Guided Specification and Analysis of a Loyalty Card System

Laurent Cuennet[1], Marc Pouly[2], and Saša Radomirović[3]

[1] Department of Informatics, University of Fribourg
[2] Lucerne University of Applied Sciences and Arts
[3] Institute of Information Security, Department of Computer Science, ETH Zürich

Abstract. We apply a graphical model to develop a digital loyalty program protocol specifically tailored to small shops with no professional or third-party-provided infrastructure. The graphical model allows us to capture assumptions on the environment the protocol is running in, such as capabilities of agents, available channels and their security properties. Moreover, the model serves as a manual tool to quickly rule out insecure protocol designs and to focus on improving promising designs. We illustrate this by a step-wise improvement of a crude but commercially used protocol to finally derive a light-weight and scalable security protocol with proved security properties and many appealing features for practical use.

1 Introduction

Paper-based ink stamp cards are a convenient and inexpensive way for small shops to improve customer loyalty. Other than an ink stamp and printed cards, no further materials nor infrastructure are required. And unlike common customer loyalty programs of large enterprises [7], such cards guarantee customer privacy. The typical example for the application of paper-based loyalty cards is the independent coffee shop around the corner that offers a free drink for every 10 stamps collected. Customers using these cards cannot be tracked and profiled, and they can easily transfer their cards to someone else.

A common problem for loyalty points hunters is the number of stamp cards that accumulate over time. With mobile devices being widely available, the straightforward idea is to implement the functionality of paper-based loyalty cards as a mobile app. With special focus on small shops, such a system must first and foremost be light-weight. The cost of an electronic loyalty points solution should not be orders of magnitude larger than the paper-based system. This precludes solutions that are based on third-party-provided infrastructure or professional check-out systems known from large retailers. A likely solution scenario is that a vendor provides loyalty points with QR codes that are scanned by the customers' mobile devices.

In this paper, we consider the problem of designing a secure loyalty points protocol along the restrictions sketched above. This problem serves as a case

study for the applicability of *communication topologies*, a graphical approach to modeling security assumptions, to guide the design of secure protocols.

The protocols we design are simple and the steps we have taken seem self-evident in retrospect. On the one hand, the imposed infrastructural constraints naturally enforce simplicity, on the other hand, this makes loyalty points protocols a perfect case study for a detailed walk-through with our design methodology. In this spirit, we encourage the reader to pause the reading of the paper at the end of Section 2.1 and to design a secure loyalty points protocol satisfying the requirements stated in that section. The reader can then analyze his or her protocol with the same methods that we apply to our first protocol in Section 3.

We have formally verified two of the protocols we design in this paper and we give a brief account of the results in Section 4. To complete our story, we discuss implementation aspects of a practical loyalty card system in Section 5. We discuss related work in Section 6 and conclude in Section 7.

2 Preliminaries

We briefly state the security requirements that an electronic loyalty points protocol should satisfy. Then we introduce the communication topology, a model on which our methodology for secure protocol design is based. The definitions given in this section are purposefully informal.

2.1 Security requirements

A classical loyalty points system consists of a vendor that issues loyalty points to a customer commensurate with the customer's purchase. In the point-per-product-purchased loyalty card system known from the coffee shop at the train station the vendor issues a loyalty point by stamping a mark on a paper card for every coffee purchased. One mark is equivalent to one loyalty point, and the customer may redeem a certain number of loyalty points for a free coffee.

In the electronic loyalty points system we replace the stamp by a computer or mobile device, to which we will refer as *server*, and the paper card by a mobile device. The loyalty points are digital information. Thus, the electronic system consists of four agents: The customer, the vendor, the mobile device of the customer, and the shop's server.

An electronic loyalty points system should ideally satisfy all the security requirements that a paper-based system satisfies, among which we consider the following as important:

Unforgeability of points: Every loyalty point accepted by the vendor has been issued by the vendor.

No double-spending of points: A loyalty point that was previously redeemed will not be accepted by the vendor.

Customer anonymity: The vendor cannot link points issued to or redeemed by a customer to the customer's identity.

Customer privacy: The vendor cannot link a returning customer's transaction to the customer's previous transactions.

Theft protection of points: Points issued to an agent can be redeemed by this agent.

