PrePaMS: Privacy-Preserving Participant Management System for Studies with Rewards and Prerequisites

Echo Meißner
echo.meissner@uni-ulm.de
Institute of Distributed Systems
Ulm University
Germany

Frank Kargl
frank.kargl@uni-ulm.de
Institute of Distributed Systems
Ulm University
Germany

Benjamin Erb
benjamin.erb@uni-ulm.de
Institute of Distributed Systems
Ulm University
Germany

Felix Engelmann
fe-research@nlogn.org
MAX-IV Laboratory
Lund University
Sweden

Abstract

Taking part in surveys, experiments, and studies is often compensated by rewards to increase the number of participants and encourage attendance. While privacy requirements are usually considered for participation, privacy aspects of the reward procedure are mostly ignored. To this end, we introduce PrePaMS, an efficient participation management system that supports prerequisite checks and participation rewards in a privacy-preserving way. Our system organizes participations with potential (dis-)qualifying dependencies and enables secure reward payoffs. By leveraging a set of proven cryptographic primitives and mechanisms such as anonymous credentials and zero-knowledge proofs, participations are protected so that service providers and organizers cannot derive the identity of participants even within the reward process. In this paper, we have designed and implemented a prototype of PrePaMS to show its effectiveness and evaluated its performance under realistic workloads. PrePaMS covers the information whether subjects have participated in surveys, experiments, or studies. When combined with other secure solutions for the actual data collection within these events, PrePaMS can represent a cornerstone for more privacy-preserving empirical research.

Keywords

practical privacy-enhancing systems, privacy-preserving systems, participation management, zero-knowledge proofs, anonymous credentials

1 Introduction and Motivation

Surveys are imperative in opinion research, empiric evaluations, and feedback in corporations and organizations. Experiments and studies are also essential for most empirical and behavioral sciences, such as psychology and sociology. In all these fields, the quality of findings heavily depends on sufficiently large numbers of participations and motivated participants. Hence, participation is often incentivized through corresponding rewards.

Most academic study programs of empirical disciplines require students to contribute to studies for a certain number of subject hours. Similarly, successful participation is rewarded by credit points, often obligatory for graduation [36]. Longitudinal studies apply repeated observations in order to explore developments over time. Hence, rewards are often tied to continual survey participation [37]. Employee surveys provide insights into workplace characteristics and organizational conditions. Here, companies often provide certain incentives to improve response rates for employee surveys [5, 35]. Overall, rewarding participations is a common incentive strategy in many areas of science, education, and business.

During most experiments, surveys, and studies, data collection is handled with safeguards to protect the privacy of participants. In particular, anonymization or pseudonymization of participants' responses is required due to research-ethical principles, data management guidelines, or compliance reasons. However, anonymization and pseudonymization mechanisms *during* a study often interfere with reward incentives provided *after* the participation. This issue becomes particularly evident when participants are required to take part in a series of studies (e.g., longitudinal studies) or to take part in a certain number of different studies (e.g., subject hours) to be eligible for a final reward. Furthermore, the inclusion criteria of a study can interfere with anonymity, as a disclosed participation by itself may already reveal some information about the participant.

While there are several commercial solutions (e.g., Sona Systems [38]) and academic tools (e.g., hroot [8], ORSEE [21]) for participant management, none of these systems consider extensive privacy requirements for participations (e.g., hiding participations from service providers) nor privacy-preserving rewarding processes (e.g., collecting rewards anonymously while preventing the transfer of rewards between users). This means that participants often have to trust the organizers and the service to keep identities and participations secret.

In this paper, we propose a novel privacy-preserving participation management system that protects the privacy of participants

Proceedings on Privacy Enhancing Technologies 2025(1), 632–653
© 2025 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2025-0034

by breaking the link between participable tasks and the participating users while maintaining the possibility of individual, non-transferable rewards. Apart from the system design with privacy and security proofs, we complement our contributions with a prototype implementation[1] of the system and a publicly available test instance[2] to demonstrate the feasibility.

Our Main Contributions: • We propose PrePaMS, a novel set of cryptographic protocols to enable participations in studies with rewards and prerequisites while preserving the privacy of participants. • We define correctness, privacy, and security properties of our scheme and prove these properties for our construction. • We provide an open-source web-based proof of concept implementation[1] to showcase the practicability of our approach. This includes a publicly hosted test deployment[2] where anyone can explore our system themselves. • We evaluate the performance of our prototypical implementation. In this evaluation, we follow the Popper convention [23] for reproducible evaluations.

Technical Overview: We describe and apply a pairing-based, multi-show unlinkable anonymous credential scheme, which allows participants to authenticate themselves to the organizers. To only allow a single participation per user and study, we derive participation tags from the user's credential using a verifiable random function. To model (dis-)qualifiers (i.e., the requirement of (not) having previously participated in referenced studies), we utilize non-interactive zero-knowledge proofs (NIZK) where a participant proves these prerequisites on participation without revealing any specific link. Participation requirements based on attributes of a user's credential (e.g., age, handedness) are also proven using NIZK proofs. Altogether, this prevents an organizer from linking multiple participations of the same participant while still allowing for prerequisites and rewards. Participation rewards are issued as blind signatures, which a participant can re-randomize for a payout request without revealing which studies they have participated in to earn the rewards. At the same time, a reward is bound to the original participant, so it cannot be passed on to another user. Double spending is prevented by additionally publishing a verifiable nullifier of every reward.

2 Requirements for Participation Management

We now illustrate the relevance of participation management as well as the associated privacy risks using the example of subject hours in academic study programs with empirical research methods. We formalize these requirements later in Section 3.

Study programs with empirical methods often mandate a certain number of subject hours from their students [36] earned through study participation. These participations serve several purposes — self-experience for participating students, recruitment of subject pools for student research projects as well as sampling for actual research studies [20]. As a concrete example, we consider an undergrad psychology program in which students are asked to take part in a certain amount of study participations. The requirements for the participation management for such subject hours can be further divided into study management requirements (*SR*) and reward management requirements (*RR*) also shown in Table 1. Study management requirements refer to administrative and organizational procedures such as a list of current ongoing studies (*SR1*), but also includes the handling of conditions for participations. In turn, reward management requirements targets the tracking of the students' study participations so that the associated credit points can be eventually rewarded once students reach the required amount (*RR3*).

In the past, study management was mainly realized by using physical bulletin boards or websites listing the studies as well as arbitrary processes to handle preconditions of individual studies. Reward management was typically implemented using log sheets (i.e., participants receive a dated signature or stamp by the study organizer after participation) or stickers (i.e., participants receive a sticker or a similar token after participation).

These analog reward approaches work well with *RR1: analog studies*, such as lab experiments. However, they lack inherent support for *RR2: digital studies* (e.g., online questionnaires, mobile-sensing studies) and required additional organizational processes for rewarding (e.g., receiving a printable participation confirmation to be traded against a signature or sticker). Also, analog reward approaches often failed to prevent misuse by malicious students. A common threat to mandatory subject hours is students selling rewards to their peers or helping their peers out by participating in a study in their name. This of course contradicts the intended didactic goals, hence the need for *RR5: non-transferable rewards*. Furthermore, researchers want to *RR4: prevent duplicate participations* of participants in a study, to not skew the gathered data.

Nowadays, most universities have superseded analog approaches with a web-based participation management systems that cover both study and reward management and accommodate these missing features. One of the most prevalent systems is the commercial SaaS product "Sona" [38], which claims to be used by 1,500 customers across 30 countries and more than 23 million registered users. Sona as well as alternative academic systems [8, 21] introduced additional new features that are not covered by their analog counterparts: Some studies have prerequisites that participants have to satisfy in order to be admitted to the study. With web-based systems, this can be done automatically even before the participation (e.g., some common characteristics of participants are surveyed already by the participation management system at registration time), saving both researchers and participants time and resources. This includes *SR4: attributed-based preconditions*, such that only participants of a certain age are *qualified* or participants with specific medical preconditions are *disqualified*. For some studies *SR5: participation-based preconditions* are also necessary, either to exclude participants of a similar experiment or previous iteration to mitigate biases or to allow for longitudinal studies in which the same participants take part in a sequence of studies over time. For lab experiments, participants often have to schedule sessions beforehand, which can be simplified by a digital system featuring e.g., a calendar-based *SR6: session scheduling*.

Some systems further enable researchers to query the database of potential participants when designing a study, to verify if the participant pool theoretically contains a suitable sample size based on their planned prerequisites. Similar to preconditions this can be

[1] *Prototype source code:* https://github.com/vs-uulm/prepams/tree/pets25.1
[2] *Public test deployment:* https://vs-uulm.github.io/prepams/

divided into *SR2: attribute-based prescreening* and *SR3: participation-based prescreening* or a combination of both.

Protecting the personal data of participants *within* studies is not only a research-ethical mandate [17], it is also often grounded in legal requirements (e.g., GDPR in European countries). Mechanisms such as pseudonymization, anonymization, and data aggregation can help mask a participant's identity *within the data set* of that study [19]. Nevertheless, the mere fact that an individual has actually taken part in a certain study represents some kind of metadata and can already leak information about that individual. Hence, privacy requirements (*PR*) also apply to participation management.

Depending on the prerequisites of a study, this can involve personal attributes such as ethnicity or a certain native language of the participant, but also physical (e.g., dexterity, pre-existing diseases) or mental preconditions, such as currently ongoing clinical treatments, and other rather sensitive characteristics. Furthermore, participation in a certain study can also imply previous participations (e.g., longitudinal study designs with multiple measurement dates) or hint to the non-participation in a former study (e.g., follow-up studies that disallow participation of subjects from previous trials). That said, an attacker with knowledge of the sequence of participations of an individual and contextual information about the corresponding studies might already be able to create a sensitive profile of a target — completely independent from the actual (and inaccessible) response data collected in these studies. As many people involved in the organizational process for studies have potential access to both the participation data (e.g., a research assistant who is signing the participation confirmations) and the contextual study information (e.g., inclusion and exclusion criteria listed on the university-internal study management system), we argue that this is a realistic threat that cannot be neglected. Given a sequence of participations of an individual, their known identity can then be linked to a set of potentially sensitive attributes deduced from these participations.

A sticker-based analog reward management — if implemented correctly — provides an inherent privacy level similar to fiat currency for money. Participants can participate anonymously, so that organizers are not able to link them to prior participations and the rewarded stickers are also not linked to the study they were rewarded for (*PR1: anonymous participation*). Similarly, when a student obtained the mandated subject hours they can exchange a full set of stickers for their course credit without revealing their participation history (*PR2: anonymous rewarding*). Log sheets on the other hand disclose the full prior participation history during participation to all study personal handling the sheet, and also when eventually handing in the completed sheet to exchange the hours for the final course credit. The existing digital systems all store the individual participations in a database, which is depending on the system accessible to all researchers or just administrators. Even if limited to just administrators of a system, this central database is still vulnerable to sensitive data leaks, e.g., through a software vulnerabilities or insider attacks.

In this paper, we present a cryptographic scheme that provides the same requirements and security properties as current web-based participation management systems, while providing similar privacy guarantees to a sticker-based analog approach. Table 1 shows the supported features and privacy properties of the discussed analog approaches and current digital systems compared to our proposed system PrePaMS.

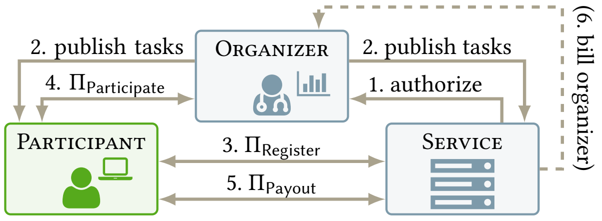


Figure 1: *Simplified system overview.* 1. organizers are authorized by the service; 2. organizer publishes tasks; 3. participant registers with service; 4. participant participates in organizer's task (retrieved either via service or organizer); 5. after multiple participations, participant requests payout of rewards from service; 6. after task is concluded, service may bill organizer for spent funds.

The listed requirements are based on an analysis of the features of state-of-the-art participation management systems and common practices for rewarding. In addition, we double-checked with contacts from our psychology department that the derived requirements are both adequate and exhaustive from their perspective and experience.

3 Formalization

In this section, we present a formal definition of a privacy-preserving participation protocol involving three types of parties, of which two collude: A single participation management service (S), colluding with one or multiple study organizers (O), and a number of participants (P). In general, the protocol enables participants to participate in various tasks, which are, e.g., studies in the scenario described above but may be other events with rewards, and to receive a reward after successful participation without leaking who participated in which task. Figure 1 shows a simplified overview.

3.1 Threat Model

Our protocol is applicable in the following setting. The *Service* acts as a covert entity during participant registration. Similarly, when a participant requests a payout of virtual credits for real-world rewards, it will honestly fulfill this exchange request. In the university study setting, the rewards may be European Credit Transfer System (ECTS) credits or fiat currency. However, it may use its available information to determine which tasks a participant has participated in, gaining confidential information about the participant. It may link this to the real identity necessary for the payout procedure, which violates the participant's privacy. We envision that this service is run by the accounting department or administration of an institution because the goals of an accounting department/administration coincide well with the goals of the service. Regarding participation privacy, we assume that the Service colludes with the Organizers.

Organizers in our model create and publish studies and offer participation rewards to participants. Furthermore, organizers may be malicious in the sense of attacking the privacy of the participants

Table 1: Functional requirements and privacy properties of common participation management approaches and related systems compared to our proposed system PrePaMS.

			Log Sheets	Sticker	hroot [8]	ORSEE [21]	Sona [38]	**PrePaMS**
study management	**SR1**	**listing of studies**			✔	✔	✔	✔
	SR2	**attribute-based prescreening**			(✔)	✔	✔	(✔) [a]
	SR3	**participation-based prescreening**			✖	✔	✔	(✖) [b]
	SR4	**attribute-based prerequisites**			✔	✔	✔	✔
	SR5	**participation-based prerequisites**			(✔)	✔	✔	✔
	SR6	**session scheduling**			✔	✔	✔	(✖) [c]
rewarding	**RR1**	**analog studies**	✔	✔	✔	✔	✔	✔
	RR2	**digital studies**	✖ [d]	✖ [d]	✔	✔	✔	✔
	RR3	**rewarding**	✔	✔	✖	✔	✔	✔
	RR4	**duplicate participation prevention**	✔	✖	✔	✔	✔	✔
	RR5	**non-transferable rewards**	✔	✖	✖	✔	(✔) [e]	✔
privacy	**PR1**	**anonymous participation**	✖	✔	✖	✖	✖ [f]	✔[g]
	PR2	**anonymous rewarding**	✖	✔	✖	✖	✖ [f]	✔[g]

Remarks: [a)] Attribute-based prescreening is currently not implemented, but possible if the service keeps track of which attributes are issued during registration. [b)] Participation-based prescreening has intrinsic privacy issues, if centrally maintained. However, an organizer knows the number of participations of their own previous studies and can use this information for most scenarios. [c)] Session scheduling does not come with strict privacy requirements and can be implemented, e.g., using the participation tag as an anonymous identity. [d)] Rewarding can only be implemented by using additional organizational processes, such as a subsequent conversion of digital participation confirmations into physical stickers or signatures. [e)] Although credits in Sona are rewarded to a persistent pseudonym of a user, it is not verified if the pseudonym belongs to the participant. [f)] Rewarding and participation uses a persistent pseudonym, which can be linked across studies and sometimes to a participant's identity. [g)] PrePaMS also supports pseudonyms to track participants over subsequent participations for longitudinal studies. The pseudonym remains unlinkable to unrelated studies.

by linking multiple participations or linking a participation to a participant's identity. A task or study T is a public, opaque object with a defined participation reward and optional prerequisites that a participant has to satisfy when participating. Such prerequisites can be divided into *qualifiers* (i.e., other tasks a participant must have participated in to qualify for this task), *disqualifiers* (i.e., previous tasks a participant must not have participated in to qualify for this task), *range constraints* (i.e., the value of an attribute associated with the participants credential must be contained in a specified range), or *set constraints* (i.e., an attribute value must be included in a specified set) a participant must fulfill.

As the Service and Organizers collude, we take advantage to simplify our formalization by modeling them as a single service S with a separate public append-only bulletin board BB.

Participants want to participate in tasks to earn rewards in the form of virtual credits while protecting their privacy. Participants can also be malicious by attempting to participate in a study they do not qualify for, participating twice, receiving payment for a task without participating, or attempting to request payouts without earning virtual credits first. Another problem is fraud between participants, selling credits to other users. It is a general limitation that users may perform a task poorly, e.g., in user studies where participants enter bogus data that needs to be sanitized by statistical tests, which we consider out of scope. De-anonymization attempts via network information and metadata are also considered out of scope. We assume participants conceal their identity when interacting with other parties through an anonymous network, such as Tor, or similar means.

3.2 Syntax

In the following, we provide a definition of five probabilistic polynomial time algorithms and interactive protocols (Setup, $\mathsf{KeyGen}_\mathsf{S}$, Π_Register, $\Pi_\mathsf{Participate}$, Π_Payout) that comprise a participation management protocol. In case of invalid inputs or failures in interaction, all algorithms have the option to abort. Our notation for lists with a specific order is $[\dots]$ to which another list is concatenated or an element is appended by $\|$. For lists, we use, e.g., $A[:n] := [A_1, \dots, A_n]$ to indicate slices and subscript to index, e.g., A_1 for the first element. For tuples $T := (a, b, c)$ we use $T.a$ for a named element of the tuple. For an integer n, we define the sequence $[n] := [1, \dots, n]$. We denote a negligible polynomial in λ as $\mathsf{negl}(\lambda)$. We use **assert** in algorithms to check a condition and abort if it evaluates to `false`. Table 2 in the appendix summarizes the variable names.

$\mathsf{Setup}(1^\lambda, m, n) \rightarrow \mathsf{p}$**:** takes the security parameter 1^λ, the number of attributes $m \in \mathbb{N}$ of a participant and the maximum number of payouts $n \in \mathbb{N}$. It outputs the public parameters p.

$\mathsf{KeyGen}_\mathsf{S}() \rightarrow (\mathsf{sk}_\mathsf{S}, \mathsf{pk}_\mathsf{S})$**:** outputs the key-pair $(\mathsf{sk}_\mathsf{S}, \mathsf{pk}_\mathsf{S})$ used by the service and organizers. pk_S is distributed to everyone for credential verification. *This means all subsequent algorithms have an implicit* $(\mathsf{p}, \mathsf{pk}_\mathsf{S})$ *input, where* p *includes* m, n.

$\Pi_\mathsf{Register}\langle \mathsf{P}(\mathsf{un}, \mathsf{attr}), \mathsf{S}(\mathsf{sk}_\mathsf{S})\rangle \rightarrow (\mathsf{cred}, \mathsf{un})$**:** is an interactive protocol between P and S where a participant P, identified by a unique username un and attributes attr, receives a secret credential cred for later participation in tasks. The service uses its secret key sk_S to sign the credential and gets the username to prevent duplicate registration, and verifiers the attributes.

$\Pi_\mathsf{Participate}\langle \mathsf{P}(\mathsf{cred}, \mathsf{BB}, \mathsf{T}), \mathsf{S}(\mathsf{sk}_\mathsf{S})\rangle \rightarrow \mathsf{tx}$**:** is an interactive protocol between P and S which allows P with credential cred to participate in a task T from an Organizer. The eligibility of P to participate is dependent on all previous participations by all participants described as the list of reward transactions $\mathsf{BB} := [\mathsf{tx}_i]_{i=1}^{|\mathsf{BB}|}$ on the bulletin board. This protocol returns a reward transaction tx to both parties that is appended to the bulletin board $\mathsf{BB} \leftarrow \mathsf{BB} \| \mathsf{tx}$. A tx includes the performed task $\mathsf{T} \in \mathbb{T}$ from a task domain, defined as a tuple $\mathsf{T} := (v, \bar{\delta} \subset \mathbb{T}, \delta \subset \mathbb{T}, \mathsf{constr}, \Delta)$ with a reward v, qualifiers $\bar{\delta}$, disqualifiers δ, attribute constraints constr, and an opaque task description $\Delta \in \{0,1\}^*$, e.g., a title, external link, and a verbose description.

$\Pi_{\mathsf{Payout}}\langle \mathsf{P}(\mathsf{cred}, \mathsf{un}, \mathsf{BB}, v, \{\mathsf{tx}_i\}_{i=1}^{k}), \mathsf{S}()\rangle \to (\{\mathcal{N}_i\}_{i=1}^{n}, v, \mathsf{un})$**:** is an interactive protocol to allow a participant P with cred, belonging to username un, to request a payout of earned rewards from a list of $k \leq n$ reward transactions $[\mathsf{tx}_i]_{i=1}^{k} \subset \mathsf{BB}$ with amount v from S. It returns payout records, also called nullifiers, $\mathcal{N}_i$ which are added to the public set NUL maintained by the service and the paid out value v to user un.

For our correctness and security definitions, we require the following four auxiliary algorithms.

$\mathsf{Receive}(\mathsf{cred}, \mathsf{BB}, \mathsf{NUL}) \to (s, R)$**:** returns the total amount s owned by cred and the list of reward transactions $R = [\mathsf{tx}_i]_{i=0}^{|R|}$ which are present in the bulletin board BB and which have not yet been paid out with a record in NUL.

$\mathsf{ChkCred}(\mathsf{cred}, \mathsf{pk_S}) \to \{0,1\}$**:** returns 1 if the credential cred is valid under the service's public key $\mathsf{pk_S}$.

