Transfer Project T2: Practical Cryptographic Techniques for Secure and Privacy-Preserving Customer Loyalty Systems

Johannes Blömer[1], Jan Bobolz[2], Fabian Eidens[1], Tibor Jager[3], Paul Kramer[1]

1 Department of Computer Science, Paderborn University, Paderborn, Germany
2 School of Informatics, University of Edinburgh, Scotland, UK [a]
3 School of Electrical, Information and Media Engineering, University of Wuppertal, Wuppertal, Germany

[a]Work done while at Paderborn University

1 Introduction

In this transfer project, we designed, implemented, and evaluated a cryptographically secure and privacy-preserving incentive system for retail stores. An incentive system is a system in which customers can participate in promotions to obtain rewards or discounts. They can be deployed to influence the shopping behavior of customers, for example, to achieve a higher customer loyalty rate or increase sales. Further, incentive systems are used as a tool for collecting customer data that serves for instance marketing purposes. Popular incentive systems, e.g., German Payback and American Express Membership Rewards, focus on the latter data-driven business model. Therefore, they are neither privacy-preserving nor do they follow a rule to minimize data collection. They essentially possess a complete record of all members' shopping history in their databases. An infamous example of what can happen if this customer data is not protected properly occurred at the loyalty program of the retailer Target, who in 2012 exposed a girl's pregnancy to her father with coupons sent by Target.[27] Not only individual incidents but also the GDPR as an effort of the EU to protect customer data motivate the need for a privacy-focused alternative.

In this transfer project with Diebold Nixdorf, we designed an incentive system that fills this gap. We visualize the main differences to "classical" incentive systems in Figure 75. The main idea is to replace the large central database that contains the users' shopping records with a token stored on the user's smartphone. Thereby, we drastically reduce the data that can be linked to a user. This token holds the state required for the business logic and is typically a point count computed from the shopping history. We use cryptographic

bloemer@upb.de (Johannes Blömer), jan.bobolz@ed.ac.uk (Jan Bobolz), fabian.eidens@uni-paderborn.de (Fabian Eidens), tibor.jager@uni-wuppertal.de (Tibor Jager), paul.kramer@upb.de (Paul Kramer)

[27]https://www.nytimes.com/2012/02/19/magazine/shopping-habits.html

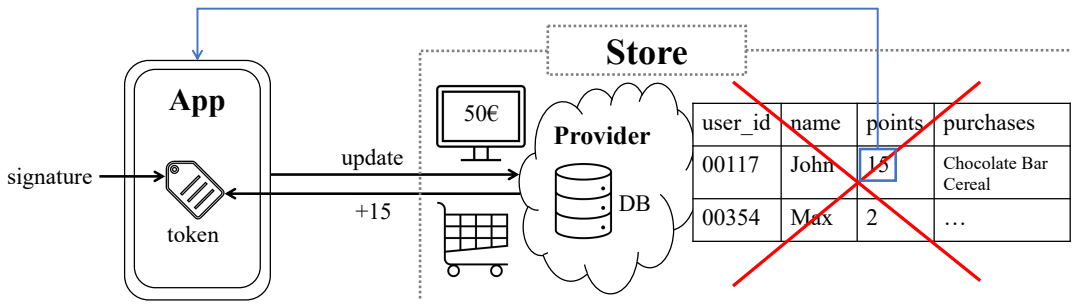

Figure 75: *Main differences between our incentive system and "classical" incentive systems. The shopping records with point count in the provider's database are replaced with signed token in app.*

protocols to change the point count of the token when a user wants to collect points for their current basket or trade points for some reward. These protocols, among other properties, ensure privacy and prevent *double-spending attacks*. Double-spending is the canonical attack on token-based systems, in which an attacker copies a digital token and redeems it multiple times.

In addition to designing the incentive system, we implemented a demonstrator and further adjusted the system to address real-world cryptographic challenges. The in-store experience with the demonstrator of the incentive system is as follows: A user goes shopping using their phone and the store's app installed (which includes the incentive system client). In the store, the user scans barcodes of items with their phone, which are then added to a digital basket. In the app, the user can see running promotions that will give them benefits, e.g., buying four chocolate bars and getting one free. The checkout process is also handled in the app and automatically handles the promotion benefits for the user, e.g., checks if the basket contains four chocolate bars and adds a discount for one of them. In our incentive system, these promotions are not limited to a single store visit. Rather, a user can, for example, buy two chocolate bars now and two next time. The incentive system stores this state in a secure token on the user's device.