Non-repudiation by vendor: The vendor cannot repudiate the validity of an unredeemed loyalty point issued to a customer.

A paper-based loyalty points system satisfies the unforgeability, no double-spending and non-repudiation requirements, but it typically does not satisfy the theft-protection requirement, since a loyalty card can be stolen. Sometimes, when plain ink stamps from a retailer are being used, unforgeability of points requires the vendor to additionally sign each point manually. Customer anonymity is also guaranteed, unless the vendor knows the customer personally, but customer privacy may only hold to a certain degree. If the vendor provides each loyalty card with additional information, some limited profiling becomes possible. The vendor may for example use a date stamp in order to profile coffee consumption of anonymous individuals and must additionally provide each loyalty card with a unique serial number, if the information from different loyalty cards is to be linked to the same anonymous individual.

As with paper-based loyalty points systems, it can be argued that an electronic system may not satisfy the theft-protection requirement if the customer's mobile device is stolen. However, in the following we assume that the agent receiving loyalty points is the mobile device. In other words, we are not protecting against theft of the mobile device, but against the case where points issued to a customer's mobile device cannot be redeemed by that device. We note that there are two ways in which the theft-protection requirement could fail: (1) Points issued to a mobile device are redeemed by an attacker's device and (2) points issued to a mobile device are corrupted or lost and thus not redeemable by the device. We therefore refine theft-protection into two classical security requirements: a confidentiality requirement to prevent scenario (1) and authenticity of loyalty points issued by the vendor to prevent scenario (2). A term x (e.g., a loyalty point or cryptographic key) is said to be *confidential* (or secret), if the attacker does not know it. A term x received in a communication apparently from Y is said to be *authentic*, if Y indeed sent x. We will focus on these two requirements in the remainder of the paper. These requirements are formalized in our models of two loyalty points protocols discussed in Section 4.

2.2 Communication Topologies

A communication topology is a graph-theoretic model of communication protocol assumptions [2]. It contains assumptions on role capabilities, initial knowledge of roles, channel availability, and security assumptions on channels. A communication topology thus represents a set of protocols: All protocols that satisfy the stated assumptions. Given a communication topology τ, we may ask whether any of the protocols that satisfy the assumptions of τ also satisfy a given security requirement, e.g., one or more of the requirements stated in the preceding section.

Formally, a communication topology is an edge- and vertex-labeled directed graph (V, E, η, μ), where V is a set of role names, $E \subseteq V \times V$ and η and μ are functions assigning labels to vertices and edges respectively. For $A, B \in V$, an edge $(A, B) \in E$ denotes the availability of a communication channel from the agent executing role A to the agent executing role B. We call a sequence of vertices $[v_1, \ldots, v_{k+1}]$ with $v_1, \ldots, v_{k+1} \in V$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k$ a *path* from v_1 to v_{k+1}.

The vertex labeling function η assigns capability, knowledge, and trust assumptions to role names, i.e., to the vertices in the graph. The edge labeling function μ assigns security assumptions to communication channels. The communication channels defined in [2] are denoted by $\circ\!\!\rightarrow\!\!\circ$, $\bullet\!\!\rightarrow\!\!\circ$, $\circ\!\!\rightarrow\!\!\bullet$, $\bullet\!\!\rightarrow\!\!\bullet$ and represent, respectively, the insecure, authentic, confidential, and secure communication channel. An insecure channel is defined as a channel that the attacker can eavesdrop on, modify messages transmitted on it, and inject arbitrary messages into it. An authentic channel prevents modification of messages. More precisely, it guarantees to the recipient of a message that the message was previously sent by the sender. The attacker can still eavesdrop on an authentic channel. The confidential channel prevents the attacker from eavesdropping on messages, but allows the attacker to inject his own messages. The secure channel is defined to be an authentic and confidential channel. That is, the attacker can neither eavesdrop on nor modify messages.

Figure 1 shows the communication topology that we refer to as the *coffee shop topology* and work with in the remainder of this paper. It contains four nodes: the customer C, the vendor V, the customer's mobile device D, and the vendor's server S. All four nodes are assumed to be honest and initially share no private information. Customer and vendor are *human* roles, which is indicated by a dashed circle. Their capabilities are restricted in that they cannot perform any computations beyond concatenating and splitting messages. The mobile device and server have no such restrictions, as indicated by a solid circle.