$\mathsf{ChkPart}(\mathsf{cred}, \mathsf{tx}) \to \{0,1\}$**:** returns 1 if the credential cred was used for the participation which resulted in the transaction tx.

$\mathsf{ChkQual}(\mathsf{cred}, \mathsf{BB}, \mathsf{T}) \to \{0,1\}$**:** returns 1 if the participant satisfies all prerequisites of T given the previous participations registered in BB which is a time ordered list of tx. Meaning a participant has neither participated in any disqualifier of T nor in the task itself, but has participated in all qualifiers of T, and fulfills the attribute constraints.

Definition 3.1 (Correctness). A participation workflow is *correct* if the following conditions apply. For any $\lambda \in \mathbb{N}$, any $\mathsf{p} \leftarrow \mathsf{Setup}(1^\lambda)$ and any of the services' generated key pairs $(\mathsf{sk_S}, \mathsf{pk_S}) \leftarrow \mathsf{KeyGen_S}()$, it then holds that:

Honest Registration: For any previously non-registered participant identified by username un and any attributes attr, it holds that $(\mathsf{cred}, \mathsf{un}) \leftarrow \Pi_{\mathsf{Register}}\langle \mathsf{P}(\mathsf{un}, \mathsf{attr}), \mathsf{S}(\mathsf{sk_S})\rangle$ implies $\mathsf{ChkCred}(\mathsf{cred}, \mathsf{pk_S}) = 1$

Honest Participation: With any valid, signed credential cred (i.e., $\mathsf{ChkCred}(\mathsf{cred}, \mathsf{pk_S}) = 1$), a participant who fulfills the prerequisites of a task T at any state BB (i.e., $\mathsf{ChkQual}(\mathsf{cred}, \mathsf{BB}, \mathsf{T}) = 1$), successfully participates in T with $\mathsf{tx} \leftarrow \Pi_{\mathsf{Participate}}\langle \mathsf{P}(\mathsf{cred}, \mathsf{BB}, \mathsf{T}), \mathsf{S}(\mathsf{sk_S})\rangle$ such that $\mathsf{Receive}(\mathsf{cred}, [\mathsf{tx}], \emptyset).v = \mathsf{T}.v$.

Honest Payout: A participant with cred, given $s, R \leftarrow \mathsf{Receive}(\mathsf{cred}, \mathsf{BB}, \mathsf{NUL})$ can get a payout of an amount $v \in \{0, \ldots, s\}$ with $(\mathsf{NUL}', v) \leftarrow \Pi_{\mathsf{Payout}}\langle \mathsf{P}(\mathsf{cred}, \mathsf{un}, \mathsf{BB}, \mathsf{NUL}, v, R), \mathsf{S}()\rangle$ such that $\mathsf{Receive}(\mathsf{cred}, \mathsf{BB}, \mathsf{NUL} \cup \mathsf{NUL}').s \leq s - v$.

We define the security of our scheme with four game-based security definitions. They all follow the logic that an adversary interacts with a game and tries to win, breaking a security property. The interaction either happens through oracles, where the adversary is instructing honest parts of the game, or a challenge to the game. For our construction, we later prove, that the existence of a probabilistic polynomial time algorithm to win the game is negligible.

3.3 Oracles

Before defining our game-based security and privacy properties of the scheme, we require oracles for the adversary to interact with. Our adversaries are allowed to call all oracles in a mixed order for a polynomial number of times in λ. This allows the adversary to instruct honest participants to perform an action without access to their secret state. The oracles presented in Figure 2 operate as one or both parties of the interactive protocols Π_{Register}, $\Pi_{\mathsf{Participate}}$, and Π_{Payout} which gives a total of nine possibilities. To highlight the adversary controlling the other party, we denote the input of this party as a part of the stateful adversary (e.g., $\mathcal{A}_\mathsf{P}$ for an adversary-controlled participant). The oracles controlling both parties of the protocols are required to allow the participant adversary to observe honest interactions and their public effects. This is important for, e.g., replay attacks. The service oracles accept any participant adversary defined tasks. Some oracles perform bookkeeping operations maintaining two lists of oracle controlled users $\mathfrak{U}^e$ with $e \in \{0,1\}$. The epoch e is required for the service oracles to separate users into two subsets. The oracle-provided service is thereby able to track new registrations. For each username, $\mathfrak{U}^e_{\mathsf{un}}$ maintains a tuple of (cred, NUL): their credential cred and spent coin nullifiers NUL from payout interactions. The oracles also track two total amounts of rewards $\mathfrak{T}^e$ depending on when the participation was rewarded. $\mathcal{O}$PParticipate and $\mathcal{O}$PPayout is parametrized with a lock set L of usernames that ban them from specific interactions after a challenge from the adversary. This is required to prevent the users from being trivially deanonymized depending on qualifying for tasks or access to rewards. However, it does not weaken the adversary, as a winning adversary could have prepared the situation beforehand and use it in the challenge. Additionally, the participant adversaries get an oracle $\mathcal{O}$state to access the public state BB, NUL.

3.4 Participation Security

Our first property is participation security, which ensures that participants can only participate in studies for which they qualify (checked by ChkQual). We capture this property by defining a security game with oracles from above. The game PartSec below sets up the service with a generic organizer and then allows the adversary $\mathcal{A}$ oracle access to register users and participate with them in studies. The oracle $\mathcal{O}$SRegister only checks that each username is registered once and $\mathcal{O}$SParticipate tracks every participation by appending the reward transaction to the list BB. The adversary also has the power to observe honest users interacting with the service through $\mathcal{O}$Register and $\mathcal{O}$Participate. There is only one epoch $e = 0$. For completeness the adversary also has access to payout oracles, however they do not help in any way. We note that a single generic organizer in this scenario has equivalent power as multiple ones. They all perform the same verification and may be controlled by a single entity.

The adversary is then challenged to present a credential cred which at any point in the list of participations successfully participated by having a reward transaction matching the credential but was not allowed to do so due to missing qualifications.

```
$\mathcal{O}\mathsf{SRegister}_{\mathsf{sk_S},e}(\mathsf{un}, \mathsf{attr}, \mathcal{A}_\mathsf{P})$:
    **Assert** $\mathsf{un} \notin \mathfrak{U}^0 \wedge \mathsf{un} \notin \mathfrak{U}^1$
    run service side of $\Pi_\mathsf{Register}\langle \mathcal{A}_\mathsf{P}, \mathsf{S}(\mathsf{sk_S})\rangle$
    $\mathfrak{U}^e_\mathsf{un} \leftarrow (\perp, \emptyset)$
$\mathcal{O}\mathsf{SParticipate}_{\mathsf{sk_S},e}(\mathcal{A}_\mathsf{P})$:
    run service side of $\mathsf{tx} \leftarrow \Pi_\mathsf{Participate}\langle \mathcal{A}_\mathsf{P}, \mathsf{S}(\mathsf{sk_S})\rangle$
    $\mathsf{BB} \leftarrow \mathsf{BB}\|\mathsf{tx}$
    $\mathfrak{T}^e \leftarrow \mathfrak{T}^e + \mathsf{tx.T}.v$
$\mathcal{O}\mathsf{SPayout}(\mathcal{A}_\mathsf{P})$:
    run S side of $\mathsf{NUL}', v, \mathsf{un} \leftarrow \Pi_\mathsf{Payout}\langle \mathcal{A}_\mathsf{P}, \mathsf{S}()\rangle$
    **if** $\mathsf{un} \in \mathfrak{U}^0$ **then**
        $\mathfrak{T}^0 \leftarrow \mathfrak{T}^0 - v$
    **else if** $\mathsf{un} \in \mathfrak{U}^1$ **then**
        $\mathfrak{T}^1 \leftarrow \mathfrak{T}^1 - v$
    $\mathsf{NUL} \leftarrow \mathsf{NUL} \cup \mathsf{NUL}'$
```

```
$\mathcal{O}\mathsf{Register}_{\mathsf{sk_S}}(\mathsf{un}, \mathsf{attr})$:
    **Assert** $\mathsf{un} \notin \mathfrak{U}^0 \wedge \mathsf{un} \notin \mathfrak{U}^1$
    run $\mathsf{cred} \leftarrow \Pi_\mathsf{Register}\langle \mathsf{P}(\mathsf{un}, \mathsf{attr}, \mathsf{pk_S}), \mathsf{S}(\mathsf{sk_S})\rangle$
    $\mathfrak{U}^0_\mathsf{un} \leftarrow (\mathsf{cred}, \emptyset)$
$\mathcal{O}\mathsf{Participate}_{\mathsf{sk_S}}(\mathsf{T}, \mathsf{un})$:
    run $\mathsf{tx} \leftarrow \Pi_\mathsf{Participate}\langle \mathsf{P}(\mathfrak{U}^0_\mathsf{un}.\mathsf{cred}, \mathsf{BB}, \mathsf{T}), \mathsf{S}(\mathsf{sk_S})\rangle$
    $\mathsf{BB} \leftarrow \mathsf{BB}\|\mathsf{tx}$
$\mathcal{O}\mathsf{Payout}(\mathsf{un}, \mathsf{BB}, v)$:
    $(\mathsf{cred}, \mathsf{NUL}') \leftarrow \mathfrak{U}^0_\mathsf{un}.\mathsf{cred}$
    $a, R \leftarrow \mathsf{Receive}(\mathsf{cred}, \mathsf{BB}, \mathsf{NUL} \cup \mathsf{NUL}')$
    run $\mathsf{NUL}'', v, \mathsf{un} \leftarrow \Pi_\mathsf{Payout}\langle \mathsf{P}(\mathsf{cred}, \mathsf{BB}, \mathsf{NUL}, v, R), \mathsf{S}()\rangle$
    $\mathfrak{U}^0_\mathsf{un}.\mathsf{NUL} \leftarrow \mathfrak{U}^0_\mathsf{un}.\mathsf{NUL} \cup \mathsf{NUL}''$
    $\mathsf{NUL} \leftarrow \mathsf{NUL} \cup \mathsf{NUL}''$
$\mathcal{O}\mathsf{state}()$:
    **return** $\mathsf{BB}, \mathsf{NUL}$
```

```
$\mathcal{O}\mathsf{PRegister}(\mathsf{un}, \mathsf{attr}, \mathsf{pk_S}, \mathcal{A}_\mathsf{S})$:
    **Assert** $\mathsf{un} \notin \mathfrak{U}^0 \wedge \mathsf{un} \notin \mathfrak{U}^1$
    run P side of $\mathsf{cred} \leftarrow \Pi_\mathsf{Register}\langle \mathsf{P}(\mathsf{un}, \mathsf{attr}, \mathsf{pk_S}), \mathcal{A}_\mathsf{S}\rangle$
    $\mathfrak{U}^0_\mathsf{un} \leftarrow (\mathsf{cred}, \emptyset)$
$\mathcal{O}\mathsf{PParticipate}_L(\mathsf{BB}, \mathsf{T}, \mathsf{un}, \mathcal{A}_\mathsf{S})$:
    **if** $\mathsf{un} \in L$ **then**
        **Assert** $\forall u \in L : \mathsf{ChkQual}(\mathfrak{U}^0_u.\mathsf{cred}, \mathsf{BB}, \mathsf{T}) = 1$
    **Assert** $\mathsf{ChkQual}(\mathfrak{U}^0_\mathsf{un}.\mathsf{cred}, \mathsf{BB}, \mathsf{T}) = 1$
    run P side of $\mathsf{tx} \leftarrow \Pi_\mathsf{Participate}\langle \mathsf{P}(\mathfrak{U}^0_\mathsf{un}.\mathsf{cred}, \mathsf{BB}, \mathsf{T}), \mathcal{A}_\mathsf{S}\rangle$
$\mathcal{O}\mathsf{PPayout}_L(\mathsf{un}, \mathsf{BB}, v, \mathcal{A}_\mathsf{S})$:
    **if** $\mathsf{un} \in L$ **then**
        **Assert** $\forall u \in L : \mathsf{Receive}(\mathfrak{U}^0_u.\mathsf{cred}, \mathsf{BB}, \mathsf{NUL}).a \geq v$
    $\mathsf{cred} \leftarrow \mathfrak{U}^0_\mathsf{un}.\mathsf{cred}$
    $a, R \leftarrow \mathsf{Receive}(\mathsf{cred}, \mathsf{BB}, \mathsf{NUL} \cup \mathfrak{U}^0_\mathsf{un}.\mathsf{NUL})$
    run P of $\mathsf{NUL}', v, \mathsf{un} \leftarrow \Pi_\mathsf{Payout}\langle \mathsf{P}(\mathsf{cred}, \mathsf{BB}, \mathsf{NUL}, v, R), \mathcal{A}_\mathsf{S}\rangle$
    $\mathfrak{U}^0_\mathsf{un}.\mathsf{NUL} \leftarrow \mathfrak{U}^0_\mathsf{un}.\mathsf{NUL} \cup \mathsf{NUL}'$
```

```
$\mathsf{PartSec}(1^\lambda)$:
    $\mathsf{p} \leftarrow \mathsf{Setup}(1^\lambda)$
    $\mathfrak{U}^0 \leftarrow \emptyset, \mathfrak{T}^0 \leftarrow 0, \mathsf{BB} \leftarrow [\,], \mathsf{NUL} \leftarrow \emptyset$
    $\mathsf{sk_S}, \mathsf{pk_S} \leftarrow \mathsf{KeyGen_S}(\mathsf{p})$
    $\mathcal{O} := \{\mathcal{O}\mathsf{SRegister}_{\mathsf{sk_S},0}, \mathcal{O}\mathsf{SParticipate}_{\mathsf{sk_S},0},$
    $\mathcal{O}\mathsf{SPayout}, \mathcal{O}\mathsf{Register}_{\mathsf{sk_S}}, \mathcal{O}\mathsf{Participate}_{\mathsf{sk_S}},$
    $\mathcal{O}\mathsf{Payout}, \mathcal{O}\mathsf{state}\}$
    $\mathsf{cred} \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{p}, \mathsf{pk_S})$
    **Assert** $\mathsf{ChkCred}(\mathsf{cred}, \mathsf{pk_S}) = 1$
    **for** $t \in [|\mathsf{BB}|]$ **do**
        **if** $\mathsf{ChkQual}(\mathsf{cred}, \mathsf{BB}[:t], \mathsf{BB}_t.\mathsf{T}) = 0$
            $\wedge\ \mathsf{ChkPart}(\mathsf{cred}, \mathsf{BB}_t) = 1$ **then**
            **return** win
```

```
$\mathsf{Balance}(1^\lambda)$:
    $\mathsf{p} \leftarrow \mathsf{Setup}(1^\lambda)$
    $\mathfrak{U}^0 \leftarrow \emptyset, \mathfrak{U}^1 \leftarrow \emptyset, \mathfrak{T}^0 \leftarrow 0, \mathfrak{T}^1 \leftarrow 0, \mathsf{BB} \leftarrow [\,], \mathsf{NUL} \leftarrow \emptyset$
    $\mathsf{sk_S}, \mathsf{pk_S} \leftarrow \mathsf{KeyGen_S}(\mathsf{p})$
    $\mathcal{O}(e) := \{\mathcal{O}\mathsf{SRegister}_{\mathsf{sk_S},e}, \mathcal{O}\mathsf{SParticipate}_{\mathsf{sk_S},e},$
    $\mathcal{O}\mathsf{SPayout}, \mathcal{O}\mathsf{Register}_{\mathsf{sk_S}}, \mathcal{O}\mathsf{Participate}_{\mathsf{sk_S}},$
    $\mathcal{O}\mathsf{Payout}, \mathcal{O}\mathsf{state}\}$
    $\mathsf{aux} \leftarrow \mathcal{A}_0^{\mathcal{O}(0)}(\mathsf{p}, \mathsf{pk_S})$
    **if** $\mathfrak{T}^0 < 0$ **then return** win
    $\mathcal{A}_1^{\mathcal{O}(1)}(\mathsf{p}, \mathsf{pk_S}, \mathsf{aux})$
    **if** $\mathfrak{T}^1 < 0 \vee \mathfrak{T}^0 + \mathfrak{T}^1 < 0$ **then return** win
```

```
$\mathsf{PartPriv}_b(1^\lambda)$:
    $\mathsf{p} \leftarrow \mathsf{Setup}(1^\lambda)$
    $\mathfrak{U}^0 \leftarrow \emptyset, \mathfrak{T}^0 \leftarrow 0, \mathsf{BB} \leftarrow [\,], \mathsf{NUL} \leftarrow \emptyset$
    $\mathcal{O}(L) := \{\mathcal{O}\mathsf{PRegister}, \mathcal{O}\mathsf{PParticipate}_L, \mathcal{O}\mathsf{PPayout}_L\}$
    $(\mathsf{BB}', \mathsf{un}_0, \mathsf{un}_1, \mathsf{T}, \mathcal{A}_\mathsf{S}, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathcal{O}(\emptyset)}(\mathsf{p})$
    **Assert** $\mathsf{BB} \subseteq \mathsf{BB}'$
    **Assert** $\forall i \in \{0,1\} : \mathsf{ChkQual}(\mathfrak{U}^0_{\mathsf{un}_i}.\mathsf{cred}, \mathsf{BB}', \mathsf{T}) = 1$
    run P side of $\mathsf{tx} \leftarrow \Pi_\mathsf{Participate}\langle \mathsf{P}(\mathfrak{U}^0_{\mathsf{un}_b}.\mathsf{cred}, \mathsf{BB}', \mathsf{T}), \mathcal{A}_\mathsf{S}\rangle$
    $b' \leftarrow \mathcal{A}_1^{\mathcal{O}(\{\mathsf{un}_0,\mathsf{un}_1\})}(\mathsf{p}, \mathsf{tx}, \mathsf{aux})$
    **if** $b = b'$ **then return** win
```

Figure 2: Oracles with Security and Privacy Games.

Definition 3.2 (Participation Security). A participation scheme is secure if for all $\lambda \in \mathbb{N}$, all ppt adversaries $\mathcal{A}$ and the game PartSec in Figure 2 it holds that: $\Pr[\mathsf{PartSec}(1^\lambda) = \mathtt{win}] \leq \mathsf{negl}(\lambda)$

3.5 Balance

In addition to assuring participation security, we require that participants can only claim as much rewards as they have rightfully earned. This balance property is captured in the Balance game. Similarly to PartSec, a service and organizer is set up and the adversary has oracle access to register users, participate in studies, and request a payout through $\mathcal{O}\mathsf{SPayout}$. The adversary may observe honest interactions through the oracles controlling both parties. The anonymous nature of the participations requires special attention to prevent users from sharing rewards. All credentials maintained by the adversary are shared and are used by the adversary to participate and get rewards. Together with the fact that the oracle cannot keep track of balances of anonymous participants, we split the adversarial actions into two epochs. During both, the adversary can register users, participate with them, and get payouts as well as observe fully honest users in their actions. In epoch 0, all adversarially registered users are tracked in $\mathfrak{U}^0$ and the total reward in $\mathfrak{T}^0$. If the adversary manages to get more paid out than was rewarded ($\mathfrak{T}^0 < 0$) it already wins. This prevents, e.g. stealing funds from honest users. In addtion, the adversary gets a second chance to win, if they are able to move coins between participants.

To test that, the game switches the epoch to $e = 1$ and now registers all new users in $\mathfrak{U}^1$ and all rewards are tracked in $\mathfrak{T}^1$. As the participation is anonymous, this means that also rewards to participations from users in $\mathfrak{U}^0$ are still tracked by $\mathfrak{T}^1$. If, after the interaction, more is paid out than was rewarded in epoch 1 ($\mathfrak{T}^1 < 0$) the adversary wins. The identity of the users is revealed at payout, which allows subtracting from the matching $\mathfrak{T}^e$ to when this user was registered. The adversary $\mathcal{A}_1$, getting secret state from A_0 through aux, also wins if the sum of both epochs is negative ($\mathfrak{T}^0 + \mathfrak{T}^1 < 0$). It is possible to have $\mathfrak{T}^0 < 0$ in the following scenario: During epoch 0, the adversary registers un but does not participate in any task ($\mathfrak{T}^0 = 0$). Then in epoch 1, the adversary asks un to participate in a study with reward v ($\mathfrak{T}^1 = v$). Then un should get a payout of v. As $\mathsf{un} \in \mathfrak{U}^0$, the deduction is from $\mathfrak{T}^0$, i.e. $\mathfrak{T}^0 = -v, \mathfrak{T}^1 = v$. Their sum is still non negative. This property also captures phishing attacks for rewards of users, which cannot be paid out by the person who stole the credential. To win our game, the adversary must find a way to get a credential paid out to a “new” username which was earned by an “old” participation.

Definition 3.3 (Balance). A participation scheme is balanced, if for all $\lambda \in \mathbb{N}$ and all ppt adversaries $(\mathcal{A}_0, \mathcal{A}_1)$, it holds that $\Pr[\mathsf{Balance}(1^\lambda) = \mathtt{win}] \leq \mathsf{negl}(\lambda)$ with Balance from Figure 2.

