services to customers.
Payment System (`PS`*)* Platforms connecting customers and merchants, offering both added value services, e.g., marketing programs to customers, and fraud prevention mechanisms to merchants. For instance, Amazon, eBay or Alibaba.
Financial Network (`FN`*)* In our abstraction, we bundle all financial entities processing and executing transactions.
Delivery Companies (`DC`*)* Responsible for deliverying physical goods.

Note that this abstraction, although may combine several “sub-roles” within the same entity (e.g., card issuers and banks within `FN`), still follows the separation of roles of the real world. This abstraction is nevertheless more specific than the one in [30], where role of the `PS` was assumed by `M`. Given the high complexity of e-shopping platforms such as Amazon, it makes sense to make the distinction herein proposed.

Main processes. Assuming users have already registered in the system, we may consider four phases: purchase, checkout, delivery and completion (see Fig. 1). First, in the *purchase phase*, `C` picks the products he wants to buy from `M`. In the *checkout phase*, the payment and delivery information specified by `C` are routed to `PS`, probably through `M`, and processed and executed by `FN`. Subsequently, in the *delivery phase*, and for physical goods, `DC` delivers them to `C`. Finally, in the *completion phase*, `C` verifies that everything is correct, maybe initiating a complaint and/or leaving feedback.

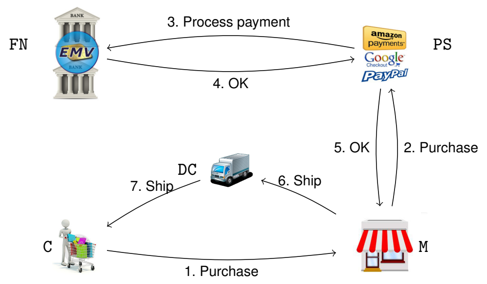


Fig. 1: The overall process of a traditional e-shopping.

The aforementioned processes are the ones that allow to fulfil the final objective of e-shopping: delivering (or providing) the customer the purchased good (or service). As we will se next, and was described in [30], there are already many privacy threats in these processes. Still, we can identify added value processes that help building a richer ecosystem. In this work, we take into account marketing tools and fraud prevention tools. These are indeed services that, while not being among the strictly necessary processes, have become essential part of the overall experience and help increase revenue (marketing tools) and reduce losses (fraud prevention). Specifically, marketing tools such as coupons are typically made available to customers since the purchase phase, either by `PS` or `M`. As for fraud prevention techniques, they are applied by `M`, `PS` and `FN` during checkout.

5.1 Identifying the threats

Once we have modelled the system and obtained its core entities and processes, the first step is to identify the main privacy risks that may appear.

For this purpose, we next review existing privacy threats in e-shopping. Table 1 summarizes the threats that have been exposed to some extent in the literature for each of the previously mentioned phases.

We note that some of them occur quite naturally in current industry systems (like threat 2.2, since M usually learns C's payment information, or threat 3.2, since DC learns both C and M addresses, being able to link C and M). However, as we will see in some of the reviewed attacks, sometimes it is enough with few additional information in order for an attacker to seriously undermine privacy. Therefore, it is advisable to keep the principle of *least information*.

Phase	Privacy threats
Purchase	1.1. Product info leaked to financial entities or 3rd parties 1.2. Link customers and merchants by 3rd parties
Checkout	2.1. Product info leaked to financial entities or 3rd parties 2.2. Payment info leaked to merchants or 3rd parties 2.3. Link customers and merchants by 3rd parties
Delivery	3.1. Shipping address leaked to merchants or 3rd parties 3.2. Link customers and merchants by delivery companies or 3rd parties
Completion	4.1. Private info leaks through feedback (All previous threats may affect completion)

Table 1: Summary from of privacy threats in the different e-shopping phases.

Threats in the purchase stage. We first note that 9 out of 13 risks outlined in [8] affect the purchase process and the communications between C and M.

Additional vulnerabilities may lie in loyalty programs; M can apply loyalty programs to lure more customers into buying products. The promotions offered by M will probably be based on C's profile, purchase history, and shopping cart. Since M will likely apply loyalty programs to increase their revenue, it is necessary to analyze how customers' personal information is treated. For this purpose, e-shopping platforms (Amazon, eBay, etc.) typically collect information such as purchase history, IP addresses, browser metadata, etc.

In [56], threat 1.1 in Table 1 is exposed for e-shops using PayPal. In this study, it was observed that 52% of the analyzed e-shops where sending product names, number of items and descriptions to PayPal. In addition, [56] also showed that PayPal leaked tracking information to Adobe's Omniture, including the referrer URL, which directly allows to link C and M (realizing threat 1.2 in Table 1). Moreover, note that in conventional e-shopping, risk 1.2 is always present, since PS, FN and DC usually learn both C and M identities [8].

Threats in the checkout stage. In this stage, C specifies the payment information and shipping address. After applying fraud prevention techniques (e.g., reject purchases of more than a predefined price), M checks the promotions presented by C, if any, and forwards the payment information to FN. After validating the payment information (along with additional fraud prevention mechanisms), FN executes the payment. When checkout is completed, M updates C's profile.

This stage handles most pieces of information; risks 1 to 6 and risk 13 of [8] directly affect this stage. Namely, either M, PS or FN may misuse C's personal or payment information. Even if it is not misused by a dishonest entity, a *honest but curious* party may still pose a serious threat.

Concerning threat 2.1 in Table 1, as pointed out in [48], the current widely deployed 3-D Secure protocol, e.g., "Verified by Visa", "MasterCard SecuriCode", or "American Express SafeKey", requires a description of the transaction to be sent to FN (more exactly, the card issuer) in order for the cardholder to see and check it later. In particular, we know that some merchants leak product information to FN [56]. As to threat 2.2, in the protocol "Verified by Visa", M receives C's PAN (Primary Account Number, i.e., the credit card number) [68].

A relevant example of threat 2.3 in Table 1, which may also imply threats 2.1 and 2.2, appears in [28]. From a large set of simply anonymized financial data (without names, addresses or obvious identifiers), [28] shows that it is possible to de-anonymize 90% of the individuals, if the data contain three items: price, when, and where.

Moreover, the payment information processed by financial entities includes card and account numbers, identifiers that persist across online and offline platforms and systems (unlike, e.g., cookies). This further implies that financial entities possess very sensitive information that paves the way to link purchases with payment transactions and perform behavioral analysis over customers' data [56]. Finally, fraud prevention is very relevant in the payment phase. It is the main mechanism that merchants and financial entities employ to prevent losses, which are far from negligible [3,4]. However, as pointed out in [3] new trends in these fraud prevention may pose a serious threat to privacy, like incorporating geolocation from mobile phones or information from social networks.

Threats in the delivery stage. Once M or PS receive the payment, it delivers the purchased goods to C. For digital goods, the files are sent via Internet, and using anonymizing networks [32] is a robust way to protect privacy. For physical goods, these will be shipped through some delivery company DC to the shipping address specified by C to M during checkout (thus, realizing threat 3.1 in Table 1). Also, as pointed out in [8], depending on the information available to DC, it may pose additional privacy threats. In the real world, the delivery company DC at least learns both C's and M's addresses (threat 3.2 in Table 1), which allows it to link them, and may also learn other data, such as product related information.

However, preventing M (or other entities) from learning C's physical address and DC to learn both C's and M's addresses is costly. Probably, physical mix networks are the most privacy respectful option [6]. Alternatively, Post Office boxes or equivalent delivery methods offer an intermediate solution between complexity and privacy, as it reveals a nearby location instead of C's address.

Threats in the completion stage. After receiving the purchased items, C verifies that everything is correct, checking the debited amount, the received items, etc. If C is satisfied, the purchase is completed. If some error is detected, C may initiate a complaint.

The situation is more complicated for purchases through e-shopping platforms (e.g., Amazon) rather than directly with the merchant; in this case, although it is usually recommended to first contact the merchant[9], it may be necessary for the e-shopping platform to mediate. In these situations, the privacy risks described for the previous stages will also be present, since `C` may need to provide product or payment information, or her contact information.

Additionally, whichever the final result is, `C` may provide online feedback about `M`, for other customers to decide whether or not to buy from him; in some platforms, such as eBay, `M` may also evaluate `C`. Concerning the possibility of leaving feedback, [47] shows how insufficient privacy controls may lead to serious privacy threats. Indeed, it is possible to infer the purchase history of a specific user by correlating the feedback she has received with the feedback received by the sellers with whom she has interacted. Also, it is possible to perform a *category attack* to obtain a list of the people that has bought an item of a specific type (e.g. guns). Other attacks explained in [47] include a *broad profiling attack* and a *side-information attack*, which also pose a serious threat to buyers (even enabling third parties to compromise their privacy in the case of the side-information attack). In a related context, [51] explains how to identify users in the Netflix database from little external information. All these attacks are realizations of threat 4.1 in Table 1. A more general model of the privacy threats related to recommendation systems is described in [57,53].

5.2 Starting point

To summarize, in this section we have modeled current industry e-shopping systems:

- Entities: customers, merchants, payment systems, financial entities and delivery companies.
- Main processes: purchase, checkout, delivery, completion.
- Added-value processes: marketing and fraud prevention tools.

We have established the information flow between entities that allows to implement the aforementioned processes. Additionally, we have identified privacy threats in each of these processes. Thus, we begin from this reference model, and proceed to address the identified privacy issues.

6 System with a High Level of Privacy and Less Functionalites

Following our methodology, we now add privacy enhancing mechanisms (*privacy* step), constraining the system's functionality to the main processes in order to achieve a high level of privacy, but otherwise minimizing the modification of core functionalities. In particular, we change neither the participating entities nor the actual information flow over the full e-shopping process. However, this comes at the cost of losing the added value processes (marketing tools and fraud prevention tools). We start by informally stating our privacy objectives.

[9] See `https://payments.amazon.com/help/5968`. Last access on April 18th, 2018.

6.1 Privacy goal

We assume that merchants can act maliciously, but PS, FN and DC are semi-honest. Informally, we aim at achieving customer privacy satisfying the following properties:

- *Hide the* identity of a customer *and reveal it only if necessary:* The identity of a customer is sometimes sensitive information, and we want to hide it from other parties as much as possible. In the overall e-shopping process, observe that parties such as merchants, PS, and DC don't really need the identity of the customer in order for the transaction to go through. However, FN must know the identity to withdraw the actual amount of money from the customer's account and to comply with current regulations.
  This addresses threats 1.2, 2.3 and 3.2 from Table 1.
- *Hide the* payment information *and reveal it only if necessary:* The information about the credit card number (or other auxiliary payment information) that a customer uses during the transaction is quite sensitive and thereby needs to be protected. In the overall e-shopping process, like the case of the customer identity, observe that only FN must know this information to complete the financial transaction.
  This addresses threat 2.2 from Table 1.
- *Hide the* product information *and reveal it only if necessary:* The information about which product a customer buys can also be sensitive. However, note that PS and FN don't really need to know what the customer is buying in order for the transaction to go through, but the merchants and DC must handle the actual product.
  This addresses threat 1.1 and 2.1 from Table 1.

6.2 Approach for privacy-enhancements

Below, we highlight our approaches to achieve our privacy goal.

Controlling the information of customer identity. We use the following privacy-enhancing mechanisms to control the information of customer identity.

- *Sender anonymous channel from customers*: Customers use sender-anonymous channels such as Tor [32] for their communications. Thus, parties interacting with a customer won't be able to know from where the customer contacts them.
- *Customer group signatures on transaction data:* The transaction data on the customer side is authenticated by the customer's group signature. In our context, FN takes the role of the group manager, issuing member keys. Thus, if a merchant M verifies the group signature included by a customer in a transaction, M is confident that the customer has an account with FN. Moreover, the identity of the customer is hidden from other parties based on security of group signatures. However, since FN takes the role of the group manager, it can identify the customer by opening the signature and complete the account management task. However, FN is otherwise not required to take any active role with respect to managing the group or processing group signatures. Note that the group manager must be a trusted entity concerning the group management tasks, although this trust can be reduced with threshold techniques like those in [11].

Controlling the payment information. Customers encrypt their payment information with $\mathtt{FN}$'s public key. Thus, only $\mathtt{FN}$ can check if the identity in the payment information matches the one extracted from the customer's group signature.

Controlling the product information. The customer encrypts the information about the product he wants to purchase using a key-private public key encryption scheme (e.g., ElGamal encryption) [9]; he generates a key pair and uses the public key to encrypt the product information. The key pair can be used repeatedly since the scheme is key-private[10], and the public encryption key is never sent to other parties. The main purpose of doing this is for logging. Once $\mathtt{FN}$ logs the transactions, the customer can check the product information in each transaction by simply decrypting the related ciphertext.

Obviously, the encryption doesn't reveal any information about the product to other parties. Of course, the merchants should obtain this information to proceed with the transaction. To handle this, the customer sends the product information in both plaintext and ciphertext forms, and then proves consistency using a ZK proof. When this step is cleared, only the ciphertext part is transferred to other entities.

Note that this system satisfies all our privacy goals. However, it reduces utility, as is not compatible with many features required by the industry (or by regulation).