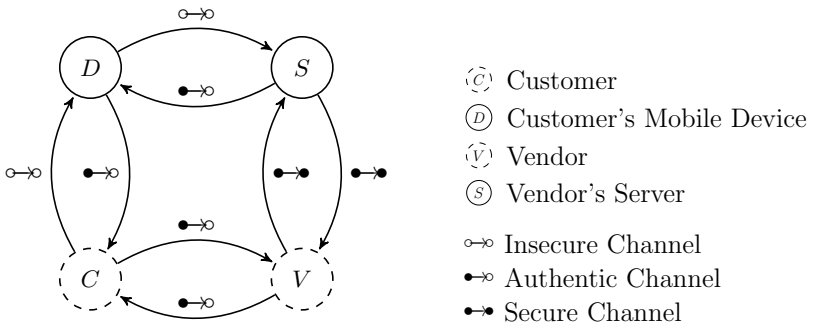

Fig. 1. The coffee shop topology.

The channel assumptions in the coffee shop topology are as follows. The communication channels between customer C and vendor V are authentic. This is justified by the fact that the customer and vendor are physically facing each other and thus able to attribute messages that they hear or paper notes that they receive to the correct source, e.g., the person in front of them. The channel from the mobile device D to the customer is assumed to be authentic, based on the assumption that the customer recognizes his own device. We assume that the channel from the customer to the device is insecure, because the customer might not be using an authentication mechanism on his device and might not always keep the device in his own possession. The channel from the device to the shop's server S is insecure, since any device could be messaging the server or eavesdrop on a communication. In particular, we assume that the server does not share any longterm secret keys with the device D, as this might clash with the privacy requirement. The channel from server to mobile device is assumed to be authentic, since the server's public key can be authentically distributed in the shop. It could be posted as a QR code on a wall that is only accessible by the vendor. However, if the reader is concerned about an unnoticeable replacement of the QR code by an attacker, we can always instruct the shop assistant to carry a shirt with an imprinted QR code. Finally, the communication channels between the server and the vendor are assumed to be secure, since this can be physically ensured.

Communication topologies can be given a semantics [2] that is aligned with the semantics of the Tamarin prover tool [12]. We have used the Tamarin tool to verify two of our protocols as discussed in Section 4.

3 Designing a simple loyalty card protocol

We start with a naive protocol and refine it in two steps with the help of the coffee shop topology into a protocol that satisfies the security requirements stated in Section 2.1. As we aim to design a scalable loyalty card system, see Section 5, we subsequently focus on the more comprehensive *point-per-euro-spent system* that incorporates the lighter variant of a *point-per-product-purchased system* from our coffee shop example, and which can also be used for retailers with a wider variety of goods.

3.1 First protocol

Consider a first electronic loyalty card protocol to issue loyalty points to a customer, shown in Figure 2. The protocol runs as follows. The customer pays the vendor a certain amount of money for a purchase. The vendor then enters the amount of money paid into the server S. The server returns a number of points that depends on the amount of money paid to the vendor (or the number of products sold). The separator / in message 3 of the protocol indicates that the server sends a message that encodes $points(money)$, but that the vendor (due to his computational restriction) is unable to parse this message and verify its

LP-1 1. $C \to V$: money
LP-1 2. $V \to S$: money
LP-1 3. $S \to V$: points(money) / QRcode
LP-1 4. $V \to C$: QRcode
LP-1 5. $C \to D$: QRcode / points(money)

Fig. 2. Protocol LP-1: A first protocol for issuing loyalty points

correctness. To the vendor, it is simply a QR code. The vendor gives the QR code to the customer. The customer then uses his mobile device to scan the code. The device can parse the scanned code, extract and verify the number of points obtained.

By the assumptions in our coffee shop topology, the channel from the vendor to the customer is authentic, but not confidential and the channel from the customer to his mobile device is an insecure channel. It follows that there are two opportunities for an attacker to observe the QR code: When the customer receives it and when it is scanned by the device. This is a problem if the QR code must be kept secret. For instance, if the information represented by the QR code is sufficient to redeem the encoded loyalty point then the protocol LP-1 has no theft protection.