We started the project with a study of existing cryptographic incentive systems from the literature [MDPD15; JR16; HHNR17] that introduced the concept to let users store their points in an authenticated form. The techniques we developed in Subproject *C1 Robustness and Security* for anonymous credentials and their implementations turned out to be well suited for this, solve open problems, and compensate for some of the downsides of the existing systems. Furthermore, we discussed real-world store infrastructure and details of the currently used hardware with the project partner. We identified the following requirements for a privacy-preserving incentive system.

- Anonymity: Providers are unable to link earn/spend transactions to users. In practice, this protects users from having their shopping history linked to their identity and point values.
- Soundness: Users cannot spend more points than they have earned.
- Online double-spending protection: Given continuous access to a central database, the provider can immediately detect double-spending.
- Offline double-spending protection: Providers can even detect double-spending

in one store without continuous access to a central database. Double-spending transactions can be detected and the perpetrating user can be identified. Losses incurred by double-spending can be reclaimed from that user.

- Partial spending: Users can choose how many points they spend in one transaction.
- Efficiency: The process of earning and spending points can be run on a consumer phone and existing store infrastructure, i.e., the privacy-preserving incentive system is ready for real-world applications.

Our privacy-preserving incentive systems [BBDE19; BEK+20] fulfill all of the goals. Previous systems from the literature do not have offline double-spending protection, they do not support partial spending or, if they support partial spending, the combination with offline double-spending protection is not securely realized. In the following, we present the incentive systems in more detail, how we adapt them in this project, and discuss highlights of the project results. Furthermore, we give important lessons that we learned in the process from theory to practice.

2 Main Contribution

The main privacy issue of current incentive systems in practice is that the incentive system provider stores user data (e.g., each user's point count) in a central database. This architecture practically forces participating users to reveal their identities in order for the correct entry to be updated in the provider's database. Unfortunately, this allows the provider to link purchases to the user's identity, enabling the creation of detailed user profiles. Our main idea to remedy this, as sketched in Figure 75, is to store personal user data (name, current point value) in a cryptographically authenticated token on the user device instead of a central database. We run privacy-preserving protocols that can update the user's data without ever revealing the data or the user's identity to the incentive system provider. For example, we can have a user with k points update their point count by +15. After that procedure, the user will hold a token that certifies $k + 15$ points, while the provider does not learn the old value k, the new value $k + 15$, or any other user data. The provider is only aware that it increments some hidden authenticated value k by +15.

The first iteration of our incentive system [BBDE19] is built from updatable anonymous credentials (UAC), which indeed store authenticated data ("attributes") within an authenticated token ("credential") that can be updated in a privacy-preserving way (see Subproject *C1 Robustness and Security* on page 145). The main idea of an update operation [BBDE19] is that the user first computes the update (e.g., +15) locally, sends this updated data in hidden form (in a cryptographic commitment) to the provider, and proves, with a zero-knowledge proof of knowledge, that the hidden data is a valid update to the user's old authenticated data. The provider then blindly authenticates the updated data.

One of the main challenges when building cryptographic incentive systems is how to handle *double-spending*. Say a user holds a token with 20 points. Of course, the user and provider can run the update protocol for the user to receive an updated token with $20 - 5 = 15$ points. However, we need a mechanism to prevent the user from using the old 20 point token *again* (which would be considered *double-spending*). Note that detecting double-spending is made difficult by our strong privacy aspirations: the act of spending a

token must not reveal any information about the token, hence the token itself is generally hidden. As a consequence, using the same token twice is, by default, not detectable by the provider. One typical way to solve the double-spending issue is *online* double-spending detection. There, every token is assigned a random ID. To invalidate a token (e.g., in the -5 scenario above), the user reveals the token's random ID. The provider checks whether that ID has already been invalidated and, if so, whether the user is trying to double-spend the token. To ensure privacy, the random ID is chosen by the user and only revealed to the provider when invalidating the token. This means that as long as the user does not double-spend, the provider only learns meaningless random numbers through this process. The downside to this approach is that it requires stores to have a consistent connection to a database containing all invalidated IDs. If a store loses the database connection, it cannot check whether a user is double-spending or legitimately spending a valid token. *Offline* double-spending protection (e.g., [HHNR17]) is an additional feature that mitigates this issue. It allows offline stores to speculatively accept a token without checking the ID. If it later turns out that the ID had already been spent (e.g., in another store), the offline double-spending protection feature guarantees that the double-spending user's identity can be revealed, allowing stores to identify misbehaving users and recoup any losses (rewards given erroneously) incurred by undetected double-spending. We combine both sorts of double-spending mitigation in [BBDE19].