6.3 System processes

Following the general e-shopping scenario described in Section 6, our privacy enhanced system (but without utility) consists of Customers $\mathtt{C}_i$, Merchants $\mathtt{M}_j$, the Payment System $\mathtt{PS}$, the Financial Network $\mathtt{FN}$ and delivery companies $\mathtt{DC}$. The processes for communicating them are described next and depicted in Fig. 2.

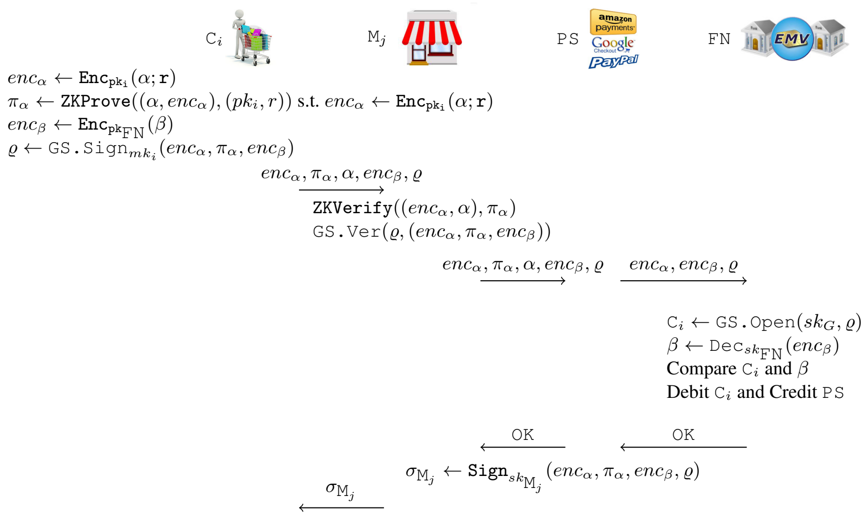
Setup. All customers belong to a group G which, for simplicity, we assume to be managed by $\mathtt{FN}$, who initially sets up this group G so that its members may issue group signatures. $\mathtt{FN}$ also generates a public key encryption key pair $(pk_{\mathtt{FN}}, sk_{\mathtt{FN}})$. $\mathtt{PS}$ and all $\mathtt{M}_j$ generate key pairs ($pk_{\mathtt{PS}}, sk_{\mathtt{PS}}$ and $pk_{\mathtt{M}_j}, sk_{\mathtt{M}_j}$, respectively). Finally, all companies (and additional entities) in $\mathtt{DC}$ are set up as in [5].

Purchase. In the purchase phase, $\mathtt{C}_i$ browses $\mathtt{M}_j$'s web, getting a description of the products he wants to buy. We use α to denote the product information.

Checkout. In the checkout stage, $\mathtt{C}_i$ initiates the transaction with $\mathtt{M}_j$ by providing the necessary information. The process ends when $\mathtt{C}_i$ receives a confirmation of the order (with a suitable receipt). The process is as follows:

1. $\mathtt{C}_i$ creates encrypted versions of the messages to submit. The product information α is encrypted with his own public key. Here, we use the public key encryption scheme (e.g., ElGamal encryption) as a private-key encryption scheme, in order to take advantage of the algebraic structure of the scheme admitting simple ZK proofs. That is, he runs $\mathsf{enc}_\alpha \leftarrow \mathtt{Enc}_{pk_i}(\alpha; r)$, and creates a ZK proof of consistency $\pi_\alpha \leftarrow \mathtt{ProveZK}(x; w)$, where $x = (\alpha, \mathsf{enc}_\alpha)$ and $w = (pk_i, r)$ such that

[10] Key-privacy security requires that an eavesdropper in possession of a ciphertext not be able to tell which specific key, out of a set of known public keys, is the one under which the ciphertext was created, meaning the receiver is anonymous from the point of view of the adversary.



$\mathtt{C}_i$
$\mathtt{M}_j$
PS
FN
$enc_\alpha \leftarrow \mathtt{Enc}_{\mathtt{pk_i}}(\alpha;\mathtt{r})$
$\pi_\alpha \leftarrow \mathtt{ZKProve}((\alpha, enc_\alpha),(pk_i,r))$ s.t. $enc_\alpha \leftarrow \mathtt{Enc}_{\mathtt{pk_i}}(\alpha;\mathtt{r})$
$enc_\beta \leftarrow \mathtt{Enc}_{\mathtt{pk_{FN}}}(\beta)$
$\varrho \leftarrow \mathtt{GS.Sign}_{mk_i}(enc_\alpha, \pi_\alpha, enc_\beta)$
$enc_\alpha, \pi_\alpha, \alpha, enc_\beta, \varrho$
$\mathtt{ZKVerify}((enc_\alpha, \alpha), \pi_\alpha)$
$\mathtt{GS.Ver}(\varrho, (enc_\alpha, \pi_\alpha, enc_\beta))$
$enc_\alpha, \pi_\alpha, \alpha, enc_\beta, \varrho$
$enc_\alpha, enc_\beta, \varrho$
$\mathtt{C}_i \leftarrow \mathtt{GS.Open}(sk_G, \varrho)$
$\beta \leftarrow \mathtt{Dec}_{sk_{\mathtt{FN}}}(enc_\beta)$
Compare $\mathtt{C}_i$ and β
Debit $\mathtt{C}_i$ and Credit PS
OK
OK
$\sigma_{\mathtt{M}_j} \leftarrow \mathtt{Sign}_{sk_{\mathtt{M}_j}}(enc_\alpha, \pi_\alpha, enc_\beta, \varrho)$
$\sigma_{\mathtt{M}_j}$

Fig. 2: The overall process of the system. Here, α and β are the product and purchase information respectively. α has been obtained previously by $\mathtt{C}_i$, browsing $\mathtt{M}_j$'s web anonymously.

$\mathtt{enc}_\alpha = \mathtt{Enc}_{pk_i}(\alpha; r)$. In addition, he fills up all the payment information β (i.e., the sensitive details such as a credit card number and the security code) and encrypts it with $\mathtt{FN}$'s public key, producing $\mathtt{enc}_\beta$. Next, $\mathtt{C}_i$ issues a group signature over all the previous tokens. That is, $\varrho \leftarrow \mathtt{GS.Sign}_{mk_i}(\mathtt{enc}_\alpha, \pi_\alpha, \mathtt{enc}_\beta)$. $\mathtt{C}_i$ sends the previous ciphertexts, group signature, and α to $\mathtt{M}_j$.

2. $\mathtt{M}_j$ (and possibly $\mathtt{PS}$ afterwards) verify ϱ and $\mathtt{enc}_\alpha$. If it is correct, then $\mathtt{PS}$ submits $\mathtt{enc}_\beta$, $\mathtt{enc}_\alpha$ and ϱ to $\mathtt{FN}$.
3. Upon receiving the previous tokens, $\mathtt{FN}$ decrypts $\mathtt{enc}_\beta$ (which was encrypted with its public key) and opens ϱ. If the identity of the customer who issued ϱ matches the identity in β, and the customer has enough funds, the payment is accepted and processed, informing $\mathtt{PS}$ of the result. $\mathtt{FN}$ also uses $\mathtt{enc}_\alpha$ as transaction description.
4. $\mathtt{PS}$ will in turn inform $\mathtt{M}_j$ of the output of the transaction and, if accepted, $\mathtt{M}_j$ issues a conventional digital signature of $\mathtt{enc}_\alpha$, π_α, $\mathtt{enc}_\beta$ and ϱ. That is, it runs $\sigma_{M_j} \leftarrow \mathtt{Sign}_{sk_{M_j}}(\mathtt{enc}_\alpha, \pi_\alpha, \mathtt{enc}_\beta, \varrho)$.
5. Finally, $\mathtt{C}_i$ receives and verifies σ_{M_j}. If correct, the transaction succeeds.

Delivery. Upon receiving the payment, $\mathtt{M}_j$ initiates the delivery. This can be done in a privacy preserving way through the anonymous physical object delivery (APOD) system [5]. In APOD, $\mathtt{C}_i$ authenticates to $\mathtt{M}_j$ using a pseudonym, and obtains a credential that entitles him to ask $\mathtt{DC}$ to ship the goods. In our case, $\mathtt{C}_i$ can be authenticated to $\mathtt{M}_j$ by claiming ownership of the group signature ϱ that is included within $\sigma_{\mathtt{M}_j}$ (using $\mathtt{GS.Claim}$), and afterwards proceed as in APOD.

Note that this approach for delivery addresses threats 3.1 and 3.2 from Table 1.

Completion. In order to perform additional actions related to some previous transaction, $\mathtt{C}_i$ has to prove having executed the transaction. For this purpose, the client runs $\mathtt{GS.Claim}$ over the group signature ϱ included in $\sigma_{\mathtt{M}_j}$. This entitles him to provide feedback over the received products, initiate a complaint and/or ask for a refund.

Note that, throughout all the processes, $\mathtt{C}_i$'s identity is never learned by either $\mathtt{M}_j$ nor $\mathtt{PS}$. Moreover, $\mathtt{FN}$ learns $\mathtt{C}_i$'s identity but does not learn $\mathtt{M}_j$'s identity and does not receive information about the products being bought. Furthermore, $\mathtt{C}_i$ receives a digital signature issued by $\mathtt{M}_j$ that he can use for initiating delivery, requesting refunds and/or leaving feedback in a privacy preserving fashion.

Finally, note that with this, we also cover threat 4.1 from Table 1.

6.4 How to proceed to the next step

In this step, we have built a system that provides the main functionality, i.e., raw purchase, checkout, delivery and completion phases; while at the same time provides a high privacy level. Still, we have eliminated important functionalities (marketing tools and fraud prevention tools). Neverthless, we have set the foundation for incoporating them back. Specifically, the introduction of group signatures and the underlying member keys will allow us to issue zero-knowledge proofs tailored to recover the lost functionality.

7 Privacy-enhanced System with Richer Functionality

Next, we add important functionalities, in particular marketing and antifraud mechanisms, to the system described in Section 6, carefully relaxing privacy (*utility again*). In particular, each customer is associated with a pseudonym by default, and fraud prevention and marketing tools are applied by aggregating certain pieces of transaction history based on the pseudonym. Moreover, we allow customers we allow customers to opt for anonymity in each transaction, which ensures that *privacy is not reduced beyond what this aggregation implies*.

Adding marketing tools: utility vs privacy. We would like the payment system $\mathtt{PS}$ (or merchants) to use marketing tools (e.g., coupons) so as to incentivize customers to purchase more products and thereby increase their revenue. For clarity of exposition, we will consider adding a feature of coupons and discuss the consequential privacy loss; other marketing features essentially follow the same framework.

When we try to add this feature to the system, $\mathtt{PS}$ *must at least have access to the amount of money each customer has spent so far*; otherwise, it's impossible for the coupons to be issued for more loyal customers. Obviously, revealing this information is a privacy loss. However, this trade-off between utility and privacy seems to be unavoidable, if the system is to be practically efficient, ruling out the use of fully-homomorphic encryptions [35] or functional encryptions [13], which are potentially promising but, as of now, prohibitively expensive to address our problem. The main question is as follows:

– Can we make the system reveal *nothing more than* the purchase history of encrypted products?

– Can we provide the customers with an option to *control the leakage of this history*? In other words, can we give the customers an option to *exclude some or all of their purchase activities* from the history?

We address both of the above questions affirmatively. In order to do so, we first allow *each customer to use a pseudonym selectively*. That is, the payment system can aggregate the customer's purchase history of encrypted products only if the customer uses his pseudonym when buying a product. If the customer wants to exclude some purchase activity from this history, he can proceed with the transaction anonymously.

Still, there are a couple of issues to be addressed. First, we would like the system to work in *a single work flow* whether a customer chooses to go *pseudonymously or anonymously*. More importantly, we want a customer to be able to *use coupons even if he buys a product anonymously*. We will show below how we address these issues, when we introduce the notion of a checkout-credential.

Adding antifraud mechanisms: utility vs privacy. Merchants need to be protected against fraudulent or risky transactions, e.g. transactions that are likely to end up in non-payments, or that are probably the result of stolen credit cards and similar cases. In current industry systems, this is typically done by having the `PS` send a risk estimation value to merchants (obtained using proprietary algorithms), who can also apply their own *filters* based on the specifics of the transaction (number of items, price, etc.). We would like to adopt this feature. Obviously, we have an utility-privacy trade-off. In particular, if the risk estimation is too specific and identifying, it will hinder the system from supporting anonymous transactions. We believe that this trade-off is inherent, and in this paper, we treat the specificity of risk estimation to be given as an appropriately-chosen system parameter, depending on the volume of the overall transactions and only mildly degrading the quality of anonymity in anonymous transactions. The main question we should ask will be as follows:

Can we relax anonymity of transactions but only to reveal *the risk estimation*?

As with the marketing tools, we use the checkout-credential for implementing this.

7.1 Our approach

Checkout credentials. Recall that we would like our system to allow customers to perform unlinkable (anonymous) purchases while we also need to provide merchants with the fraud estimation of a transaction based on each customer's previous transactions. This goal is achieved in a privacy-respectul manner through the checkout-credential retrieval process.