Remark. We have observed an even simpler system in use in Switzerland: One loyalty point is awarded by the vendor per product sold and the *same QR code* is used for every transaction. The QR code is printed on a piece of cardboard that the vendor shows to the customer. The mobile device's app essentially counts the number of times it has scanned the QR code. This system offers theft protection to everyone: Since all loyalty points are represented by the same digital information and are redeemable, nobody's points can be stolen. However, the system does not satisfy the double-spending requirement. Instead of scanning the QR code with the system's official mobile app, an attacker can take a photograph and redeem the same point over and over again. Due to the absence of a secure protocol, these merchants try to counteract such attacks with various infrastructural and legal measures.

Returning to our protocol LP-1, our goal is to ensure the secrecy of the QR code in order to satisfy the theft protection requirement. We have two options to protect the secrecy of the QR code. The first option is to strengthen the assumptions made in the coffee shop topology. We must assume that (1) the channel from the vendor to the customer is a secure channel and (2) that the channel from the customer to his mobile device is confidential. The justification for (1) could be that the QR code is given to the customer in a concealed manner, e.g., on the counter-facing side of a paper. This assumption is not uncommon: Phone credit top-ups are in some countries sold as paper-printed codes. However, we have yet to see a vendor that takes precautionary steps to keep the printed code concealed. The justification for (2) is that the customer always scans the QR

code in a private environment. We believe, however, that this is too inconvenient to be carried out in practice.[4]

The second and preferred option is to improve the protocol.

3.2 Second protocol

The QR code of protocol LP-1 does not have to be secret if it is an encryption of the loyalty points or if the loyalty points that it encodes satisfy the theft-protection requirement by another mechanism. In both cases, the server must know information related to C or D to protect the loyalty points. We must therefore solve the following problem: How to send information authentically (or even securely) from D to S?

It is not possible to send information authentically (and hence neither securely) from D to S in the coffee shop topology using *only* the two edges (D, S) and (S, D) that directly connect D and S: The edge (D, S) is labeled as insecure and (S, D) as authentic, but in the wrong direction. This impossibility can be proved formally [2, Lemma 2].

It is, however, possible to send information authentically (but not confidentially) along the path $[D, C, V, S]$, because all edges along this path are labeled as authentic or secure and C and V are honest. We thus have an authentic channel from D to S along this path and an authentic channel from S to D by the edge (S, D).

We can therefore improve upon our first protocol as follows. The mobile device creates a *points code* (an ephemeral public key) that is sent authentically to the server. The server uses this code to encrypt the loyalty points. The protocol is shown in Figure 3. We denote the encryption of a message m with the public key k by $\{m\}_k$. However, in spite of the encryption, the protocol does not pro-

LP-2 1.	$C \to D$:	GetPoints
LP-2 2.	$D \to C$:	PointsCode
LP-2 3.	$C \to V$:	money, PointsCode
LP-2 4.	$V \to S$:	money, PointsCode
LP-2 5.	$S \to V$:	$\{\text{points(money)}\}_{\text{PointsCode}}$ / QRcode
LP-2 6.	$V \to C$:	QRcode
LP-2 7.	$C \to D$:	QRcode / $\{\text{points(money)}\}_{\text{PointsCode}}$

Fig. 3. Protocol LP-2: Improved protocol for issuing loyalty points

tect against theft. The channel from the customer to the device is insecure. An attacker can therefore replace the message from customer to device by a different message, e.g., by a redeemed loyalty point encrypted (by the attacker) under the

[4] At least one of this paper's authors admits to typing phone credit top-up codes into his mobile phone in public.

device's points code. We must, however, admit that this scenario stretches the limits of our imagination. We thus have again two options: Change the channel assumption or improve the protocol. We again choose the latter.

3.3 The third protocol: A simple loyalty card protocol

Protocol LP-2 fails to satisfy a security property because the path $[S, V, C, D]$ does not provide an authentic channel from S to D. (This is because the final edge of the path (C, D) is insecure.) There is still, however, the *direct* authentic channel from S to D and the protocol can be easily modified to take advantage of this channel, as shown in Figure 4.