3.6 Participation Privacy

Next to the security properties protecting the service from malicious users, we define the privacy property which protects the users' anonymity from a malicious service. The participation privacy assures that as the service and the organizers collude they are still unable to track a user from the registration with their real identity to participations or link between participations or to payouts. In our game-based definition, the adversary controls the service and the organizers. It is given oracle access to register participants and make them participate in a given study without access to their secret state (cred). The adversary has no use of the double secret oracles (e.g. $\mathcal{O}$Participate) as all participants may be malicious. In the second invocation of the adversary (connected by aux), they have limited access to $\mathcal{O}$PParticipate and $\mathcal{O}$PPayout for the users un_0 and un_1. This prevents trivial cases to get b' from the participants' possibility to qualify for a study or access to the reward. Together with the Balance property, no rewards can be shared between the users to discriminate. As usernames are unique, we use them to identify the users. Once the adversary prepared the users, it provides a bulletin board BB′ (which is a superset of oracle participations, as otherwise it is easy to make the same user participate twice) and a Task T to fulfill by one of the two users un_0 or un_1. Both users need to have valid credentials registered by the $\mathcal{O}$PRegister oracle and fulfill the prerequisites ($\mathsf{ChkQual}(\mathsf{cred}, \mathsf{BB}, \mathsf{T}) = 1$) to participate. This ensures that the adversary gets no advantage from an aborted participation. The game proceeds to participate with the credential of the user specified by the bit b. The interactive adversary $\mathcal{A}_1$, with access to restricted oracles, needs to efficiently distinguish between the $b = 0$ and $b = 1$ game.

Definition 3.4 (Participation Privacy). A participation scheme has private participations, if for all $\lambda \in \mathbb{N}$ and all ppt adversaries $(\mathcal{A}_0, \mathcal{A}_1)$ and the game PartPriv defined in Figure 2, it holds that $|\Pr[\mathsf{PartPriv}_0(1^\lambda) = \mathtt{win}] - \Pr[\mathsf{PartPriv}_1(1^\lambda) = \mathtt{win}]| \leq \mathsf{negl}(\lambda)$.

Remark: If a series of tasks are part of a longitudinal study, a possible requirement is to link participants over a longer period of time. This is modeled as weak participation privacy, where the organizer can link a previous participation to the current one. However, anonymity regarding any other task is still maintained.

Corollary 3.5 (Fixed Participant State). *From the syntax of our formalization, it is visible that the participants get a credential through the Π_{Register} protocol, and thereafter, the credential* cred *is never updated. Such a property is desirable as users can back up their credential after registration and then in case of a recovery derive all secret state from this credential in combination with public information. One allowed exception is the need for users to store previously disclosed payout nullifiers* NUL′. *This can be circumvented if the service maintains a public append only log of* NUL.

The validation of our formalization with regards to the log sheets and stickers is deferred to Section B.

4 PrePaMS

In this section we present our protocol PrePaMS, which realizes a privacy-preserving participant management with the properties defined above.

We present the complete construction of the PrePaMS scheme in Figure 3. It utilizes multiple building blocks, which are described in detail in Section C: • a partially blind signature scheme PBS = (Setup, KeyGen, Blind, Sign, Unblind, Verify) with randomness space $\mathbb{R}_{\mathsf{PBS}}$ and an efficient NIZK for $\mathcal{L}_{\mathsf{PBS}}$ to show correct blinding. Blind allows a user to blind a secret message, which is then Sign-ed with the signer only knowing part of the message. The user then Unblind-s the result and is able to prove a valid signature with Verify. As we need this building block twice, we instantiate two, domain-separated, versions called PBSC and PBSR, • a labeled verifiable random function VRF = (Setup, KeyGen, Eval) where KeyGen generates a public private key pair and Eval takes a label from the domain of tasks $\mathbb{T}$ in addition to the secret key from φ_{VRF} and outputs a deterministic, uniformly random, verifiable tag, • zero-knowledge non-interactive arguments of knowledge $\mathsf{NIZK}[\mathcal{L}]$ = (Setup, Prove, Verify, Sim) to prove various NP languages $\mathcal{L}$, • a key derivation function KDF : $\varphi_{\mathsf{VRF}} \times \mathbb{T} \rightarrow \mathbb{S} \times \mathbb{R}_{\mathsf{PBS}}$, which takes a secret key from φ_{VRF} and a task from $\mathbb{T}$ to deterministically generate a nullifier and a PBS blinding randomness.

We explicitly indicate exchanged messages in the construction of interactive protocols with blocking **send** and **recv** calls.

Given these building blocks, we present an overview of how we combine them: To register participants, we use the PBSC scheme to blind-sign a secret key of the VRF and the users attributes attr, one of which is the user's identity un. The PBSC signature is the credential and for each participation the participant has to create a VRF tag τ for a specific task T and prove correctness with a NIZK. Along with the participation, the participant sends a partially blinded coin to the organizer using PBSR.Blind. The coin consists of the identity un and a nullifier $\mathcal{N}$, sometimes called serial number. Once the task is fulfilled, the organizer together with the service signs coin with PBSR.Sign and adds the amount of reward the task provides $\mathsf{T}.v$ to get the coin, resulting in r and creates a reward transaction tx := (T, r, τ) which is appended to the service's public log BB. On payout, the participant reveals the nullifiers $\mathcal{N}_i$ and identity un and generates a NIZK which proves they are correct and the earned reward is greater than the payout. Double spending is prevented by the service which only accepts a nullifier once. Balance per user is assured by only paying out the amount to the identity of the coins. Due to the re-randomization of the coin, it cannot link it to the transaction where it was issued.

4.1 Detailed Protocol Description

To achieve payout privacy, the participant must not disclose their number of participations. To hide the real number of participations, we assume a system parameter n denoting the maximum number of tasks to get paid out in one payout. The Setup takes a security parameter 1^λ, the number of attributes m, which is increased by one to accommodate the identity, and maximum input size for payouts n. It initializes all building blocks and databases UN, BB, NUL, and outputs a set of public parameters p. After the initial setup and key generation of the service, participants can register (c.f., $\Pi_{\mathsf{Participate}}$) to retrieve an anonymous credential cred. For this they run VRF.KeyGen to pick a random secret key sk for later use in the verifiable random function, blind it with randomness $\rho \in \mathbb{R}_{\mathsf{PBSC}}$ with sk being the blinded message and attr (including un) the public part, using the partially blind signature scheme PBSC.Blind. They send the blinded key α with their identity un and an argument of knowledge π of a correct blinding ($\mathcal{L}_{\mathsf{PBSC}}$) to the service. The service verifies this argument $\mathsf{NIZK}[\mathcal{L}_{\mathsf{PBSC}}].\mathsf{Verify}(\pi, (\alpha, \mathsf{attr})) = 1$ and if it holds, the attributes are correct and the identity has not been registered yet (i.e., $\mathsf{un} \notin \mathsf{UN}$ where UN is a set of registered users), it continues to sign the blinded seed α using its secret key sk_S. The identity un is added to the set of registered users UN and the signature σ' is send back to the participant, who is then able

General Algorithms

```
$\mathsf{Setup}(1^\lambda, m, n)$
  $\mathsf{UN} \leftarrow \emptyset, \mathsf{BB} \leftarrow [\,], \mathsf{NUL} \leftarrow \emptyset$
  $\mathsf{p}_{\mathsf{VRF}} \leftarrow \mathsf{VRF.Setup}(1^\lambda)$
  $\mathsf{p}_{\mathsf{KDF}} \leftarrow \mathsf{KDF.Setup}(1^\lambda)$
  $\mathsf{p}_{\mathsf{PBSC}} \leftarrow \mathsf{PBSC.Setup}(1^\lambda, 1, m+1)$
  $\mathsf{p}_{\mathsf{PBSR}} \leftarrow \mathsf{PBSR.Setup}(1^\lambda, 2, 1)$
  $\mathsf{p}_{\mathsf{Participate}} \leftarrow \mathsf{NIZK}[\mathcal{L}_{\mathsf{Participate}}].\mathsf{Setup}(1^\lambda)$
  $\mathsf{p}_{\mathsf{Payout}} \leftarrow \mathsf{NIZK}[\mathcal{L}_{\mathsf{Payout}}].\mathsf{Setup}(1^\lambda)$
  $\mathsf{p} := (n, \mathsf{p}_{\mathsf{VRF}}, \mathsf{p}_{\mathsf{KDF}}, \mathsf{p}_{\mathsf{PBSC}}, \mathsf{p}_{\mathsf{PBSR}}, \mathsf{p}_{\mathsf{Participate}}, \mathsf{p}_{\mathsf{Payout}})$
  return p
```

```
$\mathsf{ChkCred}(\mathsf{cred}, \mathsf{pk_S})$
  $(\mathsf{sk}, \mathsf{attr}', \mathsf{un}', \sigma) \leftarrow \mathsf{cred}$
  return $\mathsf{PBSC.Verify}(\mathsf{pk_{S,C}}, \mathsf{sk}, \mathsf{attr}' \| \mathsf{un}', \sigma)$
```

```
$\mathsf{ChkPart}(\mathsf{cred}, \mathsf{tx})$
  return $\mathsf{tx}.\tau = \mathsf{VRF.Eval}(\mathsf{cred.sk}, \mathsf{tx.T})$
```

```
$\mathsf{ChkQual}(\mathsf{cred}, \mathsf{BB}, \mathsf{T})$
  $\tau \leftarrow \mathsf{VRF.Eval}(\mathsf{cred.sk}, \mathsf{T})$
  $Q \leftarrow \{\mathsf{VRF.Eval}(\mathsf{cred.sk}, \mathsf{T}.\bar{\delta}_i)\}_{i=1}^{|\mathsf{T}.\bar{\delta}|}$
  $D \leftarrow \{\mathsf{VRF.Eval}(\mathsf{cred.sk}, \mathsf{T}.\delta_i)\}_{i=1}^{|\mathsf{T}.\delta|}$
  $X := \{\mathsf{BB}_i.\tau\}_{i=1}^{|TX|}$                  ▷ set of rewarded tags in BB
  Assert $\tau \notin X$                              ▷ already participated in T
  Assert $X \cap Q = Q$                              ▷ qualifier not satisfied
  Assert $X \cap D = \emptyset$                      ▷ participated in disqualifier
  Assert $\mathsf{Satisfy}(\mathsf{cred.attr}, \mathsf{T.constr}) = 1$     ▷ attributes fulfill
  constraints
```

```
$\mathsf{Receive}(\mathsf{cred}, \mathsf{BB}, \mathsf{NUL})$
  $a \leftarrow 0, R \leftarrow \emptyset$
  for all $\mathsf{tx} \in \mathsf{BB}$ do
    if $\mathsf{ChkPart}(\mathsf{cred}, \mathsf{tx}) = 1$ then
      $(\mathcal{N}, \rho) \leftarrow \mathsf{KDF}(\mathsf{cred}.sk, \mathsf{tx.T})$
      if $\mathsf{PBSR.Verify}(\mathsf{pk_{S,R}}, (\mathcal{N}, \mathsf{cred.un}), (\mathsf{tx.T}.v), \mathsf{tx}.r) =$
  $1$ then
        if $\mathcal{N} \notin \mathsf{NUL}$ then
          $a \leftarrow a + \mathsf{tx.T}.v$
          $R \leftarrow R \cup \{\mathsf{tx}\}$
  return $(a, R)$
```

Service Algorithms

```
$\mathsf{KeyGen_S}()$
  $(\mathsf{sk_{S,C}}, \mathsf{pk_{S,C}}) \leftarrow \mathsf{PBSC.KeyGen}()$
  $(\mathsf{sk_{S,R}}, \mathsf{pk_{S,R}}) \leftarrow \mathsf{PBSR.KeyGen}()$
  $\mathsf{sk_S} \leftarrow (\mathsf{sk_{S,C}}, \mathsf{sk_{S,R}})$
  $\mathsf{pk_S} \leftarrow (\mathsf{pk_{S,C}}, \mathsf{pk_{S,R}})$
  return $(\mathsf{sk_S}, \mathsf{pk_S})$
```

```
$\Pi_{\mathsf{Register,S}}(\mathsf{sk_S})$
  $(\mathsf{un}, \mathsf{attr}, \alpha, \pi) \leftarrow$ recv
  Assert $\mathsf{NIZK}[\mathcal{L}_{\mathsf{PBSC}}].\mathsf{Verify}(\pi, (\alpha)) = 1$
  Assert $\mathsf{un} \notin \mathsf{UN}$
  Assert attr correct
  $\sigma' \leftarrow \mathsf{PBSC.Sign}(\mathsf{sk_{S,C}}, \alpha, \mathsf{attr} \| \mathsf{un})$
  $\mathsf{UN} \leftarrow \mathsf{UN} \cup \{\mathsf{un}\}$
  send$(\sigma')$
```

```
$\Pi_{\mathsf{Participate,S}}(\mathsf{sk_S})$
  $(\pi, \mathsf{T}, \tau, r') \leftarrow$ recv.P
  $\mathsf{stmt} := (\mathsf{pk_S}, \mathsf{BB}, \mathsf{T}, \tau, r')$
  Assert $\mathsf{NIZK}[\mathcal{L}_{\mathsf{Participate}}].\mathsf{Verify}(\pi, \mathsf{stmt}) = 1$
  $r \leftarrow \mathsf{PBSR.Sign}(\mathsf{sk_{S,R}}, r', (\mathsf{T}.v))$
  $\mathsf{BB} \leftarrow \mathsf{BB} \| (\mathsf{T}, r, \tau)$
```

```
$\Pi_{\mathsf{Payout,S}}(\mathsf{BB}, \mathsf{sk_S})$
  $(\{r'_i, \pi'_i\}_{i=1}^{n}) \leftarrow$ recv
  for all $i \in n$ do                        ▷ sign padding coins
    Assert $\mathsf{NIZK}[\mathcal{L}_{\mathsf{PBSR}}].\mathsf{Verify}(\pi'_i, r'_i) = 1$
    $v_i \leftarrow 0$
    $r''_i \leftarrow \mathsf{PBSR.Sign}(\mathsf{sk_{S,R}}, r'_i, (v_i))$
  send$(\{r''_i\}_{i=1}^{n})$
  $(\pi, \{\mathcal{N}_i\}_{i=1}^{n}, \mathsf{un}, v) \leftarrow$ recv
  Assert
  $\mathsf{NIZK}[\mathcal{L}_{\mathsf{Payout}}].\mathsf{Verify}(\pi, (\{\mathcal{N}_i\}_{i=1}^{n}, \mathsf{un}, v)) = 1$
  Assert $\mathsf{NUL} \cap \{\mathcal{N}_i\}_{i=1}^{n} = \emptyset$
  $\mathsf{NUL} \leftarrow \mathsf{NUL} \cup \{\mathcal{N}_i\}_{i=1}^{n}$
  offline payment of amount $v$ to un
```

Participant Algorithms

```
$\Pi_{\mathsf{Register,P}}(\mathsf{un}, \mathsf{attr}, \mathsf{pk_S})$
  $\mathsf{sk} \leftarrow \mathsf{VRF.KeyGen}()$
  $\rho \xleftarrow{\$} \mathbb{R}_{\mathsf{PBSC}}$
  $\alpha \leftarrow \mathsf{PBSC.Blind}((\mathsf{sk}), \rho)$
  $\mathsf{stmt} := (\alpha)$
  $\mathsf{wit} := (\mathsf{sk}, \rho)$
  $\pi \leftarrow \mathsf{NIZK}[\mathcal{L}_{\mathsf{PBSC}}].\mathsf{Prove}(\mathsf{stmt}, \mathsf{wit})$
  send$(\mathsf{un}, \mathsf{attr}, \alpha, \pi)$               ▷ with external auth token
  $\sigma' \leftarrow$ recv
  $\sigma \leftarrow \mathsf{PBSC.Unblind}(\mathsf{pk_{S,C}}, \rho, \sigma')$
  $\mathsf{cred} := (\mathsf{sk}, \mathsf{attr}, \mathsf{un}, \sigma)$
  return cred
```

```
$\Pi_{\mathsf{Participate,P}}(\mathsf{cred}, \mathsf{BB}, \mathsf{T})$
  $\tau \leftarrow \mathsf{VRF.Eval}(\mathsf{cred.sk}, \mathsf{T})$
  $(\mathcal{N}, \rho) \leftarrow \mathsf{KDF}(\mathsf{cred.sk}, \mathsf{T})$
  $r' \leftarrow \mathsf{PBSR.Blind}((\mathcal{N}, \mathsf{cred.un}), \rho)$
  $\mathsf{stmt} := (\mathsf{pk_S}, \mathsf{BB}, \mathsf{T}, \tau, r')$
  $\mathsf{wit} := (\mathsf{cred}, \mathcal{N}, \rho)$
  $\pi \leftarrow \mathsf{NIZK}[\mathcal{L}_{\mathsf{Participate}}].\mathsf{Prove}(\mathsf{stmt}, \mathsf{wit})$
  send$(\pi, \mathsf{T}, \tau, r')$
```

```
$\Pi_{\mathsf{Payout,P}}(\mathsf{cred}, \mathsf{BB}, \mathsf{NUL}, v, \{\mathsf{tx}_i\}_{i=1}^{k})$
  Assert $k \leq n$
  Assert $\mathsf{Receive}(\mathsf{cred}, [\mathsf{tx}_i]_{i=1}^{k}, \mathsf{NUL}).v \geq v$
  for all $i \in [n]$ do                        ▷ prepare padding tx
    $(\mathcal{N}_i, \rho_i) \xleftarrow{\$} \mathbb{R}_{\mathsf{KDF}}$
    $r'_i \leftarrow \mathsf{PBSR.Blind}((\mathcal{N}_i, \mathsf{cred.un}), \rho_i)$
    $\pi'_i \leftarrow \mathsf{NIZK}[\mathcal{L}_{\mathsf{PBSR}}].\mathsf{Prove}((r'_i), (\mathcal{N}_i, \mathsf{cred.un}, \rho_i))$
  send$(\{(r'_i, \pi'_i)\}_{i=1}^{n}, )$
  $(\{r''_i\}_{i=1}^{n}) \leftarrow$ recv
  for all $i \in [n]$ do                        ▷ prepare spendable tx
    if $i \in [k]$ then
      $(\mathcal{N}_i, \rho_i) \leftarrow \mathsf{KDF}(\mathsf{cred}.sk, \mathsf{tx}_i.\mathsf{T})$
      $r_i \leftarrow \mathsf{PBSR.Unblind}(\mathsf{pk_{S,R}}, \rho_i, \mathsf{tx}_i.r)$
      $v_i \leftarrow \mathsf{tx}_i.\mathsf{T}.v$
    else
      $r_i \leftarrow \mathsf{PBSR.Unblind}(\mathsf{pk_{S,R}}, \rho_i, r''_i)$
      $v_i \leftarrow 0$
  $\mathsf{stmt} := (\{\mathcal{N}_i\}_{i=1}^{n}, \mathsf{cred.un}, v)$
  $\mathsf{wit} := (\{r_i, v_i\}_{i=1}^{n})$
  $\pi \leftarrow \mathsf{NIZK}[\mathcal{L}_{\mathsf{Payout}}].\mathsf{Prove}(\mathsf{stmt}, \mathsf{wit})$
  send$(\pi, \{\mathcal{N}_i\}_{i=1}^{n}, \mathsf{cred.un}, v)$
```

Figure 3: The PrePaMS Construction.

to unblind the signature with PBSC.Unblind and store it as part of its secret credential cred. The validity of the credential cred is checked by using PBSC.Verify.

On participation, the participant P provides a NIZK of a valid credential cred that is eligible to participate in the chosen task T. For this construction, the argument shows four properties (c.f., ChkQual): (1) P has not yet participated in T, (2) cred satisfies the attributes required by T, (3) P has participated in all qualifier of T, (4) P has not yet participated in any disqualifier of T. In our concrete instantiation, we decided to support range (from lower l to upper u) and set constraints (in V), such that attributes attr satisfy a set of constr defined as:

$$\mathsf{Satisfy}(\mathsf{attr}, \mathsf{constr}) = \bigwedge_{c_i \in \mathsf{constr}} \begin{cases} \mathsf{attr}_j \in [l, u] \text{ if } (l, u, j) \leftarrow c_i \\ \mathsf{attr}_j \in V \text{ if } (V, j) \leftarrow c_i \end{cases}$$

To track previous participations, we utilize a verifiable random function VRF to derive a participation tag τ for the label of a task, while allowing a participant to prove the correctness of the tag via the service's signature on the VRF seed. This combination of anonymous credentials and verifiable random functions was previously proposed by Hohenberger et al. [22] in their ANONIZE scheme. Similar to ANONIZE, a participant re-randomizes their signature and then provides a zero knowledge proof that the participation tag τ corresponding to the task T is computed correctly. Verifying that the participant has not yet participated in T is as simple as a lookup that the tag τ is not part of any reward transaction in BB.

While ANONIZE only allows participation predicates based on attributes of the credential, we additionally enable qualifiers and disqualifiers based on previous participations by computing the relevant tags for all qualifiers and disqualifiers. To not link this participation to any previous or future participations that include these tags, we construct further NIZKs to satisfy the qualifiers, disqualifiers and attribute constraints without revealing the corresponding tags and attributes. For a *qualifier* we derive the respective tag under the required task and show that it is derived using the same seed as τ and that there exists a reward transaction in BB that stored this tag. Essentially, providing a ring-signature on the set of referenced tags for this task. For *disqualifiers*, a random, secret value is picked that is used to randomize the set of referenced tags of the disqualifying task in BB. We use a homomorphic randomization and show the correctness of the randomization in the NIZK.