The checkout-credential retrieval process is carried out before the actual checkout, and it is executed between `PS` and the customer. The resulting checkout-credential is the means used by `PS` to aggregate the available information related to each pseudonym and provide the marketing and antifraud information for merchants without violating each customer's privacy.

Fig. 3 shows the augmented information flow of the *purchase* and *checkout* phases in our system. Delivery and completion are not depicted in Fig. 3 since, as we show

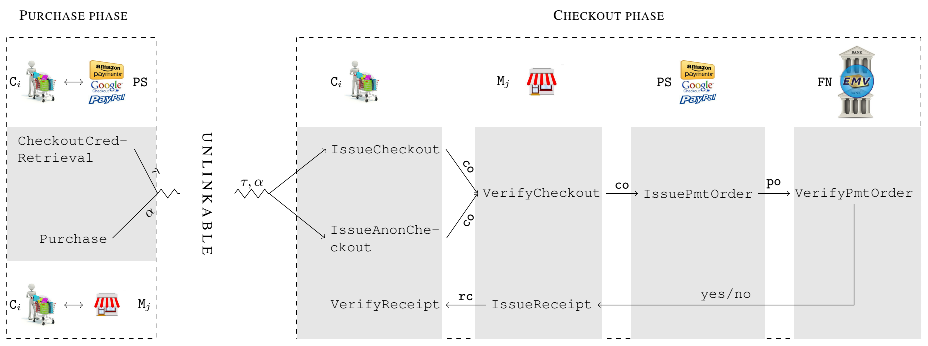


Fig. 3: System process flow. Here, τ is the checkout-credential and α is the product information.

in the following description, they are quite straightforward and do not suffer further modifications (with respect to the system in Section 6) besides integrating them with the new *purchase* and *checkout* processes. Specifically, note that while we have partitioned the main processes in multiple sub-processes, the overall flow is still the same. That is, *purchase* $\rightarrow$ *checkout* $\rightarrow$ *delivery* $\rightarrow$ *completion*. Finally, note also that the parties involved in each process are maintained compared to current systems.

Basically, a checkout-credential is a partially blind signature, requested by a customer and issued by PS. The common message of the checkout-credential includes *aggregated data related to fraud and marketing*, and the blinded message is a *commitment to the customer key*. During the actual checkout, a customer proves to merchants in ZK that he knows the committed key embedded in the checkout credential. *Since it was blindly signed,* PS *and merchants cannot establish a link* any further than what the aggregated common information allows.

At this point, when the customer decides to perform a pseudonymous checkout (in this case, the pseudonym is also shown during checkout), PS will be able to link the current checkout to the previous ones and update the customer's history (updating his eligibility to promotions and risk estimation). If he chooses an anonymous checkout, PS will not be able to link this transaction with others.

Protection against fraudulent anonymous transactions. There is an additional issue. An attacker may execute a large volume of pseudonymous transactions honestly, making its pseudonym have a low risk-estimate value, and then perform a fraudulent anonymous transaction. Note in this case, the checkout-credential will contain low risk estimate and the transaction will likely go through, but problematically, *because of unlinkability of this fraudulent transaction,* PS *cannot reflect this fraud into the pseudonym's transaction history*. Moreover, taking advantage of this, the attacker can repeatedly perform fraudulent anonymous transactions with low risk estimate.

Recall that in our previous highly-private system, the checkout is conducted using a token that contains a group signature issued by a customer and which can be opened by FN. In this system, we use traceable signatures, which augment group signature with an additional tracing feature. In particular, if an anonymous transaction proves to be fraudulent a posteriori, FN can open the signature and give PS the tracing trapdoor associated

```
$\texttt{FNSetup}(1^k)$ :
  $(pk_G, sk_G) \leftarrow \texttt{TS.Setup}(1^k)$
  $(pk_{\texttt{FN}}, sk_{\texttt{FN}}) \leftarrow \texttt{EGen}(1^k)$
  $\mathrm{PK}_{\texttt{FN}} \leftarrow (pk_{\texttt{FN}}, pk_G)$
  $\mathrm{SK}_{\texttt{FN}} \leftarrow (sk_{\texttt{FN}}, sk_G)$

$\texttt{PSSetup}(1^k)$ :
  $(pk_{\texttt{PS}}, sk_{\texttt{PS}}) \leftarrow \texttt{SGen}(1^k)$
  $(pk_{\texttt{PBS}}, sk_{\texttt{PBS}}) \leftarrow \texttt{PBS.KeyGen}(1^k)$
  $\mathrm{PK}_{\texttt{PS}} \leftarrow (pk_{\texttt{PS}}, pk_{\texttt{PBS}})$
  $\mathrm{SK}_{\texttt{PS}} \leftarrow (sk_{\texttt{PS}}, sk_{\texttt{PBS}})$

$\texttt{MSetup}(1^k)$ :
  $(pk_{\texttt{M}_j}, sk_{\texttt{M}_j}) \leftarrow \texttt{SGen}(1^k)$
  $\mathrm{PK}_{\texttt{M}_j} \leftarrow pk_{\texttt{M}_j}; \mathrm{SK}_{M_j} \leftarrow sk_{\texttt{M}_j}$

$\texttt{CSetup}(pk_G)[\texttt{C}_i(s_i), \texttt{FN}(sk_G, \ell)]$ :
  $\langle mk_i, \ell' \rangle \leftarrow \texttt{TS.Join}(pk_G)[\texttt{C}_i(s_i), \texttt{FN}(\ell, sk_G)]$
  $(pk_i, sk_i) \leftarrow \texttt{EGen}(1^k)$
  $\texttt{C}_i$ chooses $r \leftarrow \{0,1\}^*$
  $\texttt{C}_i$ computes $\varrho \leftarrow \texttt{TS.Sign}_{mk_i}(r; r_{P_i})$
  $\texttt{C}_i$ sends $P_i = (r, \varrho)$ to $\texttt{FN}$
  $\mathrm{SK}_{\texttt{C}_i} \leftarrow (P_i, mk_i, r_{P_i}, pk_i, sk_i)$
```

Fig. 4: Full system setup processes.

with the token (i.e., the traceable signature). Given this trapdoor, PS can update the risk estimation even for anonymous checkouts.

Note that customers are offered a trade-off. When customers always checkout anonymously, they have no previous record and receive worse promotions and fraud estimates. When they always checkout pseudonymously, they get better offers and probably better fraud estimates, in exchange of low privacy. But there are also intermediate options. In all cases, they can take advantage of any coupons they are eligible for and receive fraud estimates based on previous pseudonymous purchases.

In any case, our system is compatible with many antifraud techniques in the industry without needing to resort to tracing and also applicable with anonymous checkouts (some are discussed in Section 10).

7.2 System description

In this section, we describe our system. The processes composing each phase are defined next. The flow for purchase and checkout is depicted in Fig. 3.

Setup. FN, PS, and every merchant $\texttt{M}_j$ and customer $\texttt{C}_i$ run their corresponding setup processes in order to get their keys, according to the processes in Fig. 4. In particular, FN runs FNSetup to generate traceable signature and encryption keys. PS runs PSSetup to generate a key pair for partially blind signatures. $\texttt{M}_j$ runs MSetup to generate signing keys. $\texttt{C}_i$ and FN interact in order to generate key pairs for $\texttt{C}_i$, running CSetup. $\texttt{C}_i$ contacts FN, creates an account and joins a group G, obtaining a membership key mk_i using a secret s_i. In this case, $\texttt{C}_i$ also sets up a pseudonym P_i, known to FN. The pseudonym P_i is a traceable signature on a random message created using his membership key mk_i; we let $P_i.r$ denote the random message and $P_i.\varrho$ the traceable signature on $P_i.r$. During the process, FN updates its membership database ℓ into ℓ'.

Checkout-retrieval and purchase. The purchase phase includes the the processes of CheckoutCredRetrieval and Purchase. The purpose of this phase is for $\texttt{C}_i$ to obtain a description of the products to buy from $\texttt{M}_j$ and a credential authorizing him to proceed to checkout and including information necessary to apply marketing and antifraud tools.

During CheckoutCredRetrieval, $\texttt{C}_i$ interacts pseudonymously with PS. The protocol starts by having the customer $\texttt{C}_i$ send his pseudonym P_i. Then, PS retrieves the

$\langle \tau, (\top/\bot) \rangle \leftarrow \texttt{CheckoutCredRetrieval}(pk_G, pk_{\texttt{PBS}})[\texttt{C}_i(mk_i, P_i), \texttt{PS}(sk_{\texttt{PBS}})]$:

$\texttt{C}_i$ | $\texttt{PS}$

$\xrightarrow{P_i}$

Verify the signature $P_i.\varrho$ on $P_i.r$.
If verification fails, abort.
Fetch P_i's history to estimate $\texttt{rk}, \texttt{pr}$ such that:
$\texttt{rk}$: account risk for P_i
$\texttt{pr}$: promotions for P_i
Obtain $\texttt{dl}$: checkout-credential deadline

$\xleftarrow{(\texttt{rk}, \texttt{pr}, \texttt{dl})}$

$\mathsf{com} \leftarrow \mathsf{Com}(mk_i; r_{\mathsf{com}})$
$(\tilde{\mathsf{com}}, \pi) \leftarrow \texttt{PBS.Blind}_{pk_{\texttt{PBS}}}(\mathsf{com}; s)$
$\phi = \texttt{ProveZK}(x; w)$ where
$x = (\tilde{\mathsf{com}}, \pi, P_i, pk_G), w = (mk_i, r_{\mathsf{com}}, s, r_{P_i})$ such that :
$(\tilde{\mathsf{com}}, \pi) = \texttt{PBS.Blind}_{pk_{\texttt{PBS}}}(\mathsf{Com}(mk_i; r_{\mathsf{com}}); s)$,
$P_i.\varrho = \texttt{TS.Sign}_{mk_i}(P_i.r; r_{P_i})$.
$\texttt{pr}'$: customer's choice from $\texttt{pr}$

$\xrightarrow{(\texttt{pr}', \tilde{\mathsf{com}}, \pi, \phi)}$

$\texttt{VerifyZK}(\tilde{\mathsf{com}}, \pi, P_i, pk_G)$
If verification fails, abort.
$\tilde{\varrho} \leftarrow \texttt{PBS.Sign}_{sk_{\texttt{PBS}}}((\texttt{rk}, \texttt{pr}', \texttt{dl}), \tilde{\mathsf{com}}, \pi)$

$\xleftarrow{\tilde{\varrho}}$

$\varrho \leftarrow \texttt{PBS.Unblind}_{pk_{\texttt{PBS}}}(\tilde{\varrho}, \tilde{\mathsf{com}}, s)$
output $\tau = ((\texttt{rk}, \texttt{pr}', \texttt{dl}), \mathsf{com}, \varrho))$

Fig. 5: The CheckoutCredRetrieval process.

information of how loyal P_i is (i.e., $\texttt{rk}$), whether (and how) P_i is eligible for promotion (i.e., $\texttt{pr}$), and the deadline of the checkout-credential to be issued (i.e., $\texttt{dl}$), sending back $(\texttt{rk}, \texttt{pr}, \texttt{dl})$ to $\texttt{C}_i$. $\texttt{C}_i$ chooses a subset $\texttt{pr}'$ from the eligible promotions $\texttt{pr}$. Finally, $\texttt{C}_i$ will have PS create a partially blind signature such that its common message is $(\texttt{rk}, \texttt{pr}', \texttt{dl})$ and its blinded message is a commitment com to his membership key mk_i. We stress that the private member key mk_i of the customer $\texttt{C}_i$ links the pseudonym (i.e., $P_i.\varrho \leftarrow \texttt{TS.Sign}_{mk_i}(P_i.r)$) and the blinded message (i.e., $\mathsf{com} \leftarrow \mathsf{Com}(mk_i; r_{com})$). The customer is supposed to create a ZK-PoK ϕ showing this link. Upon successful execution, the checkout-credential is set to τ. We use $\tau.\texttt{rk}$, $\tau, \texttt{pr}$, $\tau.\texttt{dl}$, $\tau.\mathsf{com}$, $\tau.\varrho$ to denote the risk factor, promotion, deadline, commitment to the member key, and the resulting blind signature respectively. Refer to Fig. 5 for pictorial description. A checkout-credential issued with the process in Fig. 5 would be verified during checkout using the VerifyCheckoutCred process, defined as follows:

$$\texttt{VerifyCheckoutCred}_{\mathrm{PK}_{\texttt{PS}}}(\tau) : \textbf{return } \texttt{PBS.Ver}_{pk_{\texttt{PBS}}}(\tau.\varrho, (\tau.\texttt{pr}, \tau.\texttt{rk}, \tau.\texttt{dl}), \tau.\mathsf{com})$$