SLP 1.	$C \to D$:	GetPoints
SLP 2.	$D \to C$:	PointsCode
SLP 3.	$C \to V$:	money, PointsCode
SLP 4.	$V \to S$:	money, PointsCode
SLP 5.	$S \to D$:	$\{\text{points(money)}\}_{\text{PointsCode}}$

Fig. 4. Protocol SLP: Further improved protocol for issuing loyalty points

We have found a protocol that may plausibly satisfy the theft protection requirement. Furthermore, if the function that generates points based on the amount of the purchase is chosen appropriately, the protocol can satisfy the unforgeability requirement and, if the server keeps track of all points that were generated, the double-spending requirement can be satisfied. However, the privacy and non-repudiation requirements are not satisfied. For non-repudiation the vendor must commit to their validity by signing them for instance. In protocol SLP, no signatures are specified.

Message 5 in protocol SLP is assumed to be sent over an authentic channel from S to D. The security assumption on this channel is different from the other channel assumptions in that we have based on a cryptographic assumption: "The channel is assumed to be authentic, because the server's public key can be authentically distributed in the shop." That is, to realize the authenticity property of this channel, the server must digitally sign message 5 and the device must verify the signature with the (authentic) public key that it has received. The signature on message 5 can be used towards the non-repudiation requirement. We note that in such a case a customer would have to rely on the legal system recognizing digital signatures as a non-repudiation mechanism. A likely precondition for this is that the public key is certified for such a use and the customer would need to verify the certificate. We consider this issue to be outside of the scope of the protocol specification and accept a server's signature that is verifiable with the authentically distributed public key to be a sufficient non-repudiation token.

The protocol does not satisfy the privacy requirement, because the vendor can link the issued points when they are collectively redeemed to the shopping baskets and points in time that they were issued.

3.4 An ecash-based loyalty card protocol

We now present our final improvement on the series of protocols. To provide customer privacy, we import a solution from the digital cash domain, which is a protocol based on blind signatures. In the loyalty card version of the digital cash protocol, the roles of the mint, bank, and merchant are combined in the shop's server. To keep the customer's different transactions unlinkable, the device chooses the serial numbers of coins and the server issues a blind signature on it. To prevent cheating, the device and server need to run a cut-and-choose protocol. Such protocols are standard (we discuss a specific example in Section 5) and we can consider them a simple building block. What we therefore need to ensure is that the server and device can establish a secure channel in the coffee shop topology. The secure key establishment phase is shown in Figure 5. The protocol runs as follows. When the mobile device receives the GetPoints

PP 1.	$C \to D$:	GetPoints
PP 2.	$D \to C$:	pk(eskD) / code
PP 3.	$C \to V$:	money, code
PP 4.	$V \to S$:	money, code / money, pk(eskD)
PP 5.	$S \to D$:	sign($\{\text{SessKey}\}_{\text{pk(eskD)}}, sk(\text{S})$)
PP $\vdots$	$D \to S$:	...
PP $\vdots$	$S \to D$:	...
PP n.	$D \to C$:	number of points received

Fig. 5. Protocol PP: key setup for PrivatePoints

instruction from the customer, it generates an ephemeral private key (eskD) and displays the corresponding public key, pk(eskD), to the customer. The customer pays the vendor and shows the mobile device's display to the vendor. The vendor inputs the transaction amount into the server and scans the code displayed on the customer's mobile device with the server. The server has thus received the device's public key. The server generates a session key SessKey, encrypts it with the device's public key, signs it with its own private key, and transmits the message to the device. This transmission could be done via NFC, BLE or WiFi. The server and device can now run any protocol over the secure channel that they have just established. At the end of this protocol the device displays to the customer the number of points received.

4 Security analysis

We have verified[5] authenticity and secrecy of the session key 'SessKey' of the PP protocol with the Tamarin tool [12]. The verification considers an unbounded number of sessions and assumes that there are compromised agents in the system. Since the session key is used to provide a secure channel between the server and the device, the remaining security requirements for the PrivatePoints system follow from the security properties of the e-cash sub-protocol.

The protocol that was verified is given in Figure 6, which makes explicit what initial knowledge assumptions are made and which terms are assumed to be randomly generated. The former is denoted by the "knows" keyword and the latter by the "fresh" keyword in the figure.