Afterward, we identified that in real-world applications, the earning of points is the most frequent operation and should be further optimized. Typically, users have to earn points many times before they can spend them on a reward. In the follow-up paper [BEK+20], we present a new incentive system in which the protocol to earn points works *without* the relatively costly machinery of zero-knowledge proofs of knowledge. This is enabled by using structure-preserving signatures on equivalence classes (SPS-EQ) [FHS19] to authenticate the user data. SPS-EQ come with special randomization features that allow the provider to verify and modify a hidden version of the user's data. Additionally, we add desirable features such as (1) improved privacy for double-spending users (in [BBDE19], double-spending users would incidentally reveal their whole purchase history to the provider) and (2) support for retrying interrupted protocol executions without triggering double-spending protection. We use the ideas from [BEK+20] as the cryptographic basis of our system. However, we have significantly extended the construction and its infrastructure to match the requirements and desirable features of real-world stores.

2.1 Prototype

In the following, we describe how the in-store experience mentioned in the introduction is supported by the incentive system prototype developed in this project. Recall that a user interacts with the system using their smartphone and the app that we developed. At the first start, the app explains key features of the incentive system and then guides the user through a one-time registration process (Figure 76). The result is a join-token in the form of a digital signature on the user identity by the provider. By this, we can later guarantee for the store that the system can identify users to claim any losses, but only if they double-spend. We use the join-token in any interaction with the store in the incentive system to prove that the user is part of the system and therefore a valid user. The app then fetches any running promotions of the store. Promotions are an artifact that the store provider can configure. It

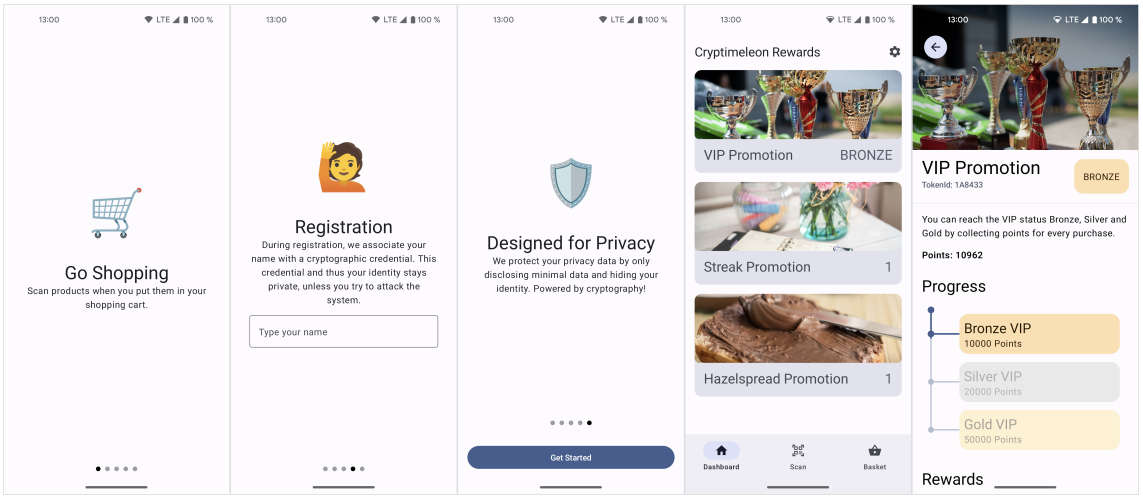

Figure 76: *1.-3. onboarding and registration screens in app. 4) dashboard with an overview of running promotions. 5) detail screen for VIP promotion.*

encapsulates the rules for how users can gain rewards. In the prototype, we support the following promotions with some examples of the rules and rewards:

- discount promotions with the rule "buy x many A" and the reward "get Y for free",
- VIP promotions with the rule "buy x currency worth of products (over multiple visits), become bronze ($x > c_1$), silver ($x > c_2$) or gold VIP ($x > c_3$)" with the reward that the user gets 2%, 5%, or 10% discount on future purchases, and
- streak promotions with the rule "go shopping at least once every 7 days" and the reward "get a free coffee after a streak of 2 weeks".