Concurrently, $\texttt{C}_i$ obtains through the Purchase process a product description of the items he wants to buy. Note that this can be done just by having $\texttt{C}_i$ browse $\texttt{M}_j$'s website using sender anonymous channels:

$$\alpha \leftarrow \texttt{Purchase}[\texttt{C}_i, \texttt{M}_j] : \textbf{return} \text{ product description from } \texttt{M}_j\text{'s website}$$

```
$\mathsf{co} \leftarrow \mathtt{IssueCheckout}(\mathrm{SK}_{\mathtt{C}_i}, \tau, \alpha, \$, \beta)$:
  Parse $\mathrm{SK}_{\mathtt{C}_i} = (P_i, mk_i, r_{P_i}, pk_i, sk_i)$
  $\mathsf{enc}_\alpha \leftarrow \mathsf{Enc}_{pk_i}(\alpha; r_\alpha)$
  $\mathsf{enc}_\beta \leftarrow \mathsf{Enc}_{pk_{\mathtt{FN}}}(\beta)$
  $\varrho \leftarrow \mathtt{TS.Sign}_{mk_i}((\$, \mathsf{enc}_\alpha, \mathsf{enc}_\beta); r_{gs})$
  $\psi \leftarrow \mathtt{ProveZK}(x, w)$ with
    $x = (P_i, \tau.\mathsf{com}, \$, \alpha, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho)$
    $w = (mk_i, r_{P_i}, r_\alpha, r_{\mathsf{com}}, r_{gs})$
    such that
      $P_i.\varrho = \mathtt{TS.Sign}_{mk_i}(P_i.r; r_{P_i})$
      $\mathsf{enc}_\alpha = \mathsf{Enc}_{pk_i}(\alpha; r_\alpha)$
      $\tau.\mathsf{com} = \mathsf{Com}(mk_i; r_{\mathsf{com}})$
      $\varrho = \mathtt{TS.Sign}_{mk_i}((\$, \mathsf{enc}_\alpha, \mathsf{enc}_\beta); r_{gs})$
  $\mathsf{co} \leftarrow (P_i, \tau, \$, \alpha, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho, \psi)$
  **return** $\mathsf{co}$

$\mathsf{co} \leftarrow \mathtt{IssueAnonCheckout}(\mathrm{SK}_{\mathtt{C}_i}, \tau, \alpha, \$, \beta)$:
  Parse $\mathrm{SK}_{\mathtt{C}_i} = (P_i, mk_i, r_{P_i}, pk_i, sk_i)$
  $\mathsf{enc}_\alpha \leftarrow \mathsf{Enc}_{pk_i}(\alpha; r_\alpha)$
  $\mathsf{enc}_\beta \leftarrow \mathsf{Enc}_{pk_{\mathtt{FN}}}(\beta)$
  $\varrho \leftarrow \mathtt{TS.Sign}_{mk_i}((\$, \mathsf{enc}_\alpha, \mathsf{enc}_\beta); r_{gs})$
  $\psi \leftarrow \mathtt{ProveZK}(x, w)$ with
    $x = (\tau.\mathsf{com}, \$, \alpha, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho)$
    $w = (mk_i, r_\alpha, r_{\mathsf{com}}, r_{gs})$
    such that
      $\mathsf{enc}_\alpha = \mathsf{Enc}_{pk_i}(\alpha; r_\alpha)$
      $\tau.\mathsf{com} = \mathsf{Com}(mk_i; r_{\mathsf{com}})$
      $\varrho = \mathtt{TS.Sign}_{mk_i}((\$, \mathsf{enc}_\alpha, \mathsf{enc}_\beta); r_{gs})$
  $\mathsf{co} \leftarrow (\tau, \$, \alpha, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho, \psi)$
  **return** $\mathsf{co}$

$\top/\bot \leftarrow \mathtt{VerifyCheckout}(\mathtt{co})$:
  Parse $\mathsf{co}$ into $([P_i,]\tau, \$, \alpha, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho, \psi)$
  $\mathtt{VerifyCheckoutCred}_{pk_{\mathtt{PS}}}(\tau)$
  Check if $(\tau.\mathtt{rk}, \tau.\mathtt{pr}, \tau.\mathtt{dl})$ is acceptable
  Check if $\tau$ is unique within $\tau.dl$
  $\mathtt{TS.Ver}_{pk_G}((\$, \mathsf{enc}_\alpha, \mathsf{enc}_\beta), \varrho)$
  $\mathtt{VerifyZK}(([P_i,]\tau.\mathsf{com}, \$, \alpha, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho), \psi)$
  If all the checks above pass, return 1
  Otherwise return 0

$\mathtt{po} \leftarrow \mathtt{IssuePmtOrder}(\mathtt{co})$:
  $\mathtt{VerifyCheckout}(\mathtt{co})$
  If verification fails, return 0
  Parse $\mathsf{co}$ into $([P_i,]\tau, \$, \alpha, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho, \psi)$
  If $P_i$ is present, update $P_i$'s history
  $\mathsf{po} \leftarrow (\$, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho)$
  **return** $\mathsf{po}$

$\mathtt{rc} \leftarrow \mathtt{IssueReceipt}(\mathrm{SK}_{\mathtt{M}_j}, \mathtt{co})$:
  $\mathtt{rc} \leftarrow \mathtt{Sign}_{sk_{\mathtt{M}_j}}(\mathtt{co})$
  **return** $\mathtt{rc}$

$\top/\bot \leftarrow \mathtt{VerifyPmtOrder}(\mathrm{SK}_{\mathtt{FN}}, \mathtt{po})$:
  Parse $\mathsf{po}$ into $(\$, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho)$
  $\mathtt{TS.Ver}_{pk_G}((\$, \mathsf{enc}_\alpha, \mathsf{enc}_\beta), \varrho)$
  $\beta = \mathtt{Dec}_{sk_{\mathtt{FN}}}(\mathsf{enc}_\beta)$
  Check if $\beta$ has not been used before
  Check if $\mathtt{TS.Open}_{sk_G}(\varrho)$ equals $C_i$ in $\beta$
  Verify the other billing info in $\beta$
  If all the checks above pass, return 1
  Otherwise return 0

$\top/\bot \leftarrow \mathtt{VerifyReceipt}(\mathtt{rc}, \mathtt{co})$:
  Find identity $M_j$ from $\alpha$ in $\mathsf{co}$
  **return** $\mathtt{SVer}_{pk_{\mathtt{M}_j}}(\mathtt{co}, \mathtt{rc})$
```

Fig. 6: Checkout algorithms.

Finally, with both the product description α and the checkout-credential τ, $\mathtt{C}_i$ can initiate the checkout phase.

Checkout. After receiving the checkout-credential τ and having obtained a product description, $\mathtt{C}_i$ decides whether to perform an anonymous (`IssueAnonCheckout`) or pseudonymous (`IssueCheckout`) checkout process. Let α be the product information with the product name, merchant, etc.; also, let \$ be the price of the product and let β be the customer's payment information containing a random number uniquely identifying each transaction. The checkout process is formed as follows (refer to Fig. 6 for a detailed description of the algorithms). Note that the information flow is equivalent to that in Fig. 2, but here we include additional cryptographic tokens.

```
$\pi \leftarrow \texttt{ShowReceiptZK}(\mathrm{SK}_{\texttt{C}_i}, \texttt{rc}, \texttt{co})$:
    Parse $\texttt{co} = ([P_i,]\tau, \$, \alpha, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho, \psi)$
    $\pi \leftarrow \texttt{TS.Claim}_{mk_i}(\varrho)$
    return $\pi$

$\top/\bot \leftarrow \texttt{VerifyReceiptZK}(\texttt{rc}, \texttt{co}, \pi)$:
    Parse $\texttt{co} = ([P_i,]\tau, \$, \alpha, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho, \psi)$
    $\texttt{VerifyReceipt}(\texttt{rc}, \texttt{co})$
    $\texttt{TS.ClaimVer}_{pk_G}(\pi, \varrho)$
    If all the checks pass, return 1
    Otherwise return 0
```

Fig. 7: Full system processes for claiming `rc` in Zero-Knowledge.

Step 1: Client issues a checkout object. A customer $\texttt{C}_i$ enters the checkout phase by creating a checkout object `co`, executing `Issue(Anon)Checkout` using the checkout-credential τ obtained during checkout-credential retrieval. In either procedure, $\texttt{C}_i$ generates a traceable signature ϱ on $(\$, \mathsf{enc}_\alpha, \mathsf{enc}_\beta)$, where enc_α is an encryption of the product information α, and enc_β is an encryption of the payment information β, and \$ is the price of the product. Then, $\texttt{C}_i$ generates a ZK proof ψ showing that the checkout-credential and the traceable signature (and the pseudonym for `IssueCheckout`) use the same mk_i. In summary, we have $\texttt{co} = ([P_i,]\tau, \$, \alpha, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho, \psi)$.

Step 2: Merchant processes checkout `co`. When $\texttt{M}_j$ receives the checkout object `co` (which includes the product information α in the clear, as well as encrypted), verifies it with `VerifyCheckout`. If verification succeeds, $\texttt{M}_j$ passes `co` to `PS`. Note that τ needs to be checked for uniqueness to prevent replay attacks. However, a used credential τ only needs to be stored up to $\tau.dl$. It is also possible for $\texttt{M}_j$ to include additional antifraud information, like an Address Verification Service value[11] (see Section 10).

Step 3: `PS` *issues a payment order po.* On receiving `co` from $\texttt{M}_j$, `PS` verifies `co`, runs `IssuePmtOrder` and issues a payment order *po* with the minimum information required by `FN` for processing the payment that is, $po = (\$, \mathsf{enc}_\alpha, \mathsf{enc}_\beta, \varrho)$.

Step 4-5: Payment confirmations. Given the payment order *po*, `FN` verifies it by running `VerifyPmtOrder`. If the verification succeeds, `FN` processes the order and notifies `PS` of the completion; `PS` in turn sends the confirmation back to $\texttt{M}_j$.

Step 6: $\texttt{M}_j$ *issues a receipt.* $\texttt{M}_j$ receives the confirmation from `PS` and runs `IssueReceipt`, issuing `rc`, a signature on `co`. Finally, $\texttt{C}_i$ verifies `rc` with `VerifyReceipt`.

Delivery. Once $\texttt{C}_i$ receives `rc`, he can use it to prove in ZK that he actually payed for some transaction `co`, and initiate additional processes, like having `DC` deliver the goods through APOD [5]. This proof is obtained with the processes in Fig. 7. In the showing process, if $\texttt{C}_i$ received a receipt `rc`, he shows `rc` along with the corresponding checkout object `co`; then, using his membership key mk_i, he claims ownership of a traceable signature contained in `co`. Even if he did not receive a receipt, he can prove ownership of ϱ to `FN` (using `ShowReceiptZK` too). Since `FN` is semi-honest, $\texttt{C}_i$ may ask `FN` to cancel the associated payment (or *force* `PS` and $\texttt{M}_j$ to reissue the receipt).

In order to interconnect with APOD, $\texttt{C}_i$ proves $\texttt{M}_j$ being the owner of `rc` (through `ShowReceiptZK`). Then, $\texttt{M}_j$ issues the credential `cred` required by APOD as in [5]. Note however that the incorporation of APOD incurs in additional costs and the need for further cryptographic tokens for merchants (who could delegate this task to `PS`). A

[11] `https://en.wikipedia.org/wiki/Address_Verification_System`. Last access on March 21st, 2018.

less anonymous delivery method, but probably good enough for many contexts, could be using Post Office boxes (or equivalent delivery methods) [30].

Completion. When $\mathtt{C}_i$ receives the goods, the completion phase may take place. In this phase, $\mathtt{C}_i$ may leave feedback or initiate a claim, for which he needs to prove having purchased the associated items. For this purpose, $\mathtt{C}_i$ can again make use of the `ShowReceiptZK` and `VerifyReceiptZK` processes, defined in Fig. 7.

8 Security

We assume that customers and merchants can act maliciously. PS is assumed to be semi-honest during checkout-credential retrieval, but malicious otherwise. FN is semi-honest. Additionally, we use the following oracles to give the adversary additional powers when necessary.

- (Add clients) This oracle, written AddC, allows the adversary to add a new client. The public/private key, pseudonym, and payment information of the client will be recorded. Finally, it returns the public key and pseudonym of the client.
- (Add merchants) This oracle, written AddM, allows the adversary to add a new merchant. However, the adversary does not observe any secret information of this merchant. The public/private key of the merchant will be recorded. Finally, it returns the public key of the merchant.
- (Corrupt clients) This oracle, written CorC, allows the adversary to corrupt a client in the system, i.e., the adversary will have all information about the target client.
- (Corrupt merchants) This oracle, written CorM, allows the adversary to corrupt a merchant in the system, that is, the adversary will have all the information about the target merchant.
- (Process checkout) This oracle, DoCheckout, is given as input a checkout *co*, and if co is valid, it will process the checkout (as would be done by PS and FN.)
- (Process confirmation) This oracle, DoConfirm, is given as input a checkout *co*, and if co is valid, returns the corresponding receipt rc (as would be done by the corresponding merchant $\mathtt{M}_j$.)
- (Process transaction) This oracle, Transaction, executes the complete process, including the checkout-credential retrieval and checkout, as would be done by a customer $\mathtt{C}_i$, producing the resulting co and rc.