The participant then computes their own tag for the disqualifying task, again showing that the same secret key is used, and applies the same randomization as for the set of disqualifying tags. The randomized set and the participant's own tag is then included in the argument to allow a verifier to check that it is not included, without linking a future participation in the disqualifying task to the current participation. For an *attribute constraint*, we prove that an intermediate blinding commitment holds the same attributes as the signed credential and then use a range or set membership bulletproof to show that a given attribute satisfies the constraint. The NIZK shows knowledge of a valid credential being used to generate a valid participation and shows fulfillment of all preconditions: $\mathcal{L}_{\mathsf{Participate}} :=$

$$\left\{ \begin{array}{l} \left(\mathsf{pk}_\mathsf{S}, \mathsf{BB}, \mathsf{T}, \tau, r'\right) \mid \exists (\mathsf{cred}, \mathcal{N}, \rho) : \\ \quad r' = \mathsf{PBS.Blind}((\mathcal{N}, \mathsf{cred.un}), \rho) \wedge \mathsf{ChkCred}(\mathsf{cred}, \mathsf{pk}_\mathsf{S}) = 1 \\ \quad \wedge\, \mathsf{ChkPart}(\mathsf{cred}, (T, \cdot, \tau)) = 1 \wedge \mathsf{ChkQual}(\mathsf{cred}, \mathsf{BB}, \mathsf{T}) = 1 \end{array} \right\}$$

To be able to later claim a reward, the participant derives a nullifier $\mathcal{N}$ and a blinding factor ρ for this participation based on their secret key cred.sk and the task T. The participant uses PBSR to blind $\mathcal{N}$ and un to r' and sends the blinded coin r' together with their participation tag τ and a NIZK of $\mathcal{L}_{\mathsf{Participate}}$ to the organizer of this task. The organizer then verifies the argument $\mathsf{NIZK}[\mathcal{L}_{\mathcal{L}_{\mathsf{Participate}}}].\mathsf{Verify}$ and if successful signs the coin of reward v using the service's secret key sk_S. In the implementation, we provide organizers access to the service signing key through an API which also keeps track of billing the organizers. The reward transaction tx includes the participation tag τ, the signed coin r, and the task T (i.e., $\mathsf{tx} := (\mathsf{T}, r, \tau)$).

After participating in one or more tasks, a participant can request a payout of earned rewards using Π_{Payout}. In this process, the participant has to disclose their identity un to receive payment outside the system. Anonymous payouts would make the balance property invalid, as rewards could be shared between participants. In order to not link their identity to previous participations, the reward transactions in BB that the participant wants to spend are kept private. A NIZK of $\mathcal{L}_{\mathsf{Payout}}$ proves knowledge of correct nullifiers $\mathcal{N}_i$ and identity un for the coins r_i of reward transactions tx_i, with rewards v_i that sum up to greater or equal to the payout amount v. This is required because a unique reward value could otherwise be used to link participations to payouts. The identity un has to be the same in all coins and equal to the payout recipient. To also keep the number of transactions that are spent private, the participant is expected to pad the spent coins with up to n fake coins $r_1, \ldots, r_n$ with a fixed reward of zero, a random nullifier $\mathcal{N}_i$ and their identity un. The participant generates the new coins r'_i, blinds them, and sends them to the service along with a proof π'_i of $\mathcal{L}_{\mathsf{PBSR}}$ to show they have zero value. The service verifies the proofs π'_i, signs these coins with PBSR.Sign, and sends them back. The padding is necessary because the proof size $|\pi|$ is proportional to the number of rewards paid out. If the participant wants to use less than n inputs, the remaining inputs are from the padded coins.

To prevent users from getting one reward paid out multiple times, the nullifier $\mathcal{N}$ is published to the service. The deterministic nullifier is detectable if the same coin is used twice. Again, showing

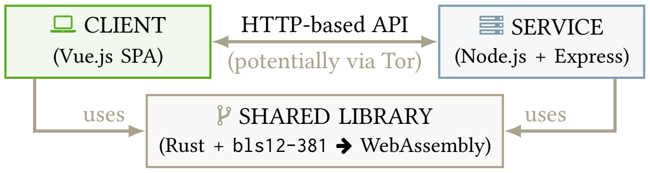


Figure 4: Graphical overview of the prototype architecture.

the correctness of the coins is performed with a NIZK. $\mathcal{L}_{\mathsf{Payout}} :=$

$$\left\{ \begin{array}{l} \left(\{\mathcal{N}_i\}_{i=1}^n, \mathsf{un}, v\right) \mid \exists (\{(r_i, v_i)\}_{i=1}^n) : \\ \forall i \in [n] : \mathsf{PBSR.Verify}(\mathsf{pk}_\mathsf{S}, (\mathcal{N}_i, \mathsf{un}), v_i, r_i) = 1 \wedge \sum_{i=1}^n v_i \geq v \end{array} \right\}$$

The service verifies the argument of knowledge π and checks that none of the nullifier have been recorded before in NUL and all identites match. Finally, it issues the payment outside of the system and adds the new nullifiers $\mathcal{N}$ to the database $\mathsf{NUL} \leftarrow \mathsf{NUL} \cup \mathcal{N}$. The detailed constructions for the NIZKs are presented in Section D. The analysis of security and privacy properties of the construction is presented in Section D.3.

5 Prototype

After showing that our scheme is cryptographically secure, we demonstrate the practicality of our approach with a proof of concept implementation. Our motivation for this prototype is to provide a realistic deployment for the use case of managing psychological study participations at a university, as introduced in Section 2. Hence, we implemented our PrePaMS system using modern web technologies (see Figure 5) to enable usage by participants on a variety of devices including mobile devices without the need to install a native application.

5.1 Implementation

Although modern browsers natively support some high-level cryptographic operations using the Web Cryptography API [43], this is insufficient for pairing-based cryptography or zero-knowledge proofs. While this cryptography could be implemented purely in JavaScript, we opted to implement the cryptographic components of our PrePaMS scheme in Rust and compile it to WebAssembly. This results in better performance and type safety, in contrast to a pure JavaScript implementation, additionally following the paradigm of a small auditable cryptography core. We chose the BLS12-381 pairing-friendly elliptic curve that is, for example, used by the privacy-focused cryptocurrency Zcash with an estimated security level of 123 Bit [42]. The main client-side application is implemented as a single page application using the Vue.js framework, and the service application is built with Node.js and the `Express` web application framework. This allows client and service to share the same WebAssembly library for cryptographic operations. The prototype architecture is shown in Figure 4.

For the proof of concept, we implemented a basic authentication system that accepts any username that is not yet registered. In a production deployment, this is meant to be replaced by an institutional authentication provider (e.g., through OAuth, SAML). Using the Fixed Participant State (Theorem 3.5), we can derive the secret keys from a user's password, which is never shared with the service. This allows users to reconstruct any private local state when switching devices without the need to store sensitive user state in a central database (e.g., the service).

In online-only studies, such as web-based surveys, the messages of the $\Pi_{\text{Participate}}$ protocol are easily submitted directly from the browser to the organizer-managed survey system. However, in-person studies enable more complex setups and human interaction with the participant. For this, we allow participation requests to be uploaded to the service in encrypted form. The corresponding secret to open the encrypted participation is then given to the organizer out of band (e.g., presenting a QR code during an offline lab experiment, c.f.,, Figure 5).

After the initial registration, participants are not required to have an authenticated session with the service. Instead, the blindly-signed VRF secret key is used as an anonymous credential to authenticate participations and payouts. This theoretically achieves the desired privacy guarantees. However, in practice, the service may use side channel information to obtain additional information about the participants. To not disclose the specific studies a participant is interested in, we use the private information retrieval [14] pattern when accessing study details by requesting all public data or a random subset from the service. Metadata, such as the client IP address of users interacting with the system, is accumulated by the service operator and potentially used to de-anonymize participants. As addressed in Section 3, this is mitigated by utilizing anonymous communication technologies such as Tor.

Even though all private state of users is recoverable from public information, a local cache is beneficial for performance reasons. This sensitive user state is only stored by the user's browser and not shared with the service. To further protect the state, it is locally encrypted with the user's password. If a user is inactive for some time, the unencrypted state is discarded, and the user will be prompted for their password. This is particularly useful when users access the service from shared devices, such as a computer lab on campus, and may forget to log out afterward.

During registration, a recovery QR code is generated, allowing users to restore their account in case they forget their password. In the future, the authentication and recovery process could be further improved using Web Authentication Credentials [24], also commonly known as Passkeys. This API enables credential generation and synchronization based on established authentication patterns, such as a device's biometric capabilities. At the time of writing, the API does not allow the export of key material for security reasons and can not be used to derive suitable key material for PrePaMS. But, for example, the proposed pseudo-random function extension [24, § 10.1.4.], could potentially be used to derive PrePaMS credentials.

In our prototype, the service also provides a partially trusted bulletin board with append-only semantics. In practice, the service could be operated by an institution's accounting department or comparable entity. Alternatively, an append-only transparency log system, similar to certificate transparency [27], can strengthen these guarantees.

In total, we implemented the PrePaMS client in 2356 lines of JavaScript code, the service in 569 lines of JavaScript code, performing little business logic but mostly storing and serving data, and the shared crypto library in 4240 lines of Rust code. Client-side entities and the service backend are decoupled and interact via an HTTP-based API. The prototype implementation of PrePaMS is available under an open-source license[1]. Additionally, we deployed a live test instance that is publicly available[2].

5.2 Performance Evaluation

With our performance evaluation, we assess two main aspects of our scheme and the associated proof of concept implementation: (i) the computational impact of prerequisite complexity on performance and (ii) processing times when users interact with the system under realistic conditions. For the first aspect, we conducted a set of microbenchmarks to explore the influence of each parameter on the performance of the protocols. The results of these microbenchmarks matched the expectations based on the amount of curve operations (c.f., Section E). For the second aspect, we run benchmarks within an actual browser using a WebAssembly compilation of our library.

We have released all evaluation artifacts[3] under an open-source license, following the Popper convention [23] for reproducibility. This features an automated, docker-based setup that executes the experiments in a remote-controlled headless browser using Puppeteer[4]. The evaluation was conducted on a desktop computer with an Intel Core i7-7700 (quad-core with SMT; 3.60 GHz) CPU and 32 GB RAM, running Ubuntu 22.04 LTS (GNU/Linux 5.4.0) with a headless Chrome 108. Additionally, we included a semi-automated build of the evaluation[5] to be used on any device featuring a modern browser. We used this artifact to evaluate the performance on different end-user devices, more precisely, an Android smartphone (Pixel 6 w/ Android 14 and Chrome 121) and an Apple tablet (3rd Gen iPad Pro 11" w/ iPadOS 17.2 and Safari 172).

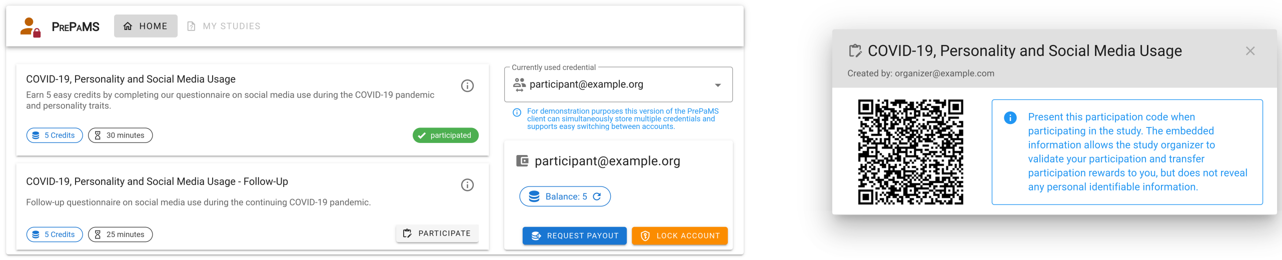
To put our system under load, we created a parametric workload generator. The generator outputs a chronological sequence of valid interactions with the system (i.e., registrations, participations, payouts) that can be replayed for reproducible evaluation runs. We defined suitable defaults and ranges for each parameter (i.e., distributions for each condition type) based on previous joint projects with psychology researchers to develop well-grounded workloads. The replay phase is preceded by an initialization phase to bootstrap a set amount of studies with randomized preconditions.

5.2.1 *Computational Impact of Prerequisite Complexity.*

In this evaluation, we wanted to assess the usage of varying amounts of different prerequisites in studies. We wanted to check whether the computational costs are reasonable enough to put our approach into practice.

Workload. We generated a corresponding series of workloads for each type of prerequisite (i.e., qualifier, disqualifier, attribute range constraint, set constraint). Each series contains a single study with a progressive increase in prerequisites, reaching a maximum of 10 per type. For qualifier and disqualifier workloads, we created additional studies necessary to reflect the dependencies of the target study. $N = 100$ random participants that satisfy the specified range and set constraints are created. If necessary, every participant first participates in all n qualifiers, or a set of n dummy participants participates in all n disqualifiers. Every participant then participates in the target study, recording the time it takes for the operation.

[3]*Evaluation artifacts:* https://github.com/vs-uulm/prepams/pets25.1/main/evaluation
[4]https://github.com/puppeteer/puppeteer
[5]*Browser benchmark suite:* https://vs-uulm.github.io/prepams/eval/



COVID-19, Personality and Social Media Usage

Earn 5 easy credits by completing our questionnaire on social media use during the COVID-19 pandemic and personality traits.

5 Credits | 30 minutes | participated

COVID-19, Personality and Social Media Usage - Follow-Up

Follow-up questionnaire on social media use during the continuing COVID-19 pandemic.

5 Credits | 25 minutes | PARTICIPATE

Currently used credential: participant@example.org

For demonstration purposes this version of the PrePAMS client can simultaneously store multiple credentials and supports easy switching between accounts.

participant@example.org

Balance: 5

REQUEST PAYOUT | LOCK ACCOUNT

COVID-19, Personality and Social Media Usage

Created by: organizer@example.com

Present this participation code when participating in the study. The embedded information allows the study organizer to validate your participation and transfer participation rewards to you, but does not reveal any personal identifiable information.

Figure 5: The system shows an overview of active studies, where a participant may participate in, and an overview of the participant's wallet with the option to request a payout of previously earned rewards. When participating in an offline study, the participant has to bring along the participation QR code, which can be scanned by the study organizer to confirm the participation and transfer the rewards without gaining additional information about the participant. For web-based studies, the participation data is directly sent to the organizer's survey system using an HTTP POST request. A public demo deployment is available[2] to interactively explore the prototype.

Before execution of the workload, a set of 50 additional participants follow the same steps—as warm-up and to fill the bulletin board. Any measurements of this warm-up phase are discarded.

Results. The results (see Figure 6) indicate a mostly linear scaling with an increasing number of (dis-)qualifiers with an exponential offset in irregular intervals.

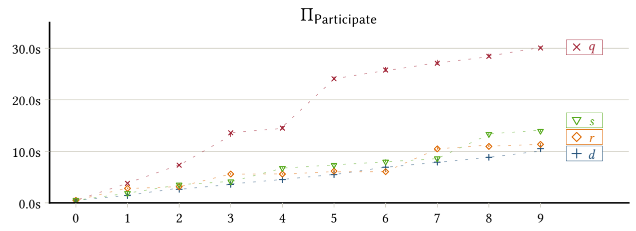


Figure 6: Plot of measured median execution times in seconds of the participation protocol based on a synthetic workload with either qualifier (×, q), disqualifier (+, d), range constraints (◇, r), or set constraints (▽, s) varied from $n \in [0..10]$ and all other parameters pinned to 0.

Discussion. This step-wise increase in computation time can be accounted for by the vector width of the inner product proof being padded to the next power of two. Each prerequisite type requires a different amount of group elements to be encoded in the inner product proof, yielding a different scaling behavior. Quantitative empirical studies require a minimum sample size to achieve statistically significant results. However, combining numerous prerequisites can make it difficult to recruit enough participants and thus impossible to achieve these sample sizes. For this reason, the number of prerequisites for psychological studies is usually in the lower single-digit range, which is also the cardinality our evaluation is grounded on. The results indicate that the scheme is also applicable for studies with non-trivial requirements that include multiple prerequisites.

5.2.2 Performance of User Operations under Realistic Conditions. In the second evaluation, we assessed the performance to be expected when users interact with our system. More precisely, we measured the execution times of three crucial operations (i.e., register, participate, and payout) based on actual user devices.

Workload. We synthesized a workload that matches our running example based on an analysis of real studies and their typical requirements. We limited the number of preconditions, i.e., (dis-)qualifier and attribute constraints, for a given study to at most 10 per type.

For the evaluation, we generated a workload with $N = 1,000$ participants, $M = 1,000$ participations, and $O = 100$ payout operations. Before execution, an additional set of $N = 200$ participants are registered and perform $M = 200$ participations and $O = 20$ payouts to fill the bulletin board and as a warm-up. Again, all warm-up measurements are discarded.

The resulting measurements provide an indication of the real-world performance of our scheme. In addition, other parameters can be easily explored with the open-source prototype and automated parametrized evaluation workflow.

Results. The execution times of the register, participate, and payout procedures are depicted in Figure 7. All participant measurements were executed separately using three different types of end-user devices (i.e., 💻 desktop, ▯ tablet, and ▯ smartphone).

Across devices the mean execution time of the registration procedure was $71.5ms$ ($\sigma = 28.6ms$). The cost of the participation procedures highly depends on the number of prerequisites of the respective study. For our workload, this resulted in a mean participation time of $624.0ms$ ($\sigma = 529.0ms$) with a mean proof size of $7.1kB$ ($\sigma = 2.6kB$). The execution time of the payout protocol is constant, depending only on the fixed number of maximum inputs, in our case, 10 study rewards. This resulted in a mean execution time of $8.4s$ ($\sigma = 5.2s$) with a mean payload size of $18.8kB$ ($\sigma = 4.1B$).

The average execution times on the service side for registration was $48.1ms$ ($\sigma = 4.25ms$), rewards issuance took an average of $26.0ms$ ($\sigma = 2.84ms$), and payout validation was performed in $1.4s$ ($\sigma = 61.3ms$) on average.

Discussion. The results indicate that the registration and most participation procedures are in the range of the typically accepted application response times of $100ms$ to $1s$ [34]. Depending on the number of prerequisites some participation proofs may take up to $4.1s$ to compute. However, we argue that this is still very acceptable in practice because the time-consuming procedures are limited to less frequent user interactions. The payout operation shows the highest execution times due to its costly NIZK. Nevertheless, it is the least frequently issued operation in the system (e.g., once in

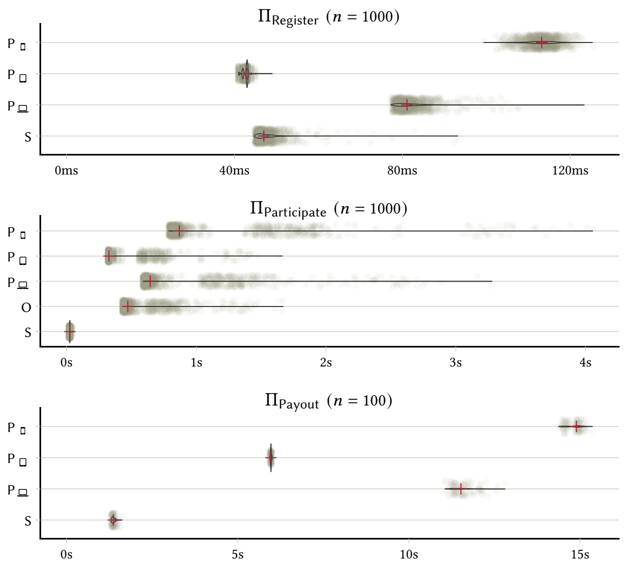


Figure 7: Combined violin/jitter plots of measured execution times (in seconds) of our PrePaMS proof of concept implementation based on a synthetic workload with $N = 1,000$ participants, $M = 1,000$ participations, and $O = 100$ payouts. The individual protocols are segmented by the role of the executing party (P: Participant, O: Organizer, S Service) and partially replicated across different device types (i.e., 💻 desktop, ▯ tablet, and ▯ smartphone).

a lifetime for a student account when claiming credits for subject hours), which makes an average of 14.8 seconds on smartphones a justifiable time. Additionally, progress bars are displayed during the expensive operations to indicate progress and bridge the waiting time. The execution time of the participation procedures scales with the amount of referenced (dis-)qualifiers. However, this is inherently bounded by the lifespan of a study.

The execution time of the service side is generally faster than the client-side. Participations, as the most commonly invoked operation, take only $26.0ms$ ($\sigma = 2.84ms$), which would enable 38.46 requests per second to the service with a single thread. Depending on the amount of users active in the systems, a multi-core CPU can be leveraged to process multiple participations simultaneously. In terms of our running example, this should allow a moderately sized university to offer this service to all students with a single server using commodity hardware.

5.3 Prototype Limitations

While our PrePaMS system enables privacy-enhanced participation management with rewards, we are aware of the following technical limitations of our web-based prototype.

Web applications have the benefit of being easily accessible without any prior installation and are supported in a wide variety of devices. However, they introduce additional attack vectors that local applications are not susceptible to. The client-side application logic — including critical code of cryptographic protocols — is fetched from the platform provider. Although transport layer security prevents other parties from modifying this code, this layer of protection does not prevent a rogue service provider from including a backdoor in the client-side code. Theoretically, the client application can be hosted separately from the service, allowing it to be hosted by a somewhat trusted external party. For example, a public GitHub repository with a continuous deployment on GitHub pages could mitigate some attacks. WAIT [31] showcases an alternative approach to verify web applications against a public transparency log, similar to certificate transparency [27]. Applying such an approach could mitigate more targeted attacks and enable detection of attacks by the platform provider.