The user (and app) is now ready for the main part of the shopping, i.e., scanning products and putting them in the digital basket (see Figure 77). Here, the store is not involved and no data is revealed since the app has all the necessary data stored locally. When users are ready for the checkout they can do so directly in the app and leave the store. During the checkout process in the app, the app checks if any promotions can be updated following the rules of the promotions. For example, the user has previously bought 2 chocolate bars (as stored in the user's token), 1 is in the basket, and there is a discount promotion that gives 1 chocolate bar for free if the user has bought at least 3. Then the app interacts with the provider to get a privacy-preserving update on the promotion.

Let us describe the update process in detail. A user has a token for every promotion certifying the status of the promotion. These tokens are obtained by proving ownership of a join-token and initialized with a starting value of 0. A token is a Pedersen commitment [Ped91] on the user's status of the promotion, a cryptographic primitive that hides the data but allows proving statements on the data. In addition, a valid token must be signed with an SPS-EQ signature by the provider. This prevents users from generating valid tokens, i.e., "printing money".

For our example, if the user simply wants to collect 2 points for buying 2 chocolate bars, the app runs the *earn protocol* with the provider: The app sends the randomized token to the provider, who adds 2 points to the token and issues an SPS-EQ signature on the token with +2 points. The app de-randomizes this new token and stores the token worth +2

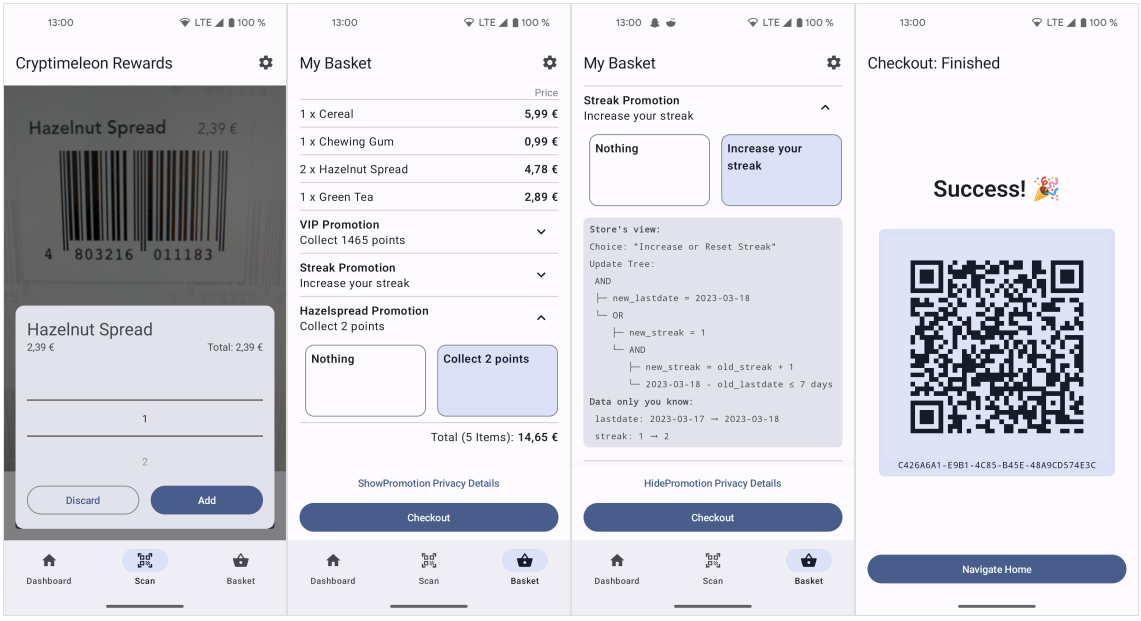

Figure 77: *Walkthrough of the shopping process: 1) Scanning products, 2) selecting updates for promotions, 3) details on privacy consequences of updates, 4) result screen with QR code as proof of payment and for claiming rewards.*

points. This earn protocol only uses the algebraic properties of the SPS-EQ signatures.