8.1 Security properties

Privacy. The system possesses the following privacy properties.

- *Customer anonymity*. Customer anonymity requires that if a customer creates an anonymous checkout co, then no coalition of merchants, PS, and other customers should be able to determine the identity or pseudonym of the customer from co. We describe this requirement by using the following game.

```
**Experiment** $\mathbf{Exp}_{\mathcal{A}}^{\texttt{CA}}[\mathrm{b}, k]$:
  $(\mathrm{PK}_{\texttt{FN}}, \mathrm{SK}_{\texttt{FN}}) \leftarrow \texttt{FNSetup}(1^k)$.
  $(\mathrm{PK}_{\texttt{PS}}, \mathrm{SK}_{\texttt{PS}}) \leftarrow \texttt{PSSetup}(1^k)$.
  $(\texttt{C}_0, \texttt{C}_1, M, \alpha, \$) \leftarrow \mathcal{A}(\mathrm{PK}_{\texttt{FN}}, \mathrm{PK}_{\texttt{PS}}, \mathrm{SK}_{\texttt{PS}} : \mathsf{AddC}, \mathsf{CorC}, \mathsf{AddM}, \mathsf{CorM}, \mathsf{DoCheckout})$
  If $\texttt{C}_0$ or $\texttt{C}_1$ is corrupted, **return** 0.
  Let $(\beta_0, P_0)$ and $(\beta_1, P_1)$ be the billing info and pseudonym of $\texttt{C}_0$ and $\texttt{C}_1$.
  $\tau_0 \leftarrow \texttt{CheckoutCredRetrieval}(\mathrm{PK}_{\texttt{PS}}, \mathrm{PK}_{\texttt{FN}}, P_0)[\texttt{C}_0(\mathrm{SK}_{\texttt{C}_0}), \mathcal{A}]$
  $\tau_1 \leftarrow \texttt{CheckoutCredRetrieval}(\mathrm{PK}_{\texttt{PS}}, \mathrm{PK}_{\texttt{FN}}, P_1)[\texttt{C}_1(\mathrm{SK}_{\texttt{C}_1}), \mathcal{A}]$
  If $\tau_0$ and $\tau_1$ have different (risk, promo, deadline)s, **return** 0.
  $\texttt{co} \leftarrow \texttt{IssueAnonCheckout}(\mathrm{SK}_{\texttt{C}_b}, \tau_b, \alpha, \$, \beta_b)$.
  $\tilde{b} \leftarrow \mathcal{A}(\texttt{co} : \mathsf{DoCheckout})$.
  **return** $\tilde{b}$.
```

We say that a private payment system has customer anonymity if, for any stateful ppt algorithm $\mathcal{A}$, $\left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\texttt{CA}}[0, k] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{\texttt{CA}}[1, k]\right|$ is negligible in k.

This property matches the the *Hide the identity of a customer* property in Section 6.1. Note however that we condition it to the customer choosing to run an anonymous checkout process.

– *Transaction privacy against merchants and* PS. No coalition of merchants, PS and other customers should be able to determine the payment information (credit card number, etc.) in a customer's transaction. The following game describes this requirement.

```
**Experiment** $\mathbf{Exp}_{\mathcal{A}}^{\texttt{TPMPS}}[\mathrm{b}, k]$:
  $(\mathrm{PK}_{\texttt{FN}}, \mathrm{SK}_{\texttt{FN}}) \leftarrow \texttt{FNSetup}(1^k)$.
  $(\mathrm{PK}_{\texttt{PS}}, \mathrm{SK}_{\texttt{PS}}) \leftarrow \texttt{PSSetup}(1^k)$.
  $(\texttt{C}, \texttt{M}, \alpha, \beta_0, \beta_1, \$) \leftarrow \mathcal{A}(\mathrm{PK}_{\texttt{FN}}, \mathrm{PK}_{\texttt{PS}}, \mathrm{SK}_{\texttt{FN}} : \mathsf{AddC}, \mathsf{AddM})$
  Let $P$ be the pseudonym of $C$.
  $\tau \leftarrow \texttt{CheckoutCredRetrieval}(\mathrm{PK}_{\texttt{PS}}, \mathrm{PK}_{\texttt{FN}}, P)[C(\mathrm{SK}_C), \texttt{PS}(\mathrm{SK}_{\texttt{PS}})]$
  $\texttt{co} \leftarrow \texttt{IssueCheckout}(\mathrm{SK}_C, \tau, \alpha, \$, \beta_b)$.
  $\texttt{po} \leftarrow \texttt{IssuePmtOrder}(\texttt{co})$.
  $\tilde{b} \leftarrow \mathcal{A}(\texttt{co}, \texttt{po})$.
  **return** $\tilde{b}$.
```

We say that a private payment system has transaction privacy against merchants and PS if, for any stateful ppt algorithm $\mathcal{A}$, it holds that $\left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\texttt{TPMPS}}[0, k] - Pr[\mathbf{Exp}_{\mathcal{A}}^{\texttt{TPMPS}}[1, k]\right|$ is negligible in k.

This property matches the *Hide the payment information* property in Section 6.1.

– *Transaction privacy against* FN.

According to this requirement, the financial network FN should not be able to determine the detail of a customer's transaction beyond what is necessary, i.e., the customer identity and the amount of payment; in particular, the product information and the merchant identity of each transaction should be hidden from FN. We describe this requirement by using the following game.

```
**Experiment** $\mathbf{Exp}_{\mathcal{A}}^{\texttt{TPFN}}[\mathsf{b}, k]$:
  $(\mathrm{PK}_{\texttt{FN}}, \mathrm{SK}_{\texttt{FN}}) \leftarrow \texttt{FNSetup}(1^k)$.
  $(\mathrm{PK}_{\texttt{PS}}, \mathrm{SK}_{\texttt{PS}}) \leftarrow \texttt{PSSetup}(1^k)$.
  $(\texttt{C}, \texttt{M}_0, \texttt{M}_1, \alpha_0, \alpha_1, \$) \leftarrow \mathcal{A}(\mathrm{PK}_{\texttt{FN}}, \mathrm{PK}_{\texttt{PS}}, \mathrm{SK}_{\texttt{FN}} : \mathsf{AddC}, \mathsf{AddM})$
  Let $(\beta, P)$ be the payment information, and pseudonym of $C$.
  $\tau \leftarrow \texttt{CheckoutCredRetrieval}(\mathrm{PK}_{\texttt{PS}}, \mathrm{PK}_{\texttt{FN}}, P)[C(\mathrm{SK}_C), \texttt{PS}(\mathrm{SK}_{\texttt{PS}})]$
  $\texttt{co} \leftarrow \texttt{IssueCheckout}(\mathrm{SK}_C, \tau, \alpha_b, \$, \beta)$.
  $\texttt{po} \leftarrow \texttt{IssuePmtOrder}(\texttt{co})$.
  $\tilde{b} \leftarrow \mathcal{A}(\texttt{po})$.
  **return** $\tilde{b}$.
```

We say that a private payment system has transaction privacy against `FN` if, for any stateful ppt algorithm $\mathcal{A}$, it holds that $\left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\texttt{TPFN}}[0, k] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{\texttt{TPFN}}[1, k]\right|$ is negligible in k.

This property matches the *Hide the product information* stated in Section 6.1.

– *Unlinkable checkout-credential retrieval and checkout*. If a customer runs an anonymous checkout, no coalition of merchants, `PS`, and other customers should be able to link the customer or his pseudonym to the corresponding checkout-credential retrieval procedure beyond what the common message in the credential reveals. We describe this requirement by using the following game.

```
**Experiment** $\mathbf{Exp}_{\mathcal{A}}^{\texttt{URC}}[\mathsf{b}, k]$:
  $(\mathrm{PK}_{\texttt{FN}}, \mathrm{SK}_{\texttt{FN}}) \leftarrow \texttt{FNSetup}(1^k)$.
  $(\mathrm{PK}_{\texttt{PS}}, \mathrm{SK}_{\texttt{PS}}) \leftarrow \texttt{PSSetup}(1^k)$.
  $(C, M, \alpha, \$) \leftarrow \mathcal{A}(\mathrm{PK}_{\texttt{FN}}, \mathrm{PK}_{\texttt{PS}}, \mathrm{SK}_{\texttt{PS}} : \mathsf{AddC}, \mathsf{CorC}, \mathsf{AddM}, \mathsf{CorM}, \mathsf{DoCheckout})$
  If $C$ is corrupted, **return** 0.
  Let $(\beta, P)$ be the billing info and pseudonym of $C$ respectively.
  $\tau_0 \leftarrow \texttt{CheckoutCredRetrieval}(\mathrm{PK}_{\texttt{PS}}, \mathrm{PK}_{\texttt{FN}}, P)[C(\mathrm{SK}_C), \mathcal{A}]$
  $\tau_1 \leftarrow \texttt{CheckoutCredRetrieval}(\mathrm{PK}_{\texttt{PS}}, \mathrm{PK}_{\texttt{FN}}, P)[C(\mathrm{SK}_C), \mathcal{A}]$
  If $\tau_0$ and $\tau_1$ have different (risk, promo, deadline)s, **return** 0.
  $\texttt{co} \leftarrow \texttt{IssueAnonCheckout}(\mathrm{SK}_C, \tau_b, \alpha, \$, \beta)$.
  $\tilde{b} \leftarrow \mathcal{A}(\texttt{co} : \mathsf{DoCheckout})$.
  **return** $\tilde{b}$.
```

We say that a private payment system has unlinkable checkout-credential retrieval and checkout if, for any stateful ppt algorithm $\mathcal{A}$, $\left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\texttt{URC}}[0, k] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{\texttt{URC}}[1, k]\right|$ is negligible in k.

Note that if the checkout-credential retrieval and checkout phases were linkable even in anonyomous checkouts, it would not be possible to achieve customer anonymity. Thus, the need to ensure this property, which is not present in the version in Section 6.1, is consequence of having divided the whole process in two phases.

Note that this properties map to the properties in Section 6.1, with some additional conditions (see Fig. 8 for a pictorial representation). It is also worth noting that there are indirect connections between them. For instance, *Transaction privacy against* `FN` and *Transaction privacy against merchants and* `PS` undoubtedly improves resistance against differential privacy attacks aimed at deanonymizing customers (hence, affecting the *Customer anonymity*). However, as stated in the conclusion, a detailed analysis of these aspects is out of the scope of this work and is left for future work.

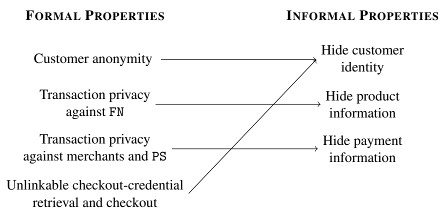

FORMAL PROPERTIES
INFORMAL PROPERTIES
Customer anonymity
Transaction privacy against FN
Transaction privacy against merchants and PS
Unlinkable checkout-credential retrieval and checkout
Hide customer identity
Hide product information
Hide payment information

Fig. 8: Mapping between informal privacy properties in Section 6.1 and formal privacy properties in this section.

Robustness. The system also ensures the following robustness properties, needed to prevent faulty executions.

- *Checkout-credential unforgeability*. A customer should not be able to forge a valid checkout-credential that contains a risk factor or a promotion or a deadline set by his own choice. We describe this requirement by using the following game.

```
**Experiment** $\mathbf{Exp}_{\mathcal{A}}^{\mathtt{CCU}}[k]$:
  $(\mathrm{PK}_{\mathtt{FN}}, \mathrm{SK}_{\mathtt{FN}}) \leftarrow \mathtt{FNSetup}(1^k)$.
  $(\mathrm{PK}_{\mathtt{PS}}, \mathrm{SK}_{\mathtt{PS}}) \leftarrow \mathtt{PSSetup}(1^k)$.
  $\tau \leftarrow \mathcal{A}(\mathrm{PK}_{\mathtt{FN}}, \mathrm{PK}_{\mathtt{PS}} : \mathsf{AddC}, \mathsf{CorC}, \mathtt{CheckoutCredRetrieval}_{\mathrm{SK}_{\mathtt{PS}}})$
  If $\mathtt{VerifyCheckoutCred}_{\mathrm{PK}_{\mathtt{PS}}}(\tau) = 1$ and $(\tau.rk, \tau.pr, \tau.dl)$ was never
  observed by $\mathtt{CheckoutCredRetrieval}_{\mathrm{SK}_{\mathtt{PS}}}$:
    **return** 1;
  Otherwise **return** 0.
```