As noted before, a service provider can also use available metadata, such as the client IP address of users, to potentially track and de-anonymize participants. When a participant is accessing the service via the institution's own network infrastructure, this metadata may already link the participant's identity (e.g., WiFi network with RADIUS-based authentication). This can be addressed using anonymous communication technologies, such as Tor. In certain circumstances, trusting a network operator might also be enough, especially when legal regulations like GDPR mandate data protection compliance.

6 Related Work

ANONIZE [22] is an anonymous survey system where a survey organizer can invite a set of participants to a survey based on their identity. Participation in a survey does not reveal their identity, and only permitted participants are accepted. PrePaMS builds on the combination of partially blind signatures and verifiable random functions used in ANONIZE and extends it with support for a privacy-preserving reward procedure and dynamic prerequisites/exclusion criteria dependent on previous participations.

ZebraLancer [29] and zkCrowd [48] are anonymous crowd-sensing systems utilizing smart contracts on a blockchain to exchange crowd-sourced data with rewards. These works are motivated by similar requirements as PrePaMS, i.e., to survey data in exchange for rewards. However, both systems only focus on rewards and do not support prerequisites and exclusion criteria of other surveys.

BLAC [39], EPID [9], PEREA [40], and FAUST [28] are anonymous credential schemes with blocklists where access to a service can be revoked for misbehaving users while maintaining privacy. PE(AR)2 [47], BLACR [3], PERM [2], EXBLACR [41], [32], and [15] extend the binary exclusion of the previous systems by a reputation-based scoring where only users with score greater than a defined threshold gain access to the service. DAC [18] and DBLACR [46] extend this with a decentralized registration. FARB [44] and Arbra [45] are centralized systems which allow more complex thresholds based on scores in different categories. Although these systems may be adapted to support dependencies on previous participations, they do not support the exclusion criteria supported by PrePaMS.

CLARC [7] was built for reviewing services after a verified purchase. It is based on anonymous credentials, which include signed attributes. Users can prove to service providers which attributes they hold (or a complex combination), allowing for flexible access control structures. The user gets a single-show token to rate the service if used correctly. Any double use to sway the opinion will make the reviews linkable through the equal token.

Untraceable Payments by David Chaum [12] introduced a payment scheme, also based on blind signatures, in which a central bank issues and accepts electronic coins. Users can receive coins and transfer them to merchants, who can then withdraw received coins from the bank without the bank learning the previous owner's identity. The bank can check for double-spending during withdrawal by comparing the coin to a list of spent coins. Electronic cash [13] and follow-up works [6, 11] introduced offline double-spending protection, where a merchant can accept payments without involving the bank. The scheme guarantees that this payment will either be honored by the bank or that the identity of the double spender will be revealed. This does not fit well with our participation protocol because the roles are reversed. Suppose an organizer double-spends a coin to two participants. In that case, the later payout operation will reveal the identity of the double-spender (i.e., the other participant or the organizer) and hence reveal the link between the participant and the study, which violates our privacy properties.

In summary, we acknowledge that there are multiple previous contributions solving adjacent or generalized problems, but all miss some functionality or trade performance for generality unnecessesary for our requirements. Our PrePaMS construction took inspiration from many of them.

7 Conclusion

In this paper, we have introduced PrePaMS, a privacy-preserving participation management system that also takes rewarding into account. We derived the requirements and features of such a system based on an analysis of existing systems as well as the involved parties, their motivations, and potential attackers. It supports analog and digital studies with single participations or follow up studies with pseudonymous participants. We then specified correctness, security, and anonymity properties for PrePaMS. Furthermore, we proposed a concrete instantiation of our PrePaMS scheme using the partially blind signature scheme by [22] and an anonymous payment system, which provably provides the security and anonymity properties of our formalization.

We implemented a proof-of-concept prototype[1] of PrePaMS and deployed a publicly available demo installation[2]. In our technical evaluation, we measured the performance impact of processing overhead and cryptographic operations and showed that the PrePaMS instantiation provides reasonable performance results.

We consider the current PrePaMS implementation a core building block for privacy-preserving participation management systems that could be extended and adapted for specific use cases. The credential management process can be simplified using proposed extensions of the Web Authentication API [24], utilizing FIDO Passkeys or other already available hardware authentication tokens. For instance, smart cards are widely available in many relevant scenarios — e.g., student identity cards at universities or employee identity cards in companies — and could be further incorporated into the system design as a source of key material.

PrePaMS only protects the participation privacy, but it does not take into account the actual study and the corresponding subject data (e.g., survey responses in an online study). Orthogonal solutions exist that target privacy-enhanced processes for data collection and analysis in empirical research, e.g., by extending the methodological practice of pre-registered studies with trusted computing concepts [30] or by enabling statistical analyses without revealing participant data to researchers using secure multiparty computation [26]. Integrating PrePaMS for participation management with a privacy-enhancing study platform facilitates the development of a comprehensive privacy-preserving study management system that covers the entire process of conducting quantitative empirical research studies while rewarding participants and upholding complete data privacy.

Outlook. One drawback of the presented implementation is that participants have to trust the organizer to pay after legitimate participation. Participants may contact the service provider and use the transcript of the interaction with an organizer to dispute the transaction, which harms their privacy. *Contingent payments* [33], where an organizer has to commit the transaction before retrieving the full response of a participant, could address this issue. For example, a two-phase protocol where a participant first commits a chunked encryption of their response, followed by opening a single uniformly random chunk to show the validity of their submission. If the decryption looks good, they proceed with the financial exchange so that spending the reward reveals the encryption keys to the organizer. However, this approach only works for online submissions and not necessarily for offline lab experiments. Nevertheless, here, a participant still has the option to not leave without a signature on the reward transaction.

Overall, PrePaMS contributes to making the scientific process more privacy friendly while maintaining all features and properties that scientists and study participants are used to in today's practice.

Acknowledgments

This work was partially supported by an internal research grant from Ulm University (*ProTrainU/21-22/PePER*). Part of this work by Felix Engelmann was done while at the ITU Copenhagen.

References

[1] Thomas Attema, Ronald Cramer, and Matthieu Rambaud. 2021. Compressed Σ-Protocols for Bilinear Group Arithmetic Circuits and Application to Logarithmic Transparent Threshold Signatures. In *Advances in Cryptology – ASIACRYPT 2021*, Mehdi Tibouchi and Huaxiong Wang (Eds.). Springer International Publishing, Cham, 526–556.

[2] Man Ho Au and Apu Kapadia. 2012. PERM: practical reputation-based blacklisting without TTPS. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security* (Raleigh, North Carolina, USA) *(CCS '12)*. Association for Computing Machinery, New York, NY, USA, 929–940. https://doi.org/10.1145/2382196.2382294

[3] Man Ho Au, Apu Kapadia, and Willy Susilo. 2012. BLACR: TTP-Free Blacklistable Anonymous Credentials with Reputation. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society. https://www.ndss-symposium.org/ndss2012/blacr-ttp-free-blacklistable-anonymous-credentials-reputation

[4] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. 2003. Constructing Elliptic Curves with Prescribed Embedding Degrees. In *Security in Communication Networks*, Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 257–267.

[5] Yehuda Baruch and Brooks C. Holtom. 2008. Survey response rate levels and trends in organizational research. *Human Relations* 61, 8 (2008), 1139–1160. https://doi.org/10.1177/0018726708094863

[6] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. 2009. Compact E-Cash and Simulatable VRFs Revisited. In *Pairing-Based Cryptography – Pairing 2009*, Hovav Shacham and Brent Waters (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 114–131. https://doi.org/10.1007/978-3-642-03298-1_9

[7] Kai Bemmann, Johannes Blömer, Jan Bobolz, Henrik Bröcher, Denis Diemert, Fabian Eidens, Lukas Eilers, Jan Haltermann, Jakob Juhnke, Burhan Otour, Laurens Porzenheim, Simon Pukrop, Erik Schilling, Michael Schlichtig, and

Marcel Stienemeier. 2018. Fully-Featured Anonymous Credentials with Reputation System. In *Proceedings of the 13th International Conference on Availability, Reliability and Security* (Hamburg, Germany) *(ARES 2018)*. Association for Computing Machinery, New York, NY, USA, Article 42, 10 pages. https://doi.org/10.1145/3230833.3234517

[8] Olaf Bock, Ingmar Baetge, and Andreas Nicklisch. 2014. hroot: Hamburg Registration and Organization Online Tool. *European Economic Review* 71 (2014), 117–120. https://doi.org/10.1016/j.euroecorev.2014.07.003

[9] Ernie Brickell and Jiangtao Li. 2007. Enhanced privacy id: a direct anonymous attestation scheme with enhanced revocation capabilities. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society* (Alexandria, Virginia, USA) *(WPES '07)*. Association for Computing Machinery, New York, NY, USA, 21–30. https://doi.org/10.1145/1314333.1314337

[10] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. 2018. Bulletproofs: Short Proofs for Confidential Transactions and More. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, Washington, D.C., USA, 315–334. https://doi.org/10.1109/SP.2018.00020

[11] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. 2005. Compact E-Cash. In *Advances in Cryptology – EUROCRYPT 2005*, Ronald Cramer (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 302–321. https://doi.org/10.1007/11426639_18

[12] David Chaum. 1983. Blind Signatures for Untraceable Payments. In *Advances in Cryptology*, David Chaum, Ronald L. Rivest, and Alan T. Sherman (Eds.). Springer US, Boston, MA, 199–203. https://doi.org/10.1007/978-1-4757-0602-4_18

[13] David Chaum, Amos Fiat, and Moni Naor. 1990. Untraceable Electronic Cash. In *Advances in Cryptology — CRYPTO' 88*, Shafi Goldwasser (Ed.). Springer New York, New York, NY, 319–327. https://doi.org/10.1007/0-387-34799-2_25

[14] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. 1998. Private information retrieval. *J. ACM* 45, 6 (nov 1998), 965–981. https://doi.org/10.1145/293347.293350

[15] Sherman S. M. Chow, Jack P. K. Ma, and Tsz Hon Yuen. 2023. Scored Anonymous Credentials. In *Applied Cryptography and Network Security*, Mehdi Tibouchi and XiaoFeng Wang (Eds.). Springer Nature Switzerland, Cham, 484–515.

[16] Yevgeniy Dodis and Aleksandr Yampolskiy. 2005. A Verifiable Random Function with Short Proofs and Keys. In *Public Key Cryptography - PKC 2005*, Serge Vaudenay (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 416–431. https://doi.org/10.1007/978-3-540-30580-4_28

[17] Susan Folkman. 2000. Privacy and confidentiality. In *Ethics in Research With Human Participants*, Bruce D Sales and Susan Folkman (Eds.). American Psychological Association, Washington, DC, USA, 49–57.

[18] Christina Garman, Matthew Green, and Ian Miers. 2014. Decentralized Anonymous Credentials. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society. https://www.ndss-symposium.org/ndss2014/decentralized-anonymous-credentials

[19] Mario Gollwitzer, Andrea Abele-Brehm, Christian Fiebach, Roland Ramthun, Anne M Scheel, Felix D Schönbrodt, and Ulf Steinberg. 2020. Data Management and Data Sharing in Psychological Science: Revision of the DGPs Recommendations. https://doi.org/10.31234/osf.io/24ncs

[20] C.J. Goodwin and K.A. Goodwin. 2016. *Research In Psychology Methods and Design*. Wiley, Hoboken, NJ, USA.

[21] Ben Greiner. 2004. *The Online Recruitment System ORSEE 2.0 - A Guide for the Organization of Experiments in Economics*. Working Paper Series in Economics 10. University of Cologne, Department of Economics. https://EconPapers.repec.org/RePEc:kls:series:0010

[22] Susan Hohenberger, Steven Myers, Rafael Pass, and abhi shelat. 2014. ANONIZE: A Large-Scale Anonymous Survey System. In *2014 IEEE Symposium on Security and Privacy*. IEEE, Washington, D.C., USA, 375–389. https://doi.org/10.1109/SP.2014.31

[23] Ivo Jimenez, Michael Sevilla, Noah Watkins, Carlos Maltzahn, Jay Lofstead, Kathryn Mohror, Andrea Arpaci-Dusseau, and Remzi Arpaci-Dusseau. 2017. The Popper Convention: Making Reproducible Systems Evaluation Practical. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, IEEE Computer Society, Washington, D.C., USA, 1561–1570. https://doi.org/10.1109/IPDPSW.2017.157

[24] Michael B. Jones, Akshay Kumar, and Emil Lundberg. 2022. *Web Authentication: An API for accessing Public Key Credential*. Editor's Draft. W3C. https://w3c.github.io/webauthn/

[25] Russell W. F. Lai, Giulio Malavolta, and Viktoria Ronge. 2019. Succinct Arguments for Bilinear Group Arithmetic: Practical Structure-Preserving Cryptography. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 2057–2074. https://doi.org/10.1145/3319535.3354262

[26] Andrei Lapets, Frederick Jansen, Kinan Dak Albab, Rawane Issa, Lucy Qin, Mayank Varia, and Azer Bestavros. 2018. Accessible Privacy-Preserving Web-Based Data Analysis for Assessing and Addressing Economic Inequalities. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies* (Menlo Park and San Jose, CA, USA) *(COMPASS '18)*. Association for Computing Machinery, New York, NY, USA, Article 48, 5 pages. https://doi.org/10.1145/3209811.3212701

[27] B. Laurie, A. Langley, and E. Kasper. 2013. *Certificate Transparency*. RFC 6962. Internet Engineering Task Force.

[28] Peter Lofgren and Nicholas Hopper. 2011. FAUST: Efficient, TTP-Free Abuse Prevention by Anonymous Whitelisting. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society* (Chicago, Illinois, USA) *(WPES '11)*. Association for Computing Machinery, New York, NY, USA, 125–130. https://doi.org/10.1145/2046556.2046572

[29] Yuan Lu, Qiang Tang, and Guiling Wang. 2018. ZebraLancer: Private and Anonymous Crowdsourcing System atop Open Blockchain. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE Computer Society, Washington, D.C., USA, 853–865. https://doi.org/10.1109/ICDCS.2018.00087

[30] Echo Meißner, Felix Engelmann, Frank Kargl, and Benjamin Erb. 2021. PeQES: A Platform for Privacy-Enhanced Quantitative Empirical Studies. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing* (Virtual Event, Republic of Korea) *(SAC '21)*. Association for Computing Machinery, New York, NY, USA, 1226–1234. https://doi.org/10.1145/3412841.3441997

[31] Echo Meißner, Frank Kargl, and Benjamin Erb. 2021. WAIT: Protecting the Integrity of Web Applications with Binary-Equivalent Transparency. In *The 36th ACM/SIGAPP Symposium on Applied Computing* (Virtual Event, Republic of Korea) *(SAC '21)*. ACM, New York, NY, USA, 1950–1953. https://doi.org/10.1145/3412841.3442143

[32] Toru Nakanishi and Takeshi Kanatani. 2020. Efficient blacklistable anonymous credential system with reputation using a pairing-based accumulator. *IET Information Security* 14, 6 (2020), 613–624. https://doi.org/10.1049/iet-ifs.2018.5505 arXiv:https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-ifs.2018.5505

[33] Ky Nguyen, Miguel Ambrona, and Masayuki Abe. 2020. WI is Almost Enough: Contingent Payment All Over Again. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, USA) *(CCS '20)*. Association for Computing Machinery, New York, NY, USA, 641–656. https://doi.org/10.1145/3372297.3417888

[34] Jakob Nielsen. 1994. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[35] Dale S. Rose, Stuart D. Sidle, and Kristin H. Griffith. 2007. A Penny for Your Thoughts: Monetary Incentives Improve Response Rates for Company-Sponsored Employee Surveys. *Organizational Research Methods* 10, 2 (2007), 225–240. https://doi.org/10.1177/1094428106294687

[36] Joan E. Sieber and Michael J. Saks. 1989. A census of subject pool characteristics and policies. *American Psychologist* 44, 7 (1989), 1053–1061. https://doi.org/10.1037/0003-066X.44.7.1053

[37] Eleanor Singer. 2018. The Use and Effects of Incentives in Surveys. In *The Palgrave Handbook of Survey Research*. Springer, Berlin, Heidelberg, 63–70. https://doi.org/10.1007/978-3-319-54395-6_9

[38] Ltd. Sona Systems. 2002–2024. Sona Systems: Cloud-based Participant Management Software. https://www.sona-systems.com/.

[39] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. 2007. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *Proceedings of the 14th ACM Conference on Computer and Communications Security* (Alexandria, Virginia, USA) *(CCS '07)*. Association for Computing Machinery, New York, NY, USA, 72–81. https://doi.org/10.1145/1315245.1315256

[40] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. 2008. PEREA: towards practical TTP-free revocation in anonymous authentication. In *Proceedings of the 15th ACM Conference on Computer and Communications Security* (Alexandria, Virginia, USA) *(CCS '08)*. Association for Computing Machinery, New York, NY, USA, 333–344. https://doi.org/10.1145/1455770.1455813

[41] Weijin Wang, Dengguo Feng, Yu Qin, Jianxiong Shao, Li Xi, and Xiaobo Chu. 2014. ExBLACR: Extending BLACR System. In *Information Security and Privacy*, Willy Susilo and Yi Mu (Eds.). Springer International Publishing, Cham, 397–412.

[42] Xiaofeng Wang, Peng Zheng, and Qianqian Xing. 2023. Security Analysis of Pairing-based Cryptography. *CoRR* abs/2309.04693 (2023), 23 pages. https://doi.org/10.48550/ARXIV.2309.04693 arXiv:2309.04693

[43] Mark Watson. 2017. *Web Cryptography API*. W3C Recommendation. W3C. https://www.w3.org/TR/2017/REC-WebCryptoAPI-20170126/.

[44] Li Xi and Dengguo Feng. 2014. FARB: Fast Anonymous Reputation-Based Blacklisting without TTPs. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society* (Scottsdale, Arizona, USA) *(WPES '14)*. Association for Computing Machinery, New York, NY, USA, 139–148. https://doi.org/10.1145/2665943.2665947

[45] Li Xi, Jianxiong Shao, Kang Yang, and Dengguo Feng. 2014. ARBRA: Anonymous Reputation-Based Revocation with Efficient Authentication. In *Information Security*, Sherman S. M. Chow, Jan Camenisch, Lucas C. K. Hui, and Siu Ming Yiu (Eds.). Springer International Publishing, Cham, 33–53.

[46] Rupeng Yang, Man Ho Au, Qiuliang Xu, and Zuoxia Yu. 2019. Decentralized blacklistable anonymous credentials with reputation. *Comput. Secur.* 85 (2019), 353–371. https://doi.org/10.1016/J.COSE.2019.05.009

Table 2: Variable names and usage.

un	participant identifier
$\mathsf{pk_S}$	service public key
attr	attributes of a participant, e.g. age
cred	anonymous credential
tx	reward transaction
BB	bulletin board, list of tx
T	task from domain $\mathbb{T}$
nul	reward nullifier
$\mathcal{N}$	set of nullifiers
$\bar{\delta}$	set of qualifiers
δ	set of disqualifiers
$\mathfrak{U}$	set of users managed by the oracles
$\mathfrak{T}$	bookkeeping of amount held by adversary
aux	channel for interactive adversary
UN	set of service registered usernames

[47] Kin Ying Yu, Tsz Hon Yuen, Sherman S. M. Chow, Siu Ming Yiu, and Lucas C. K. Hui. 2012. PE(AR)2: Privacy-Enhanced Anonymous Authentication with Reputation and Revocation. In *Computer Security – ESORICS 2012*, Sara Foresti, Moti Yung, and Fabio Martinelli (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 679–696. https://doi.org/10.1007/978-3-642-33167-1_39

[48] Saide Zhu, Zhipeng Cai, Huafu Hu, Yingshu Li, and Wei Li. 2020. zkCrowd: A Hybrid Blockchain-Based Crowdsourcing Platform. *IEEE Transactions on Industrial Informatics* 16, 6 (2020), 4196–4205. https://doi.org/10.1109/TII.2019.2941735

A Notation

In the following, we present a concrete group-level instantiation of our scheme components. Let $\mathbb{G}$ and $\mathbb{G}_2$ be paring-friendly cyclic groups of prime order q with generators g, g_2 where the discrete log assumption holds. Let $\mathbb{G}_T$ be a cyclic group of prime order q with an efficient bilinear non-degenerate mapping $e : \mathbb{G} \times \mathbb{G}_2 \to \mathbb{G}_T$, with generator $g_T := e(g, g_2)$. For our construction Setup produces a description of a Barreto, Lynn, Scott curve[4]. We follow the common multiplicative notation for group operations (e.g., $x, y \in \mathbb{G} : x \cdot y$ and $a \in \mathbb{Z}_q, x \in \mathbb{G} : x^a$). Let $\langle \vec{a}, \vec{b} \rangle = \sum_{i=1}^{n} a_i \cdot b_i$ denote the inner product of two vectors $a, b \in \mathbb{Z}_q^n$. For a vector of group elements $\vec{x}$ and a vector of scalars $\vec{x}$ we denote the element wise exponentiation as $\vec{a}^{\circ\vec{x}} = (a_1^{x_1}, a_2^{x_2}, \ldots, a_n^{x_n})$. The sum of all elements of a vector is abbreviated as $\sum \vec{a} = \sum_{a \in \vec{a}} a$, analogously the product of all elements of a vector is denoted as $\prod \vec{a}$.