At some point, the user will have collected enough points to get some reward, and the app then runs the *spend protocol*. For example, we assume the user collected 5 points, whereas 4 points can be traded in for a free chocolate bar. For this, we utilize non-interactive zero-knowledge arguments of knowledge (NIZKs), namely Schnorr-style Sigma protocols [Sch90]. The app generates a remainder token holding the remaining amount of 1 point. Then, it sends the current token with 5 points and the remainder token with 1 points together with an NIZK that proves the following statements: 1) The old token holds at least 4 points (using a so-called range proof) and 2) The remainder token holds 4 points less than the original token. After verifying the proof, the provider is convinced that the new token has been correctly updated according to the rules of the promotion and signs the remainder token, which becomes the new valid token of the user. Additionally, the reward is added to the user's basket. To prevent double-spending attacks, the old token is invalidated through this procedure, which we explain later in more detail.

All these updates are privacy-preserving, meaning that updates cannot be linked to users, and the exact points counts of the tokens are kept secret. Only necessary data is shared with the provider. We achieve this by the randomization properties of SPS-EQ signatures, the hiding properties of Pedersen commitments, and the zero-knowledge properties of the NIZKs.

So far, we only considered the provider's side as one abstract instance, which corresponds to older versions of the incentive system [BBDE19; BEK+20]. However, it turned out that applying this system to *multiple* stores, e.g., all stores of a supermarket chain, is a non-trivial task: Simply deploying copies of the incentive system at each store is not desirable, because the provider's secret key would be shared among all stores, and hence one compromised store would break the whole system. For availability reasons, we must

keep some functionality in the stores such that users can pay their baskets and update tokens in case of network outages. Our new *multi-store infrastructure* fulfills both requirements (see Figure 78):

- There is exactly one *provider* in the cloud that holds the SPS-EQ secret key. Only the provider can issue SPS-EQ signatures and thus create and update tokens. The SPS-EQ secret key could be stored in a hardware security module (HSM).
- Every *store* has some standard digital signature key pair (ECDSA) that is trusted in the incentive system's public-key infrastructure. The store uses its secret key to authorize the provider to execute a token update.