  We say that a private payment system has checkout-credential unforgeability if, for any stateful ppt algorithm $\mathcal{A}$, it holds that $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathtt{CCU}}[k] = 1]$ is negligible in k.
- *Checkout unforgeability.* With a valid checkout-credential τ issued for a customer, no coalition of other customers, merchants, and PS should be able to forge a valid checkout co containing τ. We describe this requirement by using the following game.

```
**Experiment** $\mathbf{Exp}_{\mathcal{A}}^{\mathtt{CU}}[k]$:
  $(\mathrm{PK}_{\mathtt{FN}}, \mathrm{SK}_{\mathtt{FN}}) \leftarrow \mathtt{FNSetup}(1^k)$.
  $(\mathrm{PK}_{\mathtt{PS}}, \mathrm{SK}_{\mathtt{PS}}) \leftarrow \mathtt{PSSetup}(1^k)$.
  $(C, \mathtt{co}) \leftarrow \mathcal{A}(\mathrm{PK}_{\mathtt{FN}}, \mathrm{PK}_{\mathtt{PS}}, \mathrm{SK}_{\mathtt{PS}} : \mathsf{AddC}, \mathsf{AddM}, \mathsf{CorC}, \mathsf{CorM}, \mathsf{Transaction})$
  If co has been processed before, **return** 0.
  If $\mathtt{VerifyCheckout(co)} = 0$, **return** 0.
  If $C$ has never got risk/promotion in co, but co deducts $C$'s balance
    **return** 1; otherwise **return** 0.
```

  We say that a private payment system has checkout unforgeability if, for any stateful ppt algorithm $\mathcal{A}$, it holds that $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathtt{CU}}[k] = 1]$ is negligible in k.
- *Fraudulent transaction traceability.* When a customer C performs a fraudulent transaction, FN and PS are able to trace the pseudonym that C used even if the transaction is anonymous.

```
**Experiment** $\mathbf{Exp}_{\mathcal{A}}^{\mathtt{FTT}}[k]$:
  $(\mathrm{PK}_{\mathtt{FN}}, \mathrm{SK}_{\mathtt{FN}}) \leftarrow \mathtt{FNSetup}(1^k)$.
  $(\mathrm{PK}_{\mathtt{PS}}, \mathrm{SK}_{\mathtt{PS}}) \leftarrow \mathtt{PSSetup}(1^k)$.
  $(\mathtt{C}, \mathtt{co}) \leftarrow \mathcal{A}(\mathrm{PK}_{\mathtt{FN}}, \mathrm{PK}_{\mathtt{PS}}, \mathrm{SK}_{\mathtt{PS}} : \mathsf{AddC}, \mathsf{AddM}, \mathsf{CorC}, \mathsf{CorM}, \mathsf{DoCheckout})$
  Let $P$ be C's pseudonym
  If $\mathtt{VerifyCheckout}(\mathtt{co}) = 0$, **return** 0.
  Let $\varrho$ be the traceable signature issued by C contained in co
  $b \leftarrow \mathtt{TS.Trace}(\mathtt{TS.Reveal}(\mathtt{TS.Open}(\varrho)), P.\varrho, )$
  **return** $b$.
```
We say that a private payment system has fraudulent transaction traceability if, for any stateful ppt algorithm $\mathcal{A}$, it holds that $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathtt{FTT}}[k] = 0]$ is negligible in k.

– *Receipt unforgeability.* No coalition of customers, merchants (other than the target merchant M), and PS should be able to forge a valid receipt that looks originating from M. We describe this requirement by using the following game.

```
**Experiment** $\mathbf{Exp}_{\mathcal{A}}^{\mathtt{RU}}[k]$:
  $(\mathrm{PK}_{\mathtt{FN}}, \mathrm{SK}_{\mathtt{FN}}) \leftarrow \mathtt{FNSetup}(1^k)$.
  $(\mathrm{PK}_{\mathtt{PS}}, \mathrm{SK}_{\mathtt{PS}}) \leftarrow \mathtt{PSSetup}(1^k)$.
  $M \leftarrow \mathcal{A}(\mathrm{PK}_{\mathtt{FN}}, \mathrm{PK}_{\mathtt{PS}} : \mathsf{AddC}, \mathsf{AddM}, \mathsf{CorC}, \mathsf{CorM}, \mathsf{DoCheckout})$
  If merchant $M$ is corrupted, **return** 0
  $(\mathtt{co}, \mathtt{rc}) \leftarrow \mathcal{A}(\mathrm{PK}_{\mathtt{FN}}, \mathrm{PK}_{\mathtt{PS}} : \mathsf{AddC}, \mathsf{AddM}, \mathsf{CorC}, \mathsf{CorM}, \mathsf{DoConfirm})$
  If the merchant $M$ is corrupted, **return** 0
  If co was queried to DoCheckout, **return** 0
  If $\mathtt{VerifyReceipt}(\mathtt{rc}, \mathtt{co}) = 0$, **return** 0
  **return** 1
```
We say that a private payment system has receipt unforgeability if, for any stateful ppt algorithm $\mathcal{A}$, it holds that $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathtt{RU}}[k] = 1]$ is negligible in k.

– *Receipt claimability.* For a valid receipt from an uncorrupted customer, no other customer should be able to successfully claim the ownership of the confirmation.

```
**Experiment** $\mathbf{Exp}_{\mathcal{A}}^{\mathtt{RC}}[k]$:
  $(\mathrm{PK}_{\mathtt{FN}}, \mathrm{SK}_{\mathtt{FN}}) \leftarrow \mathtt{FNSetup}(1^k)$.
  $(\mathrm{PK}_{\mathtt{PS}}, \mathrm{SK}_{\mathtt{PS}}) \leftarrow \mathtt{PSSetup}(1^k)$.
  $(\mathtt{co}, \mathtt{rc}, \pi) \leftarrow \mathcal{A}(\mathrm{PK}_{\mathtt{FN}}, \mathrm{PK}_{\mathtt{PS}}, \mathrm{SK}_{\mathtt{PS}} : \mathsf{AddC}, \mathsf{AddM}, \mathsf{CorC}, \mathsf{CorM}, \mathsf{Transaction})$
  If $rc$ is never issued by Transaction, **return** 0
  If the owner customer of $(\mathtt{co}, \mathtt{rc})$ is corrupted, **return** 0
  If $\mathtt{VerifyReceiptZK}(\mathtt{rc}, \mathtt{co}, \pi) = 0$, **return** 0
  **return** 1
```
We say that a private payment system has receipt claimability if, for any stateful ppt algorithm $\mathcal{A}$, it holds that $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathtt{RC}}[k] = 1]$ is negligible in k.

8.2 Security of our system

In this section we prove that our system satisfies the preceding security requirements. We also recall that all communications where one party is the customer are assumed to take place through an anonymizing network.

We base our proofs in the security properties of the underlying building blocks. Specifically, we mainly refer to the *blindness* and *unforgeability* properties of partially blind signatures (see e.g. [52]); the *anonymity*, *misidentification security* and *framing security* properties of traceable signatures [41]; the *zero-knowledge* and *unforgeability* properties of zero-knowledge proof systems (see e.g. [33]); the *unforgeability* property of digital signatures (see e.g. [37]); and the property of *protection against identity compromise* of pseudonym systems (see e.g. [45]).

Theorem 1. *Our system is customer anonymous.*

Proof. Recall that in $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CA}}[\mathrm{b}]$, $(P_b, \tau_b, \mathrm{SK}_{\mathsf{C}_b}, \beta_b)$ was used (other unimportant elements were omitted). We define hybrid games as follows:
Hybrid 0. This is the actual game $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CA}}[0]$.
Hybrid 1. It's the same as Hybrid 0 except that all ZK proofs are simulated by ZK simulator. From ZK properties for the ZK proofs, Hybrid 0 and Hybrid 1 are indistinguishable.
Hybrid 2. It's the same as Hybrid 1 except that in the first checkout-credential retrieval (to generate τ_0), it uses $\mathsf{com} \leftarrow \mathsf{Com}(\mathrm{SK}_{\mathsf{C}_1}; r_{\mathsf{com}})$ instead of $\mathsf{com} \leftarrow \mathsf{Com}(\mathrm{SK}_{\mathsf{C}_0}; r_{\mathsf{com}})$. From the hiding property of the commitment scheme, Hybrid 1 and Hybrid 2 are indistinguishable. Note that at this moment, the τ_0 and τ_1 are identically distributed.
Hybrid 3. It's the same as Hybrid 2 except that it generates `co` by running `IssueAnonCheckout` using τ_1 (and $\mathrm{SK}_{\mathsf{C}_0}, \beta_0$) instead of τ_0. Hybrid 2 and Hybrid 3 are indistinguishable due to blindness of the underlying partial blind signature scheme. Note that the adversary (against blindness property) first generates a key pair for the blind signature and two different messages; after signing two messages in a randomly chosen order by the game environment (for the blindness), the adversary should guess the message order (i.e., reversed or not). See [12] for more detail. We show the reduction. That is, we construct an adversary $\mathcal{B}$ breaking the blindness of the underlying partial blind scheme given an adversary distinguishing the two hybrids as follows:

> $\mathcal{B}$ runs key generation algorithm for PBS to generate a key pair, and sends out the pair to the outer environment. At the same time $\mathcal{B}$ works as Hybrid game (2 or 3). In particular, when $\mathcal{A}$ chooses two customers, $\mathcal{B}$ chooses $\mathsf{com}_0 = \mathsf{Com}(\mathrm{SK}_{\mathsf{C}_1}; r_0)$ and $\mathsf{com}_1 = \mathsf{Com}(\mathrm{SK}_{\mathsf{C}_1}; r_1)$ as the two messages to distinguish and sends out $(\mathsf{com}_0, \mathsf{com}_1)$ to the outer environment. Once the outer environment gives the blind signature ϱ, $\mathcal{B}$ uses it to generate `co`. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

When the signature ϱ is on com_0, the simulation is identical to Hybrid 2; on the other hand, when ϱ is on com_1, to Hybrid 3. Therefore, if $\mathcal{A}$ distinguishes the two hybrids with non-negligible probability, $\mathcal{B}$ also breaks the blindness property of the underlying signature with non-negligible probability.
Hybrid 4. It's the same as Hybrid 3 except that it generates `co` by running `IssueAnonCheckout` using $(\tau_1, \mathrm{SK}_{\mathsf{C}_1}, \beta_0)$ instead of $(\tau_1, \mathrm{SK}_{\mathsf{C}_0}, \beta_0)$. Hybrid 3 and Hybrid 4 are indistinguishable due to anonymity of the group signature scheme. The reduction is straightforward.
Hybrid 5. It's the same as Hybrid 3 except that it generates `co` by running `IssueAnonCheckout` using $(\tau_1, \mathrm{SK}_{\mathsf{C}_1}, \beta_1)$ instead of $(\tau_1, \mathrm{SK}_{\mathsf{C}_1}, \beta_0)$. Hybrid 4 and Hybrid 5 are indistinguishable due to semantic security of the public key encryption. The reduction is straightforward.

Hybrid 6. It's the same as Hybrid 5 except that in the first checkout-credential retrieval (to generate τ_0), it uses $\mathsf{com} \leftarrow \mathsf{Com}(\mathrm{SK}_{\mathtt{C}_0}; r_{\mathsf{com}})$ instead of $\mathsf{com} \leftarrow \mathsf{Com}(\mathrm{SK}_{\mathtt{C}_1}; r_{\mathsf{com}})$. From the hiding property of the commitment scheme, Hybrid 5 and Hybrid 6 are indistinguishable.
Hybrid 7. It's the same as Hybrid 6 except that ZK proofs are actually done instead of using simulations. From ZK properties for the ZK proofs, Hybrid 6 and Hybrid 7 are indistinguishable. Observe that Hybrid 7 is indeed $\mathbf{Exp}_{\mathcal{A}}^{\mathtt{CA}}[1]$, which concludes the proof.

Theorem 2. *Our system has the property of unlinkable checkout-credential retrieval/checkout.*

Proof. The proof is the same as showing indistinguishability of Hybrid 3 and 4 in the previous proof.

Theorem 3. *Our system has transaction privacy against* `FN`.

Proof. `FN` only views the encryption enc_α, and when `PS` creates a payment order, the payee is set to `PS` (instead of $\mathtt{M}_j$) in order to prevent `FN` to link $\mathtt{C}_i$ and $\mathtt{M}_j$. Altogether, this guarantees transaction privacy against `FN` from semantic security of the underlying encryption scheme.

Theorem 4. *Our system has transaction privacy against merchants and* `PS`.

Proof. Payment information (credit card, billing address, etc.) is encrypted by $\mathtt{C}_i$ using `FN`'s public key. Thus, transaction privacy against merchants and `PS` is guaranteed from semantic security of the underlying encryption scheme.

Theorem 5. *Our system satisfies checkout-credential unforgeability.*

Proof. Checkout-credential unforgeability simply follows from unforgeability of the partial blind signature scheme.

Theorem 6. *Our system satisfies checkout unforgeability.*

Proof. Checkout unforgeability follows from soundness of ZK proofs. In particular, $\tau.\mathsf{com}$ should be the commitment to mk_i due to the soundness of the proof in the checkout-credential retrieval. Moreover, the soundness of ZK proof ψ in the checkout object makes sure that both $\tau.\mathsf{com}$ and ϱ use the same member key mk_i.

Theorem 7. *Our system satisfies fraudulent transaction traceability.*

Proof. Fraudulent transaction traceability holds immediately from the security against misidentification of the underlying traceable signature scheme.

Theorem 8. *Our system satisfies receipt unforgeability.*

Proof. Receipt unforgeability holds immediately from unforgeability of underlying the digital signature scheme.