There exists a cryptographic hash function $\mathsf{id} : \mathbb{T} \to \mathbb{Z}_q$ that maps tasks to a unqiue scalar. We also refer to the output of this function as the identifier of a study. Further, there exist cryptographic hash functions $H_1 : \mathsf{X} \to \mathbb{Z}_q$, $H_2 : \mathsf{X} \to \mathbb{R}_{\mathsf{PBS}}$ that map arbitraty length inputs X to scalars in the respective domains.

Table 2 provides an overview of variables used in our formalization and construction.

B Model Validation

To show that our formalization captures the rewarding and privacy properties of existing systems, we show that log sheets and stickers are partial instantiations of our model. These two decentralized systems are the only interesting, as the ones with a central database simply rely on a trusted server.

The log sheet system relies on unforgeable and unclonable (manual) signatures and uses the following construction:

$\mathsf{Setup}(1^\lambda, m, n) \to \mathsf{p}$**:** Create a sheet design with space for m attributes and n lines for signatures. This is p

$\mathsf{KeyGen_S}() \to (\mathsf{sk_S}, \mathsf{pk_S})$**:** Create an unforgeable signature ($\mathsf{sk_S}$) and publish how it looks like ($\mathsf{pk_S}$).

$\Pi_{\mathsf{Register}}\langle \mathsf{P}(\mathsf{un}, \mathsf{attr}), \mathsf{S}(\mathsf{sk_S}) \rangle \to (\mathsf{cred}, \mathsf{un})$**:** The participant identifies themselves (un) and receive a log sheet (cred) from the service with with their attributes attr filled in and signed with $\mathsf{sk_S}$. The service keeps track of the identity.

$\Pi_{\mathsf{Participate}}\langle \mathsf{P}(\mathsf{cred}, \mathsf{BB}, \mathsf{T}), \mathsf{S}(\mathsf{sk_S}) \rangle \to \mathsf{tx}$**:** The participant shows the log sheet cred to the organizer who uses the previous signatures on cred to check the qualifications and upon successful participation of T signs the sheet too (tx is a line on the sheet).

$\Pi_{\mathsf{Payout}}\langle \mathsf{P}(\mathsf{cred}, \mathsf{un}, \mathsf{BB}, v, \{\mathsf{tx}_i\}_{i=1}^{k}), \mathsf{S}() \rangle \to (\{\mathcal{N}_i\}_{i=1}^{n}, v, \mathsf{un})$**:** The participant hands the log sheet cred over to the service, who keeps it and pays out the amount signed by each organizer.

$\mathsf{Receive}(\mathsf{cred}, \mathsf{BB}, \mathsf{NUL}) \to (s, R)$**:** Add all plaintext rewards with signatures on the log sheet.

$\mathsf{ChkCred}(\mathsf{cred}, \mathsf{pk_S}) \to \{0, 1\}$**:** The sheet is authentic and has attributes signed with $\mathsf{pk_S}$.

$\mathsf{ChkPart}(\mathsf{cred}, \mathsf{tx}) \to \{0, 1\}$**:** Check if the log sheet cred has the signature for tx on it.

$\mathsf{ChkQual}(\mathsf{cred}, \mathsf{BB}, \mathsf{T}) \to \{0, 1\}$**:** Check for previous participations based on signatures on the log sheet.

The log sheet only fulfills the PartSec and Balance property, but obviously not the PartPriv. With unforgeable signatures, PartSec holds, because the organizers check the qualifications in plaintext and the participation for a given study requires a signature from the organizer. Similarly, to break the balance property, signatures need to be transferred between sheets, which physically cannot be done without cloning.

Complementary, the sticker based solution is more privacy friendly. It assumes unclonable, indistinguishable stickers and fits to our model as follows with an empty Setup:

$\mathsf{KeyGen_S}() \to (\mathsf{sk_S}, \mathsf{pk_S})$**:** Create an unforgeable sticker design ($\mathsf{sk_S}$) and publish how it looks like ($\mathsf{pk_S}$).

$\Pi_{\mathsf{Register}}\langle \mathsf{P}(\mathsf{un}, \mathsf{attr}), \mathsf{S}(\mathsf{sk_S}) \rangle \to (\mathsf{cred}, \mathsf{un})$**:** The participant identifies themselves (un) and receive an empty booklet (cred) from the service.

$\Pi_{\mathsf{Participate}}\langle \mathsf{P}(\mathsf{cred}, \mathsf{BB}, \mathsf{T}), \mathsf{S}(\mathsf{sk_S}) \rangle \to \mathsf{tx}$**:** Upon successful participation of T, the participant gets a sticker (tx).

$\Pi_{\mathsf{Payout}}\langle \mathsf{P}(\mathsf{cred}, \mathsf{un}, \mathsf{BB}, v, \{\mathsf{tx}_i\}_{i=1}^{k}), \mathsf{S}() \rangle \to (\{\mathcal{N}_i\}_{i=1}^{n}, v, \mathsf{un})$**:** The participant hands the booklet with a subset of their stickers over to the service, who keeps it and pays out the number of stickers.

$\mathsf{Receive}(\mathsf{cred}, \mathsf{BB}, \mathsf{NUL}) \to (s, R)$**:** Count the stickers in cred.

ChkCred, ChkPart and ChkQual are not possible.

Given indistinguishable stickers, the system has PartPriv. In $\Pi_{\mathsf{Participate}}$, no information is dependent on b. With unclonable stickers, Balance is only satisfied up to the first winning condition. The adversary is unable to copy any sticker and get payed out more than they earned. The second winning condition is breakable as the adversary can move stickers between users.

C Deferred Preliminaries

Here we present definitions of the preliminaries used in our construction.

C.1 Key Derivation Function

We use a key derivation function $\mathsf{KDF} : \varphi_{\mathsf{VRF}} \times \mathbb{T} \to \mathbb{S} \times \mathbb{R}_{\mathsf{PBS}}$ which takes a secret key from φ_{VRF} and a task from $\mathbb{T}$ to generate a nullifier $\mathcal{N}$ and a PBS blinding randomness. It consists of two algorithms:

$\mathsf{Setup}(1^\lambda) \to \mathsf{p}$**:** takes the security parameter 1^λ and outputs public parameters p.

$\mathsf{KDF}(s, \mathsf{T}) \to (\mathcal{N}, \rho)$**:** takes a secret $s \in \varphi_{\mathsf{VRF}}$ together with a task T and generates a nullifier $\mathcal{N}$ and ρ, a PBS blinding randomness.

As concrete instantiation, we use cryptographic hash functions H_1 and H_2, set $\mathbb{S} = \mathbb{Z}_q$, and calculate $\mathcal{N} := H_1(s\|\mathsf{id}(\mathsf{T}))$, $\rho := H_2(s\|\mathsf{id}(\mathsf{T}))$, and output $\mathsf{KDF}(s, \mathsf{T}) := (\mathcal{N}, \rho)$.

C.2 VRF scheme

A VRF enables generating unique, deterministic tags for a public key and a label. It consists of the following three algorithms:

$\mathsf{Setup}(1^\lambda) \to \mathsf{p}$**:** takes the security parameter 1^λ and outputs public parameters p.

$\mathsf{KeyGen}() \to (\mathsf{sk}, \mathsf{pk})$**:** generates a public private key pair sk, pk in the domain $\varphi_{\mathsf{VRF}} \times \Phi_{\mathsf{VRF}}$ with a helper function $\mathsf{Pub} : \varphi_{\mathsf{VRF}} \to \Phi_{\mathsf{VRF}}$ to calculate the corresponding public key to a secret key.

$\mathsf{Eval}(\mathsf{sk}, x) \to \tau$**:** takes a secret key sk and a label $x \in \mathbb{T}$ and outputs a tag τ from the domain χ_{VRF}.

The VRF scheme must satisfy the following properties:

Correctness: A honestly generated NIZK with $\mathcal{L}_{\mathsf{VRF}}$ for a tag τ and witness sk generated by Eval is valid.

Tag Pseudorandomness: A tag τ generated by Eval must only be predictable for the party knowing the secret key sk and looks uniformly random to everyone else.

Tag Uniqueness: For a given secret key sk and label x, there must exist one unique, valid tag τ

The detailed definitions of these properties together with a construction is presented in [16] which is summarized as:

$\mathsf{Setup}(1^\lambda)$**:** returns $(g, \mathbb{G}, q)$.

$\mathsf{KeyGen}() \to (\mathsf{sk}, \mathsf{pk})$**:** chooses $\mathsf{sk} \in \mathbb{Z}_q$, outputs sk, $\mathsf{pk} := g^{\mathsf{sk}}$.

$\mathsf{Eval}(\mathsf{sk}, x) \to \tau$**:** computes and outputs $\tau := g^{\frac{1}{\mathsf{sk}+\mathsf{id}(x)}}$.

In addition, we require an efficient NIZK for the correct evaluation. I.e., the relation $\mathcal{L}_{\mathsf{VRF}}((\mathsf{pk}, \tau, x)\exists(\mathsf{sk}) : \mathsf{pk} = \mathsf{Pub}(\mathsf{sk}) \wedge \tau = \mathsf{Eval}(\mathsf{sk}, x))$. A concrete instantiation of $\mathcal{L}_{\mathsf{VRF}}$ is adapted from [22]. The prover first picks a random blinding $b \xleftarrow{\$} \mathbb{Z}_q$ and commits on $E \leftarrow e(g, \tau)^b$. The commitment is sent to the verifier, who then replies with a challenge $c \xleftarrow{\$} \mathbb{Z}_q$. The response is computed as $z \leftarrow b + c \cdot \mathsf{sk}$ and sent to the verifier. Then the verifier verifies the proof by checking: $E \cdot e(g, g_2)^c \cdot e(g, \tau)^{-cx} = e(g_2, \tau)^z$

C.3 PBS scheme

To issue anonymous credentials with signer defined attributes and use them in a multi-show unlinkable way, we need a partially blind signature scheme. This hybrid of a regular digital signature and a blind signature, splits the message in a part that is visible to the signer and one that is hidden from the signer. We mostly follow the ANONIZE [22] definition using six probabilistic polynomial time algorithms, but generalize the signature to allow more than one hidden message, and more than one visible message:

PBS = (Setup, KeyGen, Blind, Sign, Verify, Unblind):

$\mathsf{Setup}(1^\lambda, m, n) \to \mathsf{p}$**:** takes a security parameter 1^λ, the number of hidden messages n, the number of visible messages m, and returns public parameters p.

$\mathsf{KeyGen}() \to (\mathsf{sk}, \mathsf{pk})$**:** generates a public private key pair to sign credentials $(\mathsf{sk}, \mathsf{pk}) \in \varphi_{\mathsf{PBS}} \times \Phi_{\mathsf{PBS}}$.

$\mathsf{Blind}(S, \rho) \to \alpha$**:** takes hidden message parts $S \in \mathbb{R}^n_{\mathsf{PBS}}$ and a blinding factor $\rho \in \mathbb{R}_{\mathsf{PBS}}$, and returns a blinded message α

$\mathsf{Sign}(\mathsf{sk}, \alpha, M) \to \sigma'$**:** takes a secret key sk, a blinded message α, and the public message parts $M \in \mathbb{R}^m_{\mathsf{PBS}}$ and outputs a signature σ'.

$\mathsf{Unblind}(\rho, \sigma') \to \sigma$**:** takes the blinding factor ρ and the blinded signature σ' and outputs the unblinded version σ.

$\mathsf{Verify}(\mathsf{pk}, S, M, \sigma) \to \{0, 1\}$**:** takes the public key pk, the secret and public message parts S, M, and a signature σ on it, and outputs 1 if the signature is valid and 0 otherwise.

A PBS scheme satisfies the following three properties:

Correctness: Every honestly blinded, signed, and unblinded signature verifies.

Partial Blindness: For multiple signatures that use the same public message part, a signer cannot link these signatures to the respective signing sessions.

Unforgeability: An adversary that interacts at most l times with the signer cannot produce more than l valid message-signature pairs.

We utilize the efficient pairing-based partially blind signature scheme by Hohenberger et al. [22]. We generalized their scheme, to allow for a fixed set of public and private messages.

$\mathsf{Setup}(1^\lambda, m, n) \to \mathsf{p}$**:** sample random group generators $\vec{U} \in \mathbb{G}^m$, $\vec{V} \in \mathbb{G}^n$, $h, g \in \mathbb{G}$, $g_2 \in \mathbb{G}_2$, outputs $(\vec{U}, \vec{V}, h, g, g_2)$ and sets the randomness space $\mathbb{R}_{\mathsf{PBS}} := \mathbb{Z}_q$.

$\mathsf{KeyGen}()$**:** choose a secret $\mathsf{sk} \in \mathbb{Z}_q$, compute $\mathsf{pk} \leftarrow e(g, g_2)^{\mathsf{sk}}$, and output (sk, pk).

$\mathsf{Blind}(\vec{S}, \rho)$**:** compute $\alpha \leftarrow \prod \vec{V}^{\circ\vec{S}} \cdot g^\rho$ and output α.

$\mathsf{Sign}(\mathsf{sk}, \alpha, M)$**:** choose random $w \in \mathbb{Z}_q$, compute $\sigma_1 \leftarrow g^{\mathsf{sk}}(\alpha \cdot h \cdot \prod \vec{U}^{\circ\vec{M}})^w$, $\sigma_2 \leftarrow g^w$, $\sigma_3 \leftarrow g_2^w$, and output $(\sigma_1, \sigma_2, \sigma_3)$.

$\mathsf{Unblind}(\rho, (\sigma_1, \sigma_2, \sigma_3))$**:** compute and output $\left(\sigma_1 \cdot \sigma_2^{-\rho}, \sigma_3\right)$.

$\mathsf{Verify}(\mathsf{pk}, \vec{S}, \vec{M}, (\sigma_1, \sigma_2))$**:** check if $\mathsf{pk} \cdot e(\prod \vec{V}^{\circ\vec{S}} \cdot \prod \vec{U}^{\circ\vec{M}} \cdot h, \sigma_2) = e(\sigma_1, g_2)$ and output 1, 0 otherwise.

Proofs for correctness, partial blindness, and unforgeability in the case $|\vec{V}| = |\vec{M}| = 1$ are described in [22, (Theorem 11, Thereom 10, Theorem 12)]. Assuming the hardness of DLP in $\mathbb{G}$, an instantiation with $|\vec{V}| = 2$ and $|\vec{M}| = 2$ is computationally indistinguishable from an instantiation with $|\vec{V}| = |\vec{M}| = 1$ if no discrete logarithm relation is known between generators $\vec{V}_1, \vec{V}_2, \vec{M}_1, \vec{M}_2$ (c.f., hiding property of Pedersen Commitment). Per induction this holds for $|\vec{V}| > 2$ and $|\vec{M}| > 2$.

In [22], the authors also describe an efficient Schnorr-based NIZK $\mathcal{L}_{\mathsf{PBS}}$ for a correct blinding, and a language $\mathcal{L}_{\mathsf{VerPBS}}$ for verifying the signature without revealing any part of the message. We adapated these again for a fixed set of public message parts $\vec{M}$ and private messsage parts $\vec{S}$:

$$\mathcal{L}_{\text{PBS}} := \left\{ \ (\alpha) \mid \exists(\vec{S},\rho) : \ \alpha = \prod \vec{V}^{\circ\vec{S}} \cdot g^{\rho} \ \right\}$$

The prover first picks blinding factors $\vec{B} \xleftarrow{\$} \mathbb{Z}_q^n$, $b \xleftarrow{\$} \mathbb{Z}_q$ and commits to $\gamma \leftarrow \vec{V}^{\circ\vec{B}} \cdot g^b$. The verifier receives α, γ and responds with a challenge $c \xleftarrow{\$} \mathbb{Z}_q$. The response is computed as:

$$\vec{Y} \leftarrow \left[\vec{B}_i + c \cdot \vec{S}_i\right]_{i=1}^{n} \qquad z \leftarrow b + c \cdot \rho$$

The verifier then verifies $\prod \vec{V}^{\circ\vec{Y}} \cdot g^z = \alpha^c \cdot \gamma$.

For the anonymous verification of a signature, we use

$$\mathcal{L}_{\text{VerPBS}} := \left\{ (\text{pk}) \mid \exists(\vec{S},\vec{M},\sigma) : \ \text{PBS.Verify}(\text{pk},\vec{S},\vec{M},\sigma) = 1 \right\}$$

The prover first re-randomizes their signature $\sigma := (\sigma_1, \sigma_2)$ by picking a new random $d \xleftarrow{\$} \mathbb{Z}_q$ and computing:

$$s_1 \leftarrow \sigma_1 \cdot \left(\prod \vec{V}^{\circ\vec{S}} \prod \vec{U}^{\circ\vec{M}} \cdot h\right)^d \qquad s_2 \leftarrow \sigma_2 \cdot g_2^d$$

They then picks random values $\vec{B} \xleftarrow{\$} \mathbb{Z}_q^m$ and a random group element $J \xleftarrow{\$} \mathbb{G}$ and compute a commitment as:

$$E \leftarrow e(J, g_2) \cdot e(\prod \vec{U}^{\circ\vec{B}}, s_2)^{-1}$$

The verifier receives (s_2, E) and responds with a challenge $c \xleftarrow{\$} \mathbb{Z}_q$. The prover computes a response:

$$\vec{Y} \leftarrow \left[\vec{B}_i + c \cdot \vec{M}_i\right]_{i=1}^{m} \qquad z \leftarrow {s_1}^c \cdot J$$

This can be verified by the verifier by checking:

$$E \cdot \text{pk}^c \cdot e(h, s_2)^c = e(z, g_2) \cdot e(\prod \vec{U}^{\circ\vec{Y}}, s_2)^{-1}$$

C.4 Non-Interactive Zero-Knowledge Proofs

We use NIZKs with the following three algorithms (Setup, Prove, Verify) that satisfy simulation-extractable soundness and simulatable zero-knowledge. As a construction, we rely on the extended bulletproofs from [25], but once there is an efficient implementation of a more efficient proving system, we can easily upgrade to e.g., compressed Σ-protocols [1].

Bulletproofs[10] allow a prover to convince a verifier that the inner product of two commited scalar vectors is equal to a value with logarithmic communication. Lai et al. [25] further describe an efficient outer protocol to prove knowledge of bilinear group arithmetic relations. We use the non-pairing part of [25] which allows proofs for a proving system of the following relation $\mathcal{L}_{\text{LRM19}} :=$

$$\left\{ \begin{array}{l} \vec{K} \in \mathbb{G}^n, \{\vec{v}_i^{(\prime)} \in \mathbb{Z}_q^m, \text{cls}_i \in \{\mathfrak{mul}, \mathfrak{dir}, \mathfrak{sum}, \mathfrak{one}\}, c_i \in \mathbb{Z}_q\}_{i=1}^{o} \\ \exists(\vec{c}_L, \vec{c}_R) \in (\mathbb{Z}_q^m \times \mathbb{Z}_q^m) : \ \prod_{i=1}^n K_i^{\vec{c}_{L,i}} = I := g^0 \\ \qquad \bigwedge_{i\in\{1,\dots,o\}} : \begin{cases} \langle \vec{c}_L, \vec{v}_i \rangle = c_i \text{ if } \text{cls}_i = \mathfrak{dir} \\ \langle \vec{c}_L, \vec{c}_R \circ \vec{v}_i \rangle = c_i \text{ if } \text{cls}_i = \mathfrak{mul} \\ \langle \vec{c}_L, \vec{v}_i \rangle + \langle \vec{c}_R, \vec{v}_i' \rangle = c_i \text{ if } \text{cls}_i = \mathfrak{sum} \\ \langle \vec{c}_L - \vec{c}_R - \vec{1}^m, \vec{v}_i \rangle = c_i \text{ if } \text{cls}_i = \mathfrak{one} \end{cases} \end{array} \right\}$$

with $n \leq m$ and a proof size of $5 \cdot |\mathbb{Z}_q| + (4 + 2 \cdot \lceil log_2(|\vec{K}|)\rceil) \cdot |\mathbb{G}|$. The number of necessary point additions and scalar multiplications in relation to the length of $\vec{K}$ is shown in Equations (1) to (4).

$$|\mathsf{p}_\times| = -1 + 4\,|\vec{K}| + 2 \cdot \lceil \log_2 |\vec{K}| \rceil + 2^{\lceil \log_2 |\vec{K}| \rceil} \tag{1}$$

$$|\mathsf{p}_+| = 7 + 4\,|\vec{K}| + 2 \cdot \lceil \log_2 |\vec{K}| \rceil + 26 \cdot 2^{\lceil \log_2 |\vec{K}| \rceil} \tag{2}$$

$$|\mathsf{v}_\times| = 12 + 2\,|\vec{K}| + 2 \cdot \lceil \log_2 |\vec{K}| \rceil + 2^{\lceil \log_2 |\vec{K}| \rceil} \tag{3}$$

$$|\mathsf{v}_+| = 17 + 2\,|\vec{K}| + 2 \cdot \lceil \log_2 |\vec{K}| \rceil + 21 \cdot 2^{\lceil \log_2 |\vec{K}| \rceil} \tag{4}$$

D NIZKs for our Languages

In this section we provide a concrete construction of the languages for the PrePaMS scheme on top of the aformentioned preliminaries.