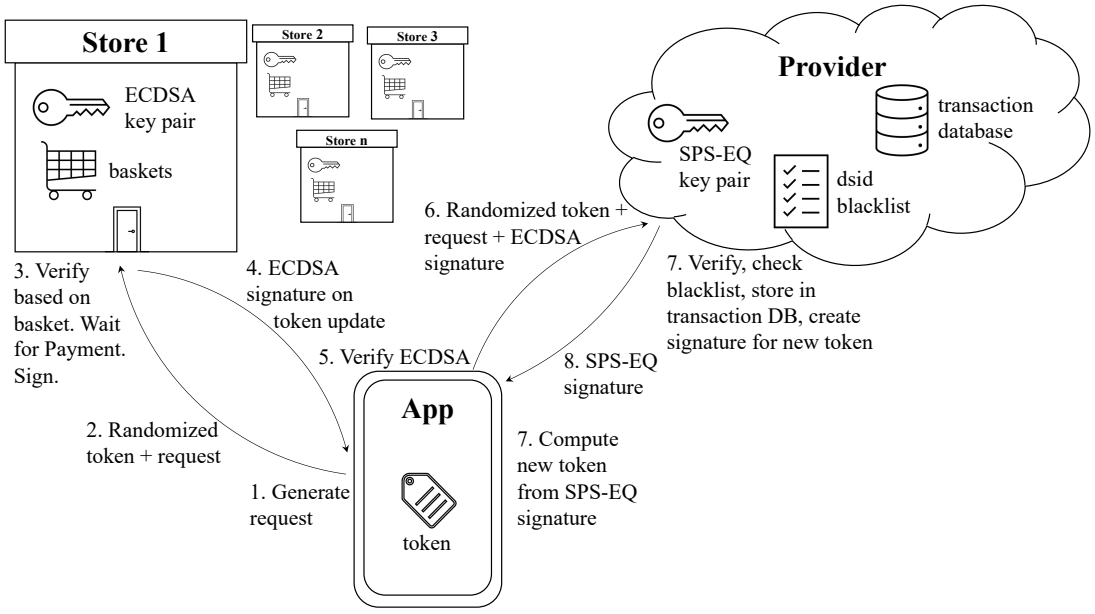

Figure 78: *Multi-store infrastructure.*

First, the user communicates with one of the stores. The store verifies the user's request based on the user's baskets and then issues an ECDSA signature to authorize a token update. For example, if a user wants to earn points for some basket, the store computes and signs the number of points to earn. This process functions in the store even if it is offline, and the user can send the signature to the provider later to obtain an updated token. Then, the provider again performs some verification, checks the store's signature, and then gives some SPS-EQ signature to the user. The user obtained a new token with a valid SPS-EQ signature.

The new multi-store infrastructure enables us to keep the transaction history of users that double-spend secret to the point of double-spending without needing the expensive and complex forward tracing technique from [BEK+20]. Furthermore, during the implementation and discussions with Diebold Nixdorf, we identified the problem of scaling the incentive system to multiple stores. Motivated by this, we introduced the multi-store infrastructure that solved both our scaling problems and enabled *clearing* (i.e., the ability to establish a pool of money that stores pay into when issuing points and can withdraw from when giving out rewards). To summarize, implementing the incentive system made real-world cryptographic challenges visible and lead to improving several aspects of the incentive system.

2.2 Implementation and the Role of Cryptimeleon

The implementation of the incentive system is provided under the open-source MIT license on GitHub.[28] It is powered by the Cryptimeleon project[29] developed at Paderborn University, a collection of cryptography-prototyping libraries written in Java. They provide all necessary primitives, for instance, basic math structures, SPS-EQ signatures, and NIZKs. Further, they natively support the MCL[30] library that ships highly optimized bilinear curves to all relevant architectures. The Android app is implemented in Kotlin with the framework Jetpack Compose. The web services use Java and Spring Boot and are deployed via Docker and Docker Compose; our web app is written in Vue.js.

While there is still room for optimizations and speedups, all implemented protocols run in well under a second and thus are more than practical. We summarize our benchmark results in Table 1.

	Time (ms)						**Size** (KB)				
Protocol	A*	S†	A*	P†	A*	Total	A→S	S→A	A→P	P→A	Total
Registration	0.0	0.5	0.7	1.9	10.0	13.0	0.4	0.9	1.0	0.8	3.1
Join			23.3	3.2	16.8	43.3			2.1	0.9	2.9
Earn	6.0	0.4	4.1	3.7	14.2	28.4	0.5	0.8	1.9	0.9	4.0
Spend	47.7	21.4	0.5	24.0	15.1	108.7	12.0	1.0	12.3	1.0	26.4

Table 1: *Benchmarks for simple point-collection promotion. Times averaged over* 1000 *runs on Google Pixel 5 *and M1 Macbook Pro †. Message sizes were recorded with Wireshark. Abbreviations: app (A), store (S), provider (P).*

3 Impact and Outlook

Right now, privacy is under attack by surveillance capitalism: Corporations collect data about people on a large scale, analyze it, create comprehensive user profiles, and then use those profiles for profit, usually via targeted advertisement. This data collection is already pervasive online. Additionally, digital incentive systems bring data collection to even more areas of everyday life, namely offline shopping.

Even if the provider of a traditional incentive system has good intentions, users have no agency over the data they hand over each time they interact with the incentive system. It is entirely possible, perhaps even likely, that the collected data will eventually be hacked or leaked, or that the provider changes its strategy and starts abusing the collected data.

Privacy-preserving incentive systems, such as the one we have developed, play a crucial role in solving the privacy issues of traditional incentive systems: Users gain cryptographic guarantees that their data *cannot* be collected by the provider. From the point of view of the provider, a privacy-preserving incentive system still offers all the (non-invasive)

[28] `https://github.com/cryptimeleon/incentive-system`
[29] `https://cryptimeleon.org`
[30] `https://github.com/herumi/mcl`

benefits for participating stores, giving them a way to reward loyalty, gamify shopping through points, or incentivize the purchase of certain products.