	C knows:	D, V
	D knows:	$C, S, \mathrm{pk}(S)$
	V knows:	S
	S knows:	$V, \mathrm{sk}(S)$
PP 1.	$C \to D$:	GetPoints
PP 2.	$D \to C$:	fresh(eskD). pk(eskD) / code
PP 3.	$C \to V$:	money, code
PP 4.	$V \to S$:	money, code / money, pk(eskD)
PP 5.	$S \to D$:	fresh(SessKey). $\mathrm{sign}(\{\mathrm{SessKey}\}_{\mathrm{pk(eskD)}}, \mathrm{sk}(S))$

Fig. 6. Protocol PP: Specification for the SessionKey exchange for PrivatePoints

Note that the same analysis shows that our simple loyalty points protocol SLP (Section 3.3) satisfies secrecy and authenticity of loyalty points. In SLP, the randomly generated number is not used as a session key, but rather as an identifier for the issued loyalty points. The protocol to *redeem points* in the simple loyalty points system is nearly identical to the protocol PP in Figure 6. The main difference is an additional sixth message in which the points to be redeemed are communicated from the device to the server. Its specification is shown in Figure 7. For this protocol, we have verified the secrecy and authenticity of the points transmitted from the device to the server.

5 Towards a practical loyalty points system

Design and formal verification of a security protocol is one part of the story; equally important, though, are aspects and protocol features that directly impact implementation in a real-world setting. We therefore discuss our experience in implementing a prototype of a loyalty points system that we call *Private Points* [10].

[5] The Tamarin specification files are available at `http://www.infsec.ethz.ch/research/projects/hisp.html`.

	C knows:	D, V
	D knows:	$C, S, \mathrm{pk}(S), points$
	V knows:	S
	S knows:	$V, \mathrm{sk}(S), points$
rSLP 1.	$C \to D$:	SpendPoints
rSLP 2.	$D \to C$:	fresh(eskD). pk(eskD) / code
rSLP 3.	$C \to V$:	redeem, code
rSLP 4.	$V \to S$:	redeem, code / redeem, pk(eskD)
rSLP 5.	$S \to D$:	fresh(SessKey). $\mathrm{sign}(\{\mathrm{SessKey}\}_{\mathrm{pk(eskD)}}, \mathrm{sk}(S))$
rSLP 6.	$D \to S$:	$\{\mathrm{points}\}_{\mathrm{SessKey}}$

Fig. 7. Protocol rSLP: Specification for redeeming loyalty points.

5.1 Private Points

We first discuss briefly how a particular digital cash solution has been used to issue and redeem points and afterwards report on implementation aspects.

Issuing loyalty points The sub-protocol to issue loyalty points we have chosen follows essentially the ecash protocol of Schoenmakers [15], with a few simplifying changes.

1. The server communicates to the device how many points will be issued.
2. For every coin C_i to be generated by the server, the device generates a secret serial number x_i.
3. The serial numbers are hashed, blinded, and transmitted to be blindly signed by the server.
4. The device verifies the signatures for all coins received.

Redeeming loyalty points As the PrivatePoints system is limited to the roles of customer, vendor, mobile device, and server, it is the server that needs to play the role of the bank (see the original ecash protocol [15] for the role specification). Most importantly, the server needs to check the validity of loyalty points and prevent double spending, as well as guarantee non-repudiation to the user as discussed above.

The following is a high-level view of the protocol.

1. The customer selects a number of loyalty points to redeem for an item.
2. All the selected points C_i are transmitted by the mobile device to the server, each with the hash of the secret serial number $h(x_i)$ in order for the server to verify the loyalty point signature.
3. The server verifies the points received by checking the signature and verifying that the points have not been previously spent. Note that if the shop maliciously claimed that a valid coin has been spent already, the user could ask for the coin number as proof of the claim. Since the shop is only in possession of the *hashed* coin number at this point of the payment process, it

is unable to uphold the claim. This non-repudiation property for individual coins is an advantage over the ecash system, where shops and banks could collude against users in rejecting valid coins.
4. The server sends a signature to the device confirming that the received points are valid for this transaction. This step is crucial for the non-repudiation requirement.
5. The device verifies the server's signature. If the signature is valid, the device sends the secret serial numbers x_i.
6. The server verifies that the serial numbers produce the previously received hashes.

5.2 Implementation

A small coffee shop around the corner could maybe still invest in some reasonably priced infrastructure, but the ice-cream man in a football stadium carrying a vendor's tray can definitely not. Consequently, the only infrastructure we can assume for customers and shops are plain mobile devices exchanging loyalty points over a near-field communication link, for example. PrivatePoints has been designed especially with this scenario in mind.