D.1 Participation Proof

On a high-level, the participation proof requires a participant to convince an organizer that they have a valid, signed credential cred signed by the service's public key pk_S, provide a correct blinding for a reward bound to the user's credential, provide a correct participation tag τ for task T (both in the tuple tx), and meet the prerequisites for this task.

$$\mathcal{L}_{\text{Participate}} := \left\{ \begin{array}{l} (\text{pk}_\text{S}, \text{BB}, \text{T}, \tau, r') \mid \exists(\text{cred}, \mathcal{N}, \rho) : \\ \quad r' = \text{PBSR.Blind}((\mathcal{N}, \text{cred.un}), \rho) \\ \quad \wedge \text{ChkCred}(\text{cred}, \text{pk}_\text{S}) = 1 \\ \quad \wedge \text{ChkPart}(\text{cred}, \text{tx}) = 1 \\ \quad \wedge \text{ChkQual}(\text{cred}, \text{BB}, \text{tx.T}) = 1 \end{array} \right\}$$

For an efficient proof, we split this language into three sub languages: $\mathcal{L}_{\text{CheckCredTag}}$ for the anonymous credential including the participation tag, $\mathcal{L}_{\text{CheckQual}}$ for the prerequisites, and $\mathcal{L}_{\text{RewardBlinding}}$ for a correct reward blinding. These proofs are joined in an AND composition by concatenation and verifying that the tag τ in both statements is equal, thereby committing to the same secret key in both parts due to tag uniqueness. A vector commitment p to the values of the user's attributes and user identity is used to bind the values used in $\mathcal{L}_{\text{CheckQual}}$ to the correct values in cred proven in $\mathcal{L}_{\text{CheckCredTag}}$ and to the blinded reward request. It is blinded by a random $b_p \xleftarrow{\$} \mathbb{Z}_q$ and computed as $p \leftarrow g^{b_p} \cdot \prod \vec{U}^{\circ\text{attr}\|\text{un}}$.

$$\mathcal{L}'_{\text{Participate}} := \left\{ \begin{array}{l} (\text{pk}_\text{S}, \text{BB}, \text{T}, \tau, r', p) \mid \exists(\text{cred}, \mathcal{N}, \rho, b_p) : \\ \quad (\text{stmt} = (\text{pk}_\text{S}, \text{T}, \tau, p), \text{wit} = (\text{cred}, b_p)) \\ \qquad \in \mathcal{L}_{\text{CheckCredTag}} \\ \quad \wedge (\text{stmt} = (\text{BB}, \text{T}, \tau, p), \text{wit} = (\text{cred}, b_p)) \\ \qquad \in \mathcal{L}_{\text{CheckQual}} \\ \quad \wedge (\text{stmt} = (r', p), \text{wit} = (\text{cred}, \mathcal{N}, \rho, b_p)) \\ \qquad \in \mathcal{L}_{\text{RewardBlinding}} \end{array} \right\}$$

Credential and Participation Tag. Participation tags τ are derived from the participant's secret key cred.sk and a task T by computing $\tau = \text{VRF.Eval}(\text{cred.sk}, \text{T})$. The correctness of the participation tag is shown by proving that the VRF evaluates to the provided participation tag. The validity of the anonymous credential cred.σ can be expressed as the validity of PBS.Verify. Both participation tag τ and the commitment p are used to bind the statements of the three sublanguages. Thereby we formulate:

$\mathcal{L}_{\mathsf{CheckCredTag}} :=$

$$\left\{\begin{array}{l} (\mathsf{pk_S}, \mathsf{T}, \tau, p) \mid \exists(\mathsf{cred}, b_p) : \\ \quad p = g^{b_p} \cdot \prod \vec{U}^{\circ \mathsf{cred.attr} \| \mathsf{cred.un}} \\ \quad \wedge \mathsf{VRF.Eval}(\mathsf{cred.sk}, \mathsf{T}) = \tau \\ \quad \wedge \mathsf{PBSC.Verify}(\mathsf{pk_S}, \mathsf{cred.sk}, \mathsf{cred}.\sigma) = 1 \end{array}\right\}$$

Concrete NIZKs for the correctness of a VRF output and the validity of PBS signature have been given in section C. The PBSC instantiation has a single private message part $\vec{S} = (\mathsf{cred.sk})$, which also serves as the seed for the VRF.

$\mathcal{L}'_{\mathsf{CheckCredTag}} :=$

$$\left\{\begin{array}{l} (\mathsf{pk_S}, \mathsf{T}, \tau, p) \mid \exists(\mathsf{cred}, b_p) : \\ \quad p = g^{b_p} \cdot \prod \vec{U}^{\circ \mathsf{cred.attr} \| \mathsf{cred.un}} \\ \quad \wedge (\mathsf{stmt} = (\mathsf{pk_S}, \tau, \mathsf{id(T)}), \mathsf{wit} = (\mathsf{cred.sk})) \in \mathcal{L}_{\mathsf{VRF}} \\ \quad \wedge \left(\begin{array}{l} \mathsf{stmt} = (\mathsf{pk_S}), \\ \mathsf{wit} = ((\mathsf{cred.sk}), \mathsf{cred.attr} \| \mathsf{cred.un}, \mathsf{cred}.\sigma) \\ (\mathsf{stmt}, \mathsf{wit}) \in \mathcal{L}_{\mathsf{VerPBS}} \end{array}\right) \end{array}\right\}$$

An AND-composition of both proofs can be created, by using the same blinding value for S_1 in $\mathcal{L}_{\mathsf{VerPBS}}$ and for sk in $\mathcal{L}_{\mathsf{VRF}}$. $\tau := g^{(\mathsf{sk}+\mathsf{id(T)})^{-1}}$ is the participation tag of a participant with credential cred (using its secret key cred.sk) for the task T with unique identifier id(T). The concatenated protocol is also extended by an additional vector pedersen commitment to bind it to the other statements in $\mathcal{L}_{\mathsf{Participate}}$. This results in the following interactive sigma-style proof. P picks $d \xleftarrow{\$} \mathbb{Z}_q$, $a_1, a_2 \xleftarrow{\$} \mathbb{Z}_q$, $\vec{B} \xleftarrow{\$} \mathbb{Z}_q^{m+1}$, and a random group element $J \xleftarrow{\$} \mathbb{G}$ and computes:

$$s_1 \leftarrow \sigma_1 \cdot (V_1^{\mathsf{sk}} h \prod \vec{U}^{\circ \mathsf{attr} \| \mathsf{un}})^d \qquad \text{(re-randomized}$$
$$s_2 \leftarrow \sigma_2 \cdot g_2^d \qquad \text{signature)}$$
$$E_1 \leftarrow e(J, g_2) \cdot e(V_1^{a_1} \cdot \prod \vec{U}^{\circ \vec{B}}, s_2)^{-1} \qquad \text{(commitment PBS)}$$
$$E_2 \leftarrow e(g, \tau)^b \qquad \text{(commitment VRF)}$$
$$E_3 \leftarrow g^{a_2} \cdot \prod \vec{U}^{\circ \vec{B}} \qquad \text{(commitment binding)}$$

The participant sends E_1, E_2, E_3, p to the organizer and gets a challenge $c \xleftarrow{\$} \mathbb{Z}_q$ back from the organizer. The last message is computed as follows:

$$\vec{Y} \leftarrow [B_i + c \cdot M_i]_{i=1}^m$$
$$z_1 \leftarrow a_1 + c \cdot d \qquad z_2 \leftarrow {s_1}^c \cdot J \qquad z_3 \leftarrow a_2 + \cdot c \cdot b_p$$

The statement is verified by the organizer with these three equations:

$$\begin{aligned} E_1 \cdot {\mathsf{pk_S}}^c \cdot e(h, s_2)^c &= e(z_2, g_2) \cdot e(\prod \vec{U}^{\circ \vec{Y}}, s_2)^{-1} \\ E_2 \cdot e(g, g_2)^c \cdot e(g, \tau)^{-c \cdot \mathsf{id(T)}} &= e(g_2, \tau)^{z_1} \\ E_3 \cdot p^c &= g^{z_3} \cdot \prod \vec{U}^{\circ \vec{Y}} \end{aligned}$$

Prerequisites. The second part of the participation proof requires a participant to show that they qualify for the participation with ChkQual. In our model this essentially means that they satisfy all attribute constraints, have previously participated in every qualifier task, and have not participated in any disqualifier task. $\mathcal{L}_{\mathsf{CheckQual}}$ takes all previous transactions BB and iterates over all qualifier tasks $\mathsf{T}_q \in \mathsf{T}.\bar{\delta}$ and selects the set $Q\tau$ as tags from transactions which contain T_q. Similarly $D\tau$ is specified for all disqualifiers. The participation tag τ is used to bind $\mathcal{L}_{\bar{\delta}}$ and $\mathcal{L}_{\delta}$ to the credential used in $\mathcal{L}_{\mathsf{CheckCredTag}}$, whereas the pedersen commitment p is used in $\mathcal{L}_{\mathsf{Satisfy}}$.

$\mathcal{L}_{\mathsf{CheckQual}} :=$

$$\left\{\begin{array}{l} (\mathsf{BB} := [(\mathsf{T}_i, \cdot, \tau_i)], \mathsf{T}, \tau, p) \mid \exists(\mathsf{cred}, b_p) : \\ \forall \mathsf{T}_q \in \mathsf{T}.\bar{\delta} : \begin{cases} Q\tau := \{\mathsf{tx}.\tau | \mathsf{tx} \in \{\mathsf{tx}' \in \mathsf{BB} | \mathsf{tx}'.\mathsf{T} = \mathsf{T}_q\}\} \\ \mathsf{stmt} = (Q\tau, \tau, \mathsf{T}, \mathsf{T}_q), \\ \mathsf{wit} = (\mathsf{cred}.sk) \\ (\mathsf{stmt}, \mathsf{wit}) \in \mathcal{L}_{\bar{\delta}} \end{cases} \\ \wedge \forall \mathsf{T}_d \in \mathsf{T}.\delta : \begin{cases} D\tau := \{\mathsf{tx}.\tau | \mathsf{tx} \in \{\mathsf{tx}' \in \mathsf{BB} | \mathsf{tx}'.\mathsf{T} = \mathsf{T}_d\}\} \\ \mathsf{stmt} = (D\tau, \tau, \mathsf{T}, \mathsf{T}_d), \\ \mathsf{wit} = (\mathsf{cred}.sk) \\ (\mathsf{stmt}, \mathsf{wit}) \in \mathcal{L}_{\delta} \end{cases} \\ \wedge \forall c \in \mathsf{T.constr} : \begin{cases} \mathsf{stmt} = (\tau, c, p), \\ \mathsf{wit} = (\mathsf{cred.attr}, \mathsf{cred}.username, b_p) \\ (\mathsf{stmt}, \mathsf{wit}) \in \mathcal{L}_{\mathsf{Satisfy}} \end{cases} \end{array}\right\}$$

To exploit the repetitive structure, we further divide this proof into sub languages for a single qualifier task T_q and a single disqualifer task T_d. To bind the conjunction of the proofs for every (dis-)qualifying task, we use the current task tag τ which assures that the participant uses the same credential secret key cred.sk to calculate the tags correctly. These statements also include the public set of participation tags ($Q\tau$, $D\tau$) from all previous participations in the (dis-)qualifier.

Qualifier. For a qualifying task T_q, the participant computes their correct participation tag for this task T_q and shows that VRF.Eval$(\mathsf{cred.sk}, \mathsf{T}_q)$ is part of the set of qualifying tags $Q\tau$. Additionally, it is necessary to show that the secret key of the participant cred.sk matches the secret key used to derive the participation tag τ for the task T where the participant wants to qualify for.

$\mathcal{L}_{\bar{\delta}} :=$

$$\left\{\begin{array}{l} (Q\tau, \tau, \mathsf{T}, \mathsf{T}_q) \mid \exists(\mathsf{sk}) : \\ \quad \mathsf{VRF.Eval}(\mathsf{sk}, \mathsf{T}_q) \in Q\tau \\ \quad \wedge \mathsf{VRF.Eval}(\mathsf{sk}, \mathsf{T}) = \tau \end{array}\right\}$$

Using the VRF construction ($\tau = g^{\frac{1}{\mathsf{sk}+\mathsf{id(T)}}}$), this corresponds to a ring signature on the set of qualifying tags $Q\tau$, where the participant has the correct sk bound by τ to equate the discrete logarithm of element $Q\tau_j$ at position j.

$\mathcal{L}'_{\bar{\delta}} :=$

$$\left\{\begin{array}{l} (Q\tau, \tau, \mathsf{T}, \mathsf{T}_q) \mid \exists(\mathsf{sk}, j) : \\ \quad \tau = g^{\frac{1}{\mathsf{sk}+\mathsf{id(T)}}} \\ \quad \wedge Q\tau_j = g^{\frac{1}{\mathsf{sk}+\mathsf{id}(\mathsf{T}_q)}} \end{array}\right\}$$

This language is directly provable in $\mathcal{L}_{\mathsf{LMR19}}$.

Disqualifier. For disqualifiers the participant also evaluates the verifiable random function for the disqualifying task T_d and shows that the tag VRF.Eval$(\mathsf{cred.sk}, \mathsf{T}_d)$ is not in the set of disqualifying tags $D\tau$. It is again necessary to to also show that the same secret key cred.sk is used to compute the disqualifying tag and the actual participation tag τ.

$$\mathcal{L}_{\delta} := \left\{ \begin{array}{l} (D\tau, \tau, \mathsf{T}, \mathsf{T}_d) \mid \exists(\mathsf{sk}) : \\ \quad \mathsf{VRF.Eval}(\mathsf{sk}, \mathsf{T}_d) \notin D\tau \\ \quad \land\ \mathsf{VRF.Eval}(\mathsf{sk}, \mathsf{T}) = \tau \end{array} \right\}$$

This is achieved by choosing a secret blinding factor $r \xleftarrow{\$} \mathbb{Z}_q$ to calculate a blinded version of the disqualifying tag $R\tau \leftarrow \tau^r$ and the set of equally blinded disqualifying tags ($\forall i \in \{1, \ldots, |D\tau|\} : RD\tau_i \leftarrow D\tau_i^r$). Hence, allowing the participant to proof in zero-knowledge that the challenge set was correctly blinded (i.e., exponentiated with the random factor r), as well as the correct computation and blinding of the disqualifying tag. The verifier is then able to check whether the blinded disqualifying tag $R\tau$ is equal to any blinded tag in the challenge set $RD\tau$, without learning the participant's real tag $\mathsf{VRF.Eval}(\mathsf{cred.sk}, \mathsf{T}_d)$ for the disqualifying task T_d. This provides participation privacy for future participations with the same disqualifying task.

$$\mathcal{L}'_{\delta} := \left\{ \begin{array}{l} (D\tau, RD\tau, R\tau, \tau, \mathsf{T}, \mathsf{T}_d) \mid \exists(\mathsf{sk}, r) : \\ \quad \tau = g^{\frac{1}{\mathsf{sk}+\mathsf{id}(\mathsf{T})}} \\ \quad \land R\tau = g^{\frac{r}{\mathsf{sk}+\mathsf{id}(\mathsf{T}_d)}} \\ \quad \land \forall i \in [|D\tau|] : RD\tau_i = D\tau_i^r \end{array} \right\}$$

Again this language is provable in the $\mathcal{L}_{\mathsf{LMR19}}$ language.

Satisfy. For attribute constraints, we differentiate between range and element constraints, which we split up into two separate languages $\mathcal{L}'_{\mathsf{Satisfy,rng}}$ and $\mathcal{L}'_{\mathsf{Satisfy,ele}}$. Based on the type of constraint c is either composed as $c_{\mathsf{rng}} := (l, u, i)$ with lower bound l, upper bound u, and index i of the referenced attribute, or as $c_{\mathsf{ele}} = (V, i)$ with set of allowed element V and referenced attribute index i. Only the statement of one language can be valid at any time, but for simplicity we denote the choice of sublanguage as a logical or in the following language. Both languages use the pedersen commitment p for showing equality of the attributes used in $\mathcal{L}_{\mathsf{CheckCredTag}}$.

$$\mathcal{L}_{\mathsf{Satisfy}} := \left\{ \begin{array}{l} (\tau, c, p) \mid \exists(\mathsf{attr}, \mathsf{un}, b_p) : \\ (\mathsf{stmt} = (c.l, c.u, c.i, p), \mathsf{wit} = (\mathsf{attr}, \mathsf{un}, b_p)) \in \mathcal{L}_{\mathsf{Satisfy,rng}} \\ \lor(\mathsf{stmt} = (c.V, c.i, p), \mathsf{wit} = (\mathsf{attr}, \mathsf{un}, b_p)) \in \mathcal{L}_{\mathsf{Satisfy,ele}} \end{array} \right\}$$

Range constraints use two binary decompositions to show that the referenced attribute value attr_i is in the interval of $[l, u]$. This can be expressed as $v_l := \mathsf{attr}_i - l$ and $v_u := u - \mathsf{attr}_i$ being a positive integers.

$$\mathcal{L}'_{\mathsf{Satisfy,rng}} := \left\{ \begin{array}{l} (l, u, i, p) \mid \exists(\mathsf{attr}, \mathsf{un}, b_p) : \\ \quad p = g^{b_p} \cdot \prod \vec{U}^{\circ \mathsf{attr} \| \mathsf{un}} \\ \quad \land\ l + v_l = \mathsf{attr}_i \land v_l \in \{0, \ldots, 2^{\lceil log_2 |u-l| \rceil + 1} - 1\} \\ \quad \land\ u = \mathsf{attr}_i + v_u \land v_u \in \{0, \ldots, 2^{\lceil log_2 |u-l| \rceil + 1} - 1\} \end{array} \right\}$$

Element constraints are essentially a ring signature similar to qualifier prerequisites, where the participant's attribute equates the discrete logarithm of the element in V with index j.

$$\mathcal{L}'_{\mathsf{Satisfy,ele}} := \left\{ \begin{array}{l} (V, i, p) \mid \exists(\mathsf{attr}, \mathsf{un}, b_p, j) : \\ \quad p = g^{b_p} \cdot \prod \vec{U}^{\circ \mathsf{attr} \| \mathsf{un}} \\ \quad \land V_j = \mathsf{attr}_i \end{array} \right\}$$

Both languages can be expressed in $\mathcal{L}_{\mathsf{LMR19}}$ and are bound to $\mathcal{L}_{\mathsf{CheckCredTag}}$ using the shared commitment p.

Reward Blinding. The correct blinding of a reward key can be shown as a statement of $\mathcal{L}_{\mathsf{PBSR}}$ (c.f., Section C). Additionally, the matching to the vector pedersen p is shown by an AND composition using the same challenge in the final message of the sigma protocol.

$$\mathcal{L}_{\mathsf{RewardBlinding}} := \left\{ \begin{array}{l} (r', p) \mid \exists(\mathsf{cred}, \mathcal{N}, \rho, b_p) : \\ \quad p = g^{b_p} \cdot \prod \vec{U}^{\circ \mathsf{cred.attr} \| \mathsf{cred.un}} \\ \quad \land\ (\mathsf{stmt} = (r'), \mathsf{wit} = ((\mathcal{N}, \mathsf{cred.un}), \rho)) \in \mathcal{L}_{\mathsf{PBSR}} \end{array} \right\}$$

Efficiency. $\mathcal{L}_{\mathsf{CheckCredTag}}$ requires a total of $13 + 5m$ point additions, $13 + 5m$ scalar multiplications, and 3 pairing operations for the proof and $9 + 2m$ additions, $10 + 2m$ multiplications, and 6 pairings for the verification. The proof of $\mathcal{L}_{\mathsf{PBSR}}$ requires 4 point additions, 6 scalar multiplications, and 0 pairing operations. The verification requires 3 additions, 4 multiplications, and 0 pairing operations.