Unfortunately, right now, there seems to be insufficient incentive for stores to deploy a privacy-preserving system instead of one that collects as much data as possible. While users would unequivocally prefer a privacy-preserving solution, in practice, sufficiently many users are ignorant or apathetic towards privacy and are willing to participate in systems without any expectation of privacy. Providers, of course, prefer collecting data over not collecting data. There are two possible ways out of this situation. First, we (as a society) can educate users about the dangers of the unmitigated collection of personal data. If sufficiently many users demand privacy, privacy-preserving systems will gain traction. Second, we can prescribe a privacy-preserving system through law. The GDPR has been incredibly impactful regarding data collection and user consent. In the same vein, mandating the use of privacy-preserving systems could be a highly effective consumer protection law. Projects such as ours take the first important step towards this, proving that such systems can be built with reasonable effort. This is crucial information for lawmakers, who would be understandably hesitant to outlaw systems with no viable alternative.

Projects such as this one are also valuable because they bring together practical aspects and academia. Often, academia only supplies the very first step for building new systems: the very basic ideas. These ideas are motivated and illustrated by idealized scenarios and aim to answer fundamental questions rather than supply a comprehensive blueprint for building concrete systems. As a second step, it is then on the industry to take the academic answers and refine them into a real system. As evidenced by this project, it is sometimes fruitful to involve academia in the second step. This allows the academic side to revisit their idealized assumptions (using insights from the industry) and to improve upon their answers. For example, our new infrastructure with support for multiple stores is a direct result of requirements from our industry partner, improving and even simplifying the system we have built based on idealized assumptions. It is also important to test and benchmark constructions in software for demonstration. Not only does the resulting software serve as a demonstrator, showing that building such systems is realistic, but the process of implementing the system also forces one to consider crucial details previously ignored. Those may even represent future research opportunities or collaboration with industry partners on related topics.

While our system proves that a privacy-preserving approach to incentive systems is viable, there are some open research questions to consider in the future: (1) How we can guard our system against adversaries with access to a hypothetical large quantum computer? (2) What are further applications for the techniques we used for incentive systems (e.g., electronic cash)? (3) How can we make the public more cognizant of the privacy issues of currently deployed systems, and of the alternatives enabled by modern cryptography?

Bibliography

[BBDE19] Blömer, J.; Bobolz, J.; Diemert, D.; Eidens, F.: Updatable Anonymous Credentials and Applications to Incentive Systems. In: *ACM CCS 2019: 26th Conference on Computer and Communications Security*. ACM Press, 2019, pp. 1671–1685

[BEK+20] Bobolz, J.; Eidens, F.; Krenn, S.; Slamanig, D.; Striecks, C.: Privacy-Preserving Incentive Systems with Highly Efficient Point-Collection. In: *ASIACCS 20: 15th ACM Symposium on Information, Computer and Communications Security*. Ed. by Sun, H.-M.; Shieh, S.-P.; Gu, G.; Ateniese, G. ACM Press, Oct. 2020, pp. 319–333

[FHS19] Fuchsbauer, G.; Hanser, C.; Slamanig, D.: Structure-Preserving Signatures on Equivalence Classes and Constant-Size Anonymous Credentials. In: *Journal of Cryptology* 32 (Apr. 2019), no. 2, pp. 498–546

[HHNR17] Hartung, G.; Hoffmann, M.; Nagel, M.; Rupp, A.: BBA+: Improving the Security and Applicability of Privacy-Preserving Point Collection. In: *ACM CCS 2017: 24th Conference on Computer and Communications Security*. Ed. by Thuraisingham, B. M.; Evans, D.; Malkin, T.; Xu, D. ACM Press, Oct. 2017, pp. 1925–1942

[JR16] Jager, T.; Rupp, A.: Black-Box Accumulation: Collecting Incentives in a Privacy-Preserving Way. In: *Proceedings on Privacy Enhancing Technologies* 2016 (July 2016), no. 3, pp. 62–82

[MDPD15] Milutinovic, M.; Dacosta, I.; Put, A.; Decker, B. D.: uCentive: An Efficient, Anonymous and Unlinkable Incentives Scheme. In: *TrustCom/BigDataSE/ISPA (1)*. IEEE, 2015, pp. 588–595.

[Ped91] Pedersen, T. P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '91. Berlin, Heidelberg: Springer-Verlag, 1991, pp. 129–140

[Sch90] Schnorr, C. P.: Efficient Identification and Signatures for Smart Cards. In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Ed. by Brassard, G. New York, NY: Springer New York, 1990, pp. 239–252