Also, the protocol must be efficient enough to be used in the train station's coffee shop during rush hour and allow for collecting multiple points in one go in case people treat each other for coffee, or if the point-per-euro-spent system is implemented. As a rule of thumb, our industry partners give a limit of 2 seconds that can be invested on issuing loyalty points during the payment process. With transmission and protocol overhead subtracted, cryptographic operations must therefore not take more than 800ms on off-the-shelf mobile phones. In case of PrivatePoints, issuing digital loyalty points consists in hashing a serial number, providing and verifying a blinded signature. Our implementation on a SAMSUNG GT-I9100 mobile phone with the Android operating system using SHA-256 and RSA-2048 produces one coin every 40ms on average, thus 20 coins in 800ms.

Especially for small shops, loyalty cards systems must not require Internet connection as these shops are usually located in places with bad or slow connection. During rush hour it is unacceptable to invest 5 seconds per customer to set up Internet connection during the payment process. From a business point of view, this probably is the most important feature of PrivatePoints that is not related to security.

Finally, PrivatePoints does not require any online registration prior to the first use of the mobile app that would spoil all the security measures taken to ensure customer anonymity.

6 Related Work

E-cash was invented and subsequently commercialized by David Chaum [5]. Since about the mid-1990's, the topic of electronic cash, payment systems, their properties, and technical foundations is extensively covered in the literature, e.g., [15,

13, 14, 1, 16, 17]. One might argue that loyalty points merely correspond to some kind of non-universal, virtual currency such that one of these protocols could directly be used. However, there is one fundamental difference between loyalty systems and virtual currencies. A currency is issued by a bank or mint that can act as a trusted-third-party in protocols between customers and vendors. In case of a loyalty points system, bank and vendor conglomerate to a single party, which breaks the trust relation between customer and bank. PrivatePoints is to a great extent based on the ecash protocol of Schoenmakers [15] without using such trust relations. In addition, a bank issuing a universal currency does not underly the same infrastructural constraints as our coffee shop.

A recently proposed loyalty points protocol is given in [3]. It has a special focus on customer privacy in that it allows customers to build and reveal their own generalized profiles from their purchase history with the idea to award more loyalty points for more precise customer profiles. Customers therefore control their own degree of privacy. This protocol requires bilinear pairing based cryptography to implement its flexible customer privacy features, whereas PrivatePoints offers only basic customer privacy protection in return for a greatly reduced complexity of the system and cryptographic primitives used. Also, this protocol is targeted to larger online and offline shops with a global taxonomy of products and makes explicit use of a certification agency.

Electronic customer loyalty systems are also related to coupon and voucher systems. The coupon systems most relevant to our work are [4, 6, 9, 8]; the multi-coupon system described in [4] has the closest resemblance to the PrivatePoints protocol. The major difference lies in the use of cryptographic tools in that [4] uses proofs of knowledge, while PrivatePoints employs digital signature and commitment schemes. Also, the implementation requirements for loyalty points systems are much stronger compared to voucher systems, since e.g. issuing loyalty points is an integral part of every single transaction.

Communication topologies have been introduced in the context of secure human-server communication [2]. They were used to classify all four-node topologies that consist of a human, a device, a corrupted computing platform belonging to the human, and a remote server. The classification distinguishes between topologies that have secure communication protocols and those for which provably no such protocols exist. The channel notation $\circ\!\!\rightarrow\!\!\circ$, $\bullet\!\!\rightarrow\!\!\circ$, $\circ\!\!\rightarrow\!\!\bullet$, $\bullet\!\!\rightarrow\!\!\bullet$ was introduced by Maurer and Schmid [11] and used to define transformation rules for secure channel establishment with cryptographic primitives.

7 Conclusion and Future Work

We have illustrated the use of *communication topologies* to guide the design of security protocols. A communication topology is a graphical tool to represent assumptions about the environment that a protocol runs in. This guided approach to designing protocols does not guarantee secure protocols. For such guarantees, pen and paper proofs or automated verification tools are still required. Nevertheless, our approach helps in reducing the search space and can be used to

sketch security protocol designs without the need for a deep understanding of the intricate details of formal security specifications.