The witness length of $\mathcal{L}_{\mathsf{CheckQual}}$ in $\mathcal{L}_{\mathsf{LMR19}}$ is as follows:

$$\begin{aligned} & 10 + m && \text{(binding)} \\ & + \textstyle\sum_{i=0}^{|\bar{\delta}|} 2 + |Q\tau_i| && \text{(qualifier)} \\ & +\ 4 \cdot |\delta| && \text{(disqualifier)} \\ & + \textstyle\sum_{i=0}^{|\mathsf{constr,ele}|} 1 + |\{e_i\}| && \text{(element constraints)} \\ & + \textstyle\sum_{i=0}^{|\mathsf{constr,rng}|} 2 \cdot \lceil 1 + log_2(|u_i - l_i|) \rceil && \text{(range constraints)} \end{aligned}$$

The necessary operations for the full composition of $\mathcal{L}_{\mathsf{Participate}}$ is strongly dependent on the complexity of the study. For better comprehensibility, we define $\kappa(\mathsf{T})$ as the nuber of group elements required to encode the prerequisites of a task T:

$$\begin{aligned} \kappa(\mathsf{T}) = & 10 + m + 2|\mathsf{T}.\bar{\delta}| + 4|\mathsf{T}.\delta| + |\mathsf{T.constr, ele}| + 2|\mathsf{T.constr, rng}| \\ & + \textstyle\sum_{i=0}^{|\mathsf{T}.\bar{\delta}|} |Q\tau_i| \\ & + \textstyle\sum_{i=0}^{|\mathsf{T.constr,ele}|} |\{e_i\}| \\ & + \textstyle\sum_{i=0}^{|\mathsf{T.constr,rng}|} \lceil 1 + log_2(|u_i - l_i|) \rceil \end{aligned}$$

Adding the necessary operations of $\mathcal{L}_{\mathsf{CheckCredTag}}$, $\mathcal{L}_{\mathsf{PBSR}}$, and $\mathcal{L}_{\mathsf{CheckQual}}$ in $\mathcal{L}_{\mathsf{LMR19}}$, we get the following for proving p and verifying v:

$$\begin{aligned} |\mathsf{p}_\times| &= 18 + 5m && + 4\,\kappa(\mathsf{T}) + 2 \cdot \lceil \log_2 \kappa(\mathsf{T}) \rceil + && 2^{\lceil \log_2 \kappa(\mathsf{T}) \rceil} \\ |\mathsf{p}_+| &= 24 + 5m && + 4\,\kappa(\mathsf{T}) + 2 \cdot \lceil \log_2 \kappa(\mathsf{T}) \rceil + && 26 \cdot 2^{\lceil \log_2 \kappa(\mathsf{T}) \rceil} \\ |\mathsf{p}_e| &= 3 \\ \\ |\mathsf{v}_\times| &= 22 + 2m && + 2\,\kappa(\mathsf{T}) + 2 \cdot \lceil \log_2 \kappa(\mathsf{T}) \rceil + && 2^{\lceil \log_2 \kappa(\mathsf{T}) \rceil} \\ |\mathsf{v}_+| &= 26 + 2m && + 2\,\kappa(\mathsf{T}) + 2 \cdot \lceil \log_2 \kappa(\mathsf{T}) \rceil + && 21 \cdot 2^{\lceil \log_2 \kappa(\mathsf{T}) \rceil} \\ |\mathsf{v}_e| &= 6 \end{aligned}$$

D.2 Payout Proof

The payout proof allows a participant to prove ownership of up to n reward coins of respective value v_i that sum up to at least the payout amount $v \leq \sum_{i=0}^{n} v_i$. Every coin is a PBS signature by the service with a value v_i and bound to a nullifier $\mathcal{N}_i$ and the username un of a participant, both hidden from the service when signing. To prevent double spending, participants have to reveal the nullifier $\mathcal{N}_i$ for each spent reward and prove their correctnes. The exact number of inputs is hidden by padding the input set to a fixed length of n inputs. Padding inputs are first requested from the service, similar to regular participations. The validity of a payout request is shown with a statement of the following language.

$$\mathcal{L}_{\mathsf{Payout}} := \left\{ \begin{array}{l} \left(\{\mathcal{N}_i\}_{i=1}^{n}, \mathsf{un}, v\right) \mid \exists\left(\{(r_i, v_i)\}_{i=1}^{n}\right) : \\ \quad \sum_{i=1}^{n} v_i \geq v \\ \quad \wedge\ \forall i \in [n] : \mathsf{PBSR.Verify}(\mathsf{pk}_{\mathsf{S,R}}, (\mathcal{N}_i, \mathsf{un}), (v_i), r_i) = 1 \end{array} \right\}$$

Substituting the concrete instantiation from our building blocks, we get the language $\mathcal{L}'_{\mathsf{Payout}}$. We allow to claim less payout than the sum of all rewards $\sum_{i=1}^{n} v_i \geq v$ which is implemented as a positive difference v^*. This helps participants to hide themselves in a larger anonymity set of e.g., students who all claim the same amount equal to the required subject hours for their study programme. Again, we split this language into two sub languages, for an efficient proof. The composition of the two sub languages is bound by adding separate pedersen commitments for each input p_i to the statement each blinded by random $a_i \xleftarrow{\$} \mathbb{Z}_q$, computed as $p_i \leftarrow g_i^{a} \cdot V_1^{\mathcal{N}_i} \cdot V_2^{\mathsf{un}} \cdot U_1^{v_i}$.

$$\mathcal{L}'_{\mathsf{Payout}} := \left\{ \begin{array}{l} \left(\{\mathcal{N}_i, p_i\}_{i=1}^{n}, \mathsf{un}, v\right) \mid \exists\left(\{(r_i, v_i, a_i)\}_{i=1}^{n}\right) : \\ \quad \mathsf{stmt} = (\{\mathcal{N}_i, p_i\}_{i=1}^{n}, \mathsf{un}, v), \\ \quad \mathsf{wit} = (\{(r_i, v_i, a_i)\}_{i=1}^{n}), \\ \quad (\mathsf{stmt}, \mathsf{wit}) \in \mathcal{L}'_{\mathsf{PayoutSum}} \\ \quad \wedge\ (\mathsf{stmt}, \mathsf{wit}) \in \mathcal{L}'_{\mathsf{CheckReward}} \end{array} \right\}$$

The correctness of the re-randomizations is shown using statements of $\mathcal{L}'_{\mathsf{CheckReward}}$. The range proof that that the sum of inputs and the positive difference v^* is equal to the payout value v is proven with a statement of $\mathcal{L}'_{\mathsf{PayoutSum}}$ using a bit decomposition of v^*. Depending on the average value of rewards, the bit range $\mathfrak{B}$ of v^* may be adapted. For reference, in our prototype we selected a range of $\mathfrak{B} = 8$, to match our use case of subject hours in an empirical study programme, where typically one payout of 30 hours equals one ECTS.

$$\mathcal{L}'_{\mathsf{PayoutSum}} := \left\{ \begin{array}{l} \left(\{\mathcal{N}_i, p_i\}_{i=1}^{n}, \mathsf{un}, v\right) \mid \exists\left(\{(r_i, v_i, a_i)\}_{i=1}^{n}\right) : \\ \quad \forall i \in [n] : p_i = {\mathsf{pk}_{\mathsf{S,R}}}^{a_i} \cdot V_1^{\mathcal{N}_i} \cdot V_2^{\mathsf{un}} \cdot U_1^{v_i} \\ \quad \wedge\ \sum_{i=1}^{n} v_i = v + v^* \wedge v^* \in \{0, \ldots, 2^{\mathfrak{B}} - 1\} \end{array} \right\}$$

The language $\mathcal{L}'_{\mathsf{PayoutSum}}$ is directly representable in $\mathcal{L}_{\mathsf{LMR19}}$. $\mathcal{L}'_{\mathsf{CheckReward}}$ can be expressed as multiple iterations of $\mathcal{L}_{\mathsf{VerPBS}}$ bound to $\mathcal{L}'_{\mathsf{PayoutSum}}$ using individual pedersen commitments.

$$\mathcal{L}'_{\mathsf{CheckReward}} := \left\{ \begin{array}{l} \left(\{\mathcal{N}_i, p_i\}_{i=1}^{n}, \mathsf{un}, v\right) \mid \exists\left(\{(r_i, v_i, a_i)\}_{i=1}^{n}\right) : \\ \quad \forall i \in [n] : \begin{cases} \mathsf{stmt} = (\mathsf{pk}_{\mathsf{S,R}}), \\ \mathsf{wit} = ((\mathcal{N}_i, \mathsf{un}), (v_i), r_i) \\ (\mathsf{stmt}, \mathsf{wit}) \in \mathcal{L}_{\mathsf{VerPBS}} \\ \wedge\ p_i = {\mathsf{pk}_{\mathsf{S,R}}}^{a_i} \cdot V_1^{\mathcal{N}_i} \cdot V_2^{\mathsf{un}} \cdot U_1^{v_i} \end{cases} \end{array} \right\}$$

For better comprehensibility, we denote the set of generators of PBSR (V_1, V_2, U_1) as $\vec{\mathfrak{G}}$, and the set of coin messages $(\mathsf{un}, \mathcal{N}_i, v_i)$ as $\vec{\mathfrak{M}}_i$. The input signatures are first re-randomized with random $d_i \in \mathbb{Z}_q$ resulting in:

$$r_i = \sigma_{i,1} \cdot (\prod \vec{\mathfrak{G}}^{\circ \vec{\mathfrak{M}}_i} \cdot h)^{d_i} \qquad s_i = \sigma_{i,2} \cdot g_2^{d,i}$$

The participant creates a committment for each input by picking random values $\vec{B}_i \xleftarrow{\$} \mathbb{Z}_q^3$, $a_i \xleftarrow{\$} \mathbb{Z}_q$, $b_i \xleftarrow{\$} \mathbb{Z}_q$, and a random group element $J_i \xleftarrow{\$} \mathbb{G}$, and computing:

$$E_i \leftarrow e(J_i, g_2) \cdot e(\prod \vec{\mathfrak{G}}^{\circ \vec{B}_i}, \vec{s}_i)^{-1} \qquad \text{(commitment PBS)}$$

$$F_i \leftarrow g^{a_i} \cdot \prod \vec{\mathfrak{G}}^{\circ \vec{B}_i} \qquad \text{(commitment binding)}$$

$$p_i \leftarrow g^{b_i} \cdot \prod \vec{\mathfrak{G}}^{\circ \vec{\mathfrak{M}}_i} \qquad \text{(binding)}$$

The participant sends $\vec{s}, \vec{E}, \vec{F}, \vec{p}$ to the service and gets a challenge $c \xleftarrow{\$} \mathbb{Z}_q$ back. The response is computed as follows:

$$Y_i \leftarrow \left[B_{i\,j} + c \cdot \vec{\mathfrak{M}}_{i\,j}\right]_{j=1}^{3} \qquad z_{E\,i} = {r_i}^{c} \cdot J_i \qquad z_{F\,i} = {F_i}^{c} \cdot b_i$$

The statements are verified by the service by checking:

$$\begin{aligned} E_i \cdot {\mathsf{pk}_{\mathsf{S}}}^{c} \cdot e(h, s_i)^c &= e(z_2, g_2) \cdot e(\prod \vec{\mathfrak{G}}^{\circ \vec{Y}_i}, s_2)^{-1} \\ {p_i}^{c} + F_i &= g^{z_{F\,i}} \cdot \prod \vec{\mathfrak{G}}^{\circ \vec{Y}_i} \end{aligned}$$

Efficiency. The n iterations of $\mathcal{L}'_{\mathsf{CheckReward}}$ each require the participant to perform 19 point additions, 16 scalar multiplications, and 2 pairing operations. The verification requires 11 additions, 11 multiplications, and 3 pairing operations. $\mathcal{L}'_{\mathsf{PayoutSum}}$ has an encoded witness length of $|\mathfrak{B}| + 5 \cdot n$, hence requiring the following operations to compute:

$$\begin{aligned} |\mathsf{p}_\times| &= -1 \quad +4\mathfrak{B} + 20n + 2 \cdot \lceil \log_2(\mathfrak{B} + 5n) \rceil \quad + 2^{\lceil \log_2(\mathfrak{B}+5n) \rceil} \\ |\mathsf{p}_+| &= 7 \quad +4\mathfrak{B} + 20n + 2 \cdot \lceil \log_2(\mathfrak{B} + 5n) \rceil \quad + 26 \cdot 2^{\lceil \log_2(\mathfrak{B}+5n) \rceil} \\ |\mathsf{p}_e| &= 2n \\ |\mathsf{v}_\times| &= 12 \quad +2\mathfrak{B} + 10n + 2 \cdot \lceil \log_2(\mathfrak{B} + 5n) \rceil \quad + 2^{\lceil \log_2(\mathfrak{B}+5n) \rceil} \\ |\mathsf{v}_+| &= 17 \quad +2\mathfrak{B} + 10n + 2 \cdot \lceil \log_2(\mathfrak{B} + 5n) \rceil \quad + 21 \cdot 2^{\lceil \log_2(\mathfrak{B}+5n) \rceil} \\ |\mathsf{v}_e| &= 3n \end{aligned}$$

Using the example parameters used for our prototype ($\mathfrak{B} = 8, n = 10$), the full $\mathcal{L}_{\mathsf{Payout}}$ proof can be computed in 2105 point additions, 531 multiplications, and 20 pairing operations. The verification requires 1599 additions, 314 multiplications, and 30 pairing operations.

D.3 Analysis

By inspection, our PrePaMS construction is correct according to Theorem 3.1.

Theorem D.1 (Participation Security). *Given an unforgeable* PBSC *scheme, a deterministic* VRF*, and a simulation-extractable (SE) NIZK, PrePaMS satisfies the participation security defined in Theorem 3.2.*

Proof. We require that a credential cred is binding to a specific secret key. This holds because of the unforgeability of the PBS scheme. Additionally, a tag must be binding in the sense, that only a unique pair of key and task result in this tag. Our deterministic VRF scheme satisfies this. We now show that the winning condition of PartSec has a negligible probability: Every reward transaction tx in the list of transactions BB is valid because organizers validate each participation NIZK in $\mathcal{O}$SParticipate or $\mathcal{O}$Participate and only append to BB if valid. The valid participation thereby implies a valid NIZK for the language $\mathcal{L}_{\mathsf{Participate}}$. By the extractability of the NIZK, there exists an efficient extractor $\mathcal{E}_{\mathsf{Participate}}$ which extracts the witness cred from π for the winning tx ($\mathsf{ChkPart}(\mathsf{cred}, \mathsf{BB}_t) = 1 \land \mathsf{ChkQual}(\mathsf{cred}, \mathsf{BB}[:t], \mathsf{BB}_t.T) = 0$). According to the language, cred satisfies at least $\mathsf{ChkPart}(\mathsf{cred}, \mathsf{tx}) = 1$ and $\mathsf{ChkQual}(\mathsf{cred}, \mathsf{BB}, \tau) = 1$. The ChkQual predicate together with the binding tag assures that the participant has not participated in the same task before. As BB in the statement is exactly $\mathsf{BB}[:n]$ containing all previous participations up to this point, the winning probability of the adversary is reduced to the negligible probability of forging a SE NIZK without knowledge of a witness cred.sk. Thereby PrePaMS has participation security. □

Theorem D.2 (Balance). *Given an unforgeable* PBSC *and* PBSR *scheme, simulation-extractable (SE) NIZKs and participation security (Def 3.2), PrePaMS is balanced according to Theorem 3.3.*

Proof. In the first epoch, the adversary is tasked to get a higher payout from the service as they earned through their participations. The system requires every reward transaction tx used for the payout to be in the list of valid transactions BB. Given participation security, the organizers only blind-sign a reward coin and add it to the BB with the correct amount for the completed task. All rewards to the adversary are also maintained by the oracle in state $\mathfrak{T}^0$. Therefore the only option to increase the adversary's balance is to steal from honest users or attack the payout protocol. Every payout in $\mathcal{O}$SPayout includes a valid proof π for the language $\mathcal{L}_{\mathsf{Payout}}$. With the NIZK extractor $\mathcal{E}_{\mathsf{Payout}}$, we extract a valid witness $\{(r_i, v_i)\}_{i=1}^{n}$. These are the coins r_i with a valid PBSR signature and the blinded identity un and nullifier $\mathcal{N}_i$, preventing repeated spending of the same key. The identifier prevents spending of rewards by someone else. Any change of the identity, nullifier or value by a participant would break the unforgeability of the PBSR signature. The extracted rewards v_i to be paid out are larger than the payout amount v. Preventing double spending and the greater or equal constraint on the amount leaves the adversary a negligible advantage to breaking the balance property by attacking the building blocks.

In the second epoch, the adversary has full access to old credentials and their rewards. Again, as $\mathcal{E}_{\mathsf{Payout}}$ extracts the identity at payout, only payouts accumulated in $\mathfrak{T}^1$ can be paid out to users in $\mathfrak{U}^1$, staying non-negative. The ideneity in every coin is binding due to the unforgeability of PBSC and $\mathcal{L}_{\mathsf{Participate}}$. □

Theorem D.3 (Participation Privacy). *Given blinding partially blind* PBSC *and* PBSR *schemes, a pseudo-random verifiable random function* VRF*, simulatable NIZKs and Balance, PrePaMS satisfies the participation privacy defined in Theorem 3.4.*

Proof. We show the participation privacy through a series of hybrids changing the experiment from $b = 0$ to $b = 1$.

Hyb1: The first hybrid is equal to $\mathsf{PartPriv}_0(\lambda)$. The information dependent on b is the communication in $\Pi_{\mathsf{Participate}}$, i.e. $(\pi_0, \mathsf{T}_0, \tau_0, r_0)$, but $\mathsf{T}_0 = \mathsf{T}_1 = \mathsf{T}$ is equal for both b. The oracles after the challenge only perform actions, if they were possible by both un_0 and un_1, thereby being independnt of b.

Hyb2: This hybrid uses the existence of the NIZK simulator to generate π_0 as π_{Sim} which is indistinguishable from the real proof and does not need a witness.

Hyb3: As the proof π_{Sim} is now simulated, this hybrid uses the credential cred_1 to generate a τ_1. The hybrid is indistinguishable by the pseudorandomness of the VRF.

Hyb4: This hybrid replaces the r_0 with r_1 as both are blinding and indistinguishable.

Hyb5: With the information about b available to the adversary being $(\pi_{\mathsf{Sim}}, \tau_1, r_1)$, it remains to change the simulated NIZK to a real one again, using cred_1 as witness. This is indistinguishable due to the simulatability of NIZKs.

Hyb5 is equal to $\mathsf{PartPriv}_1(\lambda)$, concluding that the adversary has a negligible advantage of breaking the participation privacy. □

E Efficiency

We summarize the efficiency of the concrete instantiation of our scheme in Table 3.

Table 3: Effiency of the concrete instantiation of the PrePaMS protocol. $\times$ indicates the number of scalar multiplications, $+$ the number of group element additions, e the number of pairing operations, and $|.|$ the communication size defined as multiples of group elements sizes ($|\mathbb{G}|$, $|\mathbb{G}_2|$, $|\mathbb{G}_T|$) and scalars ($|\mathbb{Z}_q|$).

	Computation Efficiency		
	$\times$	$+$	e
$\Pi_{\text{Register,P}}$	$9 + m$	$7 + m$	2
$\Pi_{\text{Register,S}}$	$8 + 3m$	$4 + m$	
$\Pi_{\text{Participate,P}}$	$18 + 5m + 4\kappa(\mathsf{T}) + 2 \cdot \lceil \log_2 \kappa(\mathsf{T}) \rceil + 2^{\lceil \log_2 \kappa(\mathsf{T}) \rceil}$	$24 + 5m + 4\kappa(\mathsf{T}) + 2 \cdot \lceil \log_2 \kappa(\mathsf{T}) \rceil + 26 \cdot 2^{\lceil \log_2 \kappa(\mathsf{T}) \rceil}$	3
$\Pi_{\text{Participate,S}}$	$22 + 2m + 2\kappa(\mathsf{T}) + 2 \cdot \lceil \log_2 \kappa(\mathsf{T}) \rceil + 2^{\lceil \log_2 \kappa(\mathsf{T}) \rceil}$	$26 + 2m + 2\kappa(\mathsf{T}) + 2 \cdot \lceil \log_2 \kappa(\mathsf{T}) \rceil + 21 \cdot 2^{\lceil \log_2 \kappa(\mathsf{T}) \rceil}$	6
$\Pi_{\text{Payout,P}}$	$-1 + 4\mathfrak{B} + 20n + 2 \cdot \lceil \log_2(\mathfrak{B} + 5n) \rceil + 2^{\lceil \log_2(\mathfrak{B}+5n) \rceil}$	$7 + 4\mathfrak{B} + 20n + 2 \cdot \lceil \log_2(\mathfrak{B} + 5n) \rceil + 26 \cdot 2^{\lceil \log_2(\mathfrak{B}+5n) \rceil}$	$2n$
$\Pi_{\text{Payout,S}}$	$12 + 2\mathfrak{B} + 10n + 2 \cdot \lceil \log_2(\mathfrak{B} + 5n) \rceil + 2^{\lceil \log_2(\mathfrak{B}+5n) \rceil}$	$17 + 2\mathfrak{B} + 10n + 2 \cdot \lceil \log_2(\mathfrak{B} + 5n) \rceil + 21 \cdot 2^{\lceil \log_2(\mathfrak{B}+5n) \rceil}$	$3n$

	Communication Size			
	$\cdot\lvert\mathbb{Z}_q\rvert$	$\cdot\lvert\mathbb{G}\rvert$	$\cdot\lvert\mathbb{G}_2\rvert$	$\cdot\lvert\mathbb{G}_T\rvert$
$\Pi_{\text{Register,P}}$	$(2 + m)$	2		
$\Pi_{\text{Register,S}}$		2	1	
$\Pi_{\text{Participate,P}}$	$\begin{pmatrix} 15 + m + \lvert\mathsf{T}.\bar{\delta}\rvert + 2\lvert\mathsf{T}.\delta\rvert \\ + 3\lvert\mathsf{T.constr, ele}\rvert + \sum_{c \in \lvert\mathsf{T.constr,rng}\rvert} \lvert c.V\rvert \end{pmatrix}$	$\begin{pmatrix} 12 + 2m + 2\lceil log_2(\lvert\vec{K}\rvert) \rceil + \sum_{i=0}^{\lvert\mathsf{T}.\bar{\delta}\rvert} \lvert Q\tau_i\rvert \\ + \sum_{i=0}^{\lvert\mathsf{T}.\delta\rvert} (1 + 2\lvert D\tau_i\rvert) \end{pmatrix}$	1	2
$\Pi_{\text{Participate,S}}$				
$\Pi_{\text{Payout,P}}$	$6 + n$	$4 + 2n + 2\lceil log_2(\mathfrak{B} + 5n) \rceil$	n	n
$\Pi_{\text{Payout,S}}$				