Theorem 9. *Our system satisfies receipt claimability.*

Proof. Receipt unforgeability holds immediately from security against framing attacks of the underlying traceable signature scheme.

9 Testing environment and results

For showing the applicability of our approach, we summarize some preliminary results obtained through a prototype incorporating the functionalities described in Section 7. We then compare them with current industry systems. Note however that the latter systems are highly optimized ones, using distributed infrastructures. The building blocks we have used are: BFPV13 [12] for blind signatures, CPY06 [23] as traceable signatures, and Pedersen commitments [55] and SKREP as defined in [19] for proving correctness in ZK of the commitments and the various ZK proofs. The code of the prototype is available upon request.

We used a laptop (Intel Core i5-480M, 4GB DDR3, with Debian Wheezy) and a desktop PC (Intel Core i7-2600, 16GB DDR3, with Debian Wheezy). We center our attention in the most frequent actions: anonymous and pseudonymous checkouts. For RSA keypairs we used a module of 1024 bits, while for ECC we used 160 bit elements. Additionally, our prototype included a MySQL database where both PS and FN keep the necessary data. We also run the following fraud prevention and marketing procedures: each time a checkout-credential was requested, PS issued a \$5 discount for customers having already spent over \$100; it also checked that no more than 10 purchases were made during the same day and ran the AVS test. Additionally, during checkout, PS rejected any transaction of more than \$1000, and FN performed billing/shipping matching. During checkout-credential retrieval, the customer role was played by the laptop (with 4 parallel threads), and the PS role by the desktop PC (with 8 parallel threads). During checkout, the laptop ran as a customer and merchant (each process spawning 2 threads), with the desktop PC acting as PS and FN (each process spawning 4 threads). Note that the PS throughput measures the average time intervals between the time PS receives a request and the time it has processed the response, ignoring the time taken to transmit them. Finally, at the moment the tests were performed, each machine was in a different network, with the `traceroute` command showing a route of 16 hops between them, and average roundtrip time of 44 ms. The results are summarized in Table 2.

To provide some context, Magento, one of the main e-commerce platform providers, claims to achieve 593 purchases per hour (~ 0.17 per second) with the 2.0 version of their system running on a 4 core domestic server or roughly 2500 orders per hour (~ 0.7 per second) using 100 parallel threads[12]. Of course, it is important to note that we have simplified some necessary parts of the process, such as the payment transaction itself (which is just simulated through a database modification). This, however, is likely to be a relatively negligible (i.e., very fast) operation compared to the overall e-shopping transaction: for instance VISA processed 141.0 billion transactions in 2016[13], which makes roughly 4500 transactions per second (this does not necessarily imply quick payments, but serves to get an idea).

Taking into account the limitations of our experiments (both in hardware equipment and software optimization), it is quite notable that we achieved a rate of $1.03\ T_{\texttt{PS}}$

[12] `https://magento.com/sites/default/files/White%20Paper%20-%20Magento%202.0%20Performance%20and%20Scalability%2003.31.16.pdf`. Last access on March 21st, 2018.

[13] `https://usa.visa.com/dam/VCOM/global/about-visa/documents/visa-facts-figures-jan-2017.pdf`. Last access on March 21st, 2018.

Action		***AL*** (secs)	$T_{\texttt{PS}}$ (reqs/sec)	***Bytes*** (sent/recv)
Checkout-credential retrieval		$1.26(\pm 0.49)$	1.03	C: $4426/793$ PS: $793/4426$
Checkout	**Anon.**	$1.68(\pm 0.47)$	3.01	C: $4212/291$ M: $4133/4358$ `PS`: $1549/3908$ `FN`: $66/1403$
	Pnym.	$1.83(\pm 0.43)$	2.95	C: $5850/291$ M: $6828/5996$ `PS`: $1549/6603$ `FN`: $66/1403$

Table 2: Summary of results. *AL* stands for round-trip Absolute Latency: total time since the customer starts the request until he receives the response. $T_{\texttt{PS}}$ stands for Throughput at the Payment System's side: total number of requests successfully answered, per second (in real time, including time spent in data transfers). *Bytes* indicates the average size in bytes of the messages sent/received during the action, for each involved entity. All results are averaged over 1000 transactions of the corresponding type.

(which may be seen as the equivalent of TPS) for checkout-credential retrieval and a rate of roughly 3 TPS for checkout (a bit higher for anonymous ones and a bit lower for pseudonymous). These figures are undoubtedly in the range of Magento's (even accounting for the mentioned payment simplification).

As for the communication costs, as seen in Table 2, the heavier load is taken by `PS` and `FN`. `PS` sends roughly $793+1549 = 2342$ Bytes and receives roughly $4426+3908 = 8334$ (anonymous checkouts) and $4426 + 6603 = 11029$ (pseudonymous checkouts) Bytes. `FN` sends roughly 66 Bytes and receives 1403 during checkout. Hence, `PS` supports approximately 10 times more communication overload than Bitcoin's peers in the best comparison. This is not bad, given the overhead of cryptography used heavily in privacy oriented communication, and further, this proportion can probably be greatly improved by further optimization.

Concerning the sizes of the groups of customers in the group signature schemes, we note that this is a highly configurable aspect. For instance, groups can be set based on geographies, based on sign up time, or any other heuristic. As for the impact on performance of the sizes of the groups, we refer to [29], which we have used to implement our prototype and offers some raw statistics about the group sizes and throughput of the main operations.

10 Additional Functionality for the Full System

Next, we discuss how to implement privacy preserving operations and support transactions that are usually added to the basic e-shopping life cycle in various scenarios.

Fraud prevention filters. Our system supports the following fraud prevention filters. Each filter is on the transaction-level (tx-level) or the account-level (acc-level), depending on whether it checks the information specific to transactions or to accounts (i.e., customers).

- *Pseudonym velocity filter (acc-level):* It measures how many recent purchases have been initiated by the same pseudonym (similar to PayPal's IP velocity filter). It can be applied during checkout-credential retrieval.
- *Suspicious shipment changes (acc-level):* This filter can be added by making the city/country of shipment visible, e.g., including it in the common message of the partially blind signature obtained during checkout-credential retrieval. Note that revealing the city/country would not be privacy sensitive in most occasions.
- *Address verification system (*`AVS`*, acc-level):* This filter can be employed by introducing the billing address within the payment information encrypted under `FN`'s public key. In this manner, only `FN` would be able to receive it and perform the AVS test as usual.
- *Billing/Shipping Address Mismatch Filter (tx-level):* With Locality Sensitive Hashing (LSH) [20,54], this filter can be added as follows. `C` computes two hashes $\tilde{b} \leftarrow H(b, r)$, $\tilde{s} \leftarrow H(s, r)$, where b and s are the billing and shipping address, respectively, and r is a random value. Then it sends $(r, \tilde{b}, \tilde{s})$ to `M` during checkout, who compares them. The probability of obtaining a match will be proportional to the similarity of b and s due to the properties of LSH. Since `FN` later checks if $\tilde{b}$ is equal to $H(billing, r)$ using the actual billing address $billing$, the filter works correctly.
- *Maximum price per order (tx-level):* This filter is trivial to apply by `M` or `PS`.
- *Maximum number of items per order (tx-level):* Trivial to apply by `M` or `PS`.
- *Other filters:* Filters like *Currency type*, *Bank Identification Number*, *Transaction type* [40] may be directly applied.

Note that the account-level filters (excluding AVS by `FN`) are applied by `PS` during the checkout-credential retrieval phase. Thus, anonymous checkouts do not affect their effectiveness. Transaction-level filters may be applied by either `PS` or merchants. Also, since an account risk estimation is passed to the checkout phase as part of the checkout-credential, both account and transaction risks may be considered jointly, even for completely anonymous checkouts.

Marketing techniques. In our system, `PS` issues promotions during checkout-credential retrieval, consulting the pseudonym's history. In [42], promotions are classified depending on for how long they can be applied (limited or unlimited) and the intended recipient (targeted or untargeted). Untargeted and unlimited (UU) promotions are trivial. The other three combinations may be achieved as follows:

- *Targeted and unlimited (TU).* `C` creates a group signature on the promotion and includes it in the blindly signed message at checkout-credential retrieval. At checkout, `C` includes this group signature in the ZK proofs (verified in `VerifyZK` in Figure 6).
- *Untargeted and limited (UL).* Simply include a deadline in the promotion.
- *Targeted and limited (TL).* Add a group signature as in TU promotions, specifying the target and deadline.

Our system could also be extended to support merchant-issued promotions. To do this, a group of merchants should be set up, and each merchant $\mathtt{M}_j$ should have a policy $\mathtt{pr}_{\mathtt{M}_j}$ for issuing promotions. Before initiating checkout-credential retrieval, $\mathtt{M}_j$ would send $(\mathtt{pr}_{\mathtt{M}_j}, \mathtt{Sign}(sk_{\mathtt{M}_j}, \alpha))$ to customer $\mathtt{C}_i$. $\mathtt{C}_i$ would then include $\mathtt{pr}_{\mathtt{M}_j}$ within the common message and $\mathtt{Sign}(sk_{\mathtt{M}_j}, \alpha)$ within the blinded message, both to be signed by PS during checkout-credential retrieval. After verifying the group signature, PS would just grant the promotions that $\mathtt{C}_i$ is eligible for, based on $\mathtt{pr}_{\mathtt{M}_j}$. During checkout, $\mathtt{M}_j$ would also verify $\mathtt{Sign}(sk_{\mathtt{M}_j}, \alpha)$ in order to prevent $\mathtt{C}_i$ using $\mathtt{pr}_{\mathtt{M}_j}$ with another $\mathtt{M}_i$.

Anonymous product returns for physical goods. Product returns may not be directly applicable, since APOD [5] only explicitly supports delivery from the merchant to the customer, but we can overcome this issue by allowing the customer to specify a return-path at the beginning of the APOD protocol and setting up an additional APOD PIN code set for the merchant to receive the returned package. Then, the customer can instruct the initial carrier to deliver back the product.

Contact customer. This might be necessary in extraordinary situations. For pseudonymous checkouts, a pseudonymous email address associated with the customer's account may be used. For anonymous checkouts, one-time email addresses should be employed, e.g. using an anonymous email service, like Bitmessage.ch. Alternatively, customers could prove in ZK ownership of a receipt, and then receive any notification related to it. However, this requires an active behavior from customers instead of just receiving an email.

Customer opinions. In order to add an opinion about a specific purchase, the customer may just generate a group-signed opinion and create a ZK proof (similar to those sent during checkout) covering this group signature and a claim (like in Fig. 7) of the receipt obtained during checkout. Any valid opinion would then just be shown in the corresponding website.

Subscriptions. If fees are paid in advance, this is easily solved. For physical goods (e.g., using APOD), customers may initially set up multiple shipment information, one for each delivery. For periodic digital goods, M may request C to claim ownership of the checkout group signatures as in Fig. 7 (adding a ZK proof with the same member key depending on some varying value for preventing pre-computations) to send the periodic digital good.

For recurring payments, the associated instruction could be sent to FN within the payment information. Also, if customer consent is required to renew the subscription, a notification could be sent to him (see *Contact customer* above).

Taxation. Assuming that revealing the city, province, state or country of shipment is not privacy threatening, and including it within the common message of the checkout-credential, our system provides all the typically necessary information for tax calculation by either PS or merchant. That is, customer (destination) locality, merchant (source) locality and type of good.

Alternative payment methods. Note that we have omitted specific details about the payment method. Therefore, our system would be easily compatible with any payment method, such as card based payments or cryptocurrencies, such as Bitcoin [49] and

Ripple[14]. Indeed, with cryptocurrencies, although the entity `FN` still exists, its role is reduced, mainly acting as an entry point to the payment system. In particular, the payment information doesn't necessarily include the real identity of a client or the billing address; instead, clients will indirectly prove ownership of an account (represented by a public key) by demonstrating that they control the corresponding private key. Now, when `FN` opens the received group signature, `FN` cannot match the signer's identity with the one in the payment information, since the payment information would not contain the real identity of the customer. However, group signatures are still useful for enabling privacy-preserving fraud prevention and marketing techniques during the rest of the process. Additionally, in Bitcoin/Ripple, the balance of each account is public given the account identifier, and verifying that the customer has enough funds is straightforward. Still, due to lack of certain pieces of information (e.g., billing addresses), some `FN`-owned filters for verifying other aspects (e.g., the AVS test) may not be applicable. Nevertheless, merchants not willing to accept alternative payment methods may just reject them by using the *Bank Identification Number* filter (Bitcoin/Ripple "acting" as banks).

Combined with Ripple, our system will have greater payment flexibility, while still maintaining good privacy guarantees. With Bitcoin, the degree of trust placed in `FN` can be greatly reduced, due to the nature of Bitcoin guaranteeing verifiable payment transactions. Moreover, Bitcoin itself does not guarantee anonymity nor unlinkability very well [7], so the privacy guarantees in our system are beneficial. Finally, both Bitcoin and Ripple would benefit from the rich set of useful features mentioned above, and this allows them to be native part of a comprehensive e-commerce system.