An ulterior motive for our work is the question how secure protocols could be designed automatically. We envision the communication topologies to be one of the inputs to the tool that a user can easily specify in a graphical environment. The other input are the security requirements that are selected from a list. An automatic tool's first step is to find possible protocol flows in a similar manner as we have found manually in Section 3. The second step is to refine the protocol flows heuristically or interactively into protocol specifications that are in turn analyzed with a theorem prover or model checking tool.

The security protocol exemplarily designed in this paper using communication topologies mimics a digital version of a paper-based customer loyalty program specifically tailored for being used in small shops with no professional or third-party-provided infrastructure available. In fact, we showed that our protocol could even be used by the ice-cream man in a football stadium carrying a vendor's tray and using his private mobile phone for issuing and redemption of loyalty points. The protocol offers the same security features as its paper-based ancestor such as unforgeability of points, double-spending protection, theft protection of points, non-repudiation by the vendor, customer anonymity and even a similar degree of customer privacy. The protocol is offline, i.e., without need for Internet connection, and scalable enough for supporting point-per-product-purchased as well as point-per-euro-spent loyalty programs.

Concerning future work, we will investigate to which extent such a simple and light-weight loyalty point system can support collaborating shops and franchising companies that expect loyalty points issued in one shop to be redeemable in other shops. Especially in franchising companies, the individual shops may be competitors, which puts the straightforward idea of key sharing between shops into question.

References

1. ECB (European Central Bank). Virtual currency schemes, October 2012.
2. David Basin, Saša Radomirović, and Michael Schläpfer. A Complete Characterization of Secure Human-Server Communication. In *28th IEEE Computer Security Foundations Symposium (CSF 2015)*. IEEE Computer Society, 2015. To appear. Full version accessible at `http://www.infsec.ethz.ch/research/projects/hisp.html`.
3. Alberto Blanco-Justicia and Josep Domingo-Ferrer. Privacy-preserving loyalty programs. *CoRR*, abs/1411.3961, 2014.
4. Sébastien Canard, Aline Gouget, and Emeline Hufschmitt. A handy multi-coupon system. In *Applied Cryptography and Network Security*, pages 66–81. Springer, 2006.
5. David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.
6. Liqun Chen, Matthias Enzmann, Ahmad-Reza Sadeghi, Markus Schneider, and Michael Steiner. A privacy-protecting coupon system. In *Financial Cryptography and Data Security*, pages 93–108. Springer, 2005.

7. Chad Cumby, Andrew Fano, Rayid Ghani, and Marko Krema. Predicting customer shopping lists from point-of-sale purchase data. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 402–409. ACM, 2004.
8. Sandra Dominikus and Manfred Aigner. mcoupons: An application for near field communication (nfc). In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 2, pages 421–428. IEEE, 2007.
9. Matthias Enzmann, Marc Fischlin, and Markus Schneider. A privacy-friendly loyalty system based on discrete logarithms over elliptic curves. In *Financial Cryptography*, pages 24–38. Springer, 2004.
10. Cuennet L. Security protocols for loyalty card systems on mobile devices. Master's thesis, University Of Fribourg, 2014.
11. U. Maurer and P. Schmid. A calculus for secure channel establishment in open networks. In Dieter Gollmann, editor, *Computer Security – ESORICS 94*, volume 875, pages 173–192. Springer, 1994.
12. Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN prover for the symbolic analysis of security protocols. In Natasha Sharygina and Helmut Veith, editors, *25th International Conference on Computer Aided Verification (CAV 2013)*, volume 8044 of *LNCS*, pages 696–701, Saint Petersburg, Russia, July 2013. Springer.
13. Birgit Pfitzmann and Michael Waidner. Properties of payment systems: General definiton scetch and classification, 1996.
14. Ahmad-Reza Sadeghi and Markus Schneider. Electronic payment systems. In *Digital Rights Management*, pages 113–137. Springer, 2003.
15. Berry Schoenmakers. Basic security of the ecash payment system. In B.Preneel & V. Rijmen, editor, *Computer Security and Industrial Cryptography: State of the Art and Evolution*, pages 342–356. Springer Verlag, 1998.
16. Simon Sprankel. Technical basis of digital currencies, 2013.
17. Sebastiaan von Solms and David Naccache. On blind signatures and perfect crimes. *Computers & Security*, 11:581–583, 1992.