11 Conclusion and Future Work

We have put forth our proposal for reaching a balance between privacy and utility in e-shopping. This is a complex scenario, where the diverse set of functionalities required by the industry makes it hard to provide them in a privacy respectful manner [30]. Moreover, the condition of having to maintain similar information flows and general architecture, greatly constraints the application of traditional privacy by design principles. For this purpose, we have set out our methodology, extrapolated from simpler cases in cryptography which begin by offering a functionality constrained primitive providing the highest privacy protections; but then are modified to incorporate back some of the lost utility. We refer to this methodology as "*utility, privacy, and then utility again*". While we apply it to the context of e-shopping, this strategy is not restricted to it, and can be applied to transition from policy to engineering in privacy protection in already deployed systems [25]. In other words, our work contributes to build up the Business, Legal, and Technical framework [38] demanded to reconcile economic interests, citizens' rights, and users' needs in today's scenario.

Concerning our proposed e-shopping solution, with respect to the related works summarized in Section 2, our proposal has the advantage of integrating all core components of e-shopping (purchase, checkout, delivery and completion) and the advanced

[14] `https://ripple.com/`. Ripple is an open system for interoperation between different payment methods, e.g., Bitcoin, real currencies, or account-based transactions.

functionality in industry systems (marketing and fraud prevention). To the best of our knowledge this is a currently unsolved problem [30,63].

Note that our system provides a basic infrastructure for building privacy respectful systems requiring user profiling. Specifically, users pseudonymously obtain customized credentials based on their history, and then anonymously prove possession of those credentials unlinkably to the pseudonymous phase. We have also implemented a prototype of our system, showing its practicability and low added costs.

Nevertheless, further work is necessary. We can also include aggregated antifraud and promotions information that is publicly accessible from the checkout-credential. Hence, an open problem is reducing the impact of this leak for reidentification, both trying to conceal that data (we sketch initial ideas in Section 11.1) and by studying optimal partitions for fraud and promotion values assuming a minimum volume of purchases.

11.1 Sketch on ZK approach for fraud

The major aspect limiting the privacy level achieved by our proposal in Section 7 is the inclusion of antifraud and marketing metainformation in the public part of the checkout-credential. Next, we informally skecth a proposal for moving fraud information to the blindly signed part of the checkout-credential and proving in ZK its correctness.

Antifraud information is actually a risk score, e.g., a number between 0 and 10, where 0 means no risk and 10 means maximum risk. Then, as in `CheckoutCredRetrieval`:

1. The customer $\mathtt{C}_i$ initiates the process, sending his pseudonym P_i.
2. $\mathtt{PS}$ estimates P_i's fraud risk to be e.g. x, and sends it back to $\mathtt{C}_i$.

Subsequently, instead of introducing x as part of the common message, $\mathtt{PS}$ and $\mathtt{C}_i$ proceed as follows:

1. $\mathtt{C}_i$ commits to x, includes the commitment in the blinded message, and issues a ZK proof of correctness.
2. $\mathtt{PS}$ verifies the proof and issues the blind signature (the checkout-credential).

Finally, during checkout, $\mathtt{C}_i$ requests $\mathtt{M}_j$ to specify its maximum acceptable fraud risk estimation. Say $\mathtt{M}_j$ accepts a risk up to y (and, for the sake of illustration, that $x < y$). Then, in addition to the current proofs, $\mathtt{C}_i$ attaches a proof showing that x lies in the interval $[0, y]$ [14]. This would effectively increase the size of possible candidates of having retrieved that credential to all those customers with a fraud risk estimation less than y, instead of equal to x.

Acknowledgements. The work of Jesus Diaz was done in part while visiting the Network Security Lab at Columbia University. The work of Seung Geol Choi was supported in part by the Office of Naval Research under Grant Number N0001415WX01232. The work of David Arroyo was supported by projects S2013/ICE-3095-CM (CIBERDINE) and MINECO DPI2015-65833-P of the Spanish Government. The work of Francisco B. Rodriguez was supported by projects MINECO TIN2014-54580-R and TIN2017-84452-R of the Spanish Government. The work of Moti Yung was done in part while visiting the Simons Institute for Theory of Computing, UC Berkeley.

References

1. Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In *ASIACRYPT*, pages 244–251, 1996.
2. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, pages 119–135, 2001.
3. Ross J. Anderson. Risk and privacy implications of consumer payment innovation. `http://www.cl.cam.ac.uk/~rja14/Papers/anderson-frb-kansas-mar27.pdf`, 2012.
4. Ross J. Anderson, Chris Barton, Rainer Böhme, Richard Clayton, Michel van Eeten, Michael Levi, Tyler Moore, and Stefan Savage. Measuring the cost of cybercrime. In *WEIS 2012, Germany, 25-26 June, 2012*, 2012.
5. Elli Androulaki and Steven M. Bellovin. Apod: Anonymous physical object delivery. In *Privacy Enhancing Technologies*, pages 202–215, 2009.
6. Elli Androulaki and Steven M. Bellovin. APOD: Anonymous Physical Object Delivery. In *Privacy Enhancing Technologies*, pages 202–215, 2009.
7. Elli Androulaki, Ghassan Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in bitcoin. In *Financial Cryptography*, pages 34–51, 2013.
8. Giannakis Antoniou and Lynn Margaret Batten. E-commerce: protecting purchaser privacy to enforce trust. *Electronic Commerce Research*, 11(4):421–456, 2011.
9. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, Heidelberg, December 2001.
10. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474, 2014. http://dx.doi.org/10.1109/SP.2014.36.
11. Vicente Benjumea, Seung Geol Choi, Javier López, and Moti Yung. Fair traceable multi-group signatures. In *FC 2008*, pages 231–246, 2008.
12. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Short blind signatures. *Journal of Computer Security*, 21(5):627–661, 2013.
13. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.
14. Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, pages 431–444, 2000.
15. Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.
16. Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. Oblivious transfer with access control. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 131–140, New York, NY, USA, 2009. ACM. http://doi.acm.org/10.1145/1653662.1653679.
17. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, pages 61–76, 2002.
18. Jan Camenisch, Jean-Marc Piveteau, and Markus Stadler. An efficient fair payment system. In *ACM Conference on Computer and Communications Security*, pages 88–94, 1996.
19. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO*, pages 410–424, 1997.

20. Moses Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002.
21. David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.
22. David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.
23. Seung Geol Choi, Kunsoo Park, and Moti Yung. Short traceable signatures based on bilinear pairings. In *IWSEC*, pages 88–103, 2006.
24. Scott E. Coull, Matthew Green, and Susan Hohenberger. Access controls for oblivious and anonymous systems. *ACM Trans. Inf. Syst. Secur.*, 14:10:1–10:28, June 2011. http://doi.acm.org/10.1145/1952982.1952992.
25. George Danezis, Josep Domingo-Ferrer, Marit Hansen, Jaap-Henk Hoepman, Daniel Le Metayer, Rodica Tirtea, and Stefan Schiffner. Privacy and data protection by design-from policy to engineering. Technical report, ENISA, 2014.
26. George Danezis, Markulf Kohlweiss, Benjamin Livshits, and Alfredo Rial. Private client-side profiling with random forests and hidden markov models. In *Privacy Enhancing Technologies - 12th International Symposium, PETS 2012, Vigo, Spain, July 11-13, 2012. Proceedings*, pages 18–37, 2012.
27. George I. Davida, Yair Frankel, Yiannis Tsiounis, and Moti Yung. Anonymity control in e-cash systems. In *Financial Cryptography*, pages 1–16, 1997.
28. Yves-Alexandre de Montjoye, Laura Radaelli, Vivek K. Singh, and Alex . Pentland. Unique in the shopping mall: On the reidentifiability of credit card metadata. *Science*, 347(6221):536–539, January 2015.
29. Jesus Diaz, David Arroyo, and Francisco de Borja Rodríguez. libgroupsig: An extensible C library for group signatures. *IACR Cryptology ePrint Archive*, 2015:1146, 2015.
30. Jesus Diaz, Seung Geol Choi, David Arroyo, Angelos D. Keromytis, Francisco B. Rodriguez, and Moti Yung. Privacy Threats in E-Shopping (Position Paper). In *Data Privacy Management*, 2015.
31. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
32. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association. http://dl.acm.org/citation.cfm?id=1251375.1251396.
33. Uriel Feige, Amos Fiat, and Adi Shamir. Zero knowledge proofs of identity. In *STOC*, pages 210–217, 1987.
34. Christina Garman, Matthew Green, and Ian Miers. Accountable privacy for decentralized anonymous payments. *IACR Cryptology ePrint Archive*, 2016:61, 2016.
35. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.
36. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
37. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
38. Daniel Greenwood, Arkadiusz Stopczynski, Brian Sweatt, Thomas Hardjono, and Alex Pentland. The new deal on data: A framework for institutional controls. *Privacy, Big Data, and the Public Good: Frameworks for Engagement*, page 192, 2014.
39. Markus Jakobsson and David M'Raïhi. Mix-based electronic payments. In *Selected Areas in Cryptography*, pages 157–173, 1998.
40. Sanjeev Jha, Montserrat Guillen, and J. Christopher Westland. Employing transaction aggregation strategy to detect credit card fraud. *Expert Syst. Appl.*, 39(16):12650–12657, 2012.

41. Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 571–589, 2004. http://www.iacr.org/cryptodb/archive/2004/EUROCRYPT/2477/2477.pdf.
42. Manoj Kumar, Anand Rangachari, Anant Jhingran, and Rakesh Mohan. Sales promotions on the internet. In *Proceedings of the 3rd conference on USENIX Workshop on Electronic Commerce - Volume 3*, WOEC98, pages 14–14, Berkeley, CA, USA, 1998. USENIX Association. http://dl.acm.org/citation.cfm?id=1267147.1267161.
43. Benoît Libert, Thomas Peters, and Moti Yung. Group signatures with almost-for-free revocation. In *CRYPTO*, pages 571–589, 2012.
44. Benoît Libert and Moti Yung. Fully forward-secure group signatures. In *Cryptography and Security*, pages 156–184, 2012.
45. Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, pages 184–199, 1999.
46. Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 397–411, 2013.
47. Tehila Minkus and Keith W. Ross. I know what you're buying: Privacy breaches on ebay. In *PETS 2014, Amsterdam, July, 2014.*, 2014.
48. Steven J. Murdoch and Ross J. Anderson. Verified by Visa and MasterCard SecureCode: Or, how not to design authentication. In *Financial Cryptography*, 2010.
49. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. http://www.bitcoin.org/bitcoin.pdf.
50. Toru Nakanishi, Nobuaki Haruna, and Yuji Sugiyama. Unlinkable electronic coupon protocol with anonymity control. In *ISW*, pages 37–46, 1999.
51. Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, 2008.
52. Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In *TCC*, pages 80–99, 2006.
53. Javier Parra-Arnau, David Rebollo-Monedero, and Jordi Forné. Optimal forgery and suppression of ratings for privacy enhancement in recommendation systems. *Entropy*, 16(3):1586–1631, 2014.
54. Kurt Partridge, Manas A. Pathak, Ersin Uzun, and Cong Wang. Picoda: Privacy-preserving smart coupon delivery architecture, 2012.
55. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.
56. Sören Preibusch, Thomas Peetz, Günes Acar, and Bettina Berendt. Purchase details leaked to PayPal (Short Paper). In *Financial Cryptography*, 2015.
57. Naren Ramakrishnan, Benjamin J. Keller, Batul J. Mirza, Ananth Grama, and George Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54–62, 2001.
58. ITU-T Recommendation. X.509. information technology - open systems interconnection - the directory: Authentication framework (june 1997).
59. Alfredo Rial. *Privacy-Preserving E-Commerce Protocols*. PhD thesis, Arenberg Doctoral School, KU Leuven, 2013.
60. Alfredo Rial, Markulf Kohlweiss, and Bart Preneel. Universally composable adaptive priced oblivious transfer. In *Pairing-Based Cryptography - Pairing 2009, Third International Conference, Palo Alto, CA, USA, August 12-14, 2009, Proceedings*, pages 231–247, 2009.
61. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

62. Phillip Rogaway. The moral character of cryptographic work. *IACR Cryptology ePrint Archive*, 2015:1162, 2015.
63. Antonio Ruiz-Martinez. Towards a web payment framework: State-of-the-art and challenges. *Electronic Commerce Research and Applications*, 2015. http://www.sciencedirect.com/science/article/pii/S1567422315000587.
64. Tomas Sander and Amnon Ta-Shma. Flow control: A new approach for anonymity control in electronic cash systems. In *Financial Cryptography*, pages 46–61, 1999.
65. S. Stolfo, Y. Yemini, and L. Shaykin. Electronic purchase of goods over a communications network including physical delivery while securing private and personal information of the purchasing party, November 2 2006. US Patent App. 11/476,304.
66. Chunfu Tan and Jianying Zhou. An electronic payment scheme allowing special rates for anonymous regular customers. In *DEXA Workshops*, pages 428–434, 2002.
67. Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. Adnostic: Privacy preserving targeted advertising. In *NDSS*, 2010.
68. Visa. Verified by Visa – acquirer and merchant implementation guide, 